Système de recommandation sur les plateformes de micro-blogging et bulles filtrantes

Quentin Grossetti

le cnam

École doctorale Informatique, Télécommunications et Électronique (Paris)

Équipe ISID

THÈSE DE DOCTORAT

présentée par : **Quentin GROSSETTI**

soutenue le : **8 novembre 2018**

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

Spécialité : **Informatique**

Système de recommandation sur les plateformes de micro-blogging et bulles filtrantes

THÈSE dirigée par

M. DU MOUZA Cédric	*Maître de conférences (HDR) , CNAM*
Mme. CONSTANTIN Camelia	*Maître de conférences, Sorbonne Université*
M. TRAVERS Nicolas	*Professeur, École supérieure d'ingénieurs Léonard-de-Vinci*

RAPPORTEURS

Mme. AMER-YAHIA Sihem	*Directrice CNRS, Laboratoire LIG*
M. CAUTIS Bogdan	*Professeur des Universités, Université d'Orsay LRI*

EXAMINATEURS

M. AMANN Bernd	*Professeur des Universités, Sorbonne Université*
M. VODISLAV Dan	*Professeur des Universités, Université de Cergy-Pontoise*
M. BOUGANIM Luc	*Directeur de recherche, INRIA Saclay-Île de France*

Abstract

With the unprecedented growth of user-generated content produced on microblogging platforms, finding interesting content for a given user has become a major issue. However due to the intrinsic properties of microblogging systems, such as the volumetry, the short lifetime of posts and the sparsity of interactions between users and content, recommender systems cannot rely on traditional methods, such as collaborative filtering matrix factorization. After a thorough study of a large `Twitter` dataset, we present a propagation model which relies on homophily to propose post recommendations. Our approach relies on the construction of a similarity graph based on retweet behaviors on top of the Twitter graph. We then conduct experiments on our real dataset to demonstrate the quality and scalability of our method. Finally, we investigate community detection algorithms and we present a metric to compute the strength of the *filter bubble*. Our results show that *filter bubble* effects are in fact limited for a majority of users. We find that, counter-intuitively, in most cases recommender systems tend to open users perspectives. However, for some specific users, the bubble effect is noticeable and we propose a model relying on communities to provide a list of recommendations closer to the user's usage of the platform.

Keywords : Recommender System, Collaborative Filtering, Microblogging, Twitter, Communities, Filter Bubble

Résumé

Avec la croissance sans précédent des publications sur les plateformes de micro-blogging, trouver du contenu intéressant pour un utilisateur est devenu un enjeu majeur. Cependant, en raison des propriétés intrinsèques des plateformes de micro–blogging, comme le flux gigantesque de messages arrivant tous les jours et leur faible durée de vie, il est difficile d'appliquer les méthodes traditionnelles de recommandation comme la factorisation matricielle. Après une étude approfondie d'un large jeu de donnée issu de `Twitter`, nous présentons un modèle de propagation qui repose sur l'homophilie présente dans le réseau pour recommander des messages aux utilisateurs. Notre approche s'appuie sur la construction d'un graphe de similarités lié aux interactions des utilisateurs. Nous présentons plusieurs expérimentations pour démontrer la qualité de prédiction de notre modèle et sa capacité à passer à l'échelle. Enfin, nous évaluons différents algorithmes de détections de communautés, qui permettent d'évaluer l'impact des systèmes de recommandations sur l'isolement communautaire des utilisateurs. Nous proposons une métrique permettant de quantifier la force des *bulles filtrantes* et nos résultats montrent que cet effet de *bulle filtrante* est en réalité limité pour une majorité d'utilisateurs. Il semble que, de façon contre intuitive, dans la majorité des cas les systèmes de recommandation ouvrent les perspectives des utilisateurs. Cependant, une minorité de personnes est concerné par l'effet de bulle et nous proposons donc un modèle reposant sur les liens entre communautés pour adapter les recommandations afin d'être plus en accord avec leur profil communautaire.

Mots clés : Recommandation, Filtrage Collaboratif, Micro-blogging, Twitter, Communautés, Bulles Filtrantes

Remerciements

Mes premiers remerciements vont naturellement à mon directeur de thèse Cédric du Mouza ainsi qu'à mes encadrants Nicolas Travers et Camelia Constantin qui m'ont accompagné tout au long de ce travail de thèse avec profonde bienveillance. Leurs conseils, encouragements ainsi que leurs qualités scientifiques et humaines ont permis l'aboutissement de cette thèse.

Je n'aurai pu rêver meilleur encadrement et je me sens extrêmement chanceux d'avoir réalisé ce travail avec eux.

Je remercie chaleureusement Sihem Amer-Yahia, Bogdan Cautis, Bernd Amann, Dan Vodislav et Luc Bouganim d'avoir accepté d'être membres de mon jury de thèse.

Un énorme merci à ma famille, mon père Michel tout d'abord pour ses conseils, ses relectures et ses encouragements constants dans les périodes de doutes qui peuvent parfois parsemer le parcours d'un doctorant. Mon frère, Jean-Baptiste, pour son aide et ses idées, toujours prompt à m'aider même à des heures tardives. Enfin ma mère, Véronique, pour son soutien sans faille.

Merci à toute l'équipe ISID de m'avoir permis de m'intégrer et de m'épanouir au sein du laboratoire. Un clin d'œil particulier à tous les doctorants, Pierre-Henri Paris, Subhi Issa, Noura Herradi, Odette Sangupamba Mwilu et Francesco Foscarin. Merci également à Ryadh Dahimene qui m'a conseillé avec sagesse lors du début de ma thèse.

Un merci tout particulier à Marie Colombier pour sa patience, son aide et sa compréhension. Particulièrement lors des périodes de soumission d'article.

Enfin je remercie mes amis Antoine Brunet, Jean Disset, Vincent Arlat, Adrien Peslerbe, Damian Bursztyn, Alessandro Solimando, Enki Baudeigne, Bouly de Martigues, François

Tardif, Marylou Vergez, Laëtitia Schlup, Lucie Broto, Matthieu Dabertrand, Maxime Blain, Yann Simeon, Vincent Laborde, Camille Dupouy, Axel Sinclair, Alexandre Chamelat, Marianne Greenwood, Hélène Fantuz, Pierre-Loup Ducout, Thomas Roziere, Clémence Crépeau, Anais Triboulet et j'en oublie beaucoup. Vous avez tous été là quand j'ai eu besoin de vous.

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Depuis son invention dans les années 1990, le web a rencontré un succès fulgurant et a bouleversé nos sociétés. On évalue qu'en 2018 plus de 4 milliards d'êtres humains sont connectés à Internet, soit 54% de la population mondiale [WeAreSocial 2017]. Le nombre de données générées par ces utilisateurs continue de croître exponentiellement, plus de 90% des données disponibles sur Internet ont été publiés après 2015 [IBM 2017]. Dans un rapport de Cisco publié en 2017 [Cisco 2017], les auteurs indiquent que le volume de données annuel transitant sur Internet a dépassé la limite du Zettabit, soit 10^{12} Gigabit.

Le web a connu un véritable essor technologique à partir des années 2000 permettant aux plateformes internet de devenir de plus en plus dynamiques. Les utilisateurs ne sont plus cantonnés à un simple rôle d'observateur mais interagissent directement avec les services en publiant ou partageant du contenu. C'est la naissance du web 2.0, déjà prédit avec beaucoup de précision par Danny DiNucci en 1999 dans la revue *Print* [DiNucci 1999]. Elle prévoyait que le web allait devenir de plus en plus interactif et s'immiscer dans tous les objets du quotidien, de la voiture au téléphone portable en passant par la télévision.

Cette transition technologique combinée à l'avènement du téléphone intelligent et à l'évolution des usages des individus, explique en partie cette énorme croissance de données. Certains acteurs ont réussi à acquérir une place privilégiée dans le trafic d'informations sur Internet. Ainsi `Google`, `YouTube`, `Facebook`, `Amazon`, `Instagram`, `Twitter` occupent les

premières places du classement des sites les plus consultés [1]. Ces plateformes proposent des services très différents, allant du réseau social à l'e-commerce en passant par le moteur de recherche.

Mais en centralisant une quantité croissante d'informations, ces plateformes font face à un défi commun, celui de stocker et de traiter cette masse d'information, ce qui a conduit à l'émergence du terme *Big Data*. Ces problématiques ont stimulé le développement d'alternatives aux bases de données relationnelles permettant de stocker le flot d'informations de manière distribuée en réalisant une série de compromis entre cohérence et disponibilité des données.

Parallèlement à ces technologies permettant de stocker l'information, des méthodes pour traiter de larges volumes de données distribuées ont vu le jour comme MapReduce ou plus récemment Spark.

Une des manières de mettre à profit cette quantité d'information consiste, pour les plateformes, à mettre en place des algorithmes de personnalisation pour maximiser l'engagement des utilisateurs. `Facebook` aide par exemple les utilisateurs à se connecter les uns aux autres en suggérant des amitiés potentielles en exploitant la topologie du réseau. Dans la même logique, le fil d'actualité est également filtré pour mettre en avant les messages des personnes avec qui l'utilisateur interagi le plus fréquemment. `Amazon` propose des produits selon l'historique d'achat de ses clients et `Google` personnalise ses actualités et ses résultats en fonction du profil de ses visiteurs, leur localisation géographique ou l'historique de leurs recherches par exemple. Les systèmes de filtrage et de recommandation se sont ainsi retrouvées au centre des préoccupations de ces services et se sont démocratisés. Plus de 80% du contenu que les utilisateurs décident de regarder sur `Netflix` a été découvert grâce au système de recommandation de la plateforme [Plummer 2017]. En 2013, `Amazon` déclarait que 35% de ses bénéfices provenait des achats réalisés via le système de recommandation [Mackenzie 2013].

Parmi ces plateformes du web 2.0, un type de réseau social particulier a connu un franc succès, celui du micro-blogging. Ainsi des services comme `Twitter`, `Pinterest`, `Instagram` ou `Weibo` ont connu une croissance très importante. `Twitter` est ainsi passé de 50 millions

1. https ://www.alexa.com/topsites

d'utilisateurs en 2010 a plus de 350 millions en 2018. On comptait environ 30 millions d'utilisateurs sur `Instagram` en 2011, et la plateforme culmine aujourd'hui à plus d'un milliard d'usagers. Plus de 500 millions de messages sont publiés tous les jours sur `Twitter` en 2018 et 100 millions de photos sont postés par jour sur `Instagram`.

Si ces plateformes ont une audience différente, elles partagent toutes les mêmes mécanismes. Un utilisateur a ainsi la possibilité de publier de courts messages, `Twitter` limite ainsi la taille d'un message a 280 caractères. `Instagram` limite jusqu'à 10 photos par publication. Les utilisateurs peuvent décider de suivre d'autres utilisateurs, c'est une forme d'abonnement qui affichera les futures publications de cette personne dans leurs propres fils d'actualité. Les utilisateurs peuvent également partager une publication et la transmettre de cette manière à leurs abonnés.

Avec ce même objectif de personnalisation, ces plateformes de micro-blogging ont mis en place des algorithmes permettant de trouver de nouveaux comptes à suivre. Cependant, pour recommander du contenu, il a fallu trouver de nouvelles stratégies s'adaptant aux caractéristiques de ces plateformes.

En effet, recommander du contenu sur ces plateformes est un réel défi en raison de la diversité des usages des utilisateurs et de la difficulté à extraire des caractéristiques à partir de très courts messages. Trouver les messages pertinents parmi les 500 millions de messages publiés chaque jour pour `Twitter` avant que ceux-ci ne soient caducs nécessite le développement de méthodes passant massivement à l'échelle.

1.2 Notre Problématique

Avec l'émergence des plateformes de micro-blogging vues précédemment et l'incroyable déluge d'information qui y est publié tous les jours combiné à la création de nouveaux utilisateurs et de nouveaux liens dans le réseau, utiliser un système de recommandation efficace qui passe à l'échelle est un véritable défi. Notre objectif est de trouver des solutions pour recommander du contenu lorsque les méthodes traditionnelles de recommandation ne permettent pas de passer à l'échelle efficacement. Nous souhaitons extraire les caractéristiques uniques des plateformes de micro-blogging afin d'améliorer notre compréhension de

ces systèmes et construire un modèle adéquat. Avec l'explosion des données sur Internet, la capacité du modèle à traiter facilement un large volume de données et à se mettre à jour est absolument cruciale.

Parallèlement, avec la place grandissante de ces systèmes de recommandation dans nos expériences numériques, analyser les changements qu'ils opèrent sur notre accès à l'information permet d'éclairer l'impact que produisent ces systèmes sur notre société. En effet, cet impact est encore mal connu et il s'agira de trouver des méthodes d'analyse permettant de quantifier la bulle filtrante chez les individus. Enfin, nous nous efforcerons de construire un modèle qui permette de lutter contre ce phénomène.

1.3 Contributions

Dans cette thèse, nous présentons plusieurs analyses permettant de mieux comprendre le fonctionnement de `Twitter`. Ces analyses portent sur la durée de vie des messages publiés par les utilisateurs, sur la topologie du réseau et sur la répartition des similarités entre utilisateurs selon leur distance dans le graphe. Nous proposons un modèle permettant de recommander efficacement du contenu aux utilisateurs. Ce modèle repose sur la construction d'un graphe de similarités entre utilisateurs, réalisable à faible coût grâce à l'homophilie présente dans le réseau. Ce modèle est inspiré des méthodes de filtrage collaboratif qui exploitent la similarité entre utilisateurs.

Nous montrons qu'en appliquant un algorithme de propagation convergent et optimisé, notre modèle est capable de passer à l'échelle et de recommander un message aux utilisateurs rapidement après sa publication. Cette propagation permet d'exploiter naturellement la transitivité entre les similarités des utilisateurs, ce qui peut être vu comme une alternative à la réduction de dimensions habituellement utilisées par les systèmes de recommandation.

Nous montrons également les différents résultats induits par l'utilisation de divers systèmes de recommandation. Nous nous concentrons sur le nombre de bonnes prédictions réalisées par chaque système ainsi qu'à leur zone de prédiction. Nous montrons que certains systèmes ont tendance à être efficaces sur la prédiction de messages populaires, quand d'autres sont à l'opposé orientés sur la recommandation de messages confidentiels. Nous

comparons également les temps de calculs nécessaires à l'initialisation et à la mise en place des différents modèles. Les plateformes de micro-blogging évoluant en permanence nous proposons une stratégie de mise à jour de notre modèle permettant de passer à l'échelle efficacement.

Afin de discerner l'impact des systèmes de recommandation sur la consommation d'information nous proposons une évaluation des bulles filtrantes sur `Twitter`. Nous comparons différentes méthodes de détection de communautés sur `Twitter` et observons finement le type des communautés produites par ces algorithmes. Cette analyse de l'aspect communautaire de `Twitter` permet par le suite de quantifier l'effet de bulle produit par les algorithmes de recommandations.

Nous proposons ainsi une métrique permettant de quantifier cette bulle pour un utilisateur donné, et proposons un modèle qui fait intermédiaire entre un système de recommandation et l'utilisateur. En utilisant les liens entre les communautés tels que les liens topologiques, les liens sémantiques et les liens d'influences, nous montrons qu'il est possible de limiter cet effet de cloisonnement des informations.

1.4 Organisation de la thèse

Après ce chapitre d'introduction, nous présentons dans le chapitre 2 l'état de l'art des systèmes de recommandations. Après un rapide historique du domaine, nous verrons les différentes approches permettant de recommander du contenu à des utilisateurs. Nous présenterons ainsi les méthodes de filtrage collaboratif, les méthodes orientées contenu et les méthodes de factorisation matricielles. Nous étayons dans cette partie les difficultés pour mettre en place les méthodes traditionnelles de recommandation sur une plateforme de micro-blogging telle que `Twitter`.

Dans le chapitre 3 nous présentons les résultats de différentes analyses réalisées sur un large jeu de données issu de `Twitter`. Ces analyses s'intéressent à la topologie du réseau, à la durée de vie des messages et à la caractérisation de l'homophilie présente dans le réseau. Ces observations permettent d'améliorer notre compréhension des plateformes de micro-blogging et de construire un modèle adapté à cette structure.

Ensuite dans le chapitre 4 nous présentons notre système de recommandation qui repose sur la combinaison d'un graphe de similarités entre utilisateurs et d'un algorithme de propagation d'informations. En comparant notre modèle à d'autres systèmes issus de l'état de l'art nous montrons que notre modèle surpasse les autres approches en terme de pertinence et de temps de calcul. Nous proposons également une approche pour mettre à jour notre modèle au gré de l'évolution des intérêts des utilisateurs et des nouveaux liens qui se tissent dans le réseau. Une stratégie se détache comme étant particulièrement efficace permettant de mettre à jour notre modèle à bas coût, garantissant ainsi l'utilisation de notre modèle en production.

Dans le chapitre 5, nous comparons différentes méthodes de détection de communautés sur notre même jeu de données issu de `Twitter`. Après une analyse des différents partitionnement du réseau produits par ces algorithmes, nous utilisons les communautés détectées pour caractériser le phénomène des bulles filtrantes. Nous tentons ainsi de discerner si les systèmes de recommandation provoquent un enfermement communautaire, dans un premier temps à l'échelle d'une communauté, puis à l'échelle d'un utilisateur. Nous proposons ensuite une méthode de reclassement des listes de prédictions issues de différents systèmes de recommandation afin de limiter l'effet de bulle pour les utilisateurs concernés.

Enfin le chapitre 6 résume les différents résultats obtenus durant ce travail de thèse et présente différentes perspectives de recherche.

Chapitre 2

État de l'art

2.1 Historique

L'histoire académique des systèmes de recommandation n'est pas évidente à retracer, mais deux principaux articles peuvent être notés comme étant à l'origine du concept. Jussi Karlgren [Karlgren 1990] alors jeune chercheur suédois en séjour à Columbia écrit un rapport technique de 11 pages dans lequel il décrit un modèle algébrique permettant de recommander des documents à des utilisateurs.

Ce rapport porte déjà la marque de nombreux questionnements qui ne cesseront de traverser la communauté des systèmes de recommandation. Il distingue ainsi les informations implicites qui sont collectées directement à partir des activités des utilisateurs, des informations explicites qui correspondent à un avis de la part de l'utilisateur. Le modèle de Karlgren utilise déjà un modèle de filtrage collaboratif qui repose sur les similarités entre utilisateurs pour faire des recommandations, avec l'intuition que des gens qui se comportent de la même façon vont avoir tendance à se comporter de la même façon dans le futur.

On sent dans ce texte déjà l'ambition de créer une bibliothèque culturelle numérique qui aiderait les utilisateurs à référencer leurs goûts et leur collection d'œuvres afin de trouver de nouveaux contenus. Il cite l'exemple d'une bibliothèque numérique de livres qui serait en mesure de connaitre les goûts de ses utilisateurs afin de leur fournir des idées de livres à lire.

De façon assez ironique son travail n'aura qu'un impact très limité dans la sphère

académique car sa publication dans la conférence INTERACT de 1990 s'est vue refuser au motif qu'utiliser les données des utilisateurs empièterait trop sur leur vie privée. Il faudra donc attendre 1992 pour voir le premier article clairement cité par la communauté internationale, article qui propose le terme "filtrage collaboratif" [Goldberg, Nichols, Oki, and Terry 1992]. Alors chercheurs à Xerox, les auteurs décrivent un système nommé Tapestry qui devrait, en théorie, remplacer le système de messagerie électronique de l'entreprise afin de réduire la quantité astronomique de messages que les employés reçoivent. L'idée est assez directe, les lecteurs de messages sont invités à les annoter ("excellent" ou "inutile" par exemple) afin de donner leurs avis. Chaque utilisateur peut filtrer ses messages par contenu, par exemple "Service Comptabilité" et par utilisateurs "Eric ayant annoté le document comme excellent". Il revient donc aux utilisateurs de faire des requêtes de filtrage, on parle alors de filtrage collaboratif à la demande.

Mais l'article le plus abouti proviendra en 1994 du laboratoire GroupLens de l'université du Minnesota [Resnick, Iacovou, Suchak, Bergstrom, and Riedl 1994]. L'idée de cet article consiste à utiliser des notes laissées par des utilisateurs sur des groupes Usenet pour recommander des articles, c'est le premier modèle à reposer sur une similarité entre utilisateurs et à travailler sur une matrice utilisateurs/contenu. Le groupe de chercheurs derrière ce premier travail s'est ensuite engouffré dans la thématique des systèmes de recommandation. Ils ont eu l'intuition que la difficulté de trouver des jeux de données d'avis d'utilisateurs allait s'avérer critique et ils ont mis en place la plateforme MovieLens, un système public de recommandation de films. Les données récoltées par cette plateforme ont eu beaucoup d'impact car elles ont stimulé les recherches sur le sujet.

Ces données sont toujours accessibles [GroupLens]. On voit comment dès le départ l'intuition pour construire des systèmes de recommandation consistait à s'appuyer sur les similarités entre utilisateurs ou entre items, c'est à dire des méthodes de filtrage collaboratif. Le succès du système de recommandation d'Amazon mis en place en 1998, dont on peut voir une capture d'écran de l'époque en figure 2.1 a également joué un rôle important dans leur popularisation industrielle et académique. Rapidement les liens entre industriels et chercheurs sont devenus extrêmement ténus dans ce champ de recherche. Il faudra attendre 2003 pour qu'Amazon rende public, succinctement, leur algorithme dans un article qui

deviendra un classique de la communauté [Linden, Smith, and York 2003]. La simplicité du modèle et sa faculté à passer à l'échelle a probablement joué un rôle important dans son essor. En résumé l'idée de l'article consiste à décaler le positionnement des systèmes de recommandation jusque-là orientés utilisateurs vers des systèmes orientés contenu. Pour chacun des objets de la matrice, c'est à dire les produits vendus par Amazon, il est possible de calculer à froid les objets les plus similaires. Si leur algorithme semble donner des résultats satisfaisants, le système présente des problèmes de calcul en temps-réel et de qualité de recommandation.

L'étape clé suivante des systèmes de recommandation a été la création du prix Netflix en 2006. A l'époque, Netflix ne fait pas encore de vidéos à la demande et propose pour un abonnement mensuel une location illimitée de DVD. Ils créent alors un prix de 1 million de dollars pour récompenser l'équipe capable d'améliorer les performances de leur algorithme de 10% avec l'idée de stimuler la recherche sur le sujet. Ce cap des 10% d'amélioration sera dépassé en 2009 par l'équipe « BellKor's Pragmatic Chaos » composée d'un mélange de chercheurs d'université publiques et privés (AT&T, Yahoo !) [Töscher, Jahrer, and Bell 2009]. La solution gagnante est une succession d'étape de factorisation imaginé par Yehuda Koren [Koren, Bell, and Volinsky 2009] qui deviendra un classique des modèles de recommandation. Netflix considèrera cependant que l'ajout énorme de complexité pour obtenir ces résultats n'est pas un compromis convenable et n'utilisera pas la solution proposée. Ce sera la dernière édition du prix Netflix, puisque des chercheurs de l'université d'Austin [Narayanan and Shmatikov 2006] démontrent que, même si les données rendues publiques par Netflix sont en apparence anonymisées, il est possible de retrouver les identités des utilisateurs du jeu de données en faisant un rapprochement avec les données IMDB. Quatre utilisateurs de Netflix intenteront même un procès estimant que le partage de leurs données est une infraction aux règles de confidentialité de la plateforme.

Loués pour avoir participé au succès d'Amazon et de Netflix, les systèmes de recommandation traversent actuellement une période de doutes de la part des citoyens : ces systèmes sont-ils responsables de notre enfermement ? De miner l'espace démocratique que représentent les réseaux sociaux actuels en nous montrant systématiquement des opinions en accord avec les nôtres ?

FIGURE 2.1: Capture d'écran du site Amazon le 13 octobre 1999, le système de recommandation a détecté 4 livres proches de Harry Potter et le prisonnier d'Azkaban dont les deux premiers livres de la saga Harry Potter

2.2 Données

Pour recommander du contenu pertinent à un individu il est nécessaire de comprendre ses goûts et donc de construire un profil de ses intérêts. Afin de construire ce profil utilisateur, les systèmes de recommandation peuvent se reposer sur deux types de données.

On distinguera ainsi les données **implicites**, par exemple le temps passé à regarder une vidéo ou bien le nombre de fois qu'un album de musique a été écouté. Une information qui est donc intrinsèquement liée à l'usage de l'utilisateur. L'alternative étant d'utiliser des données **explicites** comme par exemple des notes de 1 à 5, un *like*, un partage, bref une information volontairement déposée par l'utilisateur. Si l'information explicite est plus difficile à collecter car elle nécessite la volonté de l'utilisateur de donner son avis, elle permet d'avoir des avis négatifs sur des objets. Avec une information implicite il est plus difficile d'inférer qu'un objet ne plaît pas à l'utilisateur [Rendle, Freudenthaler, Gantner, and Schmidt-Thieme 2009]. Il existe un long débat issu des sciences sociales pour déterminer

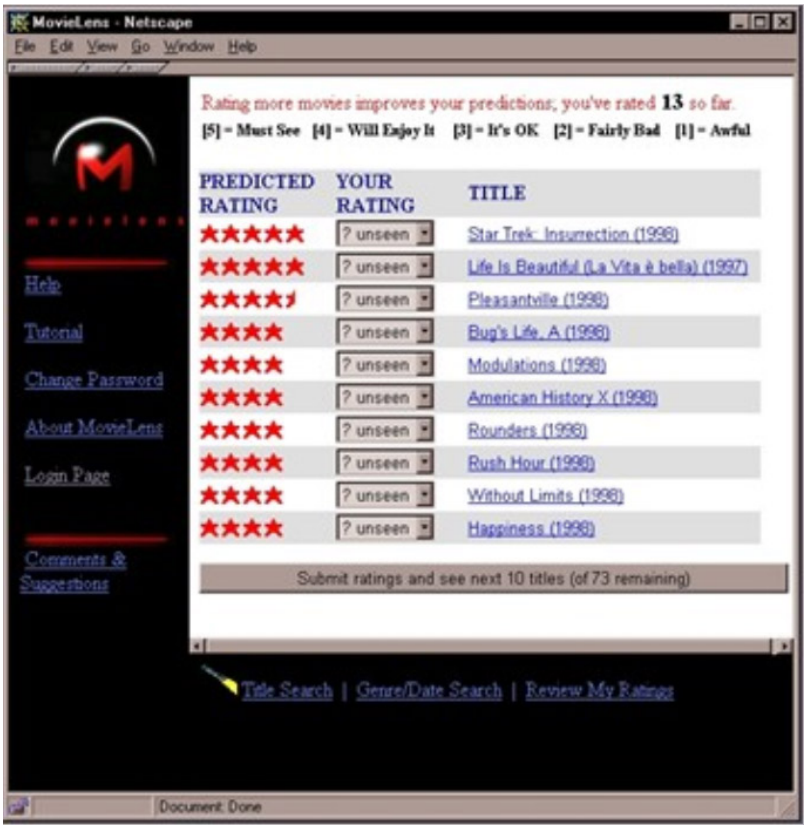


FIGURE 2.2: Capture d'écran datant de 1999 du site MovieLens, le système de recommandation de films du laboratoire de l'université du Minnesota GroupLens

quelles données sont les plus pertinentes à utiliser. En effet le problème de représentation de soi est un problème bien connu. Par exemple lors de recueil d'opinions concernant les usages télévisuels, la chaine Arte communément identifiée comme une chaine d'intellectuels, a tendance à être surreprésentée dans les résultats comparés aux audiences mesurées par télémétrie. Il y a donc un écart entre la façon dont les individus se décrivent et la façon dont ils se comportent.

La compétition pour remporter le prix Netflix, qui a contribué à dynamiser la recherche sur les systèmes de recommandation, nécessitait de traiter exclusivement des données explicites ce qui a biaisé la recherche académique sur ce type de données.

Dans les faits, il existe peu de travaux comparants directement la qualité des résultats obtenus à partir de données implicites ou explicites. Cela s'explique par la difficulté de

trouver des jeux de données contenant des données implicites, habituellement gardées précieusement par les entreprises privées qui peuvent les collecter directement sur leurs utilisateurs. Cependant, il existe quelques travaux qui s'intéressent à cette question, ainsi dans un article, Jawaheer et al [Jawaheer, Szomszor, and Kostkova 2010], comparent la qualité de prédiction d'un système de recommandation musical en se servant respectivement de données implicites ou de données explicites.

Ils obtiennent des résultats très comparables quelle que soit la source des informations utilisateurs et préconisent l'utilisation hybride des différents types de données. En revanche une étude comparative de différents algorithmes de recommandation selon s'ils utilisent des données implicites ou explicites [Zhao, Harper, Adomavicius, and Konstan 2018], explique que, de façon paradoxale, l'utilisation de données implicites nuit à la précision globale du système mais suscite davantage d'engagement de la part des utilisateurs. Le meilleur engagement des utilisateurs en utilisant des données implicites peut expliquer l'évolution des systèmes qui au départ utilisaient des données explicites qui progressivement utilisent de plus en plus de données implicites.

Dans les faits cette question passe souvent au second plan, les systèmes de recommandation étant des systèmes opportunistes utilisant le maximum des données disponibles, implicites ou explicites.

2.3 Recommandation orientée contenu

L'idéologie derrière les systèmes orientés contenu consiste à extraire des caractéristiques significatives d'un objet afin d'être en mesure de le décrire. Par exemple, Spotify [Dieleman 2014] travaille directement sur le signal audio des morceaux afin de déterminer leur genre musical, leur ambiance sonore ou leur rythme, dans le but de faire des suggestions aux utilisateurs aimant ce type de musique. S'il est possible d'avoir accès à une vision fine du contenu dans le cas de la musique, ce n'est pas toujours le cas, extraire des caractéristiques représentatives d'un *tweet* est une tâche beaucoup plus difficile par exemple. La plupart des systèmes orientés contenu passent par une représentation textuelle du contenu et des utilisateurs afin de retomber sur des problématiques de recherche d'informations plus

classiques. Au travers du modèle vectoriel, on peut représenter un utilisateur ou un document dans un espace à n-dimension, n correspondant au nombre de termes, ou le vocabulaire, de la collection de documents. Par exemple si on considère qu'un utilisateur a lu des articles avec une forte composante "Informatique, Java, Programmation" par exemple, on pourra lui recommander les documents qui s'expriment fortement sur ces dimensions. On peut retrouver de tels systèmes pour recommander des articles de journaux [Ahn, Brusilovsky, Grady, He, and Syn 2007], des sites internet [Lieberman et al. 1995], [Mladenic 1999], de la musique [Celma, Ramírez, and Herrera 2005] [Celma and Serra 2008] ou des produits d'e-commerce [Xu, Zhang, Pan, and Yang 2005]. Chaque système s'adapte suivant les informations disponibles sur le contenu, mais la logique du modèle vectoriel reste la même.

Il est possible de dépasser cette vision par mots-clés en intégrant des concepts plus globaux, par exemple en utilisant la sémantique comme dans [de Gemmis, Lops, Musto, Narducci, and Semeraro 2015]. Ces modèles tendent à avoir de bons résultats lorsque suffisamment d'informations peuvent être extraites du contenu et qu'on possède une bonne représentation des goûts des utilisateurs. Un des problèmes est la non-représentation qualitative du contenu. Si on reprend l'exemple d'un article qui aborderait les sujets "Informatique, Java et Programmation", détecter que celui-ci correspond au profil d'un utilisateur est possible, cependant il existe pléthore d'articles sur le sujet. Comment bien choisir alors ? De la même façon il est compliqué de proposer du contenu qui sort du cadre d'intérêt de l'utilisateur, ce qui pose des questions en termes de découverte de nouveaux contenus.

Les systèmes de recommandation sont rarement totalement orientés contenu, il s'agit bien souvent de combiner une approche orientée contenu avec une approche par filtrage collaboratif, les premières permettant souvent de combler des défauts des secondes. En effet, afin d'être en mesure de recommander un objet, les méthodes par filtrage collaboratif ont besoin de disposer d'informations sur les interactions entre les utilisateurs et le contenu. Dans ce cas, comment recommander un objet sur lequel il n'existe pas encore d'interaction, un morceau de musique qui vient de sortir par exemple ? C'est le problème du démarrage à froid, ou *cold-start*. En extrayant des caractéristiques de cet objet et en le proposant aux utilisateurs ayant un profil qui semble apprécier ce type d'œuvre les systèmes orientés

contenu arrivent à combler ce manque. Cette hybridation entre différents systèmes sera plus précisément abordée dans la partie 2.5.

2.4 Recommandation par filtrage collaboratif

Les systèmes de recommandation par filtrage collaboratif reposent sur l'intuition que les utilisateurs ayant eu un comportement similaire dans le passé vont avoir tendance à se comporter de manière similaire dans le futur. On peut aussi le voir comme une formalisation du bouche à oreille présent dans la vie quotidienne. Cette intuition fondatrice a été un des moteurs des premiers modèles, cités dans l'introduction. On considère que des utilisateurs interagissent avec du contenu, ce qui peut se formaliser sous la forme d'une matrice utilisateur/contenu (des films, des livres, des articles de journaux) comme illustré en figure 2.3. Dans cette matrice $\mathbf{R}$, de taille $N * M$ avec N utilisateurs et M œuvres, chaque ligne correspond à un utilisateur et chaque colonne correspond à une œuvre.

Chaque élément qui compose la matrice correspond à l'opinion d'un utilisateur sur le contenu, cette opinion peut au choix se traduire sous la forme d'une information implicite (temps passé devant une vidéo) ou explicite (une note entre 1 et 5) comme expliqué précédemment. Cette matrice est la clé de voute des systèmes de filtrage collaboratif, où l'objectif pour le modèle va être de prédire au mieux les trous de la matrice, c'est à dire d'être capable pour un utilisateur de prédire l'opinion qu'il aurait eu sur du contenu qu'il n'a pas encore évalué. Dans la matrice d'exemple, quelle serait la note de Claire sur le film Gladiator ? Une fois capable de prédire les notes manquantes, le contenu recommandé est bien souvent la liste ordonnée des œuvres non notées avec la plus forte valeur de prédiction.

Une première intuition afin de prédire les goûts des utilisateurs consiste à capter les similarités entre les utilisateurs, ou entre les œuvres. Ces modèles reposent sur la quantification de similarité entre vecteurs de la matrice $\mathbf{R}$, et nécessitent des métriques capables d'évaluer précisément la distance entre deux utilisateurs ou deux films dans notre exemple.

Si la littérature regorge de mesures de similarités différentes, les plus utilisées sont

	Harry Potter	Gladiator	Her	Titanic	Matrix	Jurrasic Park
Bob	5	3	2		1	4
Alex		5	2	3	5	1
Étienne	3	4	4	1	4	5
Marie	1	5		5	5	
Claire	5	?	5	1	1	4

FIGURE 2.3: Exemple de matrice **R** utilisateurs / contenu ici des films. Les utilisateurs ont donné un avis compris entre 1 et 5 sur des films, l'objectif est de prédire la note de Claire sur le film Gladiator.

décrites dans la table 2.1. Théoriquement chaque distance est particulièrement adaptée au type de données qu'on désire manipuler, binaires comme un pouce bleu ou un pouce rouge ou n-aire comme une note sur 10. Dans les faits, les différences entre les résultats obtenus à partir de chaque mesure de similarité restent extrêmement faibles. Cependant la similarité de Pearson apparait comme le meilleur compromis combinant une bonne qualité des prédictions et une légèreté de calcul [Gunawardana and Shani 2009] ou dans [Patra, Launonen, Ollikainen, and Nandi 2015]. Cette qualité a fait de la similarité de Pearson un standard de la communauté et une des mesures les plus utilisées dans la littérature sur le filtrage collaboratif. Une fois ces similarités calculées, il est possible de prédire un "trou" dans la matrice, c'est à dire faire une prédiction de note entre un utilisateur et un objet en utilisant au choix une somme pondérée des notes des personnes similaires ou une moyenne.

Une première méthode pour prédire ces notes, consiste à travailler sur les colonnes, c'est à dire les œuvres. On va chercher dans la matrice les œuvres les plus similaires à une œuvre cible. C'est le cas d'Amazon par exemple, qui affiche les livres aimés par les mêmes personnes qui ont aimé Harry Potter (les livres similaires). Une autre stratégie, consiste à appliquer une mesure de similarité non plus sur les colonnes mais sur les lignes, c'est le modèle orienté utilisateur ([Breese, Heckerman, and Kadie 1998], [Herlocker, Konstan, Borchers, and Riedl 1999], [Jin, Chai, and Si 2004], [Resnick, Iacovou, Suchak, Bergstrom, and Riedl 1994]). Il faut ainsi trouver dans un premier temps les utilisateurs qui ont des goûts

en commun avec Claire pour pouvoir prédire sa note sur Gladiator. A partir de là, deux stratégies sont possibles, soit de prendre en compte les similarités de tous les utilisateurs pour prédire la note, soit de prendre uniquement le top-N utilisateurs les plus proches : c'est l'approche des plus proches voisins. Un exemple de cette méthode est disponible en figure 2.4.

L'approche orientée colonne est généralement utilisée, car dans la plupart des cas, la matrice **R** possède davantage de lignes que de colonnes, se reposer sur les colonnes est donc préférable pour des questions de passage à l'échelle et de densité de la matrice [Sarwar, Karypis, Konstan, and Riedl 2001], [Deshpande and Karypis 2004]. Des modèles essayent de tirer parti des avantages des deux stratégies afin de remplir au maximum les informations manquantes dans la matrice **R** [Wang, De Vries, and Reinders 2006].

Mesure	Équation	Type de données
Corrélation de Pearson	$\frac{\sum_{i \in L_u \cap L_v} (r_{ui} - \bar{r_u}) \cdot (r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in L_u \cap L_v} (r_{ui} - \bar{r_u})^2} \cdot \sqrt{\sum_{i \in L_u \cap L_v} (r_{vi} - \bar{r_v})^2}}$	Généraliste
Jaccard	$\frac{\lvert L_u \cap L_v \rvert}{\lvert L_u \cup L_v \rvert}$	Binaire
Cosinus	$\frac{\sum_{i \in L_u \cap L_v} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in L_u \cap L_v} r_{ui}^2} \cdot \sqrt{\sum_{i \in L_u \cap L_v} r_{vi}^2}}$	Généraliste
Cosinus ajusté	$\frac{\sum_{i \in L_u \cap L_v} (r_{ui} - \bar{r_u}) \cdot (r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in L_u \cap L_v} (r_{ui} - \bar{r_i})^2} \cdot \sqrt{\sum_{i \in L_u \cap L_v} (r_{vi} - \bar{r_j})^2}}$	Généraliste
Distance Euclidienne	$\sqrt{\sum_{i \in L_u \cap L_v} (r_{ui} - r_{vi})^2}$	Généraliste
Corrélation de Spearman	$\frac{\sum_{i \in L_u \cap L_v} (\delta_{ui} - \bar{\delta_i}) \cdot (\delta_{vi} - \bar{\delta_i})}{\sqrt{\sum_{i \in L_u \cap L_v} (\delta_{ui} - \bar{\delta_i})^2} \cdot \sqrt{\sum_{i \in L_u \cap L_v} (\delta_{vi} - \bar{\delta_i})^2}}$	N-aire

TABLE 2.1: Liste non exhaustive des principales mesures de similarités utilisées par des systèmes de filtrage collaboratif. Avec L_u l'ensemble du contenu noté par u, respectivement L_v pour l'utilisateur v. r_{ui} la note de l'utilisateur u sur l'œuvre i (respectivement r_{vi})). $\bar{r_u}$ est la note moyenne de u, $\bar{r_i}$ est la note moyenne pour l'œuvre i. δ_{ui} est la position de l'œuvre i dans le classement de ses œuvres notés et $\bar{\delta_i}$ la position moyenne de i dans tous les classements.

Calculer les similarités afin de produire des recommandations est difficilement applicable lorsque la matrice **R** devient trop volumineuse. En effet calculer les similarités d'un utilisateur est de complexité O(MN) avec M le nombre d'œuvres et N le nombre d'utilisateurs. Or, pré-calculer les similarités entre utilisateurs parait peu envisageable avec n^2 résultats à conserver. De plus les profils changent en permanence ce qui nécessite de rafraichir ces

$$
\begin{array}{cc}
 & \begin{array}{cccccc} HP & G & H & T & M & JP \end{array} \\
R & \begin{bmatrix} 5 & 3 & 2 & ? & 1 & 4 \\ ? & 5 & 2 & 3 & 5 & 1 \\ 3 & 4 & 4 & 1 & 4 & 5 \\ 1 & 5 & ? & 5 & 5 & ? \\ 5 & ? & 5 & 1 & 1 & 4 \end{bmatrix}
\end{array}
\begin{array}{l} Bob \\ Alex \\ \text{Étienne} \\ Marie \\ Claire \end{array}
\Rightarrow
\begin{array}{c} \text{sim(Claire,Bob)} = 0.675 \\ \text{sim(Claire,Alex)} = \text{-}0.78 \\ \text{sim(Claire,Etienne)} = 0.45 \\ \text{sim(Claire,Marie)} = \text{-}1 \end{array}
$$

$$\widehat{R}(C,G) = \frac{sim(C,B) \times R(B,G) + sim(C,E) \times R(E,G)}{sim(C,B) + sim(C,E)}$$

$$\widehat{R}(C,G) = 3.4$$

FIGURE 2.4: Prédiction de la note de Claire sur le film Gladiator a partir de la méthode des plus proches voisins. Dans cet exemple on utilise la similarité de Pearson pour comparer les utilisateurs et une somme pondérée pour prédire la note manquante. Ici, on se limite aux 2 plus proches voisins, c'est à dire Bob et Étienne qui ont les scores de similarités les plus élevés

scores.

Une stratégie pour passer à l'échelle efficacement a été de considérer le partitionnement des utilisateurs en blocs d'utilisateurs similaires [Chee, Han, and Wang 2001] afin de réduire l'espace de recherche des plus proches voisins. Cependant cette méthode ne résout pas le problème des matrices creuses. Il n'est pas possible de comparer efficacement des utilisateurs qui ont trop peu noté de contenu en commun. Pour ces raisons, les méthodes qui consistent à décomposer la matrice R en sous matrices ont rencontré un vif succès autant académique qu'industriel.

L'intuition derrière la décomposition matricielle consiste à trouver une combinaison de matrices qui réduit les dimensions de la matrice d'origine. Autrement dit, plutôt que de prendre chaque note indépendamment on suppose que des paramètres latents existent et permettent d'expliquer les goût des utilisateurs. Ainsi on représente les intérêts des

utilisateurs dans un espace réduit. On cherche ainsi à résoudre l'équation suivante :

$$R = UI^{\intercal} \tag{2.1}$$

Avec **U** la matrice des composantes des utilisateurs et **I** la matrice des composantes des items. Au départ, des méthodes de décomposition de la matrice en valeurs singulières (SVD) ont été envisagées, car cette décomposition donnait des bons résultats dans la communauté de l'apprentissage automatique en terme de précision. Cependant, l'algorithme original de cette décomposition fonctionne uniquement sur une matrice dense, c'est à dire où aucun "trou" n'est présent. C'est une situation qui est difficilement conciliable avec la réalité des systèmes de recommandation. Une première idée a été de remplir la matrice avec des scores générés artificiellement au travers de lois probabilistes ou de moyenne [Sarwar, Karypis, Konstan, and Riedl 2000]. Mais le coût de factorisation devenait sensiblement trop élevé malgré les efforts pour développer des méthodes de factorisation incrémentale [Sarwar, Karypis, Konstan, and Riedl 2002] qui évitent de devoir factoriser depuis zéro après des modifications éventuelles de la matrice (nouvelles notes).

Une autre méthode, popularisée par Simon Funk [Funk] en se classant parmi les finalistes du concours Netflix, consiste à passer outre ce manque d'information lors de l'apprentissage et à lier l'apprentissage des vecteurs U_a et I_b pour chaque utilisateur a et chaque item b, c'est l'algorithme du gradient stochastique. En effet pour chaque note de notre matrice R nous cherchons à minimiser la marge d'erreur du modèle c'est à dire :

$$err_{ab} = \widehat{R}(a,b) - U_a \times I_b^{\intercal} \tag{2.2}$$

Pour chaque note de notre matrice **R** on essaye de réduire cette erreur en la répercutant sur chaque vecteur avec :

$$\begin{aligned} U_a &\leftarrow U_a + \gamma \times (err_{ab} \times I_b^{\intercal} - \lambda \times U_a) \\ I_b &\leftarrow I_b + \gamma \times (err_{ab} \times U_a - \lambda \times I_b^{\intercal}) \end{aligned} \tag{2.3}$$

Un exemple de cette méthode est disponible en figure 2.5, avec 2 comme taille de réduction de dimension. C'est une méthode similaire qui a été utilisée pour remporter le concours Netflix [Koren, Bell, and Volinsky 2009]. Une alternative consiste à bloquer le vecteur

U_a et I_b graduellement lors de l'apprentissage : c'est la méthode alternée des moindres carrés. En pratique si l'algorithme du gradient stochastique est plus rapide, la méthode des moindres carrés est plus facilement parallélisable et donc très pratique pour traiter des matrices volumineuses. Pour éviter le problème de sur-apprentissage, on peut ajouter un biais dans l'apprentissage ou une relaxation comme dans [Takács, Pilászy, Németh, and Tikk 2007].

$$R\begin{bmatrix}5 & 3 & 2 & ? & 1 & 4\\ ? & 5 & 2 & 3 & 5 & 1\\ 3 & 4 & 4 & 1 & 4 & 5\\ 1 & 5 & ? & 5 & 5 & ?\\ 5 & ? & 5 & 1 & 1 & 4\end{bmatrix} = U\begin{bmatrix}0.23 & 1.79\\ 0.40 & 2.06\\ 1.39 & 1.39\\ 3.11 & 0.88\\ 0.60 & 2.02\end{bmatrix} I^{\intercal}\begin{bmatrix}-0.45 & 1.07 & 1.69 & 1.30 & 1.41 & 2.12\\ 2.69 & 1.90 & 1.21 & 0.46 & 0.97 & 1.12\end{bmatrix}$$

$$UI^{\intercal} = \widehat{R}\begin{bmatrix}4.61 & 3.60 & 2.52 & 1.12 & 2.04 & 2.47\\ 5.34 & 4.34 & 3.15 & 1.48 & 2.58 & 3.15\\ 3.10 & 4.13 & 3.97 & 2.45 & 3.31 & 4.51\\ 0.97 & 5.00 & 6.20 & 4.44 & 5.25 & 7.60\\ 5.15 & 4.48 & 3.43 & 1.71 & 2.80 & 3.53\end{bmatrix}$$

$$\widehat{R}(C, G) = 4.48$$

FIGURE 2.5: Prédiction de la note de Claire sur le film Gladiator à partir d'une méthode de factorisation matricielle dans un espace latent. Ici la matrice d'origine R a été factorisée en deux matrices U (utilisateurs) et I (contenu) de rang 2.

2.5 Modèles hybrides

On a vu dans les sections précédentes que les systèmes orientés contenu étaient capables de fournir des recommandations dans des situations de démarrage à froid, lorsque peu d'interactions existent entre les utilisateurs et le contenu. Cependant, ils sont incapables de fournir des recommandations qui sortent du champ du profil de l'utilisateur, et ne donnent pas des résultats aussi précis que les méthodes filtrage collaboratif. Inversement, les systèmes par filtrage collaboratif excluent totalement le contenu du modèle de recommandation ce qui rend difficile la recommandation de nouveau contenu.

Il a donc très tôt semblé logique de combiner les avantages des deux stratégies pour faire

des systèmes de recommandation hybrides comme dans [Melville, Mooney, and Nagarajan 2002],[De Campos, Fernández-Luna, Huete, and Rueda-Morales 2010] qui ajoutent un poids dans la recommandation de films directement à partir des informations sur le film (genre, résumé, acteurs etc...) si celui-ci est proche du profil utilisateur. D'autres modèles suivent cette idée de somme pondérée comme [Miranda, Claypool, Gokhale, Mir, Murnikov, Netes, and Sartin 1999], mais il est également possible d'utiliser un système de vote ou chaque modèle vote pour une recommandation [Pazzani 1999], alterner le système responsable des recommandations [Lekakos and Caravelas 2008] une fois orienté contenu une fois orienté filtrage collaboratif, [Billsus and Pazzani 2000], ou encore de reclasser la liste de suggestions a partir des résultats issus d'un autre modèle afin de raffiner des suggestions [Burke 2002]. Une méthode plus sophistiquée reposant sur des machines de Boltzmann apprend une hybridation automatiquement et permet d'optimiser la précision du système de recommandation [Gunawardana and Meek 2009].

Si les systèmes hybrides réussissent en général à améliorer les performances de recommandation, particulièrement dans le cas où l'on dispose de peu d'informations dans la matrice $\mathbf{R}$ concernant les utilisateurs ou le contenu, ils ne sont pas exempts de défauts. En effet, les modèles hybrides sont très difficiles à généraliser du fait de la démarche extrêmement spécifique liée à chaque domaine, l'hybridation entre les systèmes sera très différente selon qu'on souhaite recommander des films ou de la musique par exemple. Cependant c'est une aubaine pour les industriels, puisque les plateformes ajoutent des nouvelles informations sur les utilisateurs en continu, que ce soit leur réseau social, leur position géographique etc. Ajouter de nouvelles caractéristiques lors de l'apprentissage de façon extrêmement flexible est la principale force des réseaux de neurones qui peuvent se voir comme une hybridation particulière, qui sera détaillé en section 7. Un autre type particulier d'hybridation qui va particulièrement nous intéresser ici est l'intégration du réseau social de l'individu, c'est à dire prendre en compte les affinités des utilisateurs pour obtenir de meilleures recommandations. Ce sont les systèmes de recommandation pour plateformes sociales.

2.6 Modèles pour plateformes sociales

Avec l'arrivée des plateformes sociales sur internet, comme Facebook, Google plus, Twitter ou encore LinkedIn, les systèmes de recommandation ont pu ajouter une dimension à leur terrain de jeu : le réseau qui connecte les individus entre eux. Dans la vie on sait qu'il existe un lien entre les conseils de notre entourage et notre comportement, on peut être incité à acheter un produit ou à regarder un film, ou au contraire en être découragé à partir des goûts de nos proches. Cette dimension parait donc évidente à prendre en compte afin de fournir des recommandations plus précises.

En sociologie et en psychologie on connait la tendance des individus à se connecter à des individus qui partagent des traits communs : CSP, âge, sexe, goûts etc... C'est le phénomène de l'homophilie [McPherson, Smith-Lovin, and Cook 2001]. Une recommandation de la part d'une personne fortement connectée dans le réseau aura peut-être ainsi plus de poids que celle d'une personne totalement étrangère. Cette notion peut même se complexifier comme dans les travaux de Yang et al. [Yang, Steck, and Liu 2012b], qui considèrent plusieurs cercles autour d'un utilisateur suivant la catégorie d'objets à recommander. Si un utilisateur peut faire confiance au goût d'un proche en matière de restaurant, il sera peut-être plus dubitatif sur ses conseils musicaux. A partir de chaque cercle d'amis liés à une catégorie, il est possible d'inférer la note manquante d'un utilisateur en prenant en compte ceux d'entre ses proches qui l'influencent dans cette même catégorie.

Une autre méthode consiste à ajouter un terme lors de la prédiction d'une note, par exemple la somme pondérée des notes des amis de l'utilisateur cible comme dans [Ma, King, and Lyu 2009].

D'autres travaux vont dans le même sens en ajoutant un terme lors de la factorisation matricielle comme [Ma, Yang, Lyu, and King 2008],[Ma, King, and Lyu 2009],[Jamali and Ester 2010],[Ma, Zhou, Liu, Lyu, and King 2011]. Si ajouter le réseau social comme biais de régularisation dans la recommandation tend en moyenne à baisser la précision des prédictions, la qualité des recommandations effectivement soumises à l'utilisateur semble augmenter paradoxalement [Yang, Steck, Guo, and Liu 2012a].

De façon un peu similaire, les auteurs de [Yang, Guo, and Liu 2013] proposent d'utiliser

une probabilité conditionnelle pour mesurer la similarité entre amis d'un réseau social, avec l'idée qu'un modèle probabiliste transcrit de façon plus réaliste les liens d'influence entre individus que la confiance. On peut donc construire un réseau Bayesien superposé au réseau des utilisateurs pour générer des recommandations.

2.7 Deep-Learning

L'explosion des modèles de deep-learning et des réseaux de neurones se retrouve de façon assez naturelle dans la communauté des systèmes de recommandation. On trouve de plus en plus d'articles tentant d'adapter les méthodes de prédiction issues de ces modèles aux systèmes de recommandation. On relève ainsi un article qui cherche à recommander des articles de journaux avec des réseaux de neurones [Oh, Lee, Lim, and Choi 2014], des citations scientifiques [Huang, Wu, Chen, Mitra, and Giles 2015] et des prédictions de notes à partir d'une critique, sur Yelp par exemple, [Tang, Qin, Liu, and Yang 2015].

Les méthodes de filtrages collaboratif peuvent s'adapter sous forme de réseaux de neurones comme dans [Wang, Wang, and Yeung 2015]. L'idée est d'utiliser des réseaux de neurones pour extraire des caractéristiques des items de façon générique, à la façon des méthodes de recommandation orientées contenu, et d'injecter ces caractéristiques dans un modèle de filtrage collaboratif. Cette approche semble particulièrement efficace dans le cas où il y a très peu d'interactions entre les utilisateurs et les items. De manière assez similaire, Burges et al. [Van den Oord, Dieleman, and Schrauwen 2013], ont utilisé des réseaux de neurones pour apprendre des représentations des morceaux de musique à partir d'un signal audio, afin d'améliorer la recommandation d'items suscitants peu d'interactions. Cette approche est donc proche des méthodes hybrides, ou la recommandation s'appuie à la fois sur le filtrage collaboratif et sur l'extraction de caractéristiques des items. Une autre stratégie pour traduire des systèmes de filtrage collaboratif consiste à implémenter des auto-encodeurs comme dans les travaux de Sedhain par exemple [Sedhain, Menon, Sanner, and Xie 2015].

L'idée des auto-encodeurs centrés utilisateurs est d'apprendre des motifs cachés à partir desquels on peut reconstruire toutes les notes manquantes d'un utilisateur donné. Elkahky

et al. ont utilisés des méthodes de deep-learning pour modéliser des utilisateurs sur plusieurs domaines [Elkahky, Song, and He 2015]. Un des articles les plus connu sur le sujet est celui décrivant le système de recommandation de Youtube paru en 2016 [Covington, Adams, and Sargin 2016]. Il s'agît d'un article industriel, donc l'accent est particulièrement mis sur le passage à l'échelle de leur approche, sur le taux d'engagement des utilisateurs et sur l'environnement logiciel utilisé, celui de Google, TensorFlow [1]. Leur approche consiste à utiliser deux réseaux de neurones séparés, un pour prédire une liste d'une centaine de vidéos candidates et l'autre pour prédire un classement. La première étape est la plus intéressante, puisqu'elle peut être vue comme une généralisation des méthodes de factorisation matricielle vues précédemment. L'architecture du réseau de neurones est représentée en figure 2.6. Le réseau de neurones prend en compte au départ énormément de paramètres : l'historique de vidéos de l'utilisateurs, l'historique de ses recherches, ses informations géographique et démographique. C'est un perceptron multicouche constitué de trois couches de neurones avec des unités de rectification linéaire comme fonctions d'activation, à qui on applique un softmax en sortie. Les classes de sorties sont très nombreuses, avec l'idée qu'une classe de sortie est équivalente à une vidéo. A noter qu'en entrée, le réseau prend pour base une représentation dense des vidéos issues des mots-clés de ces vidéos, de leurs résumés etc. Un des avantages d'utiliser un tel *perceptron multicouche* est la simplicité avec laquelle on peut brancher de nouvelles données pour construire le profil de l'utilisateur.

On peut également parler de l'article de blog de l'équipe de développement du système de recommandation de Twitter qui vise à implémenter une solution de deep-learning pour faire remonter les tweets les plus importants après une "longue" absence sur la plateforme (8h) [Koumchatzky and Andryeyev]. Habituellement, sur Twitter, les messages sont classés par ordre de publication, le modèle sert à mettre en avant des messages qui auraient été publiés il y a quelque temps et que l'utilisateur aurait manqués.

En conclusion les modèles de deep-learning aident à extraire des caractéristiques directement du contenu, à ajouter à volonté de nouvelles variables et à disposer d'une représentation plus précise des utilisateurs et du contenu.

1. https ://www.tensorflow.org/

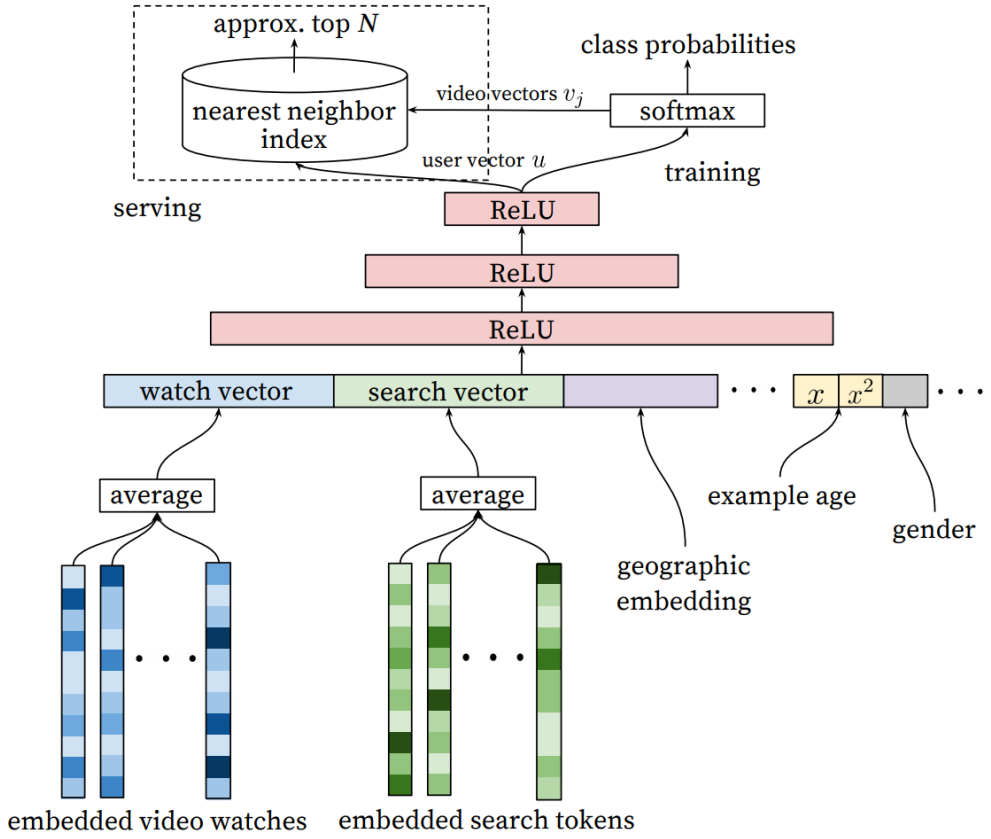


FIGURE 2.6: Architecture générale de la première étape de filtre de l'algorithme de recommandation de vidéos sur Youtube

2.8 Marches aléatoires : le cas twitter

`Twitter` dans un article de 2016 présentent leur système de recommandation de messages intitulé GraphJet [Sharma, Jiang, Bommannavar, Larson, and Lin 2016]. En réalité, GraphJet peut être vu comme une généralisation de WTF (Who To Follow), l'algorithme de recommandation d'utilisateurs de `Twitter`, proche de l'algorithme SALSA ("Stochastic Approach for Link-Structure Analysis") [Lempel and Moran 2000] utilisé pour classer les sites internet en fonction de leurs connexions à la façon du PageRank. Le calcul des recommandations de `Twitter` se fait traditionnellement par lots, environ tous les jours, et cet article témoigne de la volonté de l'entreprise d'être capable de traiter le flux en temps réels sans pré calculs. Pré-calculer des résultats paraissait contre-intuitif avec l'idée de faire de `Twitter` un système qui organise en temps réel la communication mondiale, que ce soit

des annonces politiques, du sport ou de l'activités de nos proches.
Le modèle s'appuie sur des marches aléatoires dans un graphe bipartite 2.7, une partie des nœuds de ce réseau correspond aux utilisateurs et une seconde partie des nœuds correspond aux messages. Un arc est ajouté entre les utilisateurs et les messages à chaque interaction, comme par exemple un partage (*retweet*) ou un "j'aime". Pour calculer les recommandations d'un utilisateur, les auteurs suggèrent de prendre comme point de départ le nœud correspondant dans ce graphe bipartite et de simuler un nombre X (10 000 par exemple) de marches aléatoires. Ces marches doivent obligatoirement être composées d'un nombre impair de sauts de façon à garantir un arrêt sur un message.

Il est ainsi possible d'obtenir une liste de N messages classés par fréquence d'arrêts des marches, plus on tombe sur un message plus le score de recommandation est élevé. Parmi les avantages de ce modèle on peut citer la facilité à faire des recommandations pour n'importe quel utilisateur à la volée, l'absence de pré-traitement ou de coût d'initialisation et une bonne qualité des messages recommandés. En revanche, afin de garantir la recommandation de messages relativement récents, qui est l'objectif de `Twitter`, le graphe bipartite doit progressivement retirer les liens les plus anciens. Avec ce principe de fenêtre temporelle, les interactions trop anciennes des utilisateurs sont tout simplement oubliées. Nous verrons plus loin dans la partie expérimentation qu'un autre désavantage de cette méthode est d'avoir tendance à recommander des messages populaires, ce qui est logique du fait de l'utilisation des marches aléatoires, où les messages les plus partagés ont tendance à obtenir les scores les plus importants.
Plusieurs problèmes avec les systèmes de recommandation standard ont poussé Twitter à utiliser ce modèle de marches aléatoires. Ce sont des caractéristiques liées à l'usage de la plateforme, mais qui ne sont pas exclusives à twitter et qui se retrouvent plus généralement sur toutes les plateformes de micro-blogging. Dans un premier temps, les systèmes orientés contenu ont du mal à être efficace du fait de la très courte longueur des messages (240 caractères maximum) et de la diversité du média éventuellement encapsulé dans celui-ci (vidéo, image, lien, localisation etc...).

Il est difficile d'extraire des caractéristiques pertinentes qui permettent de comprendre pourquoi le message en question suscite de l'engagement. Le second problème est la

rareté des informations disponibles ("sparsity" en anglais) de la matrice d'interactions utilisateurs/contenu. Par exemple si on compare la densité de la matrice twitter à celle de Netflix : Netflix possède un catalogue d'environ 6000 films et séries, pour 120 millions d'abonnés. En supposant que chaque utilisateur donne son avis sur en moyenne 10 œuvres, le taux de complétion de la matrice utilisateur/contenu de Netflix est de $\frac{15*120000000}{6000*120000000} = 0.025$ ce qui est déjà très faible. Comparons maintenant avec Twitter qui possède 300 millions d'utilisateurs pour 8 milliards de messages depuis la naissance du service et 100 interactions en moyenne par utilisateur. $\frac{100*300000000}{8000000000*300000000} = 0.00000001$ soit environ 200 000 fois moins. La dernière particularité est la très courte durée de vie des messages. Utiliser des systèmes de recommandation qui ont besoin d'un construire un modèle, comme une factorisation matricielle par exemple, semble inadapté car sur la totalité des 500 millions de messages entrant chaque jour il apparait comme inutile de calculer le score de recommandation de chaque message sur chaque utilisateur ou pour des messages trop anciens.

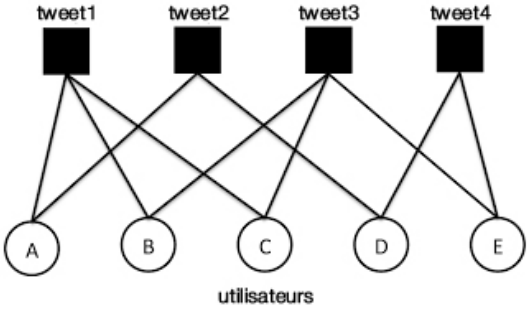


FIGURE 2.7: Graphe bipartite servant de base à GraphJet avec en bas les nœuds représentant les utilisateurs, en haut les nœuds représentant les tweets et les arcs représentant les interactions entre utilisateurs et messages

2.9 Conclusion

Nous avons vu dans ce chapitre l'évolution des techniques utilisées pour recommander du contenu à des utilisateurs. On constate la forte tendance de la recherche actuelle à s'orienter vers des méthodes d'apprentissage profond. Ces modèles permettent de réaliser un mélange subtil entre système orienté contenu et système par filtrage collaboratif tout en produisant, théoriquement, une représentation plus fine des utilisateurs par rapport aux

méthodes de factorisation matricielle.

Comprendre ces représentations reste cependant un défi, les prédictions produites par ces modèles souffrent souvent d'un effet de boite noire où comprendre les raisons qui ont poussé le système à recommander un contenu à un utilisateur est difficile à analyser. Ces méthodes, comme les modèles de factorisation matricielle souffrent également de la difficulté d'avoir des fonctions de coût capables de mesurer ce qui fait l'essence d'une bonne recommandation.

Les systèmes de recommandation continuent donc à se complexifier en même temps qu'ils se démocratisent. Cette complexité dépasse les purs aspects de pertinence des recommandations, les plateformes doivent jongler entre de nombreuses contraintes pour construire leurs algorithmes de recommandation.

Deezer par exemple utilise comme argument publicitaire le "flow", cette radio personnalisée selon les goûts de ses utilisateurs, afin de recruter de nouveaux utilisateurs. L'idée est très simple, l'utilisateur n'a qu'un bouton à cliquer pour lancer une radio qui jouera des titres censés lui plaire, alternant entre découvertes et chansons plus classiques. C'est une opportunité économique pour l'entreprise car afin de payer moins de droits d'auteurs lors des écoutes des morceaux, Deezer fait passer légalement le "flow" comme un radio, de cette manière le prix par écoute est réduit. Mais afin de légitimement pouvoir prétendre à ce statut légal, Deezer doit se soumettre à un ensemble d'exigences comme l'impossibilité de passer deux morceaux du même album en moins de 2h, devoir mettre un quota d'artistes français etc. Ainsi le modèle de recommandation doit gérer des contraintes extérieures qui n'ont pas grand-chose à voir avec la pertinence de recommandations de morceaux. Ce mélange entre exigences extérieures et "pureté" du modèle pourrait paraitre anecdotique, mais en réalité on retrouve les mêmes compromis chez toutes les plateformes entre d'un côté ce qui est pertinent pour l'utilisateur et de l'autre ce qui a un intérêt pour la plateforme.

En plus de ces contraintes économiques, les contraintes de passage à l'échelle sont cruciales. Le choix de `Twitter` d'utiliser un système de recommandation qui repose sur des marches aléatoires s'explique par la nécessité de fournir aux utilisateurs des recommandations de messages très récents. Dans un tel contexte, utiliser des méthodes de factorisation matricielle ou d'apprentissage profond reste encore une frontière. Youtube limite ainsi dans

son système de recommandation le nombre de vidéos éligibles à la recommandation avec un seuil de popularité. Une tel modèle pour `Twitter` serait incapable de recommander des messages trop récents. Trouver une méthode pour recommander efficacement du contenu sur les plateformes de micro-blogging reste un défi. Nous allons voir dans les prochains chapitres comment on peut extraire des caractéristiques essentielles de `Twitter` afin de construire un système qui passe à l'échelle, capable de recommander des messages pertinents.

Chapitre 3

Analyse de Twitter

Afin de proposer un modèle de recommandation de messages pour `Twitter` il apparait comme essentiel de disposer d'un large jeu de données représentatif de la plateforme. En effet, les jeux rendus publics sont bien souvent petits ou incomplets. Il n'existe pas de large jeu qui comporte les messages publiés par un grand nombre d'utilisateurs combiné au réseau sous-jacent qui les connecte entre eux. Ce large jeu va nous permettre dans un premier temps d'extraire des connaissances quant au fonctionnement général des plateformes de micro-blogging et dans un second temps de tester notre modèle. En effet, au vu du problème d'échelle, que ce soit en quantité d'information ou rapidité de traitement de celles-ci, il est nécessaire d'extraire les informations clés permettant de comprendre ce qui a une forte influence sur le réseau et la vie d'un tweet sur le réseau.

3.1 Construction d'un jeu de données

Collecter un jeu représentatif de Twitter est une tâche fastidieuse car le volume de données est énorme et il est nécessaire d'éviter l'effet boule de neige lors de la collecte. C'est un problème bien connu lors de l'exploration d'un réseau, où une stratégie naïve consistant à partir d'un nœud de départ et à explorer successivement les liens sortants depuis ce point produit une topologie biaisée.

Ce jeu a été constitué en plusieurs étapes. Dans un premier temps nous avons extrait une composante connexe du réseau de `Twitter` rendu disponible en 2009 par Kwak et al. [Kwak, Lee, Park, and Moon 2010]. Pour chacun des nœuds issus de cette composante,

nous avons interrogé en 2015 la plateforme afin de collecter les arcs entrants, les arcs sortants ainsi que les tweets associés aux comptes utilisateurs afin d'être à jour. En effet le réseau social réel a changé entre 2009 et aujourd'hui.

Pour ce faire, nous avons utilisé l'API [1] de `Twitter` pour récupérer les données des `followers` et des `followee` de chacun des nœuds de ce réseau. Du fait des limites de l'API de `Twitter` sur le nombre de tweets récupérés pour chaque utilisateur, nous ne pouvions récupérer que les 3 200 derniers tweets associés à chaque compte. De fait, les comptes sont bornés à cette limite, quelle que soit la fréquence de publication de ceux-ci.

Cette opération de collecte qui a duré environ 4 mois de janvier à mai 2015 et a abouti à un jeu décrit en tableau 3.1. Avec 2 millions d'utilisateurs et 3 milliards de tweets, nous avons une moyenne de 1 375 tweets par utilisateur. La composante connexe de notre réseau représente 325 millions d'arcs, avec en moyenne 69 arcs entrants par utilisateur, le maximum étant de 185. Le diamètre de ce réseau s'étend jusqu'à une distance de 15 entre deux utilisateurs avec une moyenne de parcours de 3,7. Par ailleurs, sur les 325 millions de liens unidirectionnels (followers ou followee), près de 50 millions d'entre eux sont des liens réciproques, ce qui veut dire que seulement 15% des utilisateurs se suivent mutuellement. Ce jeu est décrit plus finement au travers de diverses analyses dans les sections suivantes. A noter que la composante connexe originale du jeu de Kwak était composée de 125 millions d'arcs, le réseau s'est donc densifié avec un nombre d'arcs qui a doublé. Cela peut s'expliquer par la tendance naturelle des utilisateurs a suivre de nouvelles personnes, ou l'utilisation plus assidue de la plateforme par les utilisateurs.

Nous allons commencer notre analyse par une étude sur les tweets et retweets effectués par les utilisateurs dans ce réseau, puis nous nous intéresserons à la topologie du réseau `Twitter` ainsi qu'à la caractérisation de l'homophilie de ses utilisateurs.

3.2 Analyse des retweets

Afin de mieux comprendre les effets des tweets dans l'environnement `Twitter`, nous allons nous focaliser sur les retweets qui témoignent explicitement de l'intérêt d'un utilisateur

1. https ://dev.twitter.com/rest/public

Nombre total de nœuds	2 182 867
Nombre total d'arcs	325 451 980
Nombre total de tweet	3 001 502 711
Nombre moyen de followers	69,4
Nombre moyen de followees	57,8
Nombre maximum de followers	185 401
Nombre maximum de followees	348 595
Diamètre du réseau	15
Taille moyenne du plus court chemin	3,7
Liens reciproques	49 819 171

TABLE 3.1: Caractéristiques principales du jeu `Twitter`

pour un message. Comprendre les intérêts et les engagements des utilisateurs est une étape clé pour construire efficacement un système de recommandation.

3.2.1 Durée de vie d'un tweet

Un message peut être partagé par n'importe quel utilisateur de façon à intégrer celui-ci à ses statuts et à le mettre en avant pour ses *followers*. Dans ce contexte, on peut considérer que le temps écoulé entre la date de publication d'un message (tweet de t_1) et la dernière fois que celui-ci a été partagé (dernier retweet de t_1) peut être perçu comme sa durée de vie. Seuls les messages ayant été partagés au moins une fois sont considérés dans cette analyse (*i.e.,* durée de vie > 0). Les résultats sont visibles dans la figure 3.1.

On peut remarquer que la grande majorité des messages ont une durée de vie inférieure à une heure (40%), pour 90% elle est inférieure à 3 jours, et il est extrêmement rare pour un message d'être partagé au-delà de cette limite.

Nos résultats montrent une diminution de la durée de vie des messages comparés aux résultats de Kwak [Kwak, Lee, Park, and Moon 2010]. En 2009, cette étude montrait que des messages pouvaient être partagés jusqu'à un mois après la parution d'un message. Notre hypothèse est qu'en 2009, beaucoup moins de contenus étaient publiés sur `Twitter` et qu'il était donc encore possible d'accéder facilement à des messages anciens. De plus, les utilisateurs sont plus à l'aise avec l'utilisation de `Twitter` qu'en 2009, les comportements des utilisateurs sont davantage normalisés.

Ce qui nous paraît particulièrement intéressant, et d'autant plus lors de la conception

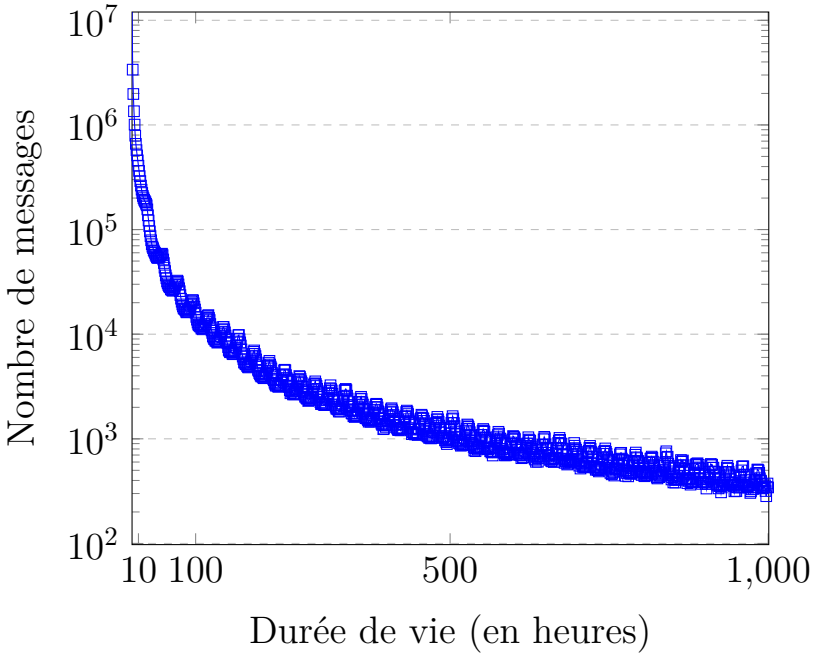


FIGURE 3.1: Durée de vie d'un message

d'un système de recommandation, c'est le caractère prépondérant de la fraîcheur d'un message pour son partage. En d'autres termes, un message perd de l'intérêt de manière exponentielle au fil du temps. Il devient alors critique de pouvoir recommander des tweets en temps réel en se fondant sur une fenêtre très courte des activités du réseau.

3.2.2 Popularité des tweets

Dans cette partie, nous nous concentrons sur la répartition de la popularité des tweets avec le nombre de retweets par message. Les résultats sont représentés dans la figure 3.2. La grande majorité des messages ne sont jamais retweetés ($\approx 90\%$), ou très peu, avec à peine 2-3 retweets ($\approx 2\%$). Ces résultats sont très cohérents par rapport à l'étude de Kwak [Kwak, Lee, Park, and Moon 2010] qui montrait également qu'une très grande partie de messages n'étaient jamais retweetés ou bien retweetés une seule fois. Seulement 0.003% des messages sont partagés plus de 50 fois (119142 messages dans notre jeu).

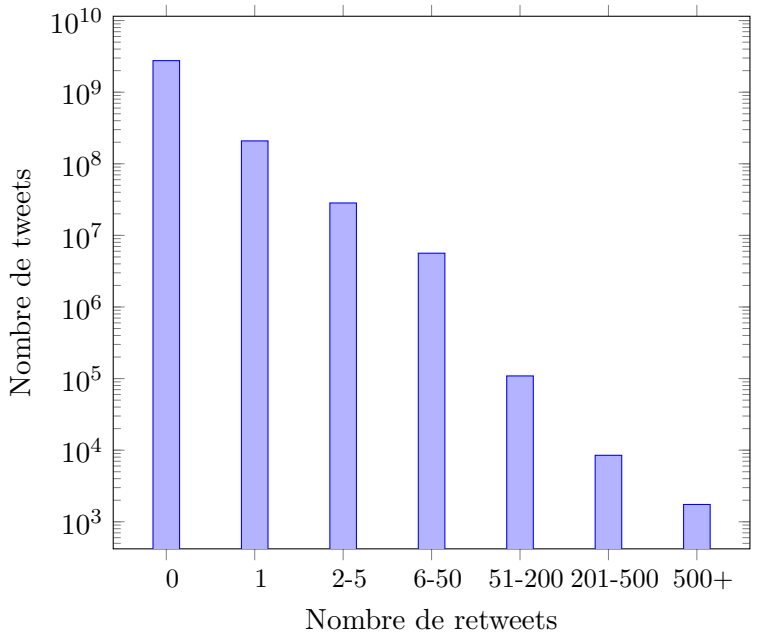


FIGURE 3.2: Répartition du nombre de retweets par message

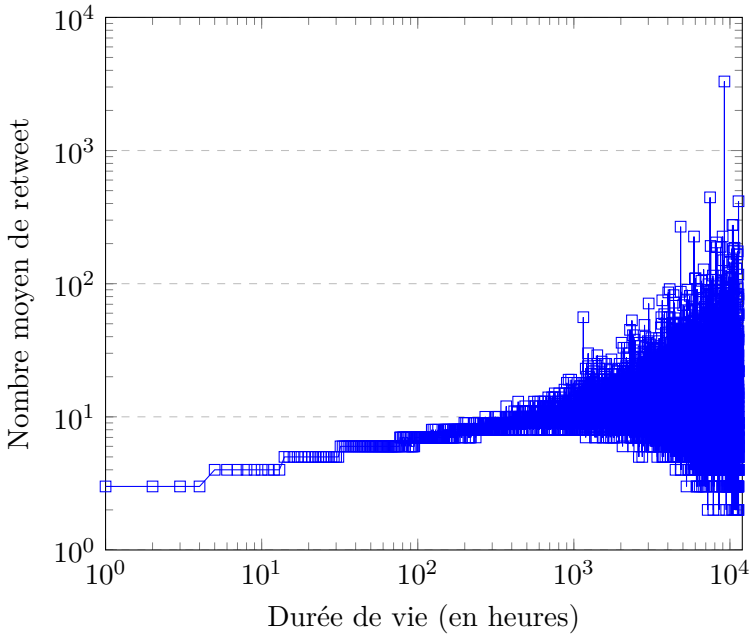


FIGURE 3.3: Corrélation entre popularité et durée de vie

3.2.3 Popularité d'un message et durée de vie

En comparant cette durée de vie en figure 3.3 avec la popularité des messages on s'aperçoit qu'il existe une forte corrélation entre les deux. De façon intuitive plus un message est partagé plus il rencontre du succès, plus il sera de nouveau partagé allongeant d'autant sa durée de vie. Si la corrélation est assez linéaire jusqu'à 1000 heures ($\approx$ un mois), la corrélation s'estompe après ce point, certains messages semblent rencontrer un succès longtemps après leur parution. Il est possible que des messages anciens redeviennent pertinents parce que le contexte a changé ou pour se remémorer un événement précis.

3.2.4 Retweets par utilisateurs

Pour terminer sur le comportement des retweets, nous allons maintenant nous intéresser à leur répartition parmi les utilisateurs. Les résultats sont présentés dans la figure 3.4, nous pouvons constater sans grande surprise que très peu d'utilisateurs font un grand nombre de retweets. En moyenne un utilisateur partage 156 messages (médiane de 37.5 messages) ce qui traduit une disparité très importante entre une masse qui retweet peu et une minorité extrêmement active. Du point de vue du système de recommandation, le problème se pose pour les utilisateurs ayant très peu de retweets, voire zéro, qui représentent une part importante des utilisateurs (un utilisateur sur 4 n'a jamais retweeté). Les méthodes qui se basent sur du filtrage collaboratif peuvent difficilement fournir des recommandations pour ces utilisateurs du fait de leur inactivité.

3.3 Analyse du réseau Twitter

Après avoir étudié le comportement des tweets et des retweets, nous allons maintenant nous intéresser à la topologie du réseau d'utilisateurs, puis aux similarités entre ces utilisateurs afin de pouvoir quantifier de façon plus précise l'homophilie du réseau.

3.3.1 Topologie générale

Nous nous intéressons dans cette section à la topologie du réseau, c'est à dire la structuration du graphe. Nous allons dans un premier temps observer la répartition des

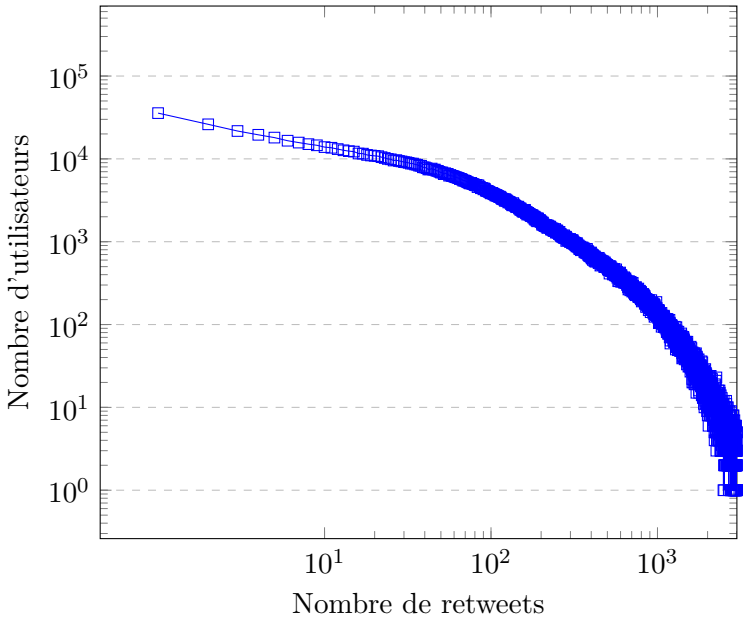


FIGURE 3.4: Répartition des retweets par utilisateur

liens sortants et des liens entrant pour chacun des nœuds de notre réseau. Les résultats sont disponibles en figure 3.5 pour les arcs entrants et en figure 3.6 pour les arcs sortants. On constate que les deux répartitions suivent des tendances très similaires avec peu de nœuds énormément connectés et beaucoup de nœuds peu connectés. Un utilisateur a tendance à davantage suivre de comptes qu'être suivi avec une médiane de 62 utilisateurs suivis et une médiane de 26 personnes suivant l'utilisateur. Ces distributions sont très similaires à d'autres études sur la topologie de Twitter comme le travail de Weng et al [Weng, Lim, Jiang, and He 2010] ou encore de Lerman [Lerman, Ghosh, and Surachawala 2012].

Une autre dimension essentielle concerne la répartition des plus courts chemins. Un plus court chemin représente le plus faible nombre de sauts nécessaires pour aller d'un nœud à un autre nœud. La répartition, en pourcentage, est disponible en figure 3.7. On observe une structuration en "petit monde", c'est à dire que n'importe quel nœud est connecté à n'importe quel nœud en moyenne avec $3,7$ sauts. Le plus grand chemin minimal sépare deux nœuds en 15 sauts, c'est le diamètre de notre réseau. Le plus court chemin moyen calculé en 2010 sur tout le graph par Kwak [Kwak, Lee, Park, and Moon 2010] était de 4.1. Toutes les distributions de notre jeu de données sont donc très proches du jeu réel ce qui valide sa représentativité pour le reste des analyses que nous avons menées.

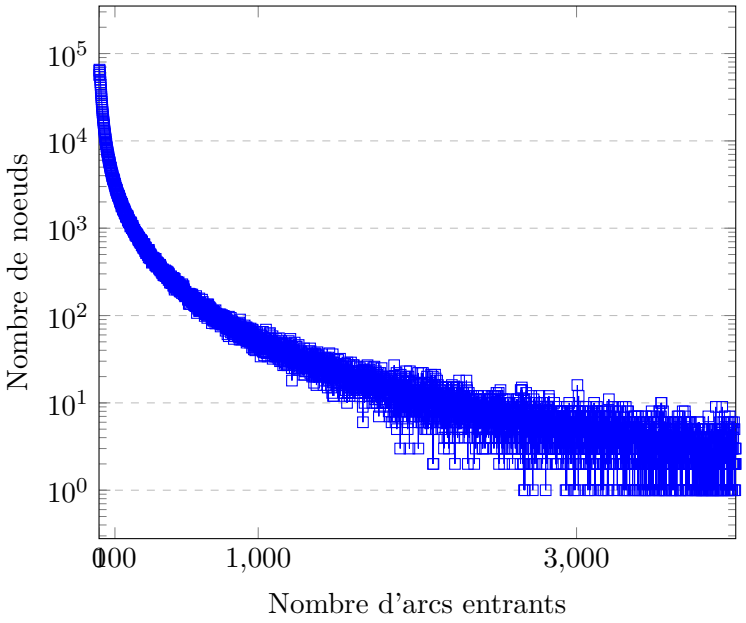


FIGURE 3.5: Répartition des arcs entrant dans le réseau

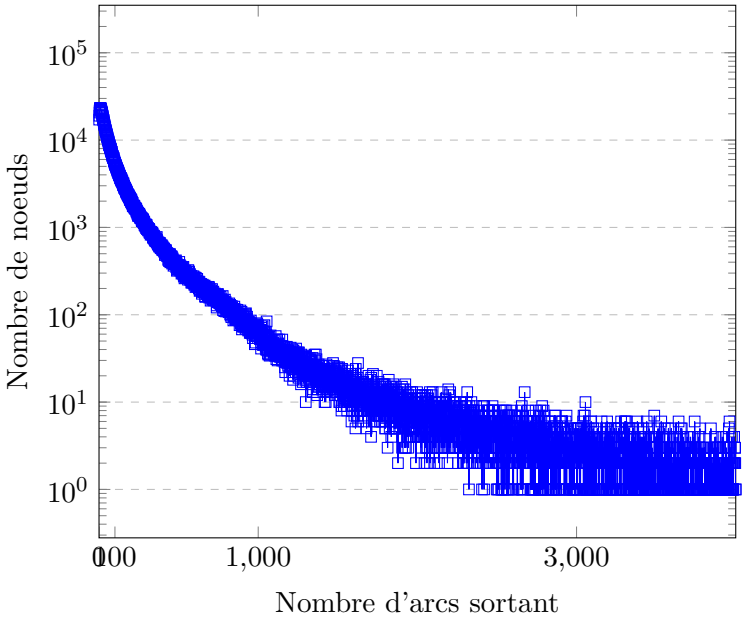


FIGURE 3.6: Répartition des arcs sortant dans le réseau

3.3.2 Influence des Top-N utilisateurs les plus similaires

Afin de mieux comprendre l'impact des retweets sur le réseau, nous allons nous intéresser à la similarité entre les utilisateurs.

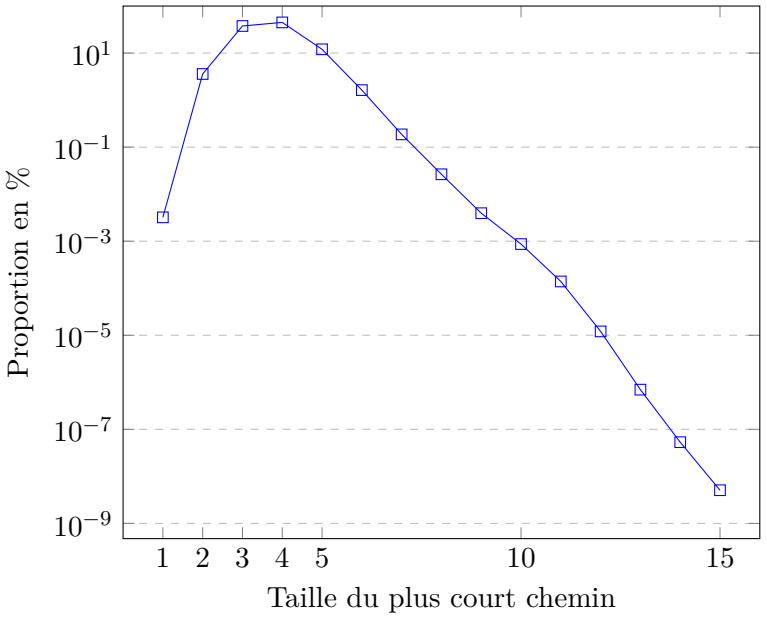


FIGURE 3.7: Répartition des plus courts chemins séparant deux noeuds dans le réseau de Twitter

Pour mesurer la similarité entre utilisateurs nous adoptons une mesure de Jaccard, adaptée pour des signaux binaires. Une interaction entre un utilisateur et un message se traduit donc par 1 et une absence d'interaction par un 0. Cette mesure est légèrement pondéré afin de prendre en considération la popularité des messages comme conseillé par Breese et al. [Breese, Heckerman, and Kadie 1998] afin d'obtenir la formule suivante :

$$sim(u,v) = \frac{\sum_{i \in L_u \cap L_v} \frac{1}{log(1+pop(i))}}{|L_u \cup L_v|} \tag{3.1}$$

$sim(u,v)$ mesure donc la similarité entre deux utilisateurs u et v. L_u est le profil de l'utilisateur u, soit l'ensemble des messages avec lesquels u a interagi (aimé et partagé). $pop(i)$ représente la popularité du message, ici le nombre total d'utilisateurs ayant interagi avec le message. L'intuition derrière l'ajout de cette pondération consiste à penser que deux utilisateurs sont d'autant plus proches que le contenu qu'ils ont aimé en commun est "confidentiel". Apprécier un message très populaire a moins de poids qu'apprécier quelque chose de plus confidentiel, aimé par uniquement quelques personnes.

La figure 3.8 montre l'évolution moyenne des scores de similarité au fur et à mesure du classement dans le *Top-N* des utilisateurs les plus similaires. Les scores de similarités

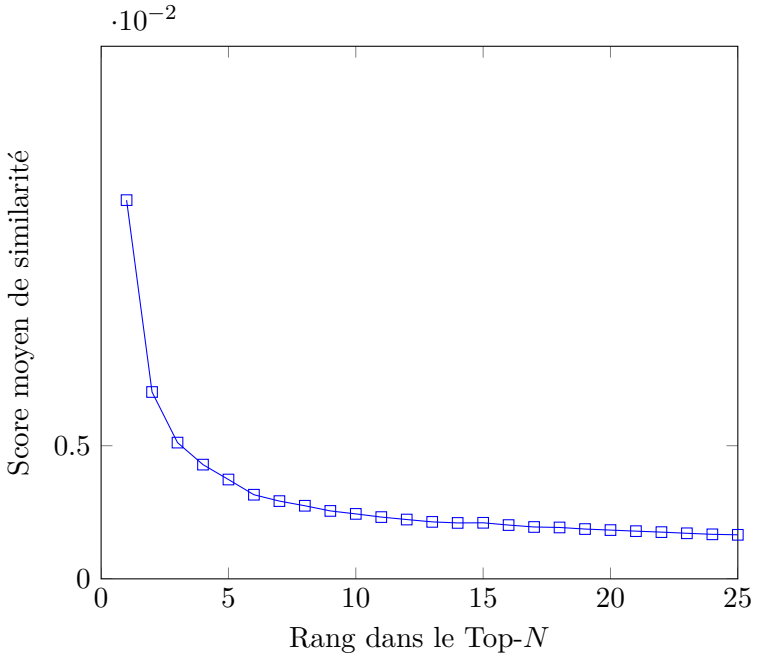


FIGURE 3.8: Score de similarité moyenne en fonction du rang dans le Top-N d'utilisateurs les plus similaires

sont très faibles, du fait de notre pondération par la popularité des messages et de la faible probabilité que deux utilisateurs apprécient les mêmes messages dans l'ensemble des tweets. Ces scores, mêmes faibles restent cependant significatifs. Ainsi, nous pouvons constater que le *top-5* reflète les utilisateurs les plus similaires à l'utilisateur u considéré.

3.3.3 Homophilie

Le phénomène d'`homophilie` appliqué aux réseaux sociaux est souvent décrit comme la tendance des utilisateurs à se connecter ensemble lorsqu'ils ont des traits en commun. Ce phénomène a déjà été largement étudié concernant les dimensions démographiques (Age, Sexe, Orientation politique) par exemple par Colleoni [Colleoni, Rozza, and Arvidsson 2014] ou Zamal [Al Zamal, Liu, and Ruths 2012]. Cette dernière étude montre qu'il est possible de prédire les informations démographiques d'un utilisateur à partir de son voisinage dans le réseau.

Des études sur l'homophilie basée sur des topics ont également été réalisées par Lerman [hyung Kang and Lerman 2012] et Bhattacharya et al. [Bhattacharya, Zafar, Ganguly,

Ghosh, and Gummadi 2014]. Elles étudient à quel point les utilisateurs ayant des centres d'intérêts (topics) similaires ont tendance à se regrouper. Cependant, aucune quantification précise de cet effet d'homophilie n'a été réalisée. C'est ce que nous proposons d'étudier dans cette section au travers du prisme de la similarité des profils.

Nous cherchons ici à mettre en lumière ce phénomène au travers des méthodes de filtrage collaboratif. Autrement dit, nous examinons le lien entre score de similarité et distance dans le réseau.

Pour des raisons de complexité et de passage à l'échelle, nous avons focalisé notre étude sur un sous ensemble de 500 utilisateurs choisis aléatoirement dans le jeu de données. Pour ces utilisateurs, nous avons calculé la distance qui les sépare de chacun des utilisateurs du graphe ayant un score de similarité non nul. La distance est mesurée en conservant l'aspect orienté du réseau initial de `Twitter`.

Le tableau 3.2 présente les résultats obtenus. Nous observons qu'en réalité seulement 6% des utilisateurs ayant une similarité non nulle sont directement connectés dans le réseau. Ces utilisateurs directement connectés possèdent cependant les scores de similarités les plus élevés, ce qui témoigne que dans le cas où des utilisateurs sont directement connectés dans le réseau et avec une similarité non-nulle, cette similarité aura tendance à être assez élevée comparativement aux autres. On constate également que ce score moyen décroit graduellement jusqu'à distance 3 et distance 4.

En faisant le rapprochement de ces résultats avec notre étude des plus courts chemins vue précédemment en figure 3.7 on s'aperçoit alors que les utilisateurs atteignables à deux sauts, qui représentent seulement 3% des nœuds du réseau, totalisent presque 40% du total des utilisateurs ayant une similarité non nulle. En revanche, les utilisateurs atteignables en 4 sauts qui représentent environ 45% des utilisateurs sont ici une infime minorité, seulement 4% des paires possèdent une similarité non nulle. En résumé on observe bien un phénomène d'homophilie, où plus des utilisateurs sont proches dans le réseau, plus leur score de similarité a tendance à être élevé. On constate également qu'une large majorité (94%) des utilisateurs similaires ne sont pas directement connectés dans le réseau.

Le tableau 3.3 se focalise cette fois sur le rang dans le *Top-N* et la distance moyenne associée. Pour chacun des 2000 utilisateurs de l'expérience précédente, nous avons conservé les 5 utilisateurs les plus proches afin de nous focaliser sur la répartition des distances pour ces utilisateurs. Ainsi, un utilisateur n'est directement connecté qu'à l'utilisateur le plus similaire de la plateforme seulement dans 53% des cas. Le phénomène d'homophilie apparait de nouveau de façon claire avec une croissance graduelle de la distance moyenne nécessaire pour atteindre les différents utilisateurs du classement en passant de $1,65$ sauts en moyenne pour l'utilisateur le plus proche à $1,99$ sauts pour le cinquième utilisateur le plus proche.

Distance	Nb d'utilisateurs	Perc.	Score de similarité moyen
1	19,163	05.96%	0.0056
2	121,857	37.91%	0.0021
3	166,633	51.84%	0.0017
4	12,070	03.76%	0.0018
5	297	00.09%	0.0016
6	6	00.01%	0.0019
Impossible	1,396	00.43%	0.0023

TABLE 3.2: Répartition des scores de similarité selon la distance avec l'utilisateur

Cette étude de l'homophilie du réseau nous sera extrêmement utile par la suite. En effet, si l'on désire appliquer des méthodes de filtrage collaboratif par plus proches voisins, il apparait possible de réduire l'espace de recherche de ces voisins à deux sauts car cette distance semble capturer la majorité des utilisateurs du top-5.

3.4 Conclusion

Au travers de ces différentes analyses nous avons une vision plus précise du fonctionnement de `Twitter`, et plus généralement des plateformes de micro-blogging. Ainsi nous avons montré qu'une large partie du contenu qui était publié tous les jours n'était jamais partagé, nous avons également montré que la durée de vie des messages est extrêmement courte et qu'il était donc essentiel pour un système de recommandation d'être en mesure de recommander du contenu peu de temps après sa publication.

Nous constatons également que peu d'utilisateurs très similaires sont en réalité présents

		Distances distribution (%)			
Rang	Distance Moyenne	1	2	3	4
1	1.65	53.30	28.20	16.65	1.45
2	1.78	43.70	34.50	20.50	1.05
3	1.88	37.99	36.04	24.37	1.35
4	1.97	33.18	36.99	27.68	1.70
5	1.99	32.01	37.93	28.20	1.56

TABLE 3.3: Distance dans le réseau en fonction du rang dans le Top-N

sur le réseau, et que dans la majorité des cas il n'existait pas de liens directs entre ces utilisateurs. Il apparait donc comme particulièrement opportun de s'appuyer sur cette distance de deux sauts pour capturer les utilisateurs les plus similaires d'un individu et ainsi réduire drastiquement l'espace de recherche, et donc de facto le temps de calcul.

Nous allons voir dans le prochain chapitre comment il est possible de construire un graphe de similarité à partir de l'homophilie présente dans le réseau. Nous montrons également comme appliquer un algorithme de propagation d'informations sur ce graphe pour calculer rapidement les recommandations des utilisateurs.

Chapitre 4

Recommandation de tweets

Nous avons montré que la durée de vie des messages était extrêmement courte et qu'il était par conséquent nécessaire de recommander des messages rapidement après leur publication. Nous voyons également comment il est possible d'exploiter le phénomène d'homophilie pour calculer a bas coût les similarités entre utilisateurs. Ces similarités sont cependant assez faibles compte tenu du faible nombre d'interactions en commun entre utilisateurs, il est donc crucial d'être en mesure de surmonter ce manque d'information.

En nous appuyant sur ces conclusions concernant le fonctionnement et les usages de `Twitter`, nous présentons dans ce chapitre notre modèle de recommandations de messages.

Ce modèle repose sur l'exploitation sur l'homophilie afin de trouver à faible coût des similarités fortes entre utilisateurs. Nous exploitons ces similarités en construisant un réseau de similarités permettant de mettre à profit facilement la transitivité entre utilisateurs. Cette transitivité est la clé pour recommander des messages sur lesquels on ne dispose que de peu d'interactions.

Nous allons développer dans une première partie le processus de construction de ce graphe de similarités. Dans une seconde partie nous présentons notre modèle de propagation qui permet de calculer rapidement le score des nouveaux messages entrants pour les utilisateurs. Enfin nous comparons notre modèle avec d'autres solutions issues de l'état de l'art pour montrer la capacité du modèle à passer à l'échelle tout en améliorant très significativement la précision des recommandations.

4.1 Graphe de similarité

Comme expliqué dans le chapitre précédent, l'homophilie naturelle du réseau, qui se traduit par des liens directs entre utilisateurs, ne permet pas de capter la majorité des utilisateurs influents. Il est très fréquent que des utilisateurs similaires ne se suivent pas dans le réseau alors qu'ils partagent de nombreux intérêts. Nous avons montré qu'une distance de 2 sauts dans le réseau orienté de `Twitter` garantissait de capturer la majorité de ces utilisateurs très similaires.

Nous nous appuyons donc sur cette distance de deux sauts pour capturer l'ensemble des utilisateurs les plus similaires pour chacun des nœuds. L'ensemble des utilisateurs atteignables à partir d'un nœud u, le voisinage à distance 2, est formalisé par l'expression $\mathcal{N}_2(u)$. Ainsi pour chaque utilisateur u de notre réseau nous calculons les similarités avec les nœuds v présents dans $\mathcal{N}_2(u)$. Pour chaque nœud w avec une similarité supérieure à un seuil τ, nous ajoutons un arc dirigé $e(u, w)$ dans notre graphe de similarité.

Notre graphe de similarité $G(V, E)$ où V est l'ensemble de nœuds, E l'ensemble des arcs et τ un seuil. Le graphe de similarité $SimGraph(V', E')$ est défini par :

$$\begin{cases} V' \subseteq V \\ e(u,v) \in E' \Rightarrow u \in V \wedge v \in \mathcal{N}_2(u) \wedge sim(u,v) \geq \tau \end{cases}$$

Un exemple du procédé de construction du graphe de similarité, pour un utilisateur u, est représenté en figure 4.1. Les caractéristiques principales de notre graphe de similarité calculé à partir du réseau de notre jeu `Twitter` sont exposés en Table 4.1. Nous observons dans un premier temps qu'environ la moitié des utilisateurs du jeu original ne sont pas présents dans le graphe de similarité. On peut facilement expliquer cette absence par l'impossibilité de trouver des utilisateurs similaires à distance 2 pour des utilisateurs qui n'ont jamais rien partagé ou bien trop peu pour avoir des similarités significatives dans ce voisinage. C'est le problème du démarrage à froid, comment recommander du contenu à des utilisateurs pour lesquels on possède trop peu d'informations.

	Jeu Twitter	Graphe de similarités
No de noeuds	2 182 867	1 149 374
No d'arcs	325,451,980	4 950 417
Score de similarité moyen	-	0.008
Nombre moyen d'arcs sortants	57.8	5.9

TABLE 4.1: Similarity Graph Characteristics

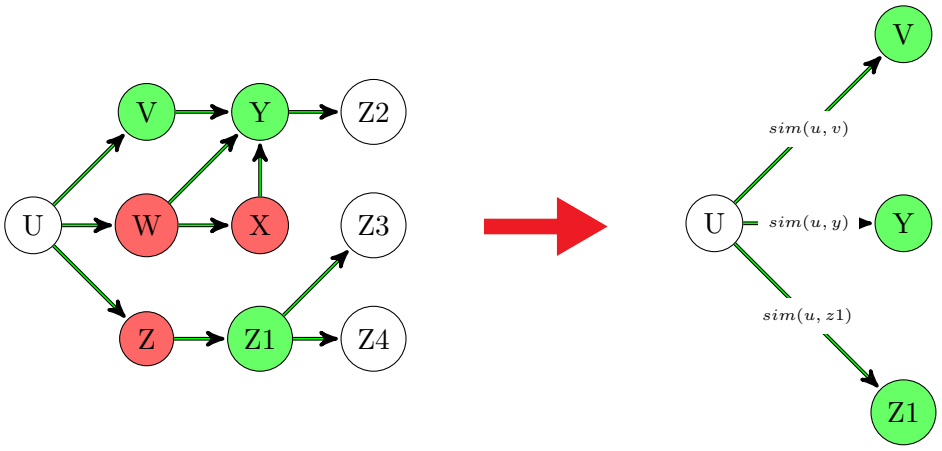


FIGURE 4.1: Pour un utilisateur, on explore à distance deux pour chaque utilisateur, ici U, afin de trouver les utilisateurs similaires, ici en vert (gauche). Pour chaque similarité trouvée on ajoute un arc dans le graphe des similarités (droite)

Si cette absence apparait comme problématique dans un premier temps, il n'est en réalité pas insurmontable de fournir des recommandations à ces utilisateurs comme l'ont montré les travaux de Sharma et al.[Sharma, Jiang, Bommannavar, Larson, and Lin 2016]. On peut par exemple envoyer les recommandations groupées provenant de l'ensemble du contenu envoyé aux personnes fortement connectés dans le réseau. Notre modèle rend ce type d'approche facile à mettre en place, mais ne sera pas l'objet de notre étude, nous délaissons dans ce chapitre les utilisateurs absents de notre graphe de similarités.

Nous observons dans un second temps, toujours à partir des caractéristiques de notre graphe de similarité, que la topologie du réseau `Twitter` original a été considérablement modifiée. La répartition du nombre de liens sortants par nœuds est disponible en figure 4.2, et la répartition des liens entrants réciproquement en figure 4.3. La répartition des plus courts chemins est visible en figure 4.4.

Le nombre médian de liens sortants par nœuds a été divisé par 12, en passant de 60

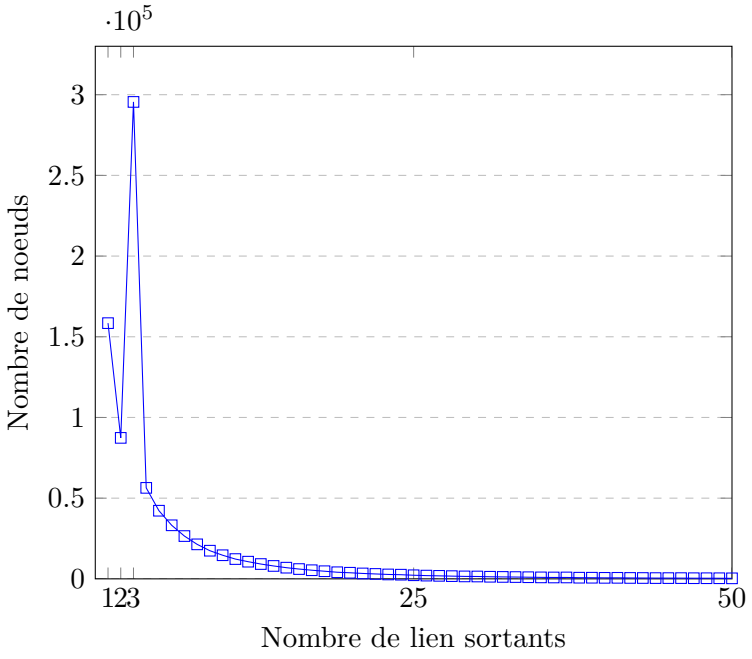


FIGURE 4.2: Répartition des liens sortants dans SimGraph

à 5 environ. De la même façon, le nombre médian de liens entrants a été divisé par 8 en passant de 26 à 3. On conserve donc toujours cette tendance des nœuds à être davantage influencés qu'influents.

Le plus court chemin moyen passe de 3.7 à 7.5 et le diamètre du réseau est allongé jusqu'à atteindre 21. D'après la catégorisation des réseaux introduit dans les travaux de Schnetrler [Schnettler 2009], le réseau reste tout de même un petit monde avec ces caractéristiques. Nous voyons clairement comment l'extraction d'un graphe de similarité à partir du réseau `Twitter` transforme la topologie originale avec un réseau beaucoup moins dense. D'une certaine manière on peut voir ce procédé comme un travail de réduction de dimensions.

On constate également qu'il n'existe pas de nœuds trop influents, le nombre de liens entrant par nœuds est assez équilibré, ce qui évite que les actions d'un seul nœud aient trop d'impact dans le réseau. *SimGraph* avec une topologie beaucoup moins dense que le réseau original `Twitter` rend l'utilisation d'algorithme de propagation particulièrement adaptée.

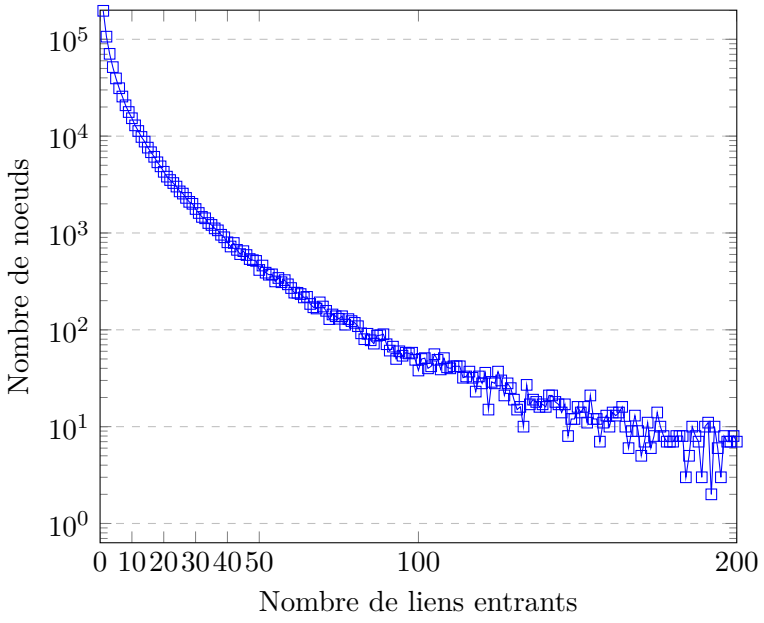


FIGURE 4.3: Répartition des liens entrants des noeuds de *SimGraph*

4.2 Modèle de propagation

Dans cette section nous présentons notre algorithme de propagation qui permet de recommander du contenu aux utilisateurs du réseau. Nous présentons également ses propriétés de convergence ainsi qu'une série d'optimisations visant à augmenter la précision et diminuer le temps de calcul du modèle.

4.2.1 Modèle

Comme nous l'avons vu précédemment, la densité des interactions entre utilisateurs et le contenu sur `Twitter` est extrêmement faible, c'est un problème majeur pour appliquer un modèle de filtrage collaboratif. Notre stratégie pour lutter efficacement contre cette faible densité consiste à s'appuyer sur la transitivité entre utilisateurs. Une stratégie similaire avait été utilisée avec succès par Huang et al. [Huang, Chen, and Zeng 2004]. Notre idée consiste donc à appliquer un algorithme de propagation sur notre graphe de similarités *SimGraph*.

En propageant l'opinion des utilisateurs dans ce graphe de similarités, nous pouvons transmettre le contenu apprécié par un utilisateur u à un utilisateur v alors qu'il n'existe

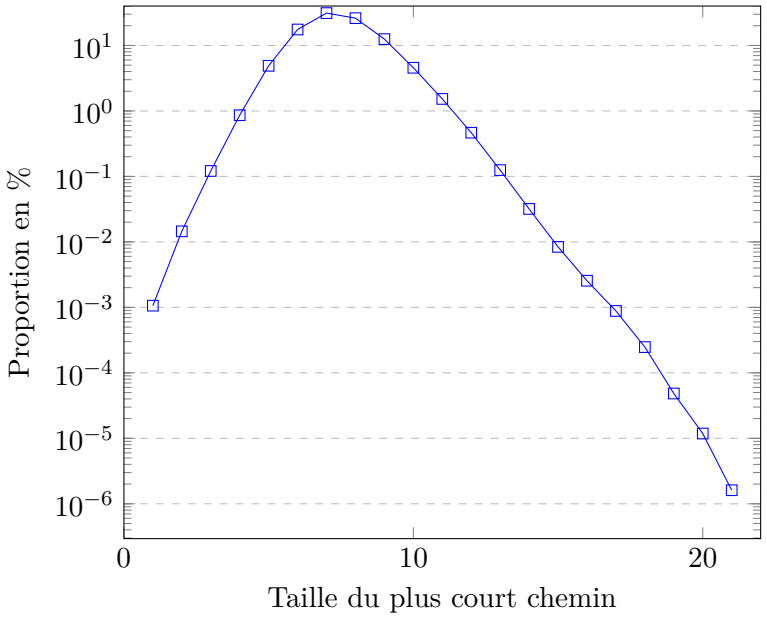


FIGURE 4.4: Répartition de la taille des plus courts chemins pour SimGraph

pas de liens direct entre eux dans le réseau. Intuitivement, si un utilisateur w a des intérêts communs avec u, le contenu apprécié par u devrait intéresser un utilisateur v si v a des intérêts communs avec w. Les seules interactions disponibles entre un utilisateur et un message sont un partage (retweet) ou un "j'aime", nous considérons ces signaux identiquement. Nous estimons qu'un utilisateur aime un message t avec :

Definition 1 *[]Probabilité d'aimer un message La probabilité qu'un utilisateur u aime (ou partage) un message t est :*

$$p(u,t) = \frac{\sum_{v \in F_u} p(u \leftarrow v, t)}{|F_u|}$$

*où F_u est l'ensemble des utilisateurs influents (similaires) pour u (*i.e.*, ses liens sortants dans le graphe de similarité) et $p(u \leftarrow v, t)$ est la probabilité que l'utilisateur u aime t selon le comportement de ses utilisateurs influents v. Cette probabilité est estimée par :*

$$p(u \leftarrow v, t) = p(v,t) \times sim(u,v)$$

Considérons le graphe de similarité présent en figure 4.5 avec 5 nœuds (u, v, w, x, y). Un arc $u \rightarrow v$ représente le fait que v est un utilisateur influent pour u. Les valeurs apposées

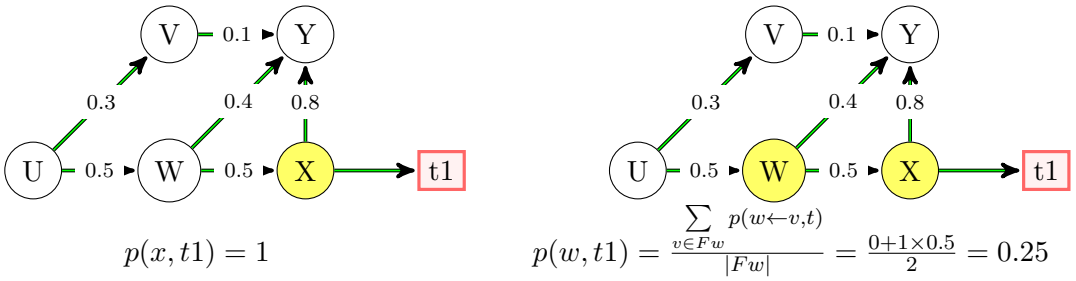


FIGURE 4.5: Exemple de propagation en deux étapes, dans un premier temps un utilisateur aime le message $t1$ (gauche), dans un second temps on peut calculer la probabilité que w aime le message (droite)

sur les arcs représentent les scores de similarité entre les nœuds $sim(u, v)$. Supposons qu'un utilisateur x aime un message $t1$, la probabilité que x aime ce message est donc inférée avec $p(x, t1) = 1$. Un influenceur dans le réseau de w a donc eu un changement de probabilité, y a toujours sa probabilité à 0 car il n'est influencé par personne. Nous pouvons donc, grâce à la définition précédente en déduire la probabilité que w aime $t1$ avec

$$p(w,t1) = \frac{\sum_{v \in F_w} p(w \leftarrow v,t)}{|F_w|} = \frac{0 \times sim(w,y) + 1 \times sim(w,x)}{2} = 0.25$$

Avec la modification de la probabilité $p(w, t1)$, il convient de mettre à jour les probabilités qui dépendent de $p(w, t1)$.

Comme la valeur de $p(w, t1)$ est modifiée, en passant de 0 à 0.25, il est nécessaire de propager ce changement de probabilité pour les utilisateurs influencés par w, dans notre cas l'utilisateur u.

$$p(u,t1) = \frac{\sum_{v \in Fu} p(u \leftarrow v,t))}{|Fw|} = \frac{0 \times sim(u,v) + 0.25 \times 0.5}{2} = 0.0625$$

Comme il n'existe pas d'utilisateurs influencés par u, l'algorithme peut s'arrêter.

Notre modèle de propagation est donc une série de calculs itératifs qui se produisent à chaque fois qu'un message est partagé. En nous appuyant sur notre graphe de similarité $SimGraph = (V, E)$ présenté dans la section précédente avec V l'ensemble des nœuds et E l'ensemble des liens. Lorsqu'un message t est partagé par un utilisateur nous propageons cette information dans le réseau $SimGraph$.

L'algorithme 1 présente les étapes du calcul de propagation. Avec D l'ensemble des utilisateurs qui ont aimé un message t. Pour chaque utilisateur $v \in D$, la probabilité $p(v,t)$ est fixée à 1 (ligne 3 de l'algorithme). Pour chacun des autres utilisateurs $u \in V \setminus D$, la probabilité initiale d'aimer le message t est égale à 0 avec $p(u,t) = 0$ (ligne 4). Lors d'une itération, nous calculons pour chaque utilisateur u de $V \setminus D$ la probabilité $p(u,t)$ à partir des probabilités des utilisateurs qui l'influence. A chaque itération k, nous calculons les probabilités à partir des probabilités de l'itération précédente $k-1$. Evidemment les utilisateurs $v \in D$ qui ont une probabilité de 1 ne sont pas mis à jour puisque cette probabilité est statique. Lorsque les scores de probabilités d'une itération k et $k-1$ sont les mêmes, la condition d'arrêt est remplie (ligne 7).

Algorithm 1: Algorithme de propagations

input : Le graphe de similarité SimGraph = (V,E), un message t et un ensemble d'utilisateurs D qui l'ont partagé

output : L'ensemble des nœuds avec leur probabilité d'aimer le message

```
1  Initialization;
2  foreach u in V do
3  |   if u ∈ D then p(u,t) = 1 ;
4  |   else p(u,t) = 0;
5  end
6  convergence = false ;
7  while convergence = false do
8  |   convergence = true ;
9  |   foreach u in V \ D do
10 |   |   p'(u,t) = ( Σ_{v∈F_u} p(v,t) × sim(u,v))/|F_u|
11 |   |   if p(u,t) ≠ p'(u,t) then convergence = false;
12 |   |   p(u,t) = p'(u,t) ;
13 |   end
14 end
15 Return ∀u ∈ V, p(u,t) ;
```

Cet algorithme produit pour chaque nouveau message un score probabiliste pour tous les utilisateurs de l'aimer. Les messages recommandés sont donc la liste des top-k messages ayant les scores de probabilité les plus élevés. Nous étudierons plus en détail l'impact de la taille de ce top-k sur la qualité des recommandations.

4.2.2 Système linéaire

Le modèle de propagation présenté dans la section précédente peut se traduire comme la résolution d'un système linéaire. Ce modèle linéaire permet de garantir la convergence du modèle et d'estimer la vitesse de cette convergence.

Soit n utilisateurs $(u_1, u_2, ..., u_n)$, les nœuds de notre réseau de similarité, notre algorithme de propagation tente de résoudre n équations :

$$\begin{cases} a_{11}p_{u_1} + a_{12}p_{u_2} + ... + a_{1n}p_{u_n} &= b_1 \\ a_{21}p_{u_1} + a_{22}p_{u_2} + ... + a_{2n}p_{u_n} &= b_2 \\ ... &= ... \\ a_{n1}p_{u_1} + a_{n2}p_{u_2} + ... + a_{nn}p_{u_n} &= b_n \end{cases}$$

Ce qui peut s'écrire sous la forme d'une multiplication matricielle $Ap = b$ avec :

- le vecteur b représente le vecteur d'initialisation, avec $b_i = 1$ si u_i a aimé le message et 0 sinon ;
- le vecteur p qui est le vecteur solution, il contient les probabilités de n'importe quel utilisateur d'aimer le message après la propagation.
- la matrice de similarité A définie par :

  $\forall i \leq n, \forall j \leq n,$

  $$a_{i,j} = \begin{cases} 1 & si\ i = j \\ \frac{-sim(u_i,u_j)}{|F_{u_i}|} & si\ il\ existe\ un\ arc\ (u_i, u_j)\ dans\ le\ graphe\ de\ similarité\ SimGraph \\ 0 & sinon \end{cases}$$

4.2.3 Convergence

La résolution d'un tel système linéaire $Ap = b$ est possible par des méthodes itératives comme Jacobi, Gauss-Seidel ou la méthode de sur relaxation successive (SOR). Si la matrice A est une matrice à diagonale dominante, ces méthodes sont prouvées convergentes. Il est possible de prouver que c'est le cas ici car $\forall u, v\ sim(u, v) \leq 1$, so $\sum_{j \neq i} |a_{ij}| < \frac{1}{F_u} \times \sum_{j \neq i} 1 = 1$. Or $|a_{jj}| = 1$, nous pouvons conclure $|a_{jj}| \geq \sum_{j \neq i} |a_{ij}|$ pour tout i i, et que donc A est à diagonale dominante. Nous pouvons donc conclure que notre modèle de propagation est convergent. Cependant, il est intéressant d'étudier plus précisement la vitesse de convergence afin d'être en mesure d'évaluer l'efficacité de notre modèle lors du pire scénario. A partir de la méthode de Jacobi, nous savons que la vitesse de convergence est bornée par la norme

de la matrice $||A||$. Hélas, il apparait impossible de quantifier la valeur de cette norme théoriquement car elle dépend de la topologie du réseau de similarité et des valeurs qui le composent, nous ne pouvons l'estimer que de manière expérimentale. A partir de notre jeu de données, nous trouvons $||A|| = 0.91$, nous savons donc que dans le pire des cas la convergence ne sera pas plus lente. Notre modèle peut donc théoriquement converger lentement dans certains cas, ce qui nous incite à appliquer différentes optimisations pour garantir le passage à l'échelle du modèle.

4.2.4 Optimisations

Nous proposons et testons différentes optimisations de notre algorithme de propagation qui servent à améliorer ses performances.

Seuils de propagation Une première optimisation repose sur l'utilisation d'un seuil statique β qui sert d'heuristique pour décider de propager un changement de probabilité ou non. Par exemple, lorsque la modification de score entre une étape $k-1$ et k est inférieure à ce seuil : $p(u,t)^n - p(u,t)^{n-1} < \beta$, il est inutile de propager ce changement car trop faible pour entrainer de réelles conséquences.

Dans le même esprit, nous ajoutons un nouveau seuil, dynamique cette fois, qui permet d'éviter la sur-propagation de messages populaires. Si un message est extrêmement populaire, de nombreuses probabilités sont à calculer ce qui peut prendre un temps considérable avant de converger. De plus, le message étant déjà extrêmement populaire, il semble donc hors du rôle d'un système de recommandation de le recommander à tous les utilisateurs de la plateforme. A contrario un message qui a été peu aimé a besoin d'un petit coup de pouce de propagation pour trouver son public. Ce seuil dynamique, lié à la popularité des messages va donc favoriser les messages moins populaires tout en améliorant les performances du système en évitant de sur-propager les messages stars. Plus précisément nous formalisons ce score dynamique $\gamma(t)$ pour un message t par :

$$\gamma(t) = \frac{(pop(t))^p}{k^p + (pop(t))^p}$$

Avec $pop(t)$ la popularité du message t, par exemple le nombre d'utilisateurs ayant

interagi avec. k et p sont fixés et supérieurs à 0, leur valeur sert à correspondre parfaitement à la distribution des messages populaires. Ainsi $\gamma(t)$ est borné entre $[0, 1]$ et proche de 0 quand peu de gens ont aimé t et très proche de 1 lorsque le message est populaire.

Calculs différés Le modèle nécessite de propager les scores à chaque nouvelle interaction entre utilisateurs et les messages, ce qui est énorme pour une plateforme de la taille de `Twitter` typiquement. Cependant plutôt que de se relancer dans un processus de propagation à chaque interaction, il est possible de faire concorder le calcul à des moments opportuns pendant la vie du message, a un intervalle δ. Par exemple, un message très populaire avec des centaines d'interactions par minutes peut bien attendre quelques minutes avant d'être recommandé. A l'opposé un message faiblement partagé peut attendre quelques heures avant de lancer la propagation.

4.3 Expériences

Nous évaluons dans cette section la qualité des recommandations obtenues en combinant notre algorithme de propagation avec le graphe de similarités *SimGraph*. Nous détaillons dans un premier temps notre protocole expérimental utilisé avec notre jeu de données `Twitter` pour mesurer la qualité des recommandations. Nous présentons ensuite plusieurs résultats en comparant notre modèle avec l'état de l'art. Enfin, nous verrons comment notre modèle peut être mis à jour à faible cout au gré des nouvelles interactions des utilisateurs.

4.3.1 Compétiteurs

Nous comparons notre modèle avec 3 solutions issues de l'état de l'art : un modèle probabiliste basé sur un réseau Bayesien introduit par Yang et al. [Yang, Guo, and Liu 2013], une méthode naïve de filtrage collaboratif basée sur les plus proches voisins [Herlocker, Konstan, Borchers, and Riedl 1999] et GraphJet la solution utilisée par `Twitter` présentée par Sharma et al. [Sharma, Jiang, Bommannavar, Larson, and Lin 2016]. La logique qui guide cette sélection découle de notre approche qui mélange l'influence des plus proches voisins avec un calcul probabiliste en s'appuyant sur la topologie du réseau. Nous avons donc parmi les compétiteurs une approche purement topologique (GraphJet), une autre

probabiliste (réseau Bayesien) et la méthode des plus proches voisins.

Le modèle d'inférence bayésienne est construit pour être utilisé sur des notes bornées entre 1 et 5, nous l'avons donc légèrement adapté à nos données binaires. De plus, en raison du très important coût de calcul nécessaire pour propager les informations sur le réseau bayésien nous ajoutons un seuil lors du calcul des probabilités afin de ne pas calculer les scores de chaque utilisateur pour chaque message. Les auteurs de GraphJet fournissent une implémentation de leur système sur github [1], nous avons donc repris leur code.

4.3.2 Protocole

Toutes nos expériences sont réalisés sur un ordinateur Linux doté de 512GB de RAM ainsi que de 80 Intel(R) Xeon(R) CPU E7-4830 v2 cadencés à 2.20GHz. Nous avons utilisé Java OpenJDK 1.8 pour réaliser toutes les expériences.

Afin de mesurer la qualité des recommandations issues des différentes méthodes, nous utilisons les messages partagés au minimum deux fois, ce qui représente environ 33 millions de messages. Nous disposons de $132,389,409$ retweets sur ces messages, ce qui constitue l'ensemble Set_{RT} de retweets. Cet ensemble Set_{RT} est ordonné temporellement avec les plus anciens retweets au début et les plus récents à la fin. Nous découpons cet ensemble en deux parties, l'une représentant 90% de Set_{RT} qui servira à construire les différents modèles et l'autre qui recouvre 10% de Set_{RT} qui permettra de tester les recommandations afin de déterminer la qualité de prédiction de chaque modèle. Cet ensemble de 10% d'actions de partage s'écoule sur une période de 66 jours. Nous sélectionnons aléatoirement 500 utilisateurs ayant moins de 100 retweets (utilisateurs peu actifs), 500 utilisateurs ayant entre 100 et 1000 retweets (utilisateurs actifs) et 500 utilisateurs avec plus de 1000 retweets (utilisateurs très actifs). En les additionnant nous obtenons un ensemble de 1500 utilisateurs pour lesquels nous testons la qualité de prédiction pour chaque méthode de recommandation. Autrement dit, lorsqu'un utilisateur a partagé un message après que celui-ci ait été présent dans ses recommandations, nous considérons qu'il s'agit d'un *hit* : une bonne prédiction.

4.3.3 Qualité des recommandations

1. https ://github.com/twitter/GraphJet

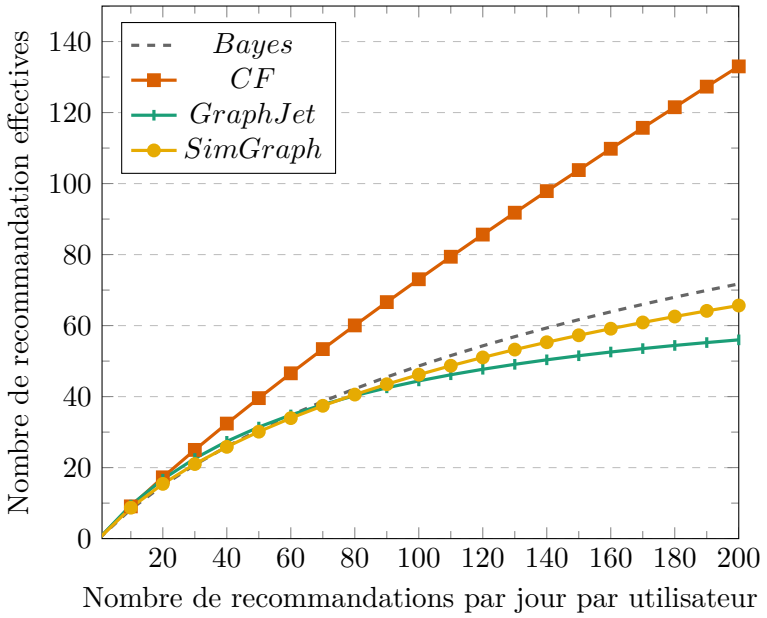


FIGURE 4.6: Capacité de nombre de recommandation

Nombre de recommandations La figure 4.6 présente le nombre moyen de recommandations par utilisateur pour les différents moteurs de recommandations en fonction de la taille maximum du top-k. Une droite linaire $y = x$ indiquerait un système capable de fournir un nombre k de recommandations pour chaque utilisateur. CF semble être le moteur qui s'approche le plus de cette tendance, c'est à dire capable de fournir un nombre élevé de recommandations. Ce comportement s'explique par le pré-calcul nécessaire de toutes les similarités entre les utilisateurs, CF est donc théoriquement capable de fournir des scores des recommandations pour un volume très important de messages. En revanche *Bayes*, *GraphJet* and *SimGraph* présentent des résultats similaires avec un nombre effectif de recommandations par utilisateur borné entre 50 et 70 par jour et par utilisateur. En théorie, *Bayes* et *SimGraph* qui reposent sur une propagation d'informations dans le réseau, doivent être capables de calculer des scores de recommandations sur un ensemble très important de messages et s'approcher ainsi de la courbe linéaire $y = x$. Ce faible nombre de recommandations effectives s'explique par l'utilisation des seuils qui empêche une propagation trop importante. Graphjet, du fait de la faible connectivité du graphe, n'est pas toujours capable de trouver un large nombre de recommandations pour les utilisateurs et bloque à 60 recommandations par jour en moyenne.

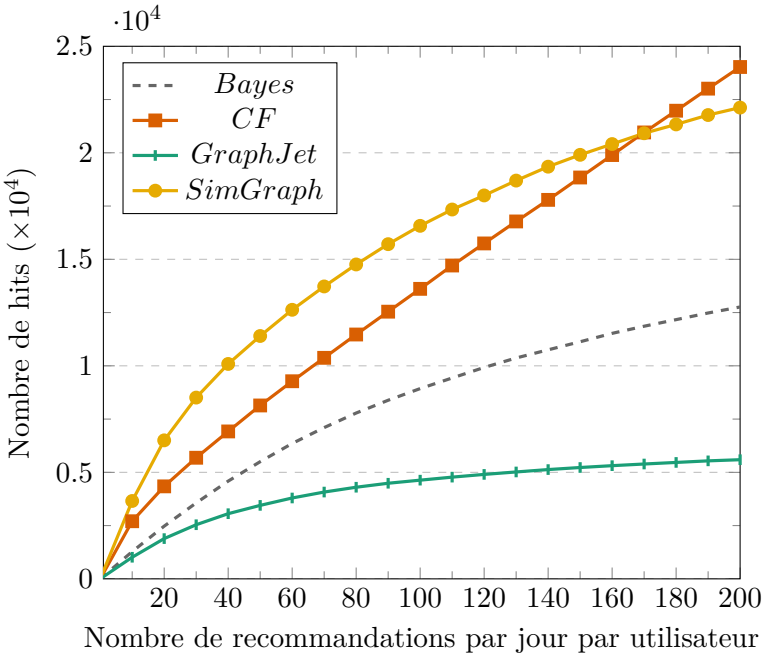


FIGURE 4.7: Nombre de hits par méthode pour 1500 utilisateurs

Prédiction de retweets Une bonne prédiction du comportement des utilisateurs correspond à un hit dans notre jeu de données, c'est à dire lorsqu'un message est recommandé à un utilisateur avant que celui-ci ne l'ait partagé. Les résultats du nombre de hits selon la taille du top-k par méthode est disponible en figure 4.7 pour notre ensemble de 1500 d'utilisateurs. Les résultats sont détaillés pour les 500 petits utilisateurs, les utilisateurs moyennement actifs et enfin les utilisateurs très actifs en figure 4.8. Les résultats semblent très stables quel que soit le profil des utilisateurs testés, les écarts entre les courbes sont diminués pour les petits utilisateurs du fait du plus faible nombre de hits possibles. Lorsque le nombre de messages recommandés est très élevé ($k > 170$), CF domine les autres méthodes et continue à faire des prédictions correctes. Cependant, comme nous l'avons vu dans la figure 4.6 c'est aussi cette méthode qui propose le plus grand nombre de messages. Avec un ratio moyen de 0.0001% de hit par message, CF peut aussi être vu comme générant beaucoup de bruit. Lorsqu'on se limite à des valeurs de k plus faibles, *Simgraph* domine les autres méthodes, la différence de score avec CF pour des valeurs plus faibles s'explique uniquement du fait de l'utilisation de la transitivité. *SimGraph* fait mieux que les autres méthodes avec 3.5 fois plus de prédictions correctes que *GraphJet* pour n'importe quelle

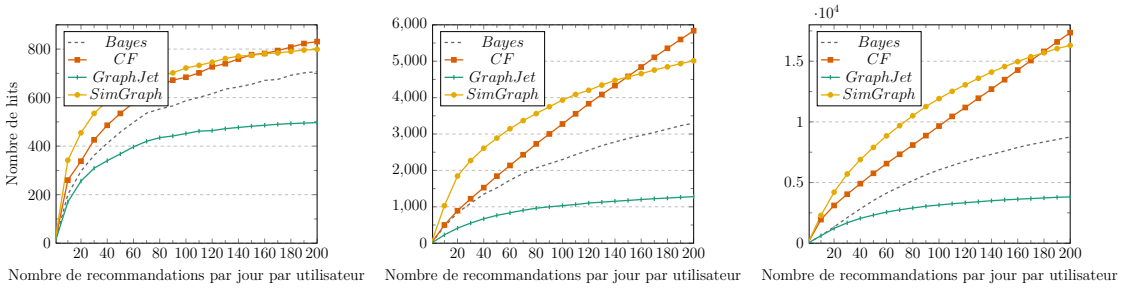


FIGURE 4.8: Nombre de hits par méthode pour 500 petits utilisateurs (gauche), 500 utilisateurs actifs (centre) et 500 utilisateurs très actifs (droite)

taille de k. De la même façon que *CF*, *SimGraph* ne repose pas directement sur le réseau d'utilisateurs pour faire des recommandations mais se concentre sur les similarités entre individus ce qui tend à fournir de meilleurs résultats. Grace à la transitivité de notre graphe de similarité, notre méthode est capable de trouver plus efficacement des messages intéressants.

F1 Scores Figure 4.9 affiche les F1 scores pour chaque méthode, d'après le nombre de hits réalisés selon le nombre de recommandations faites par jours.

$$F_1 = 2 \times \frac{précision * rappel}{précision + rappel} \tag{4.1}$$

Le F1 score est fréquemment utilisé pour évaluer la qualité d'un système de recommandation, il s'agit d'un compromis entre précision et rappel. Les résultats sont cohérents avec les précédentes figures, *SimGraph* reste en tête des résultats suivi de *CF*, *Bayes* et enfin de *GraphJet*. On constate que toutes les méthodes ont un pic de compromis entre précision et rappel autour de $k = 15$, soit 15 recommandations par jour par utilisateur. Cette valeur de 15 messages par jour parait une bonne stratégie pour envoyer un maximum de messages intéressants lors de la recommandation. *CF* en proposant beaucoup de messages voit son ratio davantage décroitre que *SimGraph*. *Bayes* est la seule méthode dont le ratio précision/rappel augmente progressivement au fur et à mesure que k augmente.

Popularité des hits Nous nous intéressons désormais à la popularité des hits de chaque solution, c'est à dire à la zone de précision de chaque système. Les résultats sont disponibles en figure 4.10. Nous observons qu'avec des marches aléatoires, *GraphJet* réalise ses hits en

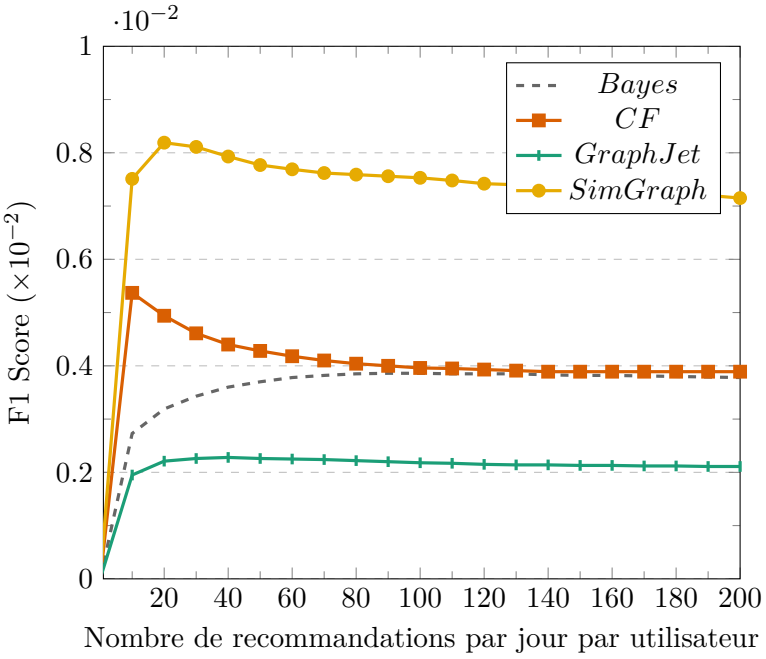


FIGURE 4.9: Compromis entre précision et rappel pour chaque méthode (F1 score)

moyenne sur des messages très populaires (113 retweets en moyenne). Plus un message est populaire, plus il existe des liens dans le réseau bi partite et donc de chances de sélectionner le message lors de marches aléatoires. Il est donc normal que *GraphJet* recommande des messages populaires. A l'opposé, le modèle Bayesien a tendance à recommander des messages peu populaires (seulement 6 retweets en moyenne). En s'appuyant sur le réseau naturel de `Twitter`, le modèle Bayesien va avoir tendance à faire des recommandations locales. Les approches qui reposent les similarités entre utilisateurs sont moins dépendantes du réseau, *CF* et *SimGraph* arrivent donc à recommander des messages populaires et plus confidentiels avec en moyenne des messages possédant respectivement 35 retweets pour *CF* et 23 retweets pour *SimGraph*. Si pour un petit nombre de recommandation, *SimGraph* se concentre davantage sur des messages populaires, le système s'oriente vers des messages moins populaires pour parvenir à trouver davantage de messages intéressants. *CF* présente un comportement opposé, en se concentrant d'abord sur des messages peu populaires puis en étendant son champ d'action progressivement. *CF* en utilisant des fortes similarités indépendamment du réseau peut recommander des messages avec très peu d'interactions lorsque les utilisateurs influents l'ont aimé.

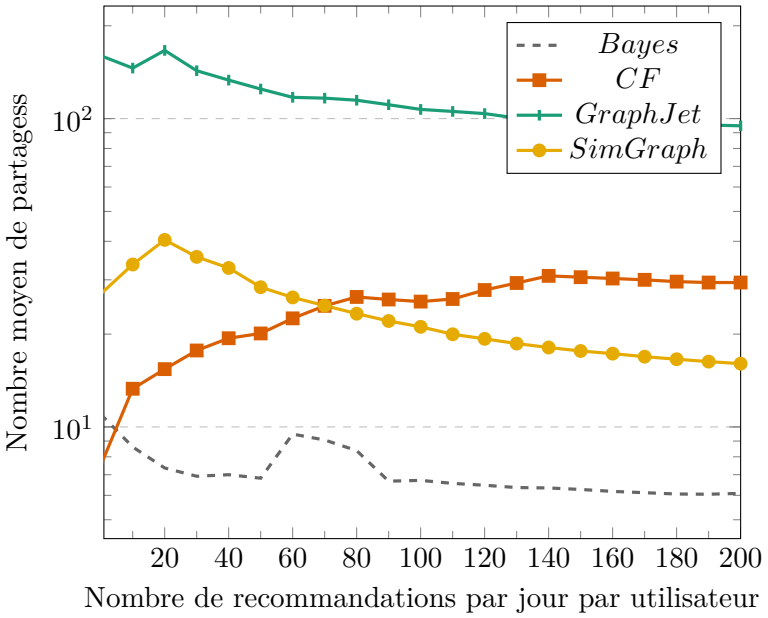


FIGURE 4.10: Popularité moyenne des messages correctement prédits (hits)

Comparaison des hits Figure 4.11 présente le nombre de hits en commun entre *SimGraph* et les autres méthodes. Chaque courbe représente le pourcentage σ de hits en commun.

$$\sigma(competitor) = \frac{hits(SimGraph) \cap hits(competitor)}{hits(competitor)} \tag{4.2}$$

Par définition donc, *SimGraph*, contient quel que soit le nombre de recommandations, c'est à dire 100% des hits en commun avec *SimGraph*. Nous observons que pour qu'un petit nombre de recommandations, *GraphJet* et *CF* réalisent des prédictions justes qui échappent à *SimGraph*. En superposant les informations contenues en figure 4.7, on se souvient que *CF* trouve des hits sur des messages plus confidentiels au départ, qui semblent échapper à *SimGraph*. Puis, progressivement, le nombre de hits en commun augmente, témoignant de l'hybridation de *SimGraph* entre les trois modèles. Nous observons que *Bayes* et *GraphJet* possèdent tous deux entre 40 et 60% de hits en commun avec *SimGraph* témoignant de la capacité de notre modèle de recommander à la fois des messages confidentiels ainsi que des messages populaires.

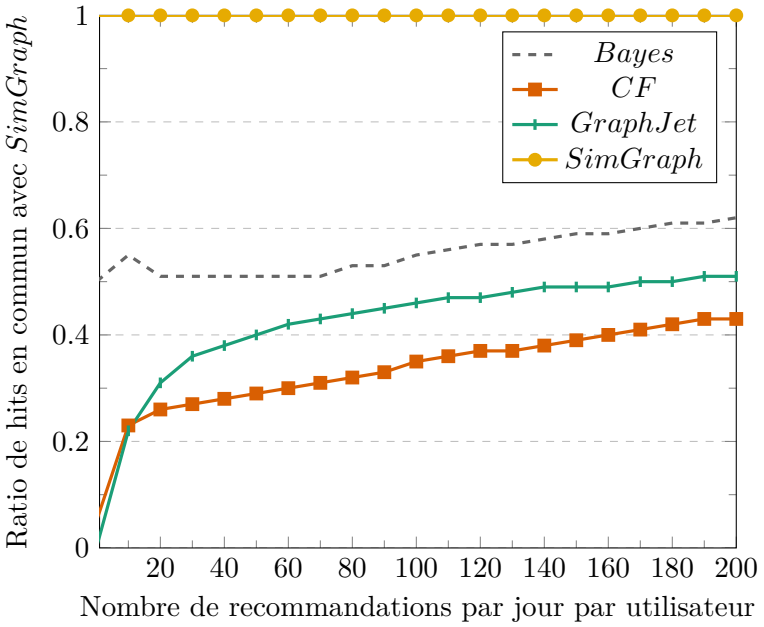


FIGURE 4.11: Parts des hits inclus dans les hits de *SimGraph*

4.3.4 Performances

Temps de calcul Nous comparons dans cette partie le temps de calcul nécessaire à l'utilisation de chaque méthode. Le temps de calcul est divisé en deux parties, le temps nécessaire pour réaliser l'apprentissage du modèle si besoin, et le temps nécessaire pour calculer le score de prédiction d'un message. A noter que *GraphJet* du fait de ses marches aléatoires calcule toutes les recommandations pour un utilisateur et non par message. On considère pour notre exemple qu'on souhaite réaliser 4 fois par jour des recommandations par utilisateur afin de maximiser le nombre de hits de *GraphJet* et ainsi d'avoir des résultats comparables aux autres méthodes. Les résultats sont compilés dans le tableau 4.2. On constate que si *CF* est capable de trouver le score de recommandation d'un message pour des utilisateurs très rapidement (en 0.5 ms), le travail d'initialisation est colossal, trouver toutes les similarités pour un utilisateur est de l'ordre de 9s. A l'inverse, *Bayes* a besoin de très peu de calculs d'initialisation, il est uniquement nécessaire de pré-calculer les probabilités conditionnelles entre personnes se suivant dans le réseau, soit environ $10ms$ par utilisateur. Pour *SimGraph*, l'initialisation consiste à parcourir le réseau de chaque utilisateur jusqu'à distance 2 et à calculer les similarités parmi cet ensemble d'utilisateurs.

	init. (par utilisateur)	**temps total init.** 1,149,374 utilisateurs	**temps (par message)**	**temps total (70 coeurs //)** 13,238,941 Tweets (10% Set_{RT})	**temps total** init + recos
Bayes	10ms	0.04h	975ms	51.22h	51.26h
CF	8,583ms	39.40h	0.5ms	0.02h	41.01h
SimGraph	311ms	1.41h	38ms	2.00h	3.41h
	init. (par user)	**temps total init.** 1,149,374 users	**temps (par utilisateur)**	**temps total (70 coeurs //)** 1,149,374 users * 66 jours (10% Set_{RT})	**total time** init + recos
GraphJet	0ms	0h	14ms	4.2h	4.2h

TABLE 4.2: Temps d'initialisation et de recommandation (en ms)

Parcourir le réseau et calculer les similarités prend environ $311ms$ en moyenne. *GraphJet* qui repose sur le graphe d'interaction entre utilisateurs et messages ne nécessite pas de travail d'initialisation.

Concernant la seconde partie du temps de calcul, le temps de calcul par message, nous constatons que *Bayes*, malgré l'utilisation d'un seuil de propagation, a besoin d'environ une seconde par message pour calculer les scores de similarité. C'est trop couteux pour envisager une mise en production réaliste. *SimGraph* a quant à lui besoin de $38ms$ par message pour calculer les scores de recommandations en moyenne. *GraphJet*, comme expliqué précédemment a lui besoin de $14ms$ pour calculer un ensemble de k recommandations pour un utilisateur.

Afin de comparer les méthodes entre elles, nous utilisons le temps de calcul combiné de l'initialisation par utilisateur et de la recommandation par message. Autrement dit, le temps total nécessaire à notre expérimentation. Ainsi *Bayes* est la méthode la plus gourmande en calculs avec $51.22h$, si l'initialisation est très rapide, c'est le temps de propagation par message qui fait exploser les compteurs. *CF* au contraire est une méthode couteuse en raison de son initialisation qui nécessite $39.40h$ de calculs. Ces deux méthodes paraissent donc inapplicables dans un contexte industriel de par leur difficulté à passer à l'échelle. *GraphJet* et *Simgraph* nécessitent des temps de calcul comparables avec $4.2h$ pour *GraphJet* et $3.41h$ pour *SimGraph*.

Temps d'avance des prédictions Cette expérience sert à illustrer l'avantage des différentes méthodes par rapport au temps d'avance des prédictions. C'est à dire combien de temps en moyenne une recommandation est faite avant d'être effectivement apprécié par l'utilisateur. Par définition on se s'intéresse donc ici qu'aux hits réalisés par chaque méthode. Les résultats sont présents en figure 4.12, on s'aperçoit que *GraphJet* est très stable avec

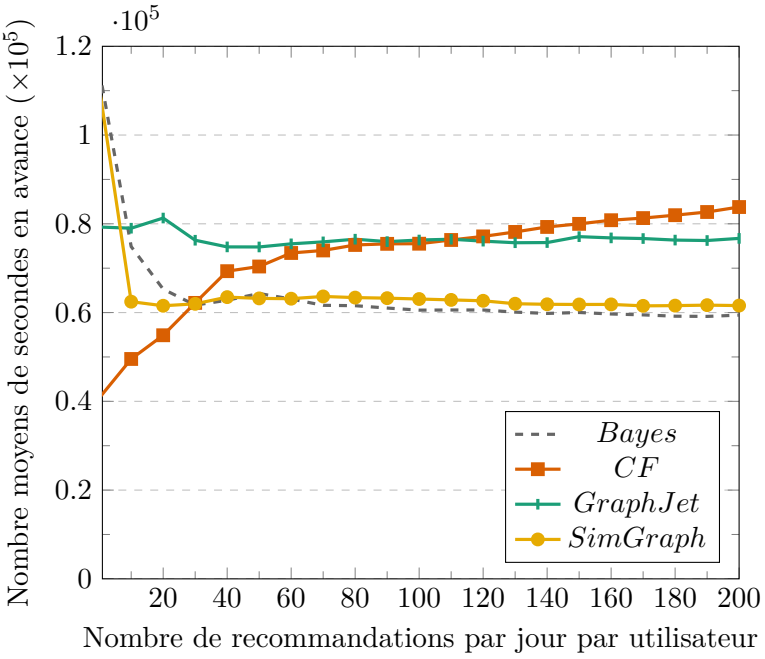


FIGURE 4.12: Average advance time before real retweets (in s)

une moyenne de temps d'avance sur ses prédictions d'environ 80 000s (soit 22 heures). En recommandant davantage de messages populaires, GraphJet maximise ses chances de faire des prédictions justes sur le long terme, puisque les messages populaires sont aussi ceux qui vivent le plus longtemps. De la même façon, la courbe de *CF* semble également corrélée à la popularité moyenne des messages recommandés (figure 4.10), ainsi plus les messages sont populaires, plus *CF* a de l'avance sur le comportement des utilisateurs. *CF* n'a pas besoin que le message se propage dans le réseau pour le recommander, il est donc particulièrement efficace sur les messages populaires car il est capable de les recommander très rapidement après leur publication. En moyenne *SimGraph* et *Bayes* parviennent à recommander un message 17 heures avant que celui-ci ne soit effectivement aimé par l'utilisateur.

Mise à jour du graphe Une plateforme de microblogging comme `Twitter` est sujette à des modifications constantes, les utilisateurs changent d'intérêt avec le temps, de nouveaux utilisateurs arrivent, des liens apparaissent ou disparaissent. Nous avons montré dans les parties précédentes les différents avantages de notre modèle de recommandation, à la fois moins couteux et multipliant les bonnes prédictions par 4, cependant il reste à évaluer sa

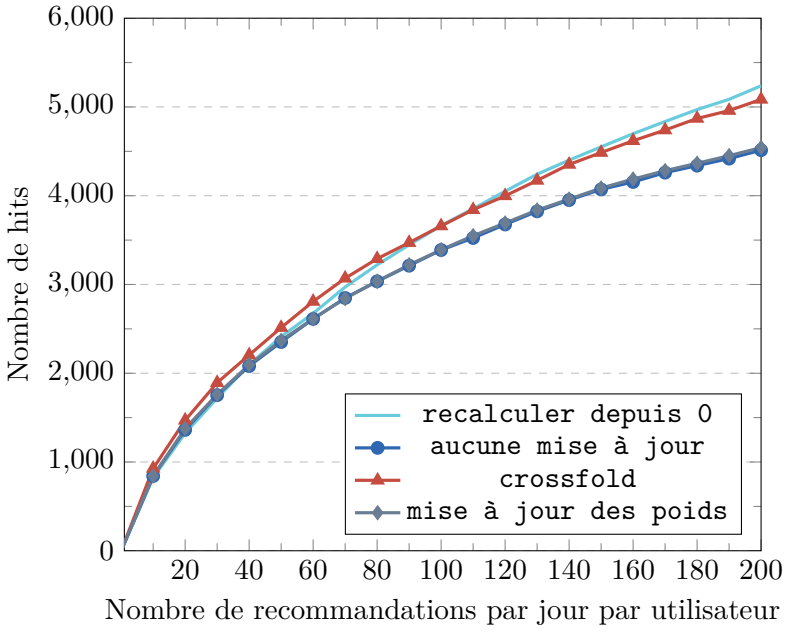


FIGURE 4.13: Nombre de hits suivant la stratégie de mise à jour employée

capacité à se mettre à jour. *GraphJet* en ne reposant sur aucun pré-calcul se met à jour continuellement au fur et à mesure que des arcs apparaissent dans le réseau bipartite entre messages et utilisateurs. Nous évaluons dans cette section plusieurs stratégies pour mettre à jour notre *SimGraph* :

— `recalculer depuis 0`, où *SimGraph* est construit a partir de rien,

— `aucune mise à jour`, où *SimGraph* n'est pas modifié

— `crossfold`, où nous appliquons notre processus de construction de *SimGraph* non pas sur le graphe `Twitter` mais sur une itération précédente de *SimGraph*

— `mise à jour de SimGraph` où nous mettons uniquement à jours les similarités entre utilisateurs d'une itération précédente du *SimGraph*

Afin d'évaluer la pertinence des diverses stratégies, nous découpons le set Set_{RT} en 95% et 10% et utilisons le *SimGraph* construit a 90% comme précédente itération du *SimGraph*. Les différents impacts des stratégies de mises à jours sur le nombre de hits sont représentés en figure 4.13. Ainsi, reconstruire le graphe depuis 0, qui représente l'opération la plus couteuse est également celle qui réussit à maximiser le nombre de hits pour de larges valeurs de k. De façon contre-intuitive, il semble que modifier les similarités dans

Itération	Nombre d'arcs
1	4 950 417
2	7 519 031
3	10 836 129
4	11 496 445
5	11 678 747

TABLE 4.3: Nombre d'arcs dans le *SimGraph* selon le nombre d'itérations

le graphe de similarités, c'est à dire les poids sur le graphe orienté, n'ait pour ainsi dire aucun impact puisque les hits réalisés sont exactement les mêmes qu'en utilisant un graphe non mis à jour. La stratégie `crossfold` apparait comme extrêmement prometteuse car l'impact sur les hits est très proche de tout recalculer depuis 0 mais pour un coût de mise à jour drastiquement réduit. En effet, `crossfold` effectue un parcours à distance 2 dans un précédent `SimGraph` et reconstruit les arcs pondérés entre utilisateurs. Si on se réfère à la répartition des arcs sortants du `SimGraph` en figure 4.2, les utilisateurs ont en moyenne 5 liens sortants, donc il faut calculer 5^2 similarités ce qui représente une petit portion des similarités qui auraient été calculées en partant de 0. Les résultats sont même améliorés par rapport à l'approche consistant à tout calculer depuis 0 pour des plus petites valeurs de k puisque cette méthode permet d'ajouter des arcs dans le réseau entre utilisateurs similaires mêmes s'ils ont une distance supérieure à 2 dans le réseau original de `Twitter`, ce que ne permet pas une construction à froid. Nous montrons donc qu'il est possible de mettre à jour à faible coût le modèle afin de continuer à faire des recommandations pertinentes pour les utilisateurs. Cette stratégie `crossfold` est rendu possible grâce à une propriété du réseau trouvée expérimentalement, la rapide convergence du système quand on applique ce parcours à distance 2 pendant plusieurs itérations. Les résultats de cette convergence sont visibles en tableau 4.3. On constate qu'après 3 or 4 itérations peu d'arcs sont ajoutés dans le graphe de similarités.

4.4 Conclusion

Nous avons présenté dans ce chapitre *SimGraph*, une méthode de recommandation particulièrement adaptée au microblogging capable de recommander rapidement des messages après leur publication et de passer à l'échelle. Il semble que la transitivité entre les similarités

d'utilisateurs soit un atout pour lutter contre le manque d'informations entre utilisateurs et messages. En comparant les résultats avec différents modèles de recommandations : *GraphJet*, *CF*, *Bayes* nous pouvons analyser le fonctionnement standard de ces méthodes avec leurs forces et leurs faiblesses. Ainsi *GraphJet* ne demande aucun travail d'initialisation, peut facilement se mettre à jour mais à tendance à recommander des messages populaires du fait des marches aléatoires. *CF* est capable de fournir un nombre très élevé de recommandations aux utilisateurs, mais ne dispose pas d'un excellent ratio précision/rappel, signe que les messages recommandés ne sont pas toujours pertinents. En revanche, le coût de calcul d'initialisation est trop élevé et ne permet pas d'envisager ce modèle de façon réaliste pour une plateforme comme `Twitter`. *Bayes*, en s'appuyant sur le réseau naturel de `Twitter` semble favoriser la recommandation de messages confidentiels, mais cette méthode produit moins de recommandations et de hits que *CF*, leurs ratio précision/rappel sont très proches. Si les couts d'initialisation sont négligeables avec *Bayes*, le coup de propagation est quant à lui très élevé, ce qui limite également l'utilisation de ce type de modèle avec de larges plateformes de microblogging. Nous montrons également comment *SimGraph* peut être mis à jour à faible coût et toujours proposer des recommandations pertinentes.

Chapitre 5

Bulle filtrante et chambre d'écho

Les réseaux sociaux sont devenus des outils extrêmement utilisés pour découvrir et partager de l'information sur Internet. Les utilisateurs s'organisent généralement autour d'intérêts communs, comme nous avons pu le voir dans le chapitre 2 avec le phénomène d'homophilie. En se connectant avec des gens similaires, les utilisateurs semblent avoir tendance à consommer de l'information du même alignement politique, c'est le phénomène des chambres d'écho. Eli Pariser [Pariser 2011] prédisait en 2011 que les systèmes de recommandation, en filtrant ou en mettant en avant des messages pertinents, aggraveraient encore ces chambres d'écho et cet isolement idéologique des individus, c'est le problème des bulles filtrantes.

Nous reviendrons dans ce chapitre sur les différentes études tentant de quantifier et d'évaluer l'existence de ces deux phénomènes. Nous étudierons ensuite différentes méthodes de détection de communautés appliqués à `Twitter`, ces communautés nous serviront de clé de voûte permettant l'analyse du phénomène des bulles filtrantes. Enfin nous présenterons un moyen de quantifier l'enfermement idéologique des utilisateurs provoqué par différents systèmes des recommandation, et nous montrons que ce phénomène ne touche qu'une minorité d'utilisateurs. Il semble que, paradoxalement, les systèmes de recommandations tendent à davantage empêcher les utilisateurs de s'enfermer dans une bulle qu'à les y pousser. Malgré tout, certains utilisateurs subissent une forme d'enfermement qu'il est possible d'atténuer en exploitant les liens de similarités entre communautés afin de modifier les recommandations qu'ils reçoivent.

5.1 Chambres d'échos et bulles filtrantes

Il convient de définir plus finement deux phénomènes théoriques concomitants agissant sur la consommation d'informations des utilisateurs au sein des réseaux sociaux, un phénomène de chambre d'échos combiné à un phénomène de bulles filtrantes. Les chambres d'échos décrivent la tendance des utilisateurs à s'entourer d'utilisateurs du même alignement idéologique et donc de ne voir de l'information qu'au travers du prisme de personnes similaires. C'est un effet qui serait donc dû à la forte homophilie présente dans le réseau.

L'étude de ce phénomène d'homophilie est ancienne et prend racine dans les sciences sociales et la philosophie. Aristote déclarait déjà, dans la Rhétorique [Bonafous 1856], "*Nous aimons ceux qui nous ressemblent et qui ont les mêmes goûts que nous*". La tendance des humains à se connecter lorsqu'ils partagent des traits communs est déjà étudié dans des travaux des années 20-30 comme dans les travaux de Wellman [Wellman 1926] par exemple qui s'intéresse aux amitiés tissées dans une école primaire selon des critères de similarités (taille, QI, âge).

Depuis, l'homophilie a été maintes fois quantifiée et démontrée, la question qui nous intéresse ici concerne davantage la traduction de cette homophilie sous forme de chambre d'échos. Les travaux qui étudient spécifiquement la circulation d'informations sur `Twitter` montrent qu'il existe un phénomène de "chambre d'échos" comme les travaux de Barbera [Barberá, Jost, Nagler, Tucker, and Bonneau 2015] ou Conover [Conover, Ratkiewicz, Francisco, Gonçalves, Menczer, and Flammini 2011]. Ils analysent le réseau de `Twitter` et montrent que le réseau des démocrates et des républicains forment deux sous-réseaux séparés. Les travaux de Himelboim et al. [Himelboim, McCreery, and Smith 2013] montrent que les utilisateurs de `Twitter` évoluent dans des sphères homogènes politiquement et ont peu de chance de voir de messages de camps opposés parmi leur flux d'actualité.

Sur Facebook les résultats sont plus contrastés. Des chercheurs détectent des chambres d'échos comme les travaux de Quattrociocchi [Quattrociocchi, Scala, and Sunstein 2016] qui montrent que les utilisateurs qui interagissent avec des messages complotistes n'interagissent qu'avec des messages issus de la même communauté. D'autres études sont plus nuancées, ainsi Goel et al. [Goel, Mason, and Watts 2010] expliquent par exemple que les utilisateurs

s'entourent également de personnes ayant des opinions très éloignées des leurs et voient des messages idéologiquement opposés régulièrement. Une large étude réalisée par Facebook [Bakshy, Messing, and Adamic 2015] va dans le même sens et indique que la diversité d'opinions présentes dans le fil d'actualité des utilisateurs est en réalité assez élevée.

Dubois et al. [Dubois and Blank 2018] étudient la consommation d'information des individus en considérant les réseaux sociaux comme un seul canal d'obtention d'informations parmi un ensemble de canaux différents. Il semble que lorsqu'elle est étudiée dans son ensemble, la consommation d'information des individus ne soit que peu sujette aux chambres d'échos. Leur enquête montre que les individus rencontrent fréquemment de l'information opposée à leur propre idéologie sur internet, qu'ils consultent plusieurs sources d'informations et qu'ils utilisent les moteurs de recherche pour confirmer ou infirmer certaines informations. Plus important, les personnes de l'enquête semblent être en mesure de découvrir des informations qui les font changer d'avis sur leurs opinions politiques.

Le consensus général est que les chambres d'échos existent sur les réseaux sociaux et qu'elles auraient pour origine ce phénomène naturel des utilisateurs à se connecter entre eux lorsqu'ils sont similaires. Toutefois l'isolement de ces communautés les unes par rapport aux autres semble avoir été surestimée. Reste la question de savoir si les algorithmes de personnalisation aggravent l'isolement des communautés, c'est l'idée même de la bulle filtrante.

On dispose de peu d'études sur le sujet. Une très large étude de Flaxman et al. [Flaxman, Goel, and Rao 2016] étudiant le comportement sur internet de 50000 personnes résidants aux Etats-Unis observe par exemple que les individus les plus isolés idéologiquement sont ceux qui utilisent le plus les systèmes de recommandation pour trouver du contenu. Mais contre intuitivement, ce sont aussi ces individus qui sont le plus au contact avec de l'information très éloignée idéologiquement.

Une autre étude importante traitant des bulles filtrantes est celle de GroupLens, dont on a vu précédemment qu'ils avaient eu un impact considérable dans les recherches sur les systèmes de recommandation. Dans leur travail [Nguyen, Hui, Harper, Terveen, and Konstan 2014] ils s'intéressent au comportement des utilisateurs sur leur plateforme, selon que les utilisateurs suivent ou non les recommandations de films. Paradoxalement,

ils observent que les utilisateurs ont tendance à s'enfermer avec le temps en visionnant des films de plus en plus similaires mais que les systèmes de recommandation freinent ce phénomène. Dans une étude sur Facebook et l'impact de l'algorithme de filtrage sur le fil d'actualité des utilisateurs, [Bakshy, Messing, and Adamic 2015] indiquent que le nombre de messages provenant d'utilisateurs ayant des points de vue opposés n'est réduit que d'un pour cent, ce qui apparaît comme négligeable.

Contrairement donc aux chambres d'échos qui sont une pure transposition de l'homophilie existante dans la vraie vie, la question de l'existence des bulles filtrantes reste ouverte.

5.2 Communautés sur Twitter

Notre approche pour quantifier les bulles filtrantes repose sur la détection des communautés sous-jacentes, présentes dans le réseau qui connecte les utilisateurs. Nous allons évaluer les résultats de différents algorithmes de détection de communautés. Nos analyses des différents partitionnements du réseau produits par les algorithmes ont pour intérêt de permettre de comprendre plus finement le type de communautés détectées et de parvenir à une vision plus globale des phénomènes communautaires présents sur `Twitter`.

5.2.1 Méthodes de détection de communautés

`Twitter` étant un réseau particulièrement large avec énormément de nœuds, nous avons par conséquent orienté notre sélection d'algorithmes de détection de communautés vers ceux capables de passer à l'échelle et de partitionner de larges réseaux. Nous avons donc conservé trois méthodes : `Infomap`, `Louvain` et `Label Propagation`. Nous allons décrire le fonctionnement général de ces trois méthodes

`Infomap` [Bohlin, Edler, Lancichinetti, and Rosvall 2014; Held, Krause, and Kruse 2016] s'appuie sur des marches aléatoires, l'idée est de maximiser les probabilités pour une marche aléatoire de rester dans la même communauté. Ainsi `Infomap`, tente de minimiser

$L(M)$ définie par :

$$L(M) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(\mathcal{P}^i) \tag{5.1}$$

$L(M)$ prend en considération la probabilité de rester dans la même communauté et la probabilité de changer lors d'une marche aléatoire.

Une autre méthode, dite la méthode de `Louvain` [Blondel, Guillaume, Lambiotte, and Lefebvre 2008], repose davantage sur l'optimisation de la modularité qui est définie comme :

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - \frac{k_i k_j}{2m}] \delta(c_i, c_j) \tag{5.2}$$

Dans un premier temps `Louvain` associe chaque nœud à une communauté. Pour chaque nœud, `Louvain` calcule les différentes modularités en associant le nœud aux différentes communautés de ses voisins. Le nœud est ensuite associé à la communauté de son voisinage qui maximise la modularité. Dans un second temps, l'algorithme regroupe tous les nœuds de la même communauté et construit un nouveau réseau en retirant les liens internes aux différents groupes.

L'alternance de ces deux phases converge et produit un partitionnement du réseau. Louvain tente ainsi de maximiser la densité des liens à l'intérieur d'une communauté par rapport aux liens à l'extérieur de la communauté.

Enfin `Label Propagation` [Raghavan, Albert, and Kumara 2007] étiquette les nœuds du réseau avec un numéro de communauté et propage cette étiquette sur le voisinage des nœuds. Ainsi des utilisateurs fortement connectés auront tendance à partager le même numéro de communauté.

5.2.2 Partitionnement

Nous allons étudier dans cette section la façon dont notre jeu de données `Twitter` a été partitionné par les différents algorithmes. Le nombre de communautés détectées et leur nombre d'utilisateurs respectif est représenté en figure 5.1. Nous observons que `Louvain` produit moins de communautés que les autres méthodes et concentre 90% des

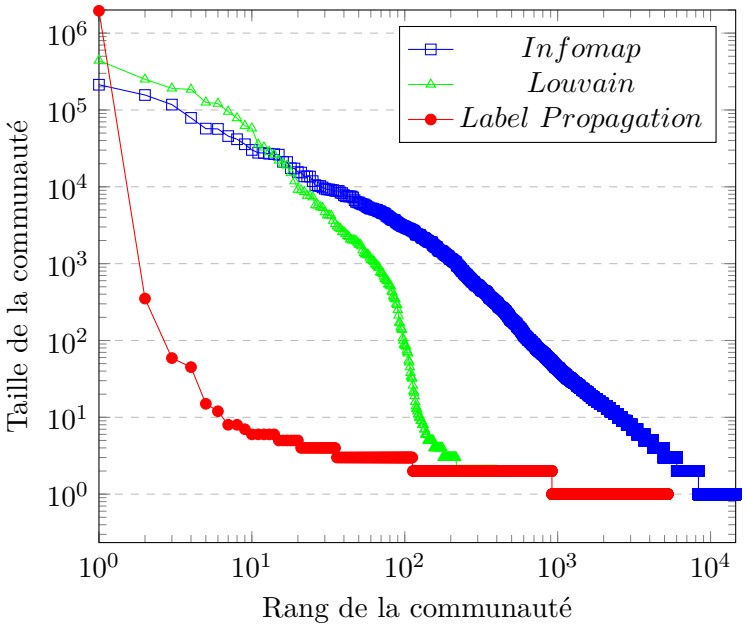


FIGURE 5.1: Distribution des communautés avec `Louvain`, `Infomap` et `Label Propagation`

utilisateurs dans 17 communautés. Les résultats produits par `Infomap` sont sensiblement similaires, avec des communautés légèrement moins peuplées, 90% des utilisateurs sont regroupés dans 247 communautés. `Infomap` détecte donc des communautés plus fines mais produit également une très longue traine de petites communautés, 12000 communautés contiennent moins de 10 utilisateurs. `Label Propagation` trouve une très large communauté regroupant plus de 99% des utilisateurs, et une très longue traine de 32000 communautés avec un seul utilisateur. Il semble que la topologie du réseau `Twitter` soit trop dense pour la propagation d'étiquettes, et ne se prête pas à l'utilisation de cette méthode.

Malgré ces différences de comportement entre `Louvain` et `Infomap`, on trouve des similarités lorsqu'on s'intéresse à l'information en commun normalisée (*Normalized Mutual Information*), une métrique très utilisée pour mesurer la similarité entre deux partitionnements [Fortunato 2010], [Strehl and Ghosh 2002] définie par :

$$NMI(A,B) = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} N_{ij} log(\frac{N_{ij}N}{N_i N_j})}{\sum_{i=1}^{C_A} N_i log(\frac{N_i}{N}) + \sum_{j=1}^{C_B} N_j log(\frac{N_j}{N})} \tag{5.3}$$

Avec A et B deux partitionnements d'un réseau. C_A représente le nombre de communautés dans le partitionnement A et C_B le nombre de communautés présentes dans B. N est le nombre de nœuds total du réseau. N_{ij} correspond au nombre de nœuds en commun présents dans la communauté i et la communauté j. N_i est le nombre total de nœuds dans la communauté i du partitionnement A et N_j le nombre de nœuds présents dans la communauté j présente dans B.

En résumé, le score NMI entre deux partitionnements de communautés est compris entre 0 et 1. 0 indique que les partitionnements sont complètement indépendants et 1 indique un partitionnement parfaitement identique.

Les différents scores NMI entre les algorithmes de détection de communautés sont disponibles en Tableau 5.1. On s'aperçoit que les partitions produites par `Louvain` et `Infomap` sont similaires à environ 50%.

	`Infomap`	`Label Propagation`	`Louvain`
`Infomap`	1	0.032	0.47
`Label Propagation`	0.032	1	0.029
`Louvain`	0.47	0.029	1

TABLE 5.1: Matrice des scores NMI entre les différentes méthodes de détection de communautés

Comme vu précédemment, `Louvain` produit moins de communautés et par conséquent évite cette longue traine de communautés minuscules particulièrement complexes à manipuler lors de nos prochaines analyses. En effet, analyser la propagation d'informations au sein de communautés de moins de 5 utilisateurs fournit des résultats difficilement exploitables. Pour cette raison nous privilégions l'utilisation de `Louvain` avec une seconde itération de l'algorithme sur les communautés trop volumineuses afin de les décomposer en sous-communautés.

5.3 Propagation d'informations dans les communautés

Nous allons étudier le lien entre le partage d'informations et les communautés, observer comment l'information se propage au travers des différentes communautés.

5.3.1 Communauté des messages

Nous avons besoin dans un premier temps d'associer un message à une communauté afin de retracer son évolution dans le réseau. Deux possibilités sont principalement envisageables : un message appartient à la communauté qui produit le message ou à la communauté qui a le plus partagé le message. Il semble qu'avec des communautés de tailles très variables, une communauté très importante aurait tendance à être considérée comme auteur de la plupart des messages, puisqu'une telle communauté aurait davantage de chances de fortement partager un message.

Nous avons comparé les deux stratégies sur les données `Twitter` et il semble que pour 90% des messages ces stratégies aboutissent au même résultat. C'est à dire qu'ils lient les messages aux mêmes communautés d'origine. Comment attendu, les 10% de messages où les stratégies produisent des résultats différents concernent des messages qui ont été publiés dans de petites communautés et qui ont rencontré un succès dans des communautés plus peuplées. La communauté associée à un message correspond donc dans la suite de nos travaux à la communauté qui a produit le message, une approche qui semble être la moins biaisée.

5.3.2 Popularité et diffusion

Nous nous intéressons ici à la corrélation entre le nombre de communautés atteintes par un message lors de sa propagation et sa popularité. Cette corrélation est représentée en figure 5.2 qui indique le nombre de communautés ayant interagi avec un message selon le nombre d'interactions totales portant sur celui-ci.

Nous constatons sans surprise que plus un message est partagé plus il touche de communautés différentes. Cependant, cette corrélation n'est pas linéaire et même des messages très populaires ne touchent pas toutes les communautés.

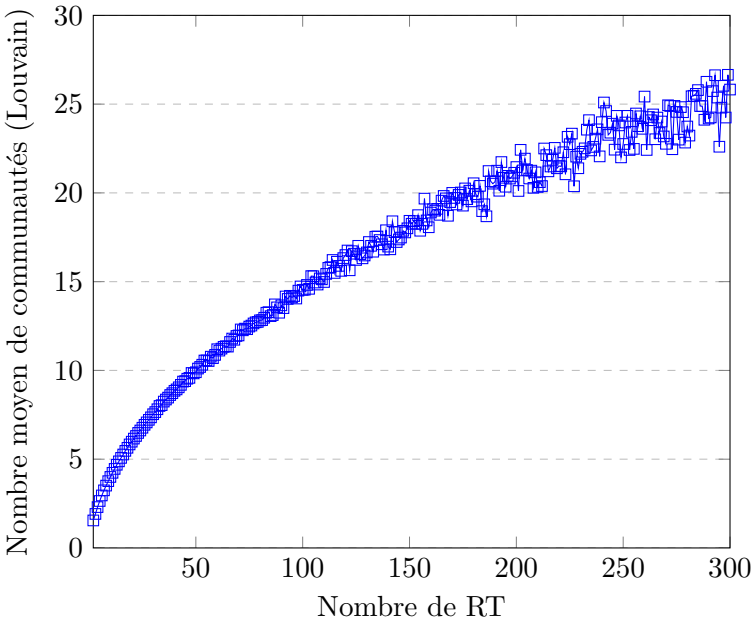


FIGURE 5.2: Corrélation entre popularité d'un message et nombre de communautés touchées

Nous examinons ensuite en figure 5.3 le nombre moyen de communautés ayant interagi avec les messages de notre jeu de données. Comme expliqué précédemment, les messages populaires sont très marginaux et ne représentent qu'un faible pourcentage de tous les messages. On voit clairement dans cette figure la tendance communautaire de `Twitter`, où dans 80% des cas un message ne touche qu'une ou deux communautés. Presque la moitié des messages publiés sur `Twitter` restent ainsi à l'intérieur de leur communauté d'origine.

5.3.3 Analyse des communautés

De façon complémentaire à notre précédente étude sur la propagation des informations dans les communautés, nous nous intéressons ici aux types de communautés détectées de façon plus qualitative.

Ainsi, pour chacune des communautés possédant plus de 100 utilisateurs, c'est à dire 105 communautés pour `Louvain` et 69 pour `Infomap`, nous sélectionnons les dix utilisateurs les plus connectés de la communauté. Manuellement, nous avons regardé leurs publications et leur profil afin d'extraire un lien commun pour la communauté comme l'âge, la géographie, la langue ou un intérêt précis. Si des méthodes d'étiquetage automatique pouvaient être

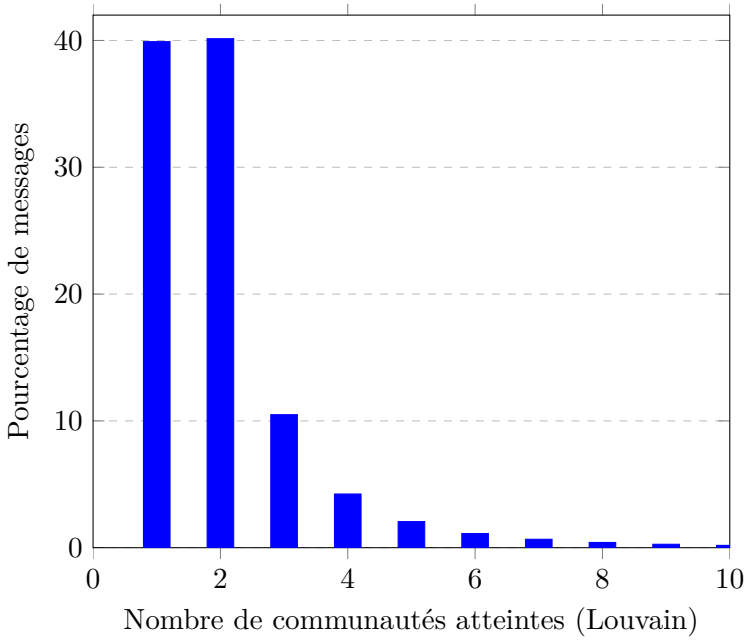


FIGURE 5.3: Répartition du nombre de communautés touchées

envisagées, nous avions besoin de comprendre la logique du regroupement des utilisateurs de manière plus fine. Notre méthode manuelle a été très efficace puisqu'une dimension commune semblait souvent se dégager de chaque sélection d'utilisateurs. Après avoir étiqueté chacune des communautés pour `Louvain` et `Infomap` , nous observons que trois principales dimensions semblent être à l'origine de la détection d'une communauté :

— *Un intérêt en commun* comme les communautés "Politique", "Start-up et technologies", "Passionnés de Tricot" etc. Ces communautés représentent 30% des communautés pour `Louvain` et 50% pour `Infomap`.

— *Géographique*. Les communautés qui regroupent des utilisateurs en fonction de leur localisation représentent 68% des communautés pour `Louvain` et 30% pour `Infomap`. `Louvain` en utilisant la modularité semble davantage détecter des enjeux géographiques comme "Royaume-Uni", le "Brésil" ou encore "Toronto". On voit également que la granularité de l'aspect géographique est flexible, allant du regroupement de tout un pays à une petite ville. Ceci est dû à notre jeu de données qui n'est qu'un extrait du réseau complet et qui donc très américano-centré. On voit ainsi que les

communautés anglophones ont été détectées avec une granularité plus fine, avec la détection de communautés correspondant à des villes ("Détroit", "Memphis").

— *Démographique.* Dans certains cas la communauté ne semble pas unie par une logique d'intérêt en commun, ni par une localisation précise mais par des traits démographiques. C'est le cas des communautés "Adolescentes", "Afro-américains". Ce type de communauté reste marginal puisqu'il ne représente que 2% des communautés détectées par `Louvain` et 4% pour `Infomap`.

Ces communautés sont bien entendu interconnectées. Afin de mieux percevoir la propagation des messages, nous nous intéressons au partage d'informations entre communautés. Comme chaque communauté partage au minimum quelques messages provenant des autres communautés, il est impossible de représenter le réseau du partage de messages entre communauté à partir des données brutes. De plus, il est nécessaire d'appliquer une normalisation selon la taille de la communauté afin de capturer plus finement certains liens entre communautés. Nous proposons donc une métrique permettant de mesurer le ratio de messages partagés par une communauté venant d'une autre communauté, cette métrique s'appuie sur la même idée que le *tf.idf* utilisé dans la communauté de recherche d'information ou le *Khi2* utilisé en statistiques :

$$score(X, Y) = \frac{nbRT(X \to Y)}{size(X)} - \frac{nbRT(Y)}{size(Twitter)} \tag{5.4}$$

Avec X et Y deux communautés, $nbRT(X \to Y)$ le nombre de messages partagés par X provenant par Y, $nbRT(Y)$ le nombre de partage total portant sur des messages provenant de Y et $size(Twitter)$ la taille de notre jeu. Autrement dit, si $score(X, Y) > 0$ alors un utilisateur de la communauté X a tendance à davantage partager les messages de Y qu'un utilisateur de la plateforme tiré au hasard.

Cette distance entre communauté combiné à l'étape d'étiquetage permet de représenter le réseau des communautés en figure 5.4. Nous ne représentons les liens entre communautés que lorsque $score(X, Y) > 1$, c'est à dire lorsqu'un utilisateur de la communauté X partage en moyenne au moins 1 message de plus qu'un utilisateur choisi aléatoirement dans le réseau.

La taille des communautés dans ce réseau est proportionnelle au nombre d'utilisateurs dans chaque communauté. La couleur des nœuds représente la proportion de messages partagés en interne, plus un nœud est foncé plus les utilisateurs de la communauté ont tendance à partager du contenu produit en interne.

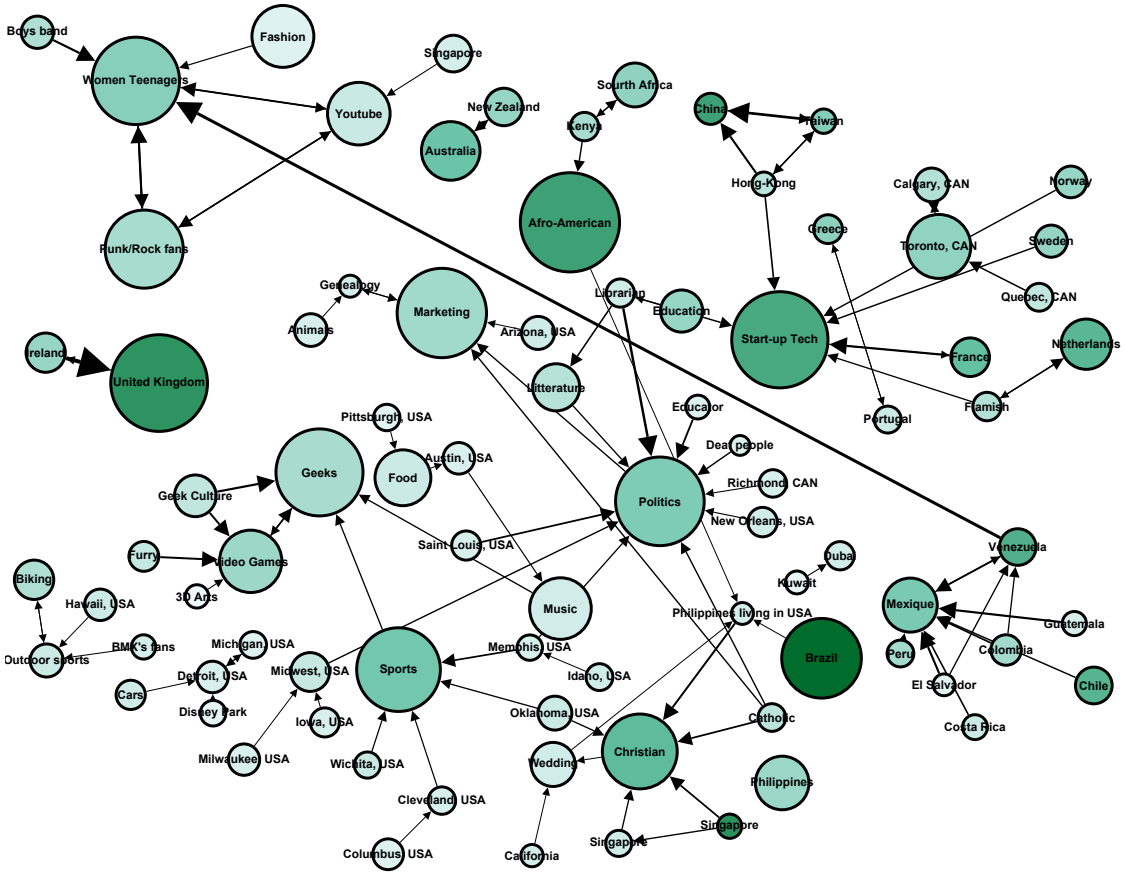


FIGURE 5.4: Connexions entre les différentes communautés

Il apparaît que des communautés construites sur une dimension géographique ont tendance à se connecter ensemble d'autant plus lorsque la langue est similaire ou qu'il existe une proximité géographique. On voit clairement émerger le bloc des communautés d'Amérique du sud par exemple. De la même façon de communautés qui partagent des intérêts relativement similaires vont avoir tendance à être proches comme "Geek" et "Jeux vidéo" ou "Éducation", "Libraire" et "Littérature".

5.3.4 Conclusion

En comparant plusieurs méthodes de détection de communautés nous montrons que `Louvain` est plus adapté à la topologie du réseau `Twitter`, `Infomap` produisant une longue traîne très importante de communautés minuscules et `Label Propagation` ne détectant qu'une seule communauté. Nous montrons que les communautés détectées regroupent les utilisateurs autour de 3 logiques : démographique, un intérêt en commun ou géographique. `Louvain` semble davantage produire de communautés géographiques quand `Infomap` produit davantage de communautés constituées autour d'un intérêt commun. Nous observons que peu de messages parviennent à se propager dans le réseau au-delà de 2 communautés, ce qui montre la tendance communautaire de la plateforme. Seuls les messages populaires, qui sont très minoritaires, parviennent à s'émanciper de leur communauté d'origine et toucher une partie plus vaste du réseau. Il semble que les communautés ont tendance à propager de l'information lorsqu'elles partagent un intérêt ou une langue en commun.

Grâce aux communautés détectées et aux liens qui les unissent, nous disposons d'un socle permettant de quantifier et de détecter les éventuelles bulles filtrantes. Nous allons étudier les effets produits par 3 des algorithmes de recommandations vus précédemment (*SimGraph*, *CF* et *GraphJet*) sur le comportement communautaire des individus.

5.4 Quantification des bulles

Nous allons désormais combiner les communautés détectées avec différents systèmes de recommandation afin d'être en mesure de quantifier un effet de bulle filtrante. Nous réalisons une première analyse globale pour tenter de déceler des communautés, qui seraient dans leur ensemble, victimes d'un effet bulle filtrante. Nous rentrerons ensuite à l'intérieur de ces communautés pour détecter des individus qui seraient personnellement prisonniers d'une bulle.

5.4.1 Approche globale

En figure 5.5 nous calculons le nombre de recommandations faites en interne aux utilisateurs, soit la proportion de messages recommandés provenant de la communauté d'origine

des utilisateurs. Nous observons que CF a tendance à faire moins de recommandations internes que $GraphJet$ et $SimGraph$, il semble qu'en éludant l'aspect topologique CF recommande davantage de messages provenant de communautés éloignés de la communauté de l'utilisateur.

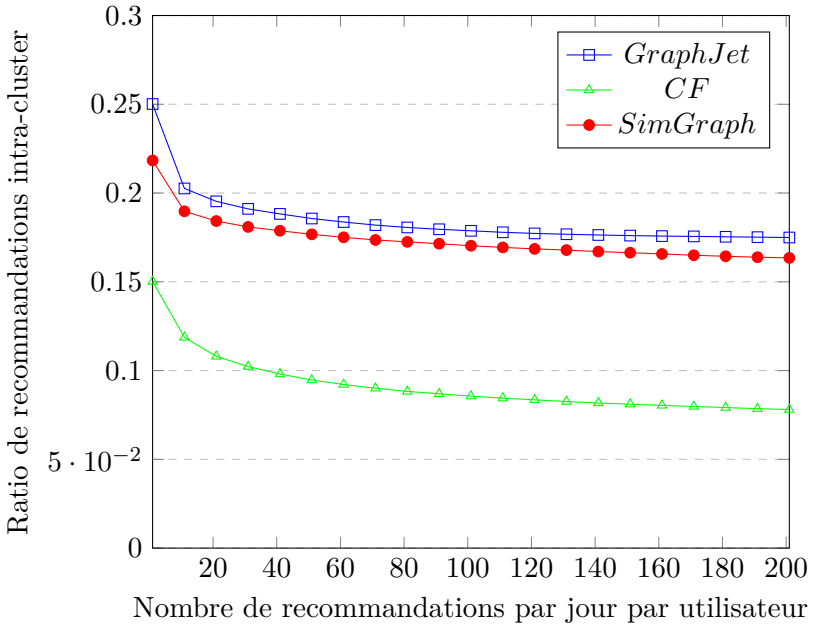
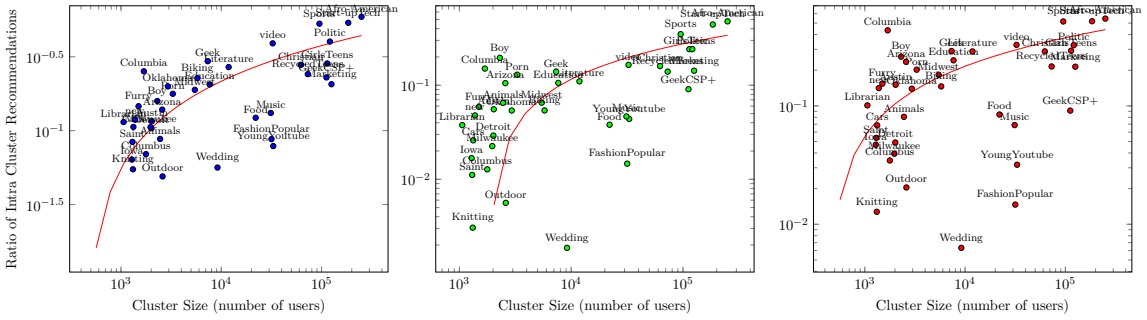


FIGURE 5.5: Ratio de recommandations intra-cluster

$GraphJet$ et $SimGraph$ obtiennent des résultats très similaires, 20% des recommandations produites par les systèmes proviennent de la communauté d'origine des utilisateurs. En effet, ces méthodes en reposant sur la topologie du réseau ont tendance à fournir davantage des recommandations de proximité. Nous remarquons également que le ratio de recommandations internes aux communautés reste stable au-delà du seuil de 20 recommandations par jour. Nous utilisons donc cette valeur de 20 recommandations par jour et par utilisateur dans le reste de nos travaux.

Existe-t-il des communautés davantage renfermées sur elle mêmes ? Nous quantifions dans cette partie le nombre de recommandations faites en interne pour chacune des communautés. Les résultats de cette étude sont visibles en figure 5.6. Il apparaît qu'il existe une relation logarithmique entre la taille d'une communauté et le nombre de recommandations faites en interne à ces utilisateurs. Plus une communauté est grande, plus elle publie de

messages, il est donc statistiquement probable qu'un message provenant d'une communauté importante se retrouve parmi les recommandations. Aucune communauté ne semble s'écarter sensiblement de cette courbe de tendance suggérant qu'il n'existe pas de communauté, traitée dans son ensemble, qui soit touchée par un effet de bulle filtrante.



GraphJet CF SimGraph

FIGURE 5.6: Proportion de recommandations interne selon les communautés (Louvain)

5.4.2 Approche locale

Comme il n'apparait pas de communauté spécialement plus renfermée que les autres, nous nous penchons sur une approche plus locale afin de détecter une bulle filtrante : des utilisateurs sont-ils eux pris au piège dans les recommandations personnelles qu'on leur envoie ?

Afin d'extraire la diversité des communautés d'origines des recommandations faites aux utilisateurs nous utilisons un coefficient de *Gini* [Gini 1912], habituellement utilisé pour mesurer l'inégalité de revenus dans une société. Nous transposons cette mesure à la distribution des communautés d'origines présentes dans l'ensemble des recommandations soumises à un utilisateur.

$$Gini = \frac{\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{\frac{2}{n}\sum_{k=1}^{n}x_k} \tag{5.5}$$

Nous analysons au travers de ce coefficient la répartition communautaire des recom-

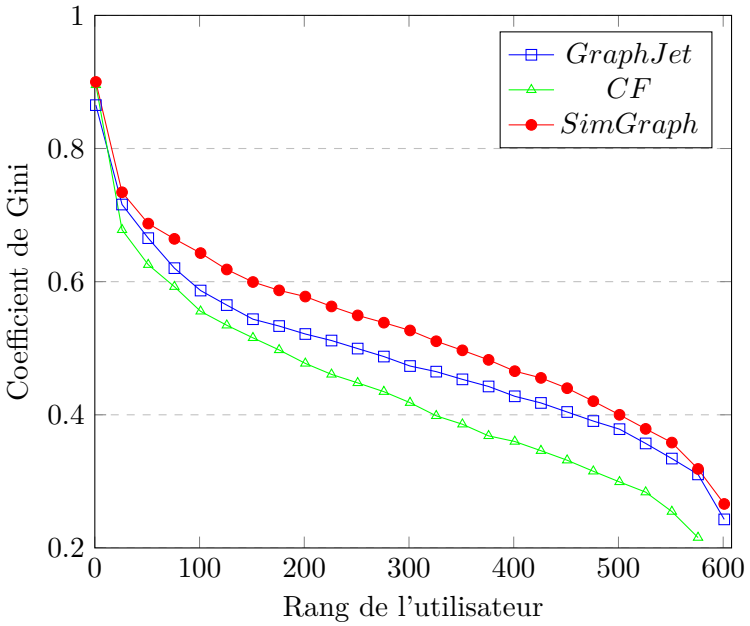


FIGURE 5.7: Distribution du coefficient de Gini des recommandations

mandations. Pour chaque utilisateur nous avons un ensemble de recommandations, et nous calculons le coefficient de Gini sur cet histogramme de communautés d'origines.

Le coefficient de Gini est borné entre 0 et 1, un coefficient proche de 0 indique une répartition équitable entre les communautés recommandées quand un coefficient proche de 1 indique au contraire une répartition inégale ou de nombreuses recommandations proviennent de la même communauté.

Figure 5.7 affiche la distribution du coefficient de Gini pour les utilisateurs de notre expérimentation. Des valeurs très élevées, supérieures à 0.6 indiquent des utilisateurs pour lesquels les recommandations sont fortement déséquilibrées et proviennent majoritairement d'un petit nombre de communautés.

Il est tentant d'affirmer que les utilisateurs avec des valeurs très élevées, par exemple l'utilisateur qui a un score Gini de 0.9, est sujet à un effet de bulle filtrante. En réalité c'est légèrement plus complexe. En effet cette affirmation est mise en doute lorsqu'on regarde plus précisément les utilisateurs possédant des scores proches de 1.

Par exemple, l'utilisateur au coefficient le plus élevé est un passionné de football qui appartient à la communauté *Sports*, qui n'interagit qu'avec des messages parlant de football.

Son usage de la plateforme est donc extrêmement spécifique, et les différents systèmes de recommandations se sont adaptés à cet usage afin de lui recommander uniquement des messages qui parlent de football. Il semble donc normal que la répartition des recommandations soit complètement inégale et largement en faveur de la recommandation de messages provenant de la communauté *Sports*.

Ainsi, il nous semble essentiel de prendre en considération l'usage de l'utilisateur afin de déterminer si celui-ci est sujet à un effet de bulle filtrante ou non. Nous proposons donc un score, qui est une différence de 2 scores Gini, et qui permet de représenter la réduction communautaire des recommandations envoyées à l'utilisateur par rapport à son usage de la plateforme.

$$BulleF(u) = Gini(u) - Gini(recos) \tag{5.6}$$

Où $Gini(u)$ correspond au coefficient de Gini de la répartition communautaire des messages aimé par un utilisateur u et $Gini(recos)$ le coefficient de Gini de la répartition communautaire des recommandations faite à l'utilisateur u.

Les résultats sont visibles en figure 5.8, des valeurs au-dessus de 0 indiquant que $Gini(u) > Gini(recos)$, autrement dit que la répartition des recommandations envoyées à un utilisateur est plus équitable que l'usage de l'utilisateur.

Il apparaît avec ces résultats que tous les systèmes de recommandations ont tendance à diversifier davantage les communautés d'origines des messages envoyés à l'utilisateur (plus 50% des utilisateurs) par rapport à l'usage de l'utilisateur qui n'aime des messages que lorsqu'ils proviennent d'une minorité de communautés.

On observe cependant des utilisateurs avec des scores en deçà de -0.2, indiquant que les différents systèmes de recommandations ont considérablement réduit la diversité des communautés d'intérêts de l'utilisateur. Il semble donc que pour environ 6% des utilisateurs, un effet de bulle filtrante soit réel. Si ce pourcentage parait particulièrement bas, à l'échelle de `Twitter` cela représente tout de même théoriquement 20 millions de personnes.

Au final, l'effet de bulle filtrante parait faible et observable uniquement à l'échelle des utilisateurs. Nous proposons un modèle qui modifie les recommandations reçus par les

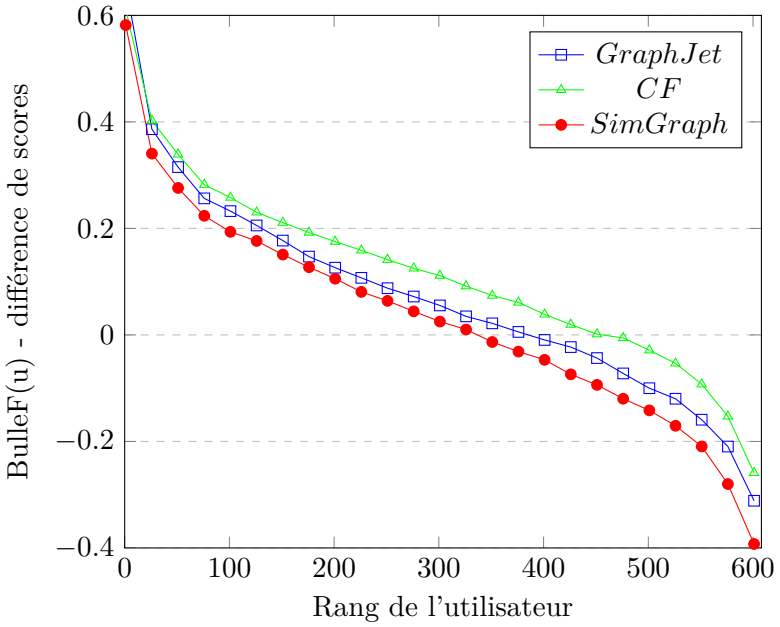


FIGURE 5.8: Différence de scores de Gini entre usage et recommandations (BullF)

utilisateurs en s'appuyant sur les similarités communautaires pour limiter l'effet de bulle pour les utilisateurs concernés.

5.5 CAM - Community Aware Model

Maintenant que nous sommes capables de détecter un effet de bulle filtrante pour les utilisateurs, nous proposons un modèle permettant de reclasser les recommandations reçues par les utilisateurs afin de réduire l'impact de la bulle filtrante pour les utilisateurs trop enclavés. Ce reclassement des recommandations, qui est une surcouche au-dessus d'un système de recommandation existant, permet de généraliser son utilisation et d'être indépendant du système de recommandation utilisé.

Considérons un utilisateur u et un réseau social où n communautés ont été détectées au travers d'un algorithme de détection de communautés.

Soit $\overrightarrow{Pu}$ le profil communautaire de l'utilisateur u avec pc_i le pourcentage de messages

aimés par l'utilisateur provenant de la communauté c_i.

$$\overrightarrow{Pu} = (pc_1, pc_2, \ldots, pc_n) \tag{5.7}$$

Le système de recommandation RS produit une liste de recommandations $LReco_u$ pour l'utilisateur u à partir duquel un top-k de messages sont extraits pour être présentés à l'utilisateur u.

Le modèle va reclasser la liste $LReco_u$ des recommandations afin de prendre en considération, pour chaque message, sa communauté d'origine, de façon à influer sur la liste des top-k messages et de proposer des messages plus proches de l'usage de l'utilisateur, c'est à dire plus proches de $\overrightarrow{Pu}$.

Un système naïf qui sélectionnerait directement les messages afin d'extraire un ensemble de messages représentatifs de $\overrightarrow{Pu}$ ne serait pas capable de fonctionner correctement. En effet, il est possible que des messages provenant d'une communauté fortement représentée dans $\overrightarrow{Pu}$ ne soient tout simplement pas présents dans la liste de messages $LReco_u$. De plus, si un message a un fort score de recommandation, mais qu'il provient d'une communauté différente de celles sur lesquelles se porte l'intérêt habituel de l'utilisateur, il semble maladroit de l'écarter.

Nous avons vu précédemment que des communautés pouvaient être très proches topologiquement ou sémantiquement, il semble donc naturel d'effectuer le reclassement des recommandations en prenant en compte la proximité des communautés.

Nous proposons donc un modèle capable de reclasser les recommandations d'un utilisateur en s'appuyant sur les scores calculés par le système de recommandation RS.

Similarités entre communautés Nous définissons la similarité entre communautés en prenant en compte la topologie du réseau, c'est à dire les liens qui relient les deux communautés, l'information sémantique, soit les thèmes des messages publiés dans une communauté et enfin la propagation d'information, à quel point une communauté partage des messages d'une autre communauté. La similarité entre les communautés c_i et c_j est

estimée par :

$$sim(c_i, c_j) = \alpha Links(c_i \rightarrow c_j) + \beta Sem(c_i, c_j) + \gamma Flow(c_i, c_j) \tag{5.8}$$

Où $Links$ est le nombre de liens orientés depuis c_i vers c_j, Sem représente la similarité sémantique entre deux communautés et $Flow$ est équivalent au $Score$ présenté précédemment pour capturer les partages entre communautés (Equation 5.4). α, β et γ sont des constantes qui permettent d'adapter le reclassement des messages en fonction du comportement général du système de recommandation.

A partir de cette mesure de similarités entre communautés, obtenons une matrice (CSM) :

$$CSM = \begin{array}{c} \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{array} \begin{array}{c} \begin{array}{cccc} c_1 & c_2 & \cdots & c_n \end{array} \\ \begin{pmatrix} 1 & sim_{12} & \dots & sim_{1n} \\ sim_{21} & 1 & \dots & sim_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ sim_{n1} & sim_{n2} & \dots & 1 \end{pmatrix} \end{array}$$

Reclassement des recommandations Pour chaque message de la liste des recommandations $LReco_u$ produite par RS pour l'utilisateur u, nous calculons le vecteur $\overrightarrow{i}$ de scores de recommandations en prenant en compte le score de recommandation original.

$$\overrightarrow{i} = \frac{1}{score(u, i)} \times \overrightarrow{CSM(c_i)} \tag{5.9}$$

La liste reclassée est donc un ensemble U de messages pour lesquels la sensibilité communautaires est $\overrightarrow{V_U}$ soit la somme normée de $\overrightarrow{i}$.

$$\overrightarrow{V_U} = \frac{\sum\limits_{i \in U} \overrightarrow{i}}{|| \sum\limits_{i \in U} \overrightarrow{i} ||} \tag{5.10}$$

Percer la bulle filtrante Notre modèle reclasse donc les propositions de messages issues du système de recommandation pour extraire un sous-ensemble respectant davantage la distribution communautaire du profil de l'utilisateur $\overrightarrow{Pu}$. L'objectif est donc de trouver

dans l'ensemble des suggestions, un sous-ensemble qui minimise la distance entre $\overrightarrow{V_U}$ et $\overrightarrow{Pu}$. Soit :

$$distance = |\overrightarrow{Pu} - \overrightarrow{V_U}| \tag{5.11}$$

Calculer l'ensemble des combinaisons d'un ensemble x de messages est de complexité $O(C_y^x)$ avec y la taille de la liste $LReco_u$ de recommandations d'origine. Il est possible d'optimiser considérablement la complexité de sélection des messages en ne testant qu'un sous-ensemble de solutions. A partir de la liste de messages recommandés par le système, nous extrayons les x les plus hauts scores de recommandations, nous parcourons ensuite la liste des messages restants du classement en testant x possibilités de changement. Cette opération permet de réduire la complexité à $O((x - y) \times y)$.

5.5.1 Modèle jouet

Afin d'illustrer le modèle de reclassement de recommandations CAM et de mieux comprendre les différentes étapes qui aboutissent à un rééquilibrage des recommandations, nous proposons un petit modèle jouet. Soit un utilisateur Joe pour lequel un système de recommandation a calculé une liste de messages intéressants.

Rang	Communauté d'origine	Score
1	A	0.8
2	A	0.7
3	A	0.5
4	B	0.4
5	C	0.1

TABLE 5.2: Liste de recommandations pour Joe

Dans cet exemple, nous allons supposer que seulement le $top-3$ des messages recommandés sont effectivement montrés à Joe. Dans notre cas, Joe reçoit donc 3 recommandations issues de la même communauté A. Nous allons également supposer qu'il n'existe que 3 communautés dans le réseau et qu'il n'existe aucune similarité entre elles. La matrice CSM est donc égale à la matrice identité. Dans notre exemple, Joe interagit uniformément avec ces 3 communautés d'où :

$$\overrightarrow{P_{Joe}} = (0.33, 0.33, 0.33)$$

Nous reclassons la liste des recommandations en prenant en compte son profil et les scores de recommandations calculés par le système pour minimiser la distance présentée en équation 5.11 avec pour vecteur communautaire celui calculé grâce à l'équation 5.10. Il existe donc $\binom{5}{3} = 10$ possibilités, l'objectif est donc de trouver la combinaison qui minimise la distance entre les deux vecteurs présenté en équation 5.11. Le tableau des combinaisons possibles est présenté dans le tableau 5.3.

Combinaison	Score
{1,2,3}	0.81
{1,2,4}	0.40
{1,2,5}	0.57
{1,3,4}	0.41
{1,3,5}	0.54
{1,4,5}	0.48
{2,3,4}	0.42
{2,3,5}	0.47
{2,4,5}	0.47
{3,4,5}	0.43

TABLE 5.3: Scores pour toutes les combinaisons de 3 recommandations

Dans notre exemple il apparait que la combinaison de trois recommandations qui minimise la distance présentée en équation 5.11 correspond à la combinaison des messages 1, 2 et 4. C'est-à-dire, un compromis entre qualité de recommandation (les scores de recommandation) et la diversité des communautés d'origine de ces messages. La façon dont ce reclassement est effectué est donc directement lié aux similarités calculés entre communautés, et plus spécifiquement aux poids α, β et γ qui permettent de donner plus ou moins d'importance à certaines communautés. Nous allons détailler les détails du protocole expérimental employé pour mesurer l'impact de ces poids avec notre stratégie de reclassement sur la bulle filtrante en utilisant différents algorithmes de recommandations.

5.6 Expérimentation

5.6.1 Protocole

Nous conservons donc cette différence entre le coefficient de Gini du profil de l'utilisateur et des recommandations décrit en section 5.4.2.

Afin de déterminer les conséquences du reclassement des recommandations suivant le profil des utilisateurs, pour chaque communauté détectée par *Louvain*, nous extrayons 16 utilisateurs comprenant 4 utilisateurs présentant un petit profil (entre 10 et 50 RT), 4 utilisateurs ayant un moyen profil (50 et 90RT), 4 utilisateurs avec un profil actif (120 et 300 RT) et enfin 4 utilisateurs très actifs (plus de 300RT).

Nous réutilisons notre ensemble Set_{RT} de messages partagés au moins 2 fois décrit en section 4.3.2. Pour chaque système de recommandation (CF, $SimGraph$ et $GraphJet$) nous évaluons les recommandations faites à ces utilisateurs en appliquant ou non notre méthode de reclassement.

Comme formulé précédemment, le modèle de reclassement des recommandations repose sur la similarité entre communautés (Equation 5.8). Or, ce score de similarité dépend de 3 poids α, β et γ respectivement associés à la similarité topologique, la similarité sémantique et la similarité d'influence. Nous réalisons donc une expérimentation pour étudier l'impact de chaque poids sur la qualité du reclassement produit.

Chaque poids est situé entre 0 et 1 et nous utilisons un pas de 0.1 entre les valeurs pour nos expérimentations. Chaque poids peut donc adopter 11 valeurs, ce qui donne pour nos trois poids $11^3 = 1331$ configurations possibles. Pour des questions de visibilités des résultats, nous ne présentons que les cas où α est égal à 0.1, 0.5 et 1 tandis que toutes les configurations de β et de γ sont présentées.

Afin d'évaluer la similarité sémantique entre les communautés nécessaires à notre algorithme de reclassement, nous regroupons l'ensemble des messages produits par chaque communauté pour former un document par communauté. Nous utilisons ensuite les outils de CoreNLP [Manning, Surdeanu, Bauer, Finkel, Bethard, and McClosky 2014] pour extraire les entités nommées de chaque communauté et ne conservons que les entités les

plus représentées. Pour chacune de ces entités nous utilisons leur représentation vectorielle extraite à partir des données produites par Word2Vec entrainés sur Google News (100 milliards de mots) [1]. Nous obtenons ainsi une représentation vectorielle qui correspond au profil sémantique de chaque communauté permettant ainsi de les comparer en utilisant une mesure de similarité comme un cosinus.

Le nombre de liens directs entre communautés est utilisé pour calculer la similarité topologique *Links*. La similarité d'influence *Flow* est calculé avec la même métrique utilisée pour construire la cartographie des communautés en équation 5.4.

5.6.2 Impact des différents poids

Les résultats de notre étude sont présentés en figure 5.9 pour *GraphJet*. Les trois tableaux du haut représentent le nombre de bonnes prédictions parmi les recommandations (*hits*) réalisées par le système pour $\alpha = 0.1$ à gauche, $\alpha = 0.5$ au centre et $\alpha = 1$ à droite. Les trois tableaux du bas représentent quant à eux le nombre d'utilisateurs souffrant d'un effet de bulle filtrante. C'est à dire les utilisateurs pour lesquels la différence de Gini présenté précédemment est inférieur au seuil de -0.2. Les mêmes résultats sont présentés en figure 5.10 pour *SimGraph* et en figure 5.11 pour CF.

Nous observons dans un premier temps que quel que soit le système de recommandation envisagé, les résultats sont très variables autant sur le nombre de *hits*, allant de 492 à 653 pour *GraphJet* par exemple. Ce qui représente une différence de 32%. On constate même un ordre de variabilité supérieur pour le nombre d'utilisateurs affectés par un effet de bulle filtrante. Pour CF, par exemple, au minimum 6 utilisateurs sont concernés pour la meilleure configuration et dans le pire des cas 128 utilisateurs sont concernés.

Il apparait qu'en jouant avec les différentes valeurs nous avons identifié des leviers permettant d'augmenter sensiblement la pertinence des recommandations ou de baisser le nombre d'utilisateurs sujets à un effet de bulle. Il y a de manière évidente une corrélation entre la pertinence du système et le nombre d'utilisateurs affectés par un effet de bulle. Plus on cherche à recommander des messages concernant un intérêt précis de l'utilisateurs, plus la pertinence va augmenter au détriment des utilisateurs pris dans une bulle, qui verront

1. https ://code.google.com/archive/p/word2vec/

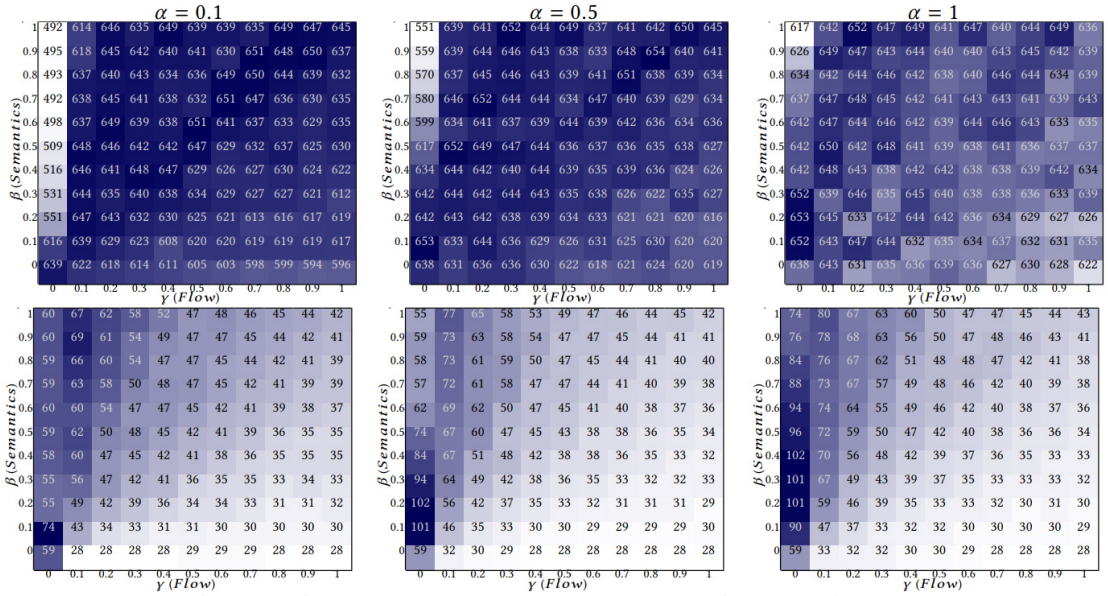


FIGURE 5.9: # de *hits* (en haut) et # d'utilisateurs affectés par une bulle filtrante (en bas) pour *GraphJet* selon γ et β

moins de recommandations provenant de certains de leurs intérêts secondaires.

Une tendance générale se dégage de l'analyse des trois systèmes de recommandations. En donnant moins de poids à la dimension sémantique (β), le reclassement des messages donne leur chance à des messages traitant d'un sujet différent de l'intérêt principal de l'utilisateur, ce qui réduit également le nombre d'utilisateurs dans une bulle en diversifiant les communautés d'origines de messages recommandés. Mais en réduisant ce poids, certaines recommandations pertinentes passent à la trappe et ne sont pas montrés à l'utilisateur, ce qui implique une baisse dans la qualité globale de prédiction.

Donner du poids à la similarité d'influence (γ) produit un effet inverse. Avec de fortes valeurs, le reclassement va avoir tendance à donner du poids à des messages de communautés différentes que celles avec lequel l'utilisateur interagi habituellement baissant également la qualité globale de prédiction.

Cette expérimentation montre qu'il n'existe pas de configuration qui maximise à la fois la qualité des recommandations tout en minimisant le nombre d'utilisateurs affectés par une bulle. Les différentes valeurs données aux poids servent ainsi d'outil afin d'atteindre un objectif d'amélioration de pertinence du système, de faire décroitre le nombre d'utilisateurs

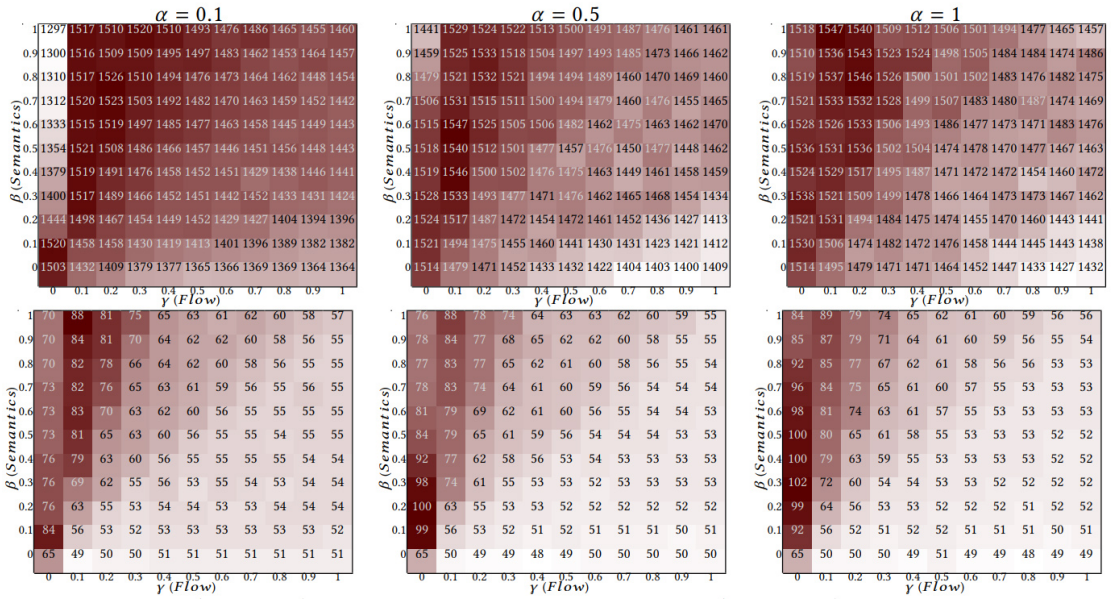


FIGURE 5.10: # de *hits* (en haut) et # d'utilisateurs affectés par une bulle filtrante (en bas) pour *SimGraph* selon γ et β

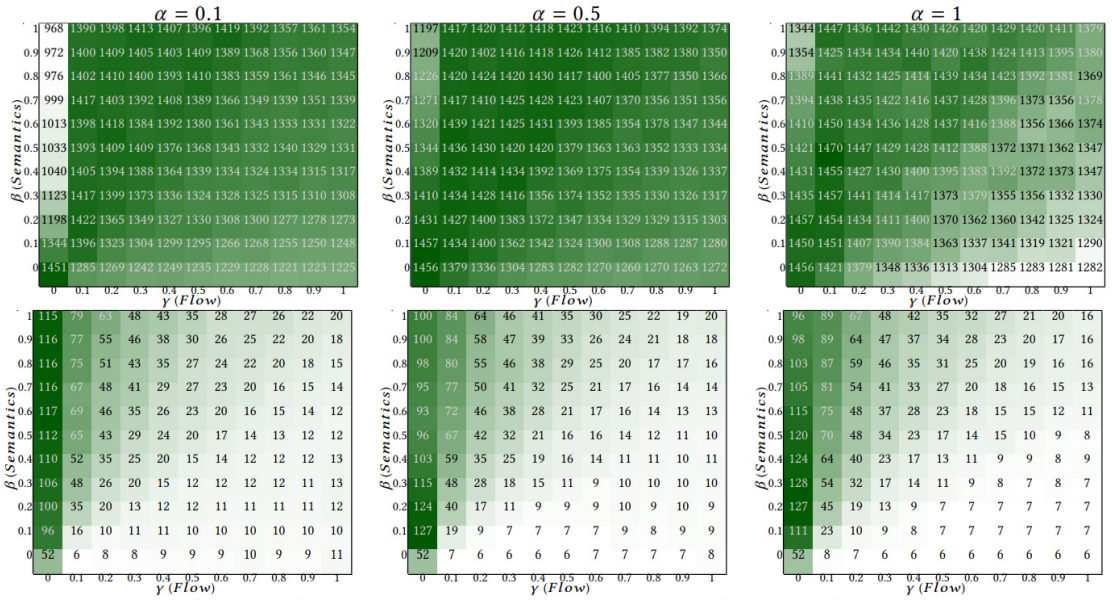


FIGURE 5.11: # de *hits* (en haut) et # d'utilisateurs affectés par une bulle filtrante (en bas) pour *CF* selon γ et β

dans une bulle ou bien de trouver un compromis entre les deux. Nous nous concentrons dans notre cas sur les configurations minimisant le nombre d'utilisateurs affectés par une bulle.

5.6.3 Meilleures configurations

Cette expérimentation a pour but d'illustrer les gains réalisés en utilisant notre stratégie de reclassement CAM au-dessus des différents systèmes de recommandations. Pour chaque tableau de résultats présentés dans les figures 5.9, 5.10 et 5.11, nous choisissons la configuration de poids minimisant le nombre d'utilisateurs sujets à une bulle. Le tableau 5.4 présente le nombre d'utilisateurs parmi nos 608 utilisateurs sujets à la bulle filtrante avec ou sans l'utilisation de CAM ainsi que le nombre de *hits* réalisés. Ainsi, notre stratégie de reclassement fait décroitre avec succès le nombre d'utilisateurs affectés par un effet de bulle pour 15% avec $GraphJet$, 64% avec CF et 23% avec $SimGraph$.

Parallèlement, en utilisant la dimension communautaire pour reclasser les recommandations et être plus en accord avec le profil des utilisateurs, nous améliorons en même temps la pertinence globale du système. Pour $Graphjet$ on observe une hausse de 15% des bonnes prédictions et pour CF une hausse de 2%. C'est seulement avec CF que le reclassement fait décroitre drastiquement le nombre d'utilisateurs concernés par une bulle filtrante au prix d'une baisse de la pertinence du système (-3% de $hits$).

Il semble que notre modèle arrive mieux à enlever des utilisateurs affectés par une bulle filtrante pour CF. On peut expliquer ce phénomène car en adoptant une stratégie de reclassement, le modèle est limité par la quantité des recommandations qui se trouvent dans cette liste. Or nous avons vu précédemment que CF est capable de calculer de large listes de recommandations, laissant ainsi plus de marche de manœuvre à une stratégie de reclassement de choisir les messages adaptés.

5.6.4 Impact suivant l'usage des utilisateurs

En figure 5.12, nous nous intéressons au lien entre l'activité des utilisateurs et la bulle filtrante. Nous observons qu'une majorité des utilisateurs concernés par un effet de bulle sont les utilisateurs ayant une faible activité sur la plateforme. En effet, les utilisateurs ayant

	État Initial		Meilleure configuration	
	Hits	Nb Bulle Filtrante	Hits	Nb Bulle Filtrante
GraphJet	552	33	630 (+14%)	28 (-15%)
CF	1400	17	1348 (-3%)	6 (-64%)
SimGraph	1468	62	1491 (+2%)	48 (-23%)

TABLE 5.4: Impact des configurations minimisant le nombre d'utilisateurs affectés par une bulle pour *CAM*

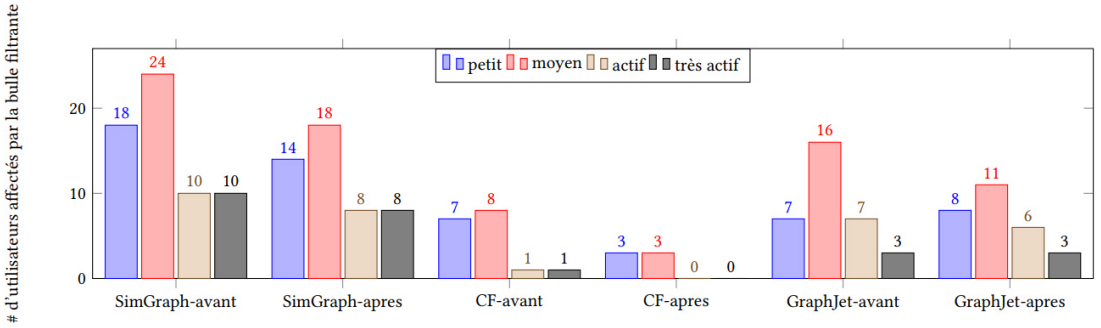


FIGURE 5.12: Nombre d'utilisateurs affectés par une bulle filtrant avec ou sans utiliser *CAM*

un petit profil (moins de 50 RT) ou un profil moyen (moins de 90RT) représentent plus de 70% des utilisateurs concernés par une bulle. En disposant de moins d'informations sur ces comptes, les systèmes de recommandations vont se concentrer sur un principal intérêt détecté dans leur profil. Ainsi utiliser *CAM* permet de mettre en valeur des messages qui étaient peu considérés au départ du fait de la faible présence de cet intérêt dans le profil de ces utilisateurs.

5.7 Conclusion

En utilisant plusieurs méthodes de détection de communautés sur `Twitter` nous avons montré que les communautés détectées avaient tendance à recouper des logiques géographiques ou d'intérêt commun. Il semble que l'aspect communautaire soit très présent dans `Twitter` avec très peu de messages capables de se propager au-delà de 2 communautés, mais nous trouvons peu d'impact concernant le renforcement communautaire dû aux systèmes de recommandation. Nous observons plutôt une tendance inverse dans laquelle, paradoxalement, les systèmes de recommandation ont proposé des messages plus diversifiés comparé à

l'usage spontané des utilisateurs. La bulle filtrante ne semble concerner finalement qu'un faible pourcentage d'utilisateurs.

Cependant, même si ce pourcentage est faible, à l'échelle d'une plateforme de taille mondiale cela représente tout de même de nombreux utilisateurs et nous proposons une méthode de reclassement des recommandations en s'appuyant sur les liens entre communautés et le profil de l'utilisateur pour éviter une distribution de messages trop déséquilibrée. En se basant sur une mesure de similarité pondérée pour reclasser les recommandations, nous montrons qu'il est possible au choix, de maximiser la pertinence globale du système, de minimiser le nombre d'utilisateurs dans une bulle ou de réaliser un compromis entre ces deux variables. Nous observons que donner beaucoup de poids à la sémantique lors du reclassement, tend à accroitre la pertinence du système au détriment du nombre d'utilisateurs dans une bulle. A l'inverse donner de l'importance à la similarité d'influence fait décroitre le nombre d'utilisateurs dans une bulle mais réduit également la pertinence globale du système. Nous montrons qu'en reclassant les propositions avant de soumettre les recommandations à l'utilisateur, avec la meilleure configuration des poids, nous arrivons à sensiblement limiter l'effet de bulle filtrante pour 66% avec CF, 23% avec $SimGraph$ et 15% avec $GraphJet$. Cette différence d'efficacité s'explique principalement par la capacité de CF à produire de longues listes de recommandations pour les utilisateurs, permettant ainsi davantage de possibilités de reclassement. A l'inverse, les courtes listes produites par $SimGraph$ et $GraphJet$ sont plus difficiles à exploiter par notre méthode.

En parallèle de cette chute du nombre d'utilisateurs affectés par une bulle, nous observons que dans la plupart des cas en proposant des messages plus en accord avec les différentes communautés d'interactions de l'utilisateur, le système accroit également sa prédiction. Ainsi, $GraphJet$ augmente ses *hits* de 14% et $SimGraph$ de 2%. Seul CF voit une légère baisse de sa qualité de prédiction, en réalisant 2% de *hits* en moins.

Chapitre 6

Conclusion

Dans un premier temps l'objectif de ce travail de thèse est de mieux comprendre les caractéristiques intrinsèques des plateformes de micro-blogging. Ces éclairages permettent de proposer un système de recommandation adapté qui passe à l'échelle. Le second objectif de ce travail consiste à analyser le comportement communautaires des utilisateurs sur ces mêmes plateformes et de discerner l'impact des systèmes de recommandation sur la consommation d'information des individus. Ce chapitre revient sur les différentes contributions de cette thèse et propose différentes perspectives de recherche.

6.1 Contributions

Après avoir collecté un jeu de données représentatif de `Twitter` nous avons réalisé différentes analyses approfondies pour mieux cerner les caractéristiques d'une plateforme de micro-blogging.

Nous montrons ainsi que la topologie de `Twitter` est restée stable avec les années. Le réseau reste un petit monde, avec une distance moyenne qui sépare deux nœuds d'un peu plus de 3 sauts. Les distributions de liens entrants et liens sortants sont également similaires à celles trouvées par de précédentes études réalisés il y a une dizaine d'années. On distingue ainsi une forte disparité entre une majorité d'utilisateurs très peu suivis et quelques utilisateurs énormément suivis. Nous montrons cependant qu'avec l'augmentation du nombre de messages publiés chaque jour, la durée de vie des messages s'est considérablement raccourcie [Grossetti, Du Mouza, and Travers 2017]..

Nous analysons également l'homophilie présente dans le réseau et découvrons que des utilisateurs peuvent partager beaucoup de contenu en commun sans pour autant se suivre mutuellement. Il apparaît que les utilisateurs présentant les scores de similarités les plus élevées sont majoritairement situés dans un voisinage proche, jusqu'à deux sauts.

Grâce à cette compréhension plus fine de la plateforme nous proposons une méthode reposant sur la construction d'un réseau orienté pondéré capturant efficacement les liens de similarités entre utilisateurs. *SimGraph*, ce graphe de similarité, permet également d'exploiter la transitivité entre utilisateurs similaires et d'être ainsi en mesure de faire de recommandations lorsqu'un message a rencontré peu d'interactions [Grossetti, Constantin, du Mouza, and Travers 2018].

Nous comparons notre modèle à des solutions issues de l'état de l'art, dont une méthode de filtrage collaboratif naïve, une méthode d'inférence probabiliste et *GraphJet* la méthode utilisée par `Twitter`. Nos résultats montrent que notre système est capable de surpasser les autres méthodes en qualité de prédiction ainsi qu'en temps de calcul. En combinant la transitivité entre utilisateurs, la topologie du réseau et les similarités entre individus, *SimGraph* est capable de faire des recommandations de messages populaires et de messages plus confidentiels. Nous montrons que seules deux solutions sont capables de passer à l'échelle, *SimGraph* et *GraphJet* et que notre solution est légèrement moins couteuse pour une précision multipliée par 4.

Nous évaluons également plusieurs méthodes de mise à jour de notre graphe de similarité et montrons qu'il est possible de le mettre à jour à bas coût. Expérimentalement ce procédé, qui consiste à réappliquer la découverte de similarité entre utilisateurs à distance 2 dans une itération précédente du graphe de similarités converge rapidement, ce qui rend l'utilisation de cette stratégie réaliste.

Nous présentons également une analyse quantitative et qualitative des communautés détectées par différents algorithmes détection de communautés. Il apparaît que parmi les méthodes de partitionnement qui passent à l'échelle, `Louvain` et `Infomap` produisent des résultats comparables malgré une longue traîne de minuscules communautés détectées par `Infomap`. `Label Propagation` en revanche ne parvient pas à partitionner le réseau efficacement du fait de sa forte connectivité et ne détecte qu'une seule communauté

comprenant tous les utilisateurs du réseau.

Nous montrons que `Louvain` a tendance à détecter des communautés suivant une logique géographique quand `Infomap` trouve des communautés moins volumineuses, davantage orientés par les intérêts des utilisateurs [Grossetti, Constantin, du Mouza, and Travers].

Ces communautés permettent de confirmer le comportement communautaire des utilisateurs, qui partagent principalement de l'information provenant de leur propre communauté. En observant l'impact des systèmes de recommandation sur l'enfermement communautaire des utilisateurs nous arrivons à la conclusion qu'il n'existe pas de communauté particulièrement isolée et que peu d'individus sont touchés.

Ces résultats vont dans le sens de différentes études sur les bulles filtrantes qui montrent qu'un faible nombre d'utilisateurs est concerné par ce phénomène. Les systèmes de recommandation ont tendance à davantage diversifier les messages reçus par les utilisateurs.

Nous proposons un modèle qui agit comme une couche intermédiaire entre le système de recommandation et l'utilisateur. Nous montrons qu'en utilisant les différents liens qui unissent les communautés comme les liens topologiques, les liens de propagation et les liens sémantiques, il est possible de modifier les recommandations envoyées aux utilisateurs en amont afin de limiter l'effet de bulle filtrante.

En parallèle de cette baisse de nombre d'utilisateurs affectés par un effet de bulle, proposer des messages plus en accord avec le profil des utilisateurs augmente également la qualité globale des systèmes de recommandation.

Après une analyse du profil des utilisateurs affectés par un effet de bulle, il apparait que les utilisateurs concernés sont principalement des utilisateurs ayant peu interagi avec la plateforme. En disposant de moins d'informations sur les utilisateurs, les systèmes de recommandation se concentre sur l'un des intérêt détectés chez l'utilisateur réduisant ainsi l'assiette communautaire des recommandations.

6.2 Perspectives

Les progrès réalisés en apprentissage profond commencent déjà à se répercuter sur le domaine de recherche des systèmes de recommandation. Nous avons vu dans l'état l'art des

exemples de très importantes plateformes internet qui ont adoptés ces modèles reposant sur l'utilisation de réseaux de neurones.

Cette évolution technologique ouvre la porte à de nouvelles problématiques de recherche en terme de passage à l'échelle et d'extraction d'informations provenant des interactions des utilisateurs et du contenu.

Si ces méthodes par apprentissage profond ont fait leurs preuves pour des plateformes disposant d'un catalogue de contenu réduit, il n'est pas évident que le compromis entre le coût d'apprentissage nécessaire pour construire le modèle et la qualité des recommandations soit favorable lorsque le catalogue est dantesque. Nous avons montré que des modèles simples pouvaient être très efficaces lorsqu'ils exploitaient finement les caractéristiques d'un type de plateformes. Il semble donc particulièrement pertinent d'adapter les réseaux de neurones en y intégrant ces caractéristiques pour tenter d'obtenir de meilleures performances.

Les perspectives ouvertes par les réseaux de neurones sont très stimulantes mais ces systèmes vont être confrontés à des problématiques générales déjà présentes chez leurs prédécesseurs.

Les systèmes de recommandation ont en effet la particularité d'être utilisés par des humains, aussi est-il vital de comprendre les critères qui font d'une recommandation, une « bonne » recommandation. Plus généralement, réussir à comprendre plus finement les attentes qu'ont les utilisateurs de ces systèmes. C'est l'enjeu des recherches sur les recommandations contextuelles par exemple, qui tentent de mieux discerner les conditions qui vont faire d'une recommandation une proposition très pertinente. Faire de l'utilisateur un acteur conscient et maître des options du système de recommandation semble être une autre option exploitable.

Nous envisageons ainsi plusieurs pistes afin d'améliorer ce travail de thèse et dépasser certaines limites concernant sa généralisation, la qualité des analyses et la qualité de recommandation de notre modèle.

Évaluer plus finement les différents systèmes. En effet, nous évaluons les systèmes de recommandation au travers d'une métrique de prédiction, c'est une limite importante

de nos résultats. Un système capable de mieux prédire le comportement des utilisateurs n'est pas synonyme d'un meilleur système de recommandation. Réaliser une enquête auprès d'un panel d'individus pour savoir quelles sont les recommandations qu'ils trouvent les plus pertinentes serait une source d'enseignement très enrichissante. Déterminer les critères de qualité des recommandations parait essentiel afin de bâtir des systèmes qui améliorent l'expérience des utilisateurs. Nos expériences sont également limitées par l'utilisation d'un seul jeu de données issue de `Twitter`. Évaluer la stabilité de la qualité des recommandations pour les autres plateformes de micro-blogging nécessiterait de réaliser les mêmes expérimentations sur des plateformes concurrentes.

Maximiser les points de comparaison entre utilisateurs. Dans nos travaux nous exploitons une similarité entre utilisateurs directement à partir de leur historique d'interactions. Or, nous avons vu précédemment qu'avec cette similarité, les scores entre utilisateurs étaient très faible. L'intersection de deux ensembles d'interactions est souvent limité car les utilisateurs n'ont pas forcément interagi avec exactement le mêmes message, alors que ceux-ci peuvent avoir des contenus très similaires. Une solution consisterait à utiliser des méthodes de traitement automatique du texte des messages afin de déterminer si deux messages ont des contenus similaires et de les grouper. Des méthodes d'extractions d'entités nommées combinées à des classificateurs peuvent ainsi être envisagées. Une autre option consiste à trouver un espace restreint de représentation des profils utilisateurs, ce qui permettrait de maximiser les scores de similarité et de capturer des liens d'influence entre utilisateurs qui échappent actuellement à notre modèle.

Caractériser l'évolution dans le temps des bulles et les utilisateurs. Notre étude des bulles filtrantes repose sur une vision statique des données. Comprendre si un utilisateur reste dans une bulle dans la durée ou si c'est un comportement erratique de la part du système de recommandation permettrait de décider quand reclasser les recommandations envoyées à l'utilisateur ainsi de réduire les coûts de l'algorithme de reclassement. Conjointement, à partir de nos données nous pouvons analyser le comportement et le profil des utilisateurs ayant des indices de cloisonnement élevés afin d'y détecter un motif commun. Comprendre s'il existe une corrélation entre le profil de l'utilisateur, par exemple le nombre

de personne auxquelles il est abonné, la fréquence de ses publications et son indice de cloisonnement, permettrait également de déterminer quels utilisateurs seront les plus affectés par les algorithmes de recommandations.

Liste des publications

CONSTANTIN Camelia, DAHIMENE Ryadh, GROSSETTI Quentin, DU MOUZA Cédric. Finding Users of Interest in Micro-blogging Systems. In : *International Conference on Extending Database Technology*, EDBT. 2016.

CONSTANTIN Camelia, DAHIMENE Ryadh, GROSSETTI Quentin, DU MOUZA Cédric. Recommandation contextuelle d'utilisateurs pour les plateformes de micro-blogging. In : *Ingénierie des Systèmes d'Information*, ISI. 2016.

GROSSETTI Quentin, DU MOUZA Cédric, TRAVERS Nicolas. Tweet, Retweet et Follower : que recommander et à qui ?. In : *AISR2017*. 2017.

GROSSETTI Quentin, DU MOUZA Cédric, CONSTANTIN Camelia, TRAVERS Nicolas. An Homophily-based Approach for Fast Post Recommendation in Microblogging Systems. In : *International Conference on Extending Database Technology*, EDBT. 2018.

GROSSETTI Quentin, DU MOUZA Cédric, CONSTANTIN Camelia, TRAVERS Nicolas. Community-based Recommendations on Twitter : Avoiding The Filter Bubble. Under Review : *International Conference on Extending Database Technology*, EDBT. 2019.

Bibliographie

Jae-wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Yeon Syn. Open user profiles for adaptive news systems : help or harm ? In *Proceedings of the 16th international conference on World Wide Web*, pages 11–20. ACM, 2007.

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. Homophily and latent attribute inference : Inferring latent attributes of twitter users from neighbors. *ICWSM*, 270 :2012, 2012.

Eytan Bakshy, Solomon Messing, and Lada A Adamic. Exposure to ideologically diverse news and opinion on facebook. *Science*, 348(6239) :1130–1132, 2015.

Pablo Barberá, John T Jost, Jonathan Nagler, Joshua A Tucker, and Richard Bonneau. Tweeting from left to right : Is online political communication more than an echo chamber ? *Psychological science*, 26(10) :1531–1542, 2015.

Parantapa Bhattacharya, Muhammad Bilal Zafar, Niloy Ganguly, Saptarshi Ghosh, and Krishna P. Gummadi. Inferring User Interests in the Twitter Social Network. In *Proc. Intl. Conf. on Recommender Systems (RECSYS)*, pages 357–360, 2014.

Daniel Billsus and Michael J Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2-3) :147–180, 2000.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics : theory and experiment*, 2008(10) :P10008, 2008.

Ludvig Bohlin, Daniel Edler, Andrea Lancichinetti, and Martin Rosvall. *Community Detection and Visualization of Networks with the Map Equation Framework*, pages 3–34. Springer International Publishing, Cham, 2014.

N.A. Bonafous. *La Rhétorique d'Aristote. Traduite en français avec le texte en regard, et suivie de notes philologiques et littéraires par Norbert Bonafous.* 1856. URL `https://books.google.fr/books?id=DGlZAAAAcAAJ`.

John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

Robin Burke. Hybrid recommender systems : Survey and experiments. *User modeling and user-adapted interaction*, 12(4) :331–370, 2002.

Òscar Celma and Xavier Serra. Foafing the music : Bridging the semantic gap in music recommendation. *Web Semantics : Science, Services and Agents on the World Wide Web*, 6(4) :250–256, 2008.

Òscar Celma, Miquel Ramírez, and Perfecto Herrera. Foafing the music : A music recommendation system based on rss feeds and user preferences. In *in ISMIR*. Citeseer, 2005.

Sonny Han Seng Chee, Jiawei Han, and Ke Wang. Rectree : An efficient collaborative filtering method. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 141–151. Springer, 2001.

Cisco. The zettabyte era : Trends and analysis. `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html`, 2017.

Elanor Colleoni, Alessandro Rozza, and Adam Arvidsson. Echo Chamber or Public Sphere ? Predicting Political Orientation and Measuring Political Homophily in Twitter Using Big Data. *Jour. of Communication*, 64(2) :317–332, 2014.

Michael Conover, Jacob Ratkiewicz, Matthew R Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini. Political polarization on twitter. *Icwsm*, 133 :89–96, 2011.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM, 2016.

Luis M De Campos, Juan M Fernández-Luna, Juan F Huete, and Miguel A Rueda-Morales. Combining content-based and collaborative recommendations : A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7) :785–799, 2010.

Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In *Recommender Systems Handbook*, pages 119–159. Springer, 2015.

Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1) :143–177, 2004.

Sander Dieleman. Recommending music on spotify with deep learning. `http://benanne.github.io/2014/08/05/spotify-cnns.html`, 2014.

Darcy DiNucci. Fragmentedfuture. *Print Magazine*, 4 :32, 1999.

Elizabeth Dubois and Grant Blank. The echo chamber is overstated : the moderating effect of political interest and diverse media. *Information, Communication & Society*, 21(5) : 729–745, 2018.

Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288. International World Wide Web Conferences Steering Committee, 2015.

Seth Flaxman, Sharad Goel, and Justin M Rao. Filter bubbles, echo chambers, and online news consumption. *Public Opinion Quarterly*, 80(S1) :298–320, 2016.

Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5) :75–174, 2010.

Simon Funk. Netflix update : Try this at home. `http://sifter.org/simon/journal/20061211.htmll`.

Corrado Gini. Variabilità e mutabilità. *Libreria Eredi Virgilio Veschi*, 1912.

Sharad Goel, Winter Mason, and Duncan J Watts. Real and perceived attitude agreement in social networks. *Journal of personality and social psychology*, 99(4) :611, 2010.

David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12) :61–70, 1992.

Quentin Grossetti, Camélia Constantin, Cédric du Mouza, and Nicolas Travers. Community-based recommendations on twitter : Avoiding the filter bubble. In *Under Review*.

Quentin Grossetti, Cédric Du Mouza, and Nicolas Travers. Tweet, retweet et follower : que recommander et à qui ? In *AISR2017*, 2017.

Quentin Grossetti, Camélia Constantin, Cédric du Mouza, and Nicolas Travers. An homophily-based approach for fast post recommendation in microblogging systems. In *International Conference on Extending Database Technology, EDBT*, pages 1–12, Austria, 2018. ACM.

GroupLens. Movielens dataset. `https://grouplens.org/datasets/movielens/`.

Asela Gunawardana and Christopher Meek. A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 117–124. ACM, 2009.

Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10(Dec) :2935–2962, 2009.

Pascal Held, Benjamin Krause, and Rudolf Kruse. Dynamic clustering in social networks using louvain and infomap method. In *Third European Network Intelligence Conference, ENIC*, pages 61–68, Poland, 2016. IEEE Computer Society.

Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

Itai Himelboim, Stephen McCreery, and Marc Smith. Birds of a feather tweet together : Integrating network and content analyses to examine cross-ideology exposure on twitter. *Journal of Computer-Mediated Communication*, 18(2) :154–174, 2013.

Wenyi Huang, Zhaohui Wu, Liang Chen, Prasenjit Mitra, and C Lee Giles. A neural probabilistic model for context based citation recommendation. In *AAAI*, pages 2404–2410, 2015.

Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. *ACM Trans. Inf. Syst.*, 22 (1) :116–142, January 2004.

Jeon hyung Kang and Kristina Lerman. Using Lists to Measure Homophily on Twitter. In *AAAI work. on Intelligent Techniques for Web Personalization and Recommendation*, 2012.

IBM. 10 key marketing trends for 2017 and ideas for exceeding customer expectations. `https://public.dhe.ibm.com/common/ssi/ecm/wr/en/wrl12345usen/watson-customer-engagement-watson-marketing-wr-other-papers-and-reports-wrl12345usen-1pdf`, 2017.

Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems*, pages 47–51. ACM, 2010.

Rong Jin, Joyce Y Chai, and Luo Si. An automatic weighting scheme for collaborative filtering. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 337–344. ACM, 2004.

Jussi Karlgren. An algebra for recommendations : using reader data as a basis for measuring document proximity, 1990.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

Nicolas Koumchatzky and Anton Andryeyev. Using deep learning at scale in twitter's timelines. `https://blog.twitter.com/engineering/en_us/topics/insights/2017/using-deep-learning-at-scale-in-twitters-timelines.html`.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media ? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

George Lekakos and Petros Caravelas. A hybrid approach for movie recommendation. *Multimedia tools and applications*, 36(1-2) :55–70, 2008.

Ronny Lempel and Shlomo Moran. The stochastic approach for link-structure analysis (salsa) and the tkc effect1. *Computer Networks*, 33(1-6) :387–401, 2000.

Kristina Lerman, Rumi Ghosh, and Tawan Surachawala. Social Contagion : An Empirical Study of Information Spread on Digg and Twitter Follower Graphs. *CoRR*, abs/1202.3162, 2012.

Henry Lieberman et al. Letizia : An agent that assists web browsing. *IJCAI (1)*, 1995 : 924–929, 1995.

Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations : Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1) :76–80, 2003.

Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec : social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM, 2008.

Hao Ma, Irwin King, and Michael R Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210. ACM, 2009.

Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

Ian Mackenzie. How retailers can keep up with consumers. `https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers`, 2013.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics : system demonstrations*, pages 55–60, 2014.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather : Homophily in social networks. *Annual review of sociology*, 27(1) :415–444, 2001.

Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *In Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002*, 2002.

Tim Miranda, Mark Claypool, Anuja Gokhale, Tim Mir, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*. Citeseer, 1999.

Dunja Mladenic. Text-learning and related intelligent agents : a survey. *IEEE intelligent systems and their applications*, 14(4) :44–54, 1999.

Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

Tien T. Nguyen, Pik-Mai Hui, F. Maxwell Harper, Loren G. Terveen, and Joseph A. Konstan. Exploring the filter bubble : the effect of using recommender systems on content diversity. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 677–686, Seoul, Korea, 2014. ACM.

Kyo-Joong Oh, Won-Jo Lee, Chae-Gyun Lim, and Ho-Jin Choi. Personalized news recommendation using classified keywords to capture user preference. In *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, pages 1283–1287. IEEE, 2014.

Eli Pariser. Beware online "filter bubbles", 2011. `https://www.ted.com/talks/eli_pariser_beware_online_filter_bubbles`,.

Bidyut Kr Patra, Raimo Launonen, Ville Ollikainen, and Sukumar Nandi. A new similarity measure using bhattacharyya coefficient for collaborative filtering in sparse data. *Knowledge-Based Systems*, 82 :163–177, 2015.

Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, 13(5-6) :393–408, 1999.

Libby Plummer. This is how netflix's top-secret recommendation system works. `https://www.wired.co.uk/article/how-do-netflixs-algorithms-work-machine-learning-helps-to-predict-what-viewers-will-like`, 2017.

Walter Quattrociocchi, Antonio Scala, and Cass R Sunstein. Echo chambers on facebook. 2016.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76 :36–47, 2007. doi : 10.1103/PhysRevE.76.036106.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr : Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens : an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

Sebastian Schnettler. A structured overview of 50 years of small-world research. *Social Networks*, 31(3) :165 – 178, 2009.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec : Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM, 2015.

Aneesh Sharma, Jerry Jiang, Praveen Bommannavar, Brian Larson, and Jimmy Lin. GraphJet : Real-time Content Recommendations at Twitter. *Proc. VLDB Endow.*, 9 (13) :1281–1292, 2016.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec) :583–617, 2002.

Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major components of the gravity recommendation system. *Acm Sigkdd Explorations Newsletter*, 9(2) :80–83, 2007.

Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. User modeling with neural network for review rating prediction. In *IJCAI*, pages 1340–1346, 2015.

Andreas Töscher, Michael Jahrer, and Robert M Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, pages 1–52, 2009.

Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2015.

Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

WeAreSocial. Digital, social media, mobile et e-commerce en 2018. `https://wearesocial.com/fr/blog/2018/01/global-digital-report-2018`, 2017.

Beth Wellman. The school child's choice of companions. *The Journal of Educational Research*, 14(2) :126–132, 1926.

Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. TwitterRank : Finding Topic-sensitive Influential Twitterers. In *Proc. Intl. Conf. on Web Search and Data Mining (WSDM)*, pages 261–270, 2010.

Bing Xu, Mingmin Zhang, Zhigeng Pan, and Hongwei Yang. Content-based recommendation in e-commerce. In *International Conference on Computational Science and Its Applications*, pages 946–955. Springer, 2005.

Xiwang Yang, Harald Steck, Yang Guo, and Yong Liu. On top-k recommendation using social networks. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 67–74. ACM, 2012a.

Xiwang Yang, Harald Steck, and Yong Liu. Circle-based recommendation in online

social networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1267–1275. ACM, 2012b.

Xiwang Yang, Yang Guo, and Yong Liu. Bayesian-inference-based recommendation in online social networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(4) : 642–651, 2013.

Qian Zhao, Maxwell Harper, Gediminas Adomavicius, and Joseph Konstan. Explicit or Implicit Feedback ? Engagement or Satisfaction ? *SAC 2018*, 2018.

le cnam

Quentin GROSSETTI

Système de recommandation sur les plateformes de micro-blogging et bulles filtrantes

le cnam

Abstract :
With the unprecedented growth of user-generated content produced on microblogging platforms, finding interesting content for a given user has become a major issue. However due to the intrinsic properties of microblogging systems, such as the volumetry, the short lifetime of posts and the sparsity of interactions between users and content, recommender systems cannot rely on traditional methods, such as collaborative filtering matrix factorization. After a thorough study of a large `Twitter` dataset, we present a propagation model which relies on homophily to propose post recommendations. Our approach relies on the construction of a similarity graph based on retweet behaviors on top of the Twitter graph. We then conduct experiments on our real dataset to demonstrate the quality and scalability of our method. Finally, we investigate community detection algorithms and we present a metric to compute the strength of the *filter bubble*. Our results show that *filter bubble* effects are in fact limited for a majority of users. We find that, counter-intuitively, in most cases recommender systems tend to open users perspectives. However, for some specific users, the bubble effect is noticeable and we propose a model relying on communities to provide a list of recommendations closer to the user's usage of the platform.

Keywords :
Recommender System, Collaborative Filtering, Microblogging, Twitter, Communities, Filter Bubble

Résumé :
Avec la croissance sans précédent des publications sur les plateformes de micro-blogging, trouver du contenu intéressant pour un utilisateur est devenu un enjeu majeur. Cependant, en raison des propriétés intrinsèques des plateformes de micro–blogging, comme le flux gigantesque de messages arrivant tous les jours et leur faible durée de vie, il est difficile d'appliquer les méthodes traditionnelles de recommandation comme la factorisation matricielle. Après une étude approfondie d'un large jeu de donnée issu de `Twitter`, nous présentons un modèle de propagation qui repose sur l'homophilie présente dans le réseau pour recommander des messages aux utilisateurs. Notre approche s'appuie sur la construction d'un graphe de similarités lié aux interactions des utilisateurs. Nous présentons plusieurs expérimentations pour démontrer la qualité de prédiction de notre modèle et sa capacité à passer à l'échelle. Enfin, nous évaluons différents algorithmes de détections de communautés, qui permettent d'évaluer l'impact des systèmes de recommandations sur l'isolement communautaire des utilisateurs. Nous proposons une métrique permettant de quantifier la force des *bulles filtrantes* et nos résultats montrent que cet effet de *bulle filtrante* est en réalité limité pour une majorité d'utilisateurs. Il semble que, de façon contre intuitive, dans la majorité des cas les systèmes de recommandation ouvrent les perspectives des utilisateurs. Cependant, une minorité de personnes est concerné par l'effet de bulle et nous proposons donc un modèle reposant sur les liens entre communautés pour adapter les recommandations afin d'être plus en accord avec leur profil communautaire.

Mots clés :
Recommandation, Filtrage Collaboratif, Micro-blogging, Twitter, Communautés, Bulles Filtrantes