Data management in social networks

Silviu Maniu

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2012-ENST-053

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique »

présentée et soutenue publiquement par

Silviu MANIU

le 28 Septembre 2012

Gestion des données dans les reseaux sociaux

Directeur de thèse : **Bogdan CAUTIS**
Co-encadrement de la thèse : **Talel ABDESSALEM**

Jury

Mme Anne DOUCET, Professeur, LIP6, Université Paris 6	Examinateur
Mme Ioana MANOLESCU, Directeur de Recherche DR2, INRIA	Examinateur
M. David GROSS-AMBLARD, Professeur, IRISA, Université Rennes 1	Examinateur
Mme Tova MILO, Professeur, Université de Tel Aviv	Rapporteur
M. Michael BENEDIKT, Professeur, Université de Oxford	Rapporteur

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

THÈSE

Abstract

We address in this thesis some of the issues raised by the emergence of social applications on the Web, focusing on two important directions: efficient social search in online applications and the inference of signed social links from interactions between users in collaborative Web applications.

We start by considering social search in tagging (or bookmarking) applications. This problem requires a significant departure from existing, socially agnostic techniques. In a network-aware context, one can (and should) exploit the social links, which can indicate how users relate to the seeker and how much weight their tagging actions should have in the result build-up. We propose an algorithm that has the potential to scale to current applications, and validate it via extensive experiments.

As social search applications can be thought of as part of a wider class of context-aware applications, we consider context-aware query optimization based on views, focusing on two important sub-problems. First, handling the possible differences in context between the various views and an input query leads to view results having uncertain scores, i.e., score ranges valid for the new context. As a consequence, current top-k algorithms are no longer directly applicable and need to be adapted to handle such uncertainty in object scores. Second, adapted view selection techniques are needed, which can leverage both the descriptions of queries and statistics over their results.

Finally, we present an approach for inferring a signed network (a "web of trust") from user-generated content in Wikipedia. We investigate mechanisms by which relationships between Wikipedia contributors - in the form of signed directed links - can be inferred based their interactions. Our study sheds light into principles underlying a signed network that is captured by social interaction. We investigate whether this network over Wikipedia contributors represents indeed a plausible configuration of link signs, by studying its global and local network properties, and at an application level, by assessing its impact in the classification of Wikipedia articles.

Keywords

Social applications, collaborative applications, social search, threshold algorithms, context-aware search, query processing, cached results, views, signed networks, Wikipedia, link inference/prediction, Web-scale algorithms.

Resumé

Nous abordons dans cette thèse quelques-unes des questions soulevées par l'émergence d'applications sociales sur le Web, en se concentrant sur deux axes importants: l'efficacité de recherche sociale dans les applications Web et l'inférence de liens sociaux signés à partir des interactions entre les utilisateurs dans les applications Web collaboratives.

Nous commençons par examiner la recherche sociale dans les applications de "tagging". Ce problème nécessite une adaptation importante des techniques existantes, qui n'utilisent pas des informations sociaux. Dans un contexte ou le réseau est importante, on peut (et on devrait) d'exploiter les liens sociaux, ce qui peut indiquer la façon dont les utilisateurs se rapportent au demandeur et combien de poids leurs actions de "tagging" devrait avoir dans le résultat. Nous proposons un algorithme qui a le potentiel d'évoluer avec la taille des applications actuelles, et on le valide par des expériences approfondies.

Comme les applications de recherche sociale peut être considérée comme faisant partie d'une catégorie plus large des applications sensibles au contexte, nous etudions le probleme de repondre au requetes a partir des vues, en se concentrant sur deux sous-problèmes importants. En premier, la manipulation des éventuelles différences de contexte entre les différents points de vue et une requête d'entrée conduit à des résultats avec des score incertains, valables pour le nouveau contexte. En conséquence, les algorithmes top-k actuels ne sont plus directement applicables et doivent être adaptées aux telle incertitudes dans les scores des objets. Deuxièmement, les techniques adaptées de sélection de vue sont nécessaires, qui peuvent s'appuyer sur les descriptions des requêtes et des statistiques sur leurs résultats.

Enfin, nous présentons une approche pour déduire un réseau signé (un "réseau de confiance") à partir de contenu généré dans Wikipedia. Nous étudions les mécanismes pour deduire des relations entre les contributeurs Wikipédia - sous forme de liens dirigés signés - en fonction de leurs interactions. Notre étude met en lumière un réseau qui est capturée par l'interaction sociale. Nous examinons si ce réseau entre contributeurs Wikipedia représente en effet une configuration plausible des liens signes, par l'étude de ses propriétés globaux et locaux du reseau, et en évaluant son impact sur le classement des articles de Wikipedia.

Mots clés

Applications sociales, applications collaboratifs, receherche sociale, algorithmes de seuil, vues, reseaux signées, Wikipedia, algorithmes de grande echelle.

Resumé en Français

La possibilité d'interroger et d'analyser la quantité sans précédent de données présentes sur le World Wide Web, par des algorithmes rapides et efficaces, a largement contribué à la croissance rapide de l'Internet, ce qui en fait tout simplement irremplaçable dans notre vie quotidienne. Un développement récent sur le Web est représentée par le Web social, c'est à dire, les applications qui sont centrées autour des utilisateurs, leurs relations et leurs données.

En effet, les grandes communautés en ligne ou les utilisateurs contribuent et partagent leur contenu, font aujourd'hui une partie importante et très qualitative du Web. Elles peuvent être construites comme explicitement sociales (Facebook, Google+, Flickr ou Twitter), ou comme des applications dans lesquelles les communautés sociales représentent le "moteur" de la création de contenu (Wikipedia). Pour ces deux types d'applications Web, la présence des interactions entre utilisateurs est un élément crucial de leur succès. Pour ne citer que deux chiffres éloquents, dans les 3 dernières années, le nombre total des utilisateurs sur Facebook est passé d'environ 400 millions à près d'un milliard, tandis que le nombre de tweets par jour sur Twitter a augmenté de 40 millions à 340 millions. Avec des bases d'utilisateurs plus larges, plus de participation et de plus grands volumes de données vient la nécessité de garanties fraÃőcheur et la pertinence des données, la nécessité de conduire pour une récupération rapide et une analyse efficace. En effet, les techniques de recherche et de la croissance du Web sont clairement inter-dépendants.

Malgré leur promesse, le potentiel des outils fournis par le Web social d'aujourd'hui n'est pas encore pleinement réalisé. Systèmes de recherche dans lequel les relations entre les utilisateurs d'un site Web sont pris en compte sont encore relativement rares, et ils utilisent principalement des simples approches de filtrage afin de servir le contenu. La recherche sur les mesures de réputation dans les applications sociales qui permettent des sémantiques entre liens qui sont plus riches, au-delà de modeles liaison / non-liaison, comme la similitude et antagonisme dans les réseaux signés, est encore dans une phase très embryonnaire. En outre, le manque d'outils contributeurs, tels que les classificateurs capables de différencier les contributeurs valables de ceux sans valeur (par exemple, les vandales) est l'une des raisons possibles derrière le déclin lent de

l'activité contributeur sur la Wikipédia.

Cela indique que, pour réaliser le potentiel du Web social et collaboratif, il ya un réel besoin de systèmes de gestion des données sociales, et en particulier pour des techniques capables de: (i) déduire des liens utilisateur, éventuellement avec des semantiques plus riches, même dans des applications où ces liens n'existent pas, (ii) classifier les utilisateurs ou les collaborateurs du réseau social, pour une utilisation dans les systèmes de recommandation, dans des meilleurs modèles de classement de contenu, et (iii) classer et servir efficacement le contenu, en tenant en compte le fait que le classement des contenus ne dépend pas seulement des proprietes des données – comme c'est le cas dans les systemes IR – mais aussi sur la relation entre l'utilisateur demandant des données et les propriétaires ou producteurs de contenu pertinent.

Dans cette thèse, nous nous concentrons principalement sur deux des trois aspects mentionnés ci-dessus: la déduction des liens sociaux implicits et le problème de la recherche efficace dans les applications sociales, avec des techniques qui représentent aussi les premiers pas vers le classement pertinent des donnees, ou vers des systemes de recommandation precises.

Donc, nous abordons trois problèmes importants dans le domaine de la gestion des données sociales:

1. Nous étudions le problème de la recherche sociale, dans son interprétation centre reseau. Les scénarios de recherche dans lequel le réseau est une partie intégrante du modèle de classement contenu exigent des algorithmes qui vont au-delà de simples adaptations des algorithmes top-k courants. La sémantique des requêtes change de maniere fondamentale: le modèle de notation doit prendre en compte non seulement les mots clés de la requête, mais aussi l'identite de l'utilisateur qui a initie la requête. Ce fait nécessité d'algorithmes adaptés, capables de calculer efficacement des réponses à ces requêtes. Plus important encore, ils doivent être robustes en présence des mises à jour dans le réseau d'utilisateurs et de leurs données.

2. Deuxièmement, comme une extension naturelle du problème ci-dessus, nous étudions le scénario où les résultats des recherches peuvent être mis en cache, une fonctionnalité importante pour tous les types d'applications de recherche. Nous étudions les requetes top-k en présence de résultats requête précalculées (que nous appelons vues), dans le cadre plus vaste des systemes sensibles au contexte, couvrant non seulement la recherche sociale, mais aussi la recherche spatiale. Contrairement aux algorithmes top-k classiques qui utilisent les vues, dans notre contexte les données

dans les vues ne peuvent pas être utilisés tel que. Le contexte dans lequel les vues ont été calculées doitêtre transposé dans le contexte de la requête actuelle. Cela introduit une incertitude dans les scores des données dans les vues, et conduit a des nouveaux semantiques de requête et des algorithmes.

3. Troisièmement, nous étudions le problème de l'inférence des liens sociaux a partir des interactions, en allant au-delà de prédiction lien simple. Nous étudions les reseaux signes dans lesquels les liens peuvent avoir non seulement une une interpretation positive de l'attitude inter-utilisateurs, mais aussi un cÃt'té négatif. Plus précisément, nous montrons qu'il existe un reseau signe implicite des contributeurs de Wikipedia, et nous decrivons une approche de la déduire. La sémantique plus riche des liens dans ce réseau signé, que nous appelons WikiSigned, peut aider à des tÃćches comme la classification des contributeurs, des pages ou meme du contenu. Comme une preuve de concept, nous montrons que le classement des articles de Wikipedia peut être considérablement améliorée par la prise en compte du réseau signé des contributeurs.

Recherche sociale

Une classe importante d'applications sociales sont les applications de marquage collaboratif, aussi connus comme des applications de bookmarking,avec des exemples populaires, y compris Delicious, StumbleUpon ou Flickr. Leur fonctionnement général est le suivant:

- former les utilisateurs dans un réseau social, ce qui reflète peut-être la proximité, similitude, l'amitié, la proximité, etc,
- des éléments d'une archive publique d'articles (par exemple: un document, des URL, des photos, etc) sont marqués par les utilisateurs avec des mots clés, à des fins telles que la description et la classification, ou pour faciliter la récupération ultérieure,
- des utilisateurs recherchent des articles ayant certains mots clés (tags) ou qu'ils sont des recommandations, basées sur la proximité dans le reseau.

Les applications de marquage collaboratif et les applications sociales en général - peuvent offrir une perspective entièrement nouvelle à la façon dont on recherche et retrouvent les informations. La raison principale de ceci est que les utilisateurs peuvent (et le font souvent) jouer un rÃt'le aux deux extrémités de la

circulation de l'information, en tant que producteurs et aussi en tant que demandeurs d'information. Par conséquent, trouver le articles les plus pertinents qui sont marqués avec quelques mots clés doit être fait d'une sensible au reseau. En particulier, les éléments qui sont marqués par les utilisateurs qui sont "plus proches" de l'origine devraient avoir plus de poids que les articles qui sont marqués par plusieurs utilisateurs distants.

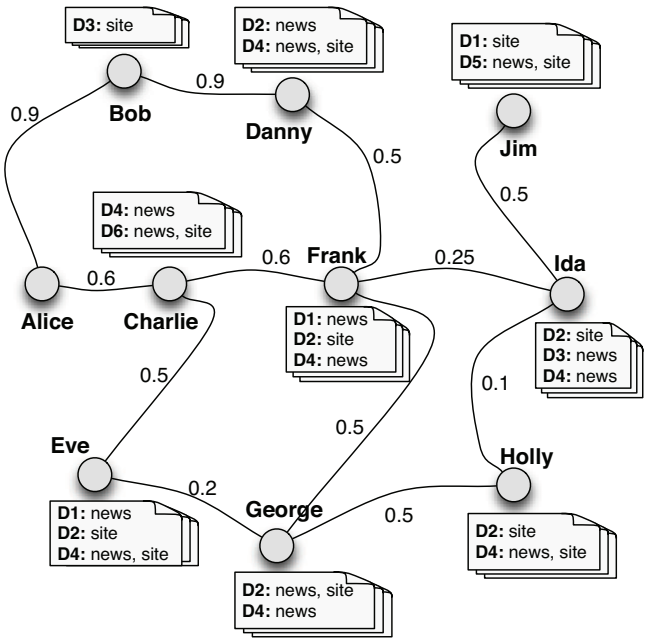
Dans ce travail, nous etudions le problème de la récupération top-k dans les systèmes de marquage collaboratif. Bien que l'accent sur les applications de bookmarking peut paraÃőtre restrictif, ceux-ci représentent une bonne abstraction pour d'autres types d'applications sociales, à laquelle nos techniques peuvent etre appliques directement.

Nous étudions ce problème en mettant l'accent sur l'efficacité, les techniques de ciblage qui ont le potentiel de passage a l'échelle pour les applications actuels sur le Web, dans un contexte, les données de marquage et même les préférences de recherche des demandeurs peuvent changer à tout moment. Dans ce contexte, un des principaux sous-problème pour requetes top-k que nous devons considerer est le calcul de scores candidats pour le top-k par considerer non seulement les éléments les plus pertinents à l'égard de la requête, mais aussi (ou surtout) en consultant les utilisateurs les plus proches et de leurs articles marquees.

Nous associer à la notion de réseau social une interprétation assez générale, sous forme de utilisateurs dont les liens sont marqués par des scores sociaux,qui donnent une mesure de la proximité ou de la similitude entre deux utilisateurs. Ceux-ci sont ensuite exploitables dans les recherches, car ils disent à quel point les actions marquage doivent etres relevants. Par exemple, même pour les applications de marquage où si un réseau social explicite n'existe pas ou n'est pas exploitable, on peut utiliser l'histoire de marquage pour construire un réseau de similarité.

Example 1. *Dans la reuseau de la figure, les utilisateurs ont associées des documents et ils sont reliés entre eux par des liens sociaux. Chaque lien est marquée par son score social,dans l'intervalle* $[0, 1]$. *Prenons l'utilisateur Alice dans le rÃt'le du chercheur. Le graphe n'est pas complète, comme le montre la figure, et seuls deux utilisateurs ont un score sociale explicite en ce qui concerne Alice. Pour ceux qui restent : Danny, . . . , Jim, seulement un score social implicite pourrait être calculée à partir des liens existants si une mesure précise de leur pertinence par rapport a la requete d'Alice est donnee.*

Supposons qu'Alice cherche les deux premiers documents qui sont marqués avec "news" et "site". En regardant les voisins immédiats d'Alice et de leurs documents



respectifs, intuitivement, D3 doit avoir un score plus élevé que D4,puisque le premier est marqué par une utilisateur plus pertinent (Bob, ayant le score maximal sociale par rapport à Alice). Si nous étendrons la recherche à l'ensemble du graphe, le score de D4 peuvent toutefois bénéficier du fait que les autres, comme Holly, ont également marqué des documents avec "news" ou "site". En outre, des documents tels que D2 et D1 peut également être pertinent pour le meilleur résultat top-2, même si elles ont été marqués uniquement par les utilisateurs qui sont indirectement liés à Alice.

Sous certaines hypothèses a être clarifiés prochainement, le top-2 pour la requête d'Alice sera D4 et D2. Nous allons détailler le modèle sous-jacent et les algorithmes qui nous permettent de construire cette réponse dans ce qui suit.

L'algorithme NRA de Fagin s'appuye sur des listes inversees précalculées avec des scores exacts de chaque terme de la requête (dans notre contexte, un terme est une étiquette). Fans la Figure 1.1, nous avons deux listes inversés $IL(news) = \{D4 : 7, D2 : 2, D1 : 2, D3 : 1, D6 : 1, D5 : 1\}$ et $IL(site) = \{D2 : 5, D4 : 2, D3 : 1, D6 : 1, D1 : 1, D5 : 1\}$, qui donne le nombre de fois qu'un document a été marque.

Lorsque la proximité utilisateur est un élément supplémentaire dans le top-k, un adaptation de l'algorithme de seuil et ses variantes auraient besoin de listes inverses précalculées pour chaque paire (utilisateur, mot-clé). Par exemple, si nous interprétons les liens explicites dans le graphe comme l'amitié, ignorant les scores lien et consideront le marquage uniquement par des amis directs les

listes sont $IL_{Alice}(news) = \{D4 : 1, D6 : 1\}$ et $IL_{Alice}(site) = \{D3 : 1, D6 : 1\}$. 18 autres listes seraient exigees et, évidemment, cela aurait un espace prohibitif et des coûts d'execution trop grands dans un contexte reel. Amer-Yahia et al. est le premier à répondre à cette question mais dans un maniere simplifiée. Les auteurs considèrent une extension des algorithmes top-k classiques dans laquelle la proximite utilisateur est considérée comme une fonction binaire (0-1 proximité): seul un sous-ensemble des utilisateurs du réseau sont sélectionnés et peuvent influencer let top-k. Cela introduit deux fortes restrictions et simplifications: (i) seuls les documents etiquetes par les utilisateurs sélectionnés doivent être pertinents pour la recherche, et (ii) tous les utilisateurs ainsi sélectionnés sont egalement importants. La solution de base de est à conserver pour chaque paire de utilisateur-etiquette, au lieu des listes détaillées, seulement une valeur limite supérieure sur le nombre de tagueurs. Par exemple, la limite supérieure de (news, D4) serait de 2, puisque pour n'importe quel utilisateur il ya au plus deux voisins qui taggént D4 avec news. C'est ce qu'on appelle le algorithme GLOBAL-UPPERBOUND. Une version plus raffinée, qui echange l'espace pour l'efficacité, garde de la limite supérieure des valeurs au sein des communites d'utilisateurs, au lieu du réseau dans son ensemble.

Seulement dans Schenkel et al., le problème de récupération sensible au réseau pour le marquage collaboratif est considérée selon une interprétation générale, celle que nous avons également adopte dans ce travail. Il estime que même les utilisateurs qui sont indirectement liés à l'origine peut être pertinents pour le top-k Leur algorithme ContextMerge suit l'intuition que les utilisateurs les plus proches du demandeur ont plus d'influence dans le score d'un élément, et donc ils maximizent la chance que l'article restera dans le top-k final.Les auteurs décrivent une approche hybride dans laquelle, à chaque étape, l'algorithme choisit soit de consulter les documents marqués par le plus proche utilisateur ou sur les listes inversees pour chaque etiquette. Afin d'obtenir le prochain utilisateur l'algorithme précalcule à l'avance la valeur de proximité pour toutes les paires possibles d'internautes. Ces valeurs sont ensuite stockées dans des listes classées (une liste par l'utilisateur), et un incrément pointeur permet d'obtenir le prochain utilisateur concerné.

Example 2. *Considérons le réseau de la figure. 1,1. En ce qui concerne le demandeur Alice, la liste des utilisateurs selon leur indice de proximité serait* $\{Bob : 0,9, Danny : 0,81, Charlie : 0,6, Frank : 0,4, Eve : 0,3, George : 0,2, Holly : 0,1, Ida : 0,1, Jim : 0,05\}$*,avec la proximité entre deux utilisateurs construits comme le produit maximal de liens sur un chemin entre eux (formalisée dans la section 2.2.1).*

Les principaux inconvénients de sont la scalabilite et l'applicabilité. De toute évidence, le précalcul d'une fermeture transitive pondérée sur l'ensemble du réseau a un coût élevé en termes d'espace et de calcul en même modérée de taille des réseaux sociaux. Plus important encore, le maintien de ces proximité listes à jour quand ils reflètent la similitude de marquage, serait tout simplement impossible dans le monde réel, très dynamique. (Nous revisitons ces considérations en section 2.6.)

Contributions Nous proposons un algorithme top-k dans le application marquage collaboratif, qui a le potentiel de passer a l'échelle pour les applications actuelles et au-delà, dans un contexte où les changements du réseau et des actions marquage sont fréquents. Pour cet algorithme, nous avons aborde un aspect essentiel: l'accès efficace les utilisateurs les plus proches pour un chercheur donné. Nous décrivons comment cela peut être fait à la volée - (sans des calculs préalables) pour une grande famille de fonctions pour le calcul de proximité dans un réseau social, y compris les plus naturel. L'intérêt de le faire est triple:

- nous pouvons avoir une personnalisation complete de notation, où chaque utilisateur peut definir sa propre voie d'exécuter des requêtes, grÃćce à des paramètres et des choix de fonctions de score,
- nous pouvons itérer sur les utilisateurs concernés de manière plus efficace, car un réseau typique peut facilement entrer dans la memoire principale ce qui peut épargner les volumes de disque potentiellement énormes exigés par l'algorithmed (voir section 2.6), tout en ayant la potentiel de s'executer plus rapidement,
- les mises de liens social ne sont plus un problème, par exemple, si elle est fondée sur la similitude des actions de marquage.

Nos algorithmes sont correctes et complets. Nous montrons que, lorsque la recherche se fonde exclusivement sur le poids social de ces données, il est optimale dâĂŹinstance pour une grande classe des algorithmes. Des expériences approfondies sur des données réelles montrent que notre algorithme s'execute beaucoup mieux que les techniques existantes, avec jusqu'à 50% d'amélioration (voir section 2.7).

Pour plus d'efficacité encore, nous considérons alors des statistiques pour des résultats approximatifs. Nos approches présentent les avantages de la consommation de mémoire négligeable (ils s'appuient sur des statistiques concises

sur le réseau de l'utilisateur) et les frais généraux de calcul réduite. En plus, ces statistiques peuvent être tenus à jour avec un effort limité, même quand le réseau social est construit sur la base de l'histoire de marquage. Les expériences montrent que les techniques de recherche approximatives peuvent considérablement améliorer le temps de réponse, pour atteindre environ 25% du temps de fonctionnement de l'approche exacte, sans sacrifier la précision.

L'objectif principal de notre travail est sur les aspects sociaux de la recherche top-k dans les applications de marquage collaboratif et nos techniques sont conçus pour un meilleur reponse dans des scenarios où les actions marquage sont pour la plupart (sinon exclusivement) vue à travers le prisme de la pertinence sociale.

Reseaux signées

Une tendance importante dans les plates-formes sociales vise à exploiter les relations d'utilisateurs déjà existants, les liens entre les utilisateurs (par exemple, les liens sociaux), afin d'améliorer les fonctionnalités de base du système. Ceci est particulièrement le cas lorsque les liens peuvent être considérés comme étant signé,indiquant une attitude positive ou négative; les significations possibles des liens positifs pourrait être la confiance, l'amitié ou de la similitude, tandis que les liens négatifs pourraient representer la méfiance, de l'opposition ou d'antagonisme. Dans les situations où des relations explicites n'existent pas, sont rares ou sont des indicateurs inadéquats des attitudes envers les autres membres de la communauté, il devient donc important de découvrir connexions implicites, des liens positifs ou négatifs, a partir des activités d'utilisateurs concernés et de leurs interactions.

Contributions Ce travail représente une étude sur les modes d'interaction entre les contributeurs de Wikipedia et des relations qui peuvent être déduites de leur activite.

Nous avons extrait le réseau signée par l'historique de révision totale de la Wikipédia en anglais, et nous présentons aussi une étude sur une plus petite échelle - une collection de 563 articles du domaine politique. A partir de l'historique des révisions, nous étudions les mécanismes par lesquels les relations entre les contributeurs - sous forme de liens dirigés signés - peuvent être déduites de leurs interactions. Nous prenons en compte les modifications plus souvent par des auteurs des articles, des activités telles que la voix pour les postes d'administrateur, la restauration d'un article à une version antérieure,

ou l'attribution d'un barnstar (un prix, tout en reconnaissant les contributions précieuses).

Vu que Wikipédia en anglais contient environ 5 million d'articles et plus de 260 millions de révisions, les algorithmes d'extraction décrits dans notre travail doivent être adapté à un environnement distribué, comme le paradigme MapReduce et son implementation open-source, Hadoop . En outre, les primitives importantes telles que l'enumeration de triangles dans le graphe et algorithmes d'apprentissage machine ont besoin des algorithmes adaptés.

Le réseau signé que nous construisons est basé sur un modèle local pour les relations utilisateur: une paire ordonnée entre des membres de la communauté en ligne - entre le générateur et le destinataire - assigne une valeur positive ou négative, quand cette valeur peut être déduite. Cela pourrait être interprété comme la confiance subjective / méfiance dans la capacité d'un editeur pour améliorer la Wikipedia, et nous appelons l'ensemble de ces valeurs dans le réseau le "web de confiance". En bref, notre approche vise à transformer les interactions en indicateurs d'affinité d'utilisateur ou de compatibilité: pour donner une breve intuition, la suppression de texte ou restauration des modifications d'un autre editeur (retour en arrière dans le thread des versions) serait favorable à une relation négative, tandis que la modification du texte ou la restauration d'une version précédente conduit vers une expérience positive.

Example 3. *Pour illustrer, prenons six editeurs des articles de Wikipedia, ayant interagi sur le texte de l'article de la manière indiquée dans le tableau 1.1. Nous détaillons dans la section 4.2 comment ces données agrégées interaction est construit en vecteurs d'interaction.*

Generateur	Destinataire	Interactions	inserées	effacés	remplancés
Dodo19	Loopy	10	247	0	10
Capt_Jim	Zscout370	5	6	120	0
99.227.60.251	VolkovBot	1	0	1	0

En regardant la première paire de contributeurs $(Dodo19, Loopy)$*, on peut noter que, à travers 10 interactions,* $Dodo19$ *a ajouté 247 mots près du texte rédigé par* $Loopy$*, alors que seulement 10 mots on ete modifiees. Intuitivement, cela laisse entendre que* $Dodo19$ *considere le texte de* $Loopy$ *comme relevant. Par conséquence, un lien dirige positif pourra être crée à de* $Dodo19$ *à* $Loopy$*.*

La deuxième paire de contributeurs, $(Capt\_Jim, Zscout370)$ *illustre la situation inverse. Dans 5 interactions,* $Capt\_Jim$ *a supprimé 120 mots de* $Zscout370$*'s alors quâĂŹil a insere seulement 6 mots. Il peut être raisonnablement dit (voir la section 4.4 pour une évaluation empirique de cet argument) que le premier contributeur se méfie*

en général (ou n'aime pas) le texte de la deuxième contributeur. Dans ce cas, un lien dirige négatif pourrait être créé entre $Capt_Jim$ et Zscout3709.

A cas où aucun lien ne peut être créé avec une bonne confiance est illustré par la troisième paire de contributeurs, $(99.227.60.251, VolkovBot)$*. Dans ce cas, comme une seule interaction composée d'un mot supprimé a eu lieu, il n'y a pas assez de preuves pour décider de l'existence et de la polarité d'un lien.*

Outre les interactions sur le texte Wikipedia, d'autres types d'interactions existent entre les contributeurs, par exemple, des votes ou des prix , et pour une interprétation positive ou négative, ils sont également pris en considération pour étayer ces liens signés.

Notre travail fournit des informations précieuses sur les principes qui sous-tendent signé un réseau qui est capturée par les interactions sociales. Nous examinons si le réseau sur Wikipedia contributeurs, appelé ci-après WikiSigned, représente en effet une configuration plausible des signes des lien. Tout d'abord, nous évaluons les connexions aux théories sociales telles que l'équilibre structurel et de l'état, qui ont été testés dans les mêmes communautés en ligne . Deuxièmement, nous évaluons le WikiSigned la précision d'une méthode d'apprentissage pour la prédiction des liens signes. Cela revient à exploiter les liens existants - en particulier les triades lien - pour déduire de nouveaux liens et pourrait être considéré comme la propagation des relations signés. En utilisant des techniques d'apprentissage automatique qui ont été appliquées dans la littérature précédente , dans des reseaux signes explicites, comme:

- Slashdot, dans lequel les utilisateurs peuvent étiqueter les autres comme des amis ou des ennemis,
- Epinions, dans lequel les utilisateurs peuvent indiquer si la confiance ou la méfiance dans dâĂŹautres utilisateurs,
- élection adminship Wikipedia, où les cotisants peuvent soutenir ou s'opposer à l'élection dâĂŹun autre contributeur à des postes de responsabilité plus élevés dans Wikipedia,

on obtient une bonne précision sur le réseau WikiSigned (meilleur que celle obtenu avant dans une reseau election Wikipedia). Par cross-validations, nous obtenons des preuves solides que notre réseau révèle vraiment une configuration signe implicite et que ces réseaux ont des caractéristiques similaires au niveau local, même si WikiSigned est inférée à partir des interactions tandis que les autres réseaux sont explicitement déclarées.

Il existe de nombreuses possibilités qui présentent à nous pour exploiter un tel réseau au niveau de l'application, par exemple, dans les tÃćches de gestion des contributeurs. Nous discutons une application qui sâĂŹappuye également sur les lecteurs, specifiquement le classement des articles de Wikipedia par ordre d'importance et de qualité. L'intuition est que ces caractéristiques dâĂŹun article dépendra de la façon contributeurs se rapportent les uns aux autres.

La contribution de base de ce travail est une thèse: les interactions des utilisateurs des applications sociales en ligne peuvent fournir de bons indicateurs de relations implicites - dans un sens plus riche que des simples relations génériques - et devraient être exploitées comme telles.

Contents

1. Introduction

The ability to query and analyze the unprecedented amount of data present on the World Wide Web, by fast and effective algorithms, has largely contributed to the rapid growth of the Web, making it simply irreplaceable in our every day life. A recent development to the Web is represented by the *social Web*, i.e., applications that are centered around users, their relationships and their data.

Indeed, large online communities that contribute and share content account nowadays for a significant and highly qualitative portion of the data on the Web. They may be built as explicitly social (Facebook[1], Google+[2], Flickr[3] or Twitter[4]) or as applications in which social communities represent the "engine" creating content (Wikipedia[5]). For both these types of Web applications, the presence of user-to-user interactions is a crucial part of their success, with one quantifiable measure of this being the tremendous extent to which they have expanded lately. To give just two telling figures, in the last 3 years, the total number of users on Facebook has grown from around 400 million to almost a billion, while the number of tweets per day in Twitter has grown from 40 million to 340 million [76]. With larger user bases, more participation and larger volumes of data comes the need for freshness guarantees and data relevance, driving the need for fast retrieval and effective analysis. Indeed, search techniques and the growth of the Web are undoubtly inter-dependent, as few would publish data on the Web if it could not be found in relevant searches.

For all their promise, the potential of tools provided by today's socially-enabled Web applications is not yet fully realized. Search systems in which the relationships between the users of a website are taken into account are still relatively rare, and they mainly use simple filtering approaches in order to serve content [33]. Research on reputation measures in social applications that allow for richer link semantics, beyond link/no-link models, like similarity and antagonism in signed networks [75, 60], is still in a very incipient phase. Moreover, the lack of contributor tools, such as classifiers able to differentiate worthwhile contributors from the worthless ones (e.g., vandals) is one of the possible reasons behind the slow decline in contributor activity on the

[1] `http://www.facebook.com`
[2] `http://www.google.com/plus`
[3] `http://www.flickr.com`
[4] `http://www.twitter.com/`
[5] `http://www.wikipedia.org`

Wikipedia [73] [6].

This indicates that, in order to fulfill the potential of the social and collaborative Web, there is a real need for social data management systems, and especially for techniques able to: (i) *infer implicit user links*, possibly supporting richer, more informative semantics, even in applications where links do not exist, (ii) effectively *rank and classify the users or contributors* in the social network, for use in better recommendation systems and content ranking models, and (iii) *efficiently and effectively rank and serve content*, accounting for the fact that the ranking of content does not depend only on its data – as is the case in document based IR systems [56] – but also on the relationship between the user requesting data and the owners or producers of relevant content.

In this thesis, we focus mainly on two of the three aspects outlined above: the inference of implicit social links and the problem of efficient search in social applications, with techniques that also represent first steps towards relevance or reputation ranking for recommendation systems.

More precisely, we approach three important problems in the area of social data management:

1. First, we study the problem of *social search*, in its network-aware interpretation. Search scenarios in which the network is an integral part of the scoring model require algorithms that go beyond simple adaptations of the current top-k processing techniques [43, 84]. The semantics of queries changes fundamentally: the scoring model needs to take into account not only the keywords of the query, but also the identity of *the user who initiated the query*. This brings the need for adapted algorithms that are able to efficiently compute answers for such query-seeker pairs. More importantly, they need to be robust in the presence of massive update rates in the network of users and their data.

   The relevant publications for this work are [MCA12] and the corresponding demonstration [MC12].

2. Second, as a natural extension of the above problem, for efficiency and scalability, we study the scenario where results of searches can be *cached*, an important functionality supported by all types of search applications. We investigate top-k query answering in the presence of precomputed query results (that we call *views*), in the larger scope of *context-aware* search systems, covering not only social keyword search but also spatial search. Unlike context-unaware top-k computations using views [23, 49], in context-aware scenarios, the data in the views can not be used as is. Crucially, the context in which the views were computed

[6] The *amount of content* on the Wikipedia continues to increase, but the *number of content creators* is declining.

has to be transposed to the context of the current query. This introduces uncertainty in the views' data, calling for new query semantics and algorithms.

The relevant publication for this work is [MC13].

3. Third, we study the problem of *link inference* from interactions in social applications, going beyond simple link prediction. We study *signed networks* [52] in which links admit not only a *positive* interpretation of the inter-user attitude, but also a *negative* one. More precisely, we show that there exists an *implicit* signed network of contributors in Wikipedia, and describe an approach to infer it, at the scale of the full English Wikipedia. The richer semantics of links in the resulting signed network, that we call *WikiSigned*[7], can help with tasks like classifying nodes (contributors), pages, or more fined grained content in the network. As a proof-of-concept, we show that the classification of articles in Wikipedia can be significantly improved by taking into account the signed network of contributors.

   The relevant publications for this work are [MCA11] and [MAC11].

We continue in the next three sections by giving an overview of the motivations and our contributions to these problems, and we detail them in the subsequent chapters.

1.1. Social-Aware Search

An important class of social applications are the *collaborative tagging applications*, also known as *social bookmarking applications*, with popular examples including Delicious[8], StumbleUpon or Flickr. Their general setting is the following:

- users form a *social network*, which may reflect proximity, similarity, friendship, closeness, etc,
- items from a public pool of items (e.g., document, URLs, photos, etc) are *tagged* by users with keywords, for purposes such as description and classification, or to facilitate later retrieval,
- users *search* for items having certain keywords (i.e., tags) or they are *recommended* items, e.g., based on proximity at the level of tags.

Collaborative tagging – and social applications in general – can offer an entirely new perspective to how one searches and accesses information. The main reason for this is that users can (and often do) play a role at both ends of the information flow, as producers and also as seekers of information. Consequently, finding the most relevant

[7] `http://perso.telecom-paristech.fr/~maniu/wikisigned/`
[8] `http://www.delicious.com`

items that are tagged with some keywords should be done in a *network-aware* manner. In particular, items that are tagged by users who are "closer" to the seeker – where the term closer depends on model assumptions that will be clarified shortly – should be given more weight than items that are tagged by more distant users.

We consider in this work the problem of top-k retrieval in collaborative tagging systems. While the focus on bookmarking applications may seem restrictive, these represent a good abstraction for other types of social applications, to which our techniques could directly apply.

We investigate this problem with a focus on efficiency, targeting techniques that have the potential to scale to current applications on the Web[9], in an online context where the social network, the tagging data and even the seekers' search preferences can change at any moment. In this context, a key sub-problem for top-k retrieval that we need to address is computing scores of top-k candidates by iterating not only through the most relevant items with respect to the query, but also (or mostly) by looking at the closest users and their tagged items.

We associate with the notion of social network a rather general interpretation, as a user graph whose edges are labeled by *social scores*, which give a measure of the proximity or similarity between two users. These are then exploitable in searches, as they say how much weight one's tagging actions should have in the result build-up. For example, even for tagging applications where an explicit social network does not exist or is not exploitable, one may use the tagging history to build a network based on similarity in tagging and items of interest.

Example 1. C*onsider the collaborative tagging configuration of Figure 1.1. Users have associated lists of tagged documents and they are interconnected by social links. Each link is labeled by its (social) score, assumed to be in the* $[0, 1]$ *interval. Let us consider user Alice in the role of the seeker. The user graph is not complete, as the figure shows, and only two users have an explicit social score with respect to Alice. For the remaining ones,* $Danny, \ldots, Jim$*, only an implicit social score could be computed from the existing links if a precise measure of their relevance with respect to Alice's queries is necessary in the top-k retrieval.*

Let us assume that Alice looks for the top two documents that are tagged with both news and site. Looking at Alice's immediate neighbors and their respective documents, intuitively, $D3$ *should have a higher score than* $D4$*, since the former is tagged by a more relevant user (Bob, having the maximal social score relative to Alice). If we expand the search to the entire graph, the score of* $D4$ *may however benefit from the fact that other users, such as Eve or even Holly, also tagged it with news or site. Furthermore, documents such as* $D2$ *and* $D1$ *may also be relevant for the top-*2 *result, even though they were tagged only by users who are indirectly linked to Alice.*

[9]The most popular ones have user bases of the order of millions and huge repositories of data; today's most accessed social Web application, which also provides tagging and searching functionalities, has almost reached a billion registered users.

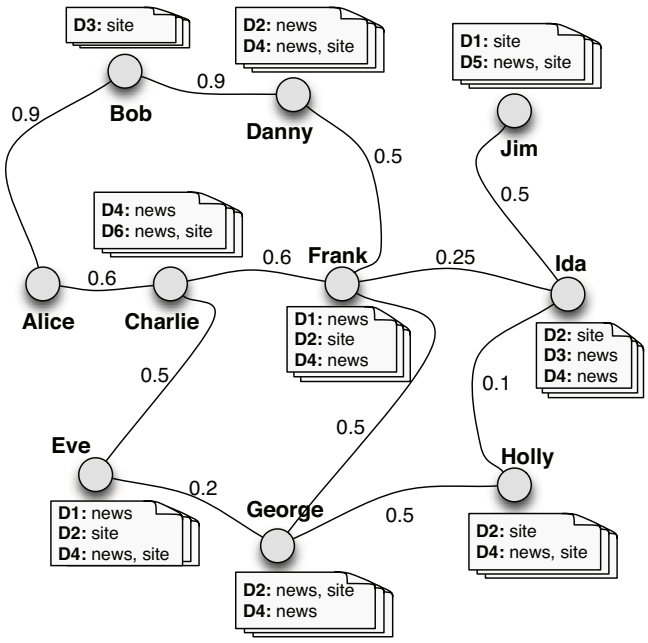


Figure 1.1.: A collaborative tagging scenario and its social network.

Under certain assumptions to be clarified shortly, the top-2 documents for Alice's query will be, in descending score order, D4 and D2. We will detail the underlying model and algorithms that allow us to build this answer in the following.

Relation to the most relevant research Classic top-k retrieval algorithms, such as Fagin's threshold algorithm [31] and the no random access (NRA) algorithm, rely on precomputed inverted-index lists with exact scores for each query term (in our setting, a term is a tag). Revisiting the setting in Figure 1.1, we would have two per-tag inverted lists $IL(news) = \{D4 : 7, D2 : 2, D1 : 2, D3 : 1, D6 : 1, D5 : 1\}$ and $IL(site) = \{D2 : 5, D4 : 2, D3 : 1, D6 : 1, D1 : 1, D5 : 1\}$, which give the number of times a document has been tagged with the given tag.

When user proximity is an additional ingredient in the top-k retrieval process, a direct network-aware adaptation of the threshold algorithm and variants would need precomputed inverted-index lists for each $(user, tag)$ pair. For instance, if we interpret explicit links in the user graph as friendship, ignoring the link scores, and only tagging by direct friends matters, $Alice$'s lists would be $IL_{Alice}(news) = \{D4 : 1, D6 : 1\}$ and $IL_{Alice}(site) = \{D3 : 1, D6 : 1\}$. Other 18 such lists would be require and, clearly, this would have prohibitive space and computing costs in a real-world setting. Amer-Yahia et al. [4] is the first to address this issue, considering the problem of network-aware search in collaborative tagging sites, though by a simplified flavor. The authors consider an extension to classic top-k retrieval in which user proximity is seen as a

binary function (0-1 proximity): only a subset of the users in the network are selected and can influence the top-k result. This introduces two strong simplifying restrictions: (i) only documents tagged by the selected users should be relevant in the search, and (ii) all the users thus selected are equally important. The base solution of [4] is to keep for each tag-item pair, instead of the detailed lists per user-tag pair, only an upper-bound value on the number of taggers. For instance, the upper-bound for $(news, D4)$ would be 2, since for any user there are at most two neighbors who tagged $D4$ with *news*. This is called the GLOBAL UPPER-BOUND strategy. A more refined version, which trades space for efficiency, keeps such upper-bound values within *clusters* of users, instead of the network as a whole.

Only in Schenkel et al. [69], the network-aware retrieval problem for collaborative tagging is considered under a general interpretation, the one we also adopt in this work. It considers that even users who are indirectly connected to the seeker may be relevant for the top-k result. Their CONTEXTMERGE algorithm follows the intuition that the users closest to the seeker will contribute more to the score of an item, thus maximizing the chance that the item will remain in the final top-k. The authors describe a hybrid approach in which, at each step, the algorithm chooses either to look at the documents tagged by the closest unseen user or at the tag-document inverted lists (a seeker agnostic choice). In order to obtain the next (unseen) closest user at any given step, the algorithm *precomputes* in advance the proximity value for all possible pairs of users. These values are then stored in ranked lists (one list per user), and a simple pointer increment allows to obtain the next relevant user.

Example 2. *Consider the network of Fig. 1.1. With respect to seeker Alice, the list of users ranked by proximity would be* $\{Bob : 0.9, Danny : 0.81, Charlie : 0.6, Frank : 0.4, Eve : 0.3, George : 0.2, Holly : 0.1, Ida : 0.1, Jim : 0.05\}$*, with proximity between two users built as the* maximal product of scores over paths linking them *(formalized in Section 2.2.1).*

The main drawbacks of [69] are scalability and applicability. Clearly, precomputing a weighted transitive closure over the entire network has a high cost in terms of space and computation in even moderate-size social networks. More importantly, keeping these proximity lists up to date when they reflect tagging similarity[10] (as advocated in [69]), would simply be unfeasible in real-world settings, which are highly dynamic. (We revisit these considerations in Section 2.6.)

Contributions We propose an algorithm for top-k answering in collaborative tagging, which has the potential to scale to current applications and beyond, in an online context where network changes and tagging actions are frequent. For this algorithm, we

[10] Tagging similarity may indeed be a more pertinent proximity measure than friendship for top-k search in bookmarking applications.

first address a key aspect: accessing efficiently the closest users for a given seeker. We describe how this can be done *on-the-fly* (without any pre-computations) for a large family of functions for proximity computation in a social network, including the most natural ones (and the one assumed in [69]). The interest in doing this is threefold:

- we can support full scoring personalization, where each user issuing queries can define her own way to rank items, through parameters and score function choices – see also Section 2.7 for a detailed description,
- we can iterate over the relevant users in more efficient manner, since a typical network can easily fit in main-memory[11]; this can spare the potentially huge disk volumes required by [69]'s algorithm (see Section 2.6), while also having the potential to run faster.
- social link updates are no longer an issue; for example, if it is based on similarity in tagging actions, we can keep it up-to-date and, by it, all the proximity values at any given moment, with little overhead.

Based on the on-the-fly visit of the relevant network space, our top-k algorithm TOPKS is sound and complete. We show that, when the search relies exclusively on the social weight of the data, it is *instance optimal* within a large and important class of algorithms. Extensive experiments on real world data show that our algorithm performs significantly better than existing techniques, with up to 50% improvement (see Section 2.8).

For further efficiency, we then consider directions for approximate results. Our approaches present the advantages of negligible memory consumption (they rely on concise statistics about the user network) and reduced computation overhead. Moreover, these statistics can be maintained up to date with limited effort, even when the social network is built based on tagging history. Experiments show that approximate search techniques can drastically improve the response time, reaching around 25% of the running time of the exact approach, without sacrificing precision.

The main focus of our work is on the social aspects of top-k retrieval in collaborative tagging applications, and our techniques are designed to perform best in settings where tagging actions are mostly (if not exclusively) viewed through the lens of social relevance.

1.2. Context-Aware Query Processing Using Views

Retrieving the k best data objects for a given query, under a certain scoring model, is one of the most common problems in database systems and on Web. In many applica-

[11] In our view, social networks, excluding the data published by their users, are not "Big Data", since the extent of their growth has obvious limits.

tions, and in particular in current Web search engines, tens of thousands of queries per second need to be answered over massive amounts of data. Significant research effort has been put into addressing the performance of top-k processing, towards optimal algorithms – such as TA and NRA [31, 43] – or highly-efficient data structures [84] (e.g., inverted lists). In recent research, the use of *pre-computed results* (also called *views*) has been identified as a promising avenue for improving efficiency [49, 23].

At the same time, with the advent of location-aware devices, geo-tagging, bookmarking applications, or online social applications in general, as a way to improve the result quality and the user experience, new kinds of top-k search applications are emerging, which can be simply described as *context-aware*. The context of a query may represent the geographic *location* where the query was issued or the *identity* – within a social network – of the user who issued it. Indeed, the setting described in the previous section, i.e. network-aware social search, is an example of a context-aware search application.

More generally, a context could represent certain score parameters that can be defined or personalized at query time. For example, a query for *top-class vegetarian restaurants* should not give the same results if issued in *Paris* or in *Berlin*, as it should not give the same results if issued within a social community of culinary reviewers or within a student community.

Unsurprisingly, taking into account a query context in top-k processing represents a new source of complexity, and many of the common approaches employed in context-agnostic scenarios need to be revisited [22, 69, 55]. In context-aware scenarios, query processing usually entails an exploration of a "neighborhood" space for the closest or most relevant objects, which is often interleaved with some of the classic, context independent top-k processing steps, such as scans over inverted lists.

Consequently, materializing and exploiting in searches the results of previous queries can play an even more important role for efficient, online processing of queries with context. However, in this direction, a broader view-based answering problem than in the context-agnostic setting needs to be addressed, in which the cached results are modeled as unranked lists of objects having only *uncertain scores* or *score ranges*, instead of exact scores. The rationale is that, even when the cached results in views do have exact scores with respect to one context, we should expect these to evolve into score ranges when a *context transposition* is necessary. For example, answers to the previous query, for the *Paris* context, may be useful – but only to a certain extent – when the same query is issued in a nearby *Versailles* context, as one has to adapt the scores of restaurants from the parisian perspective to the versaillaise one; this, inherently, introduces uncertainty.

The potential impact of view-based algorithms that can cope with such uncertainty is highly relevant but not limited to the context-aware setting. Indeed, even when queries are not parameterized by a context, some of the most efficient algorithms, such as NRA or TA_Z [31] can support early-termination and output unranked results

with only score ranges (instead of a precise ranking).

We give next two example scenarios – mostly self-explanatory – from *location-aware* and *social-aware* search. They illustrate, on one hand, the fact that previous (cached) results pertaining to one context may be interpretable only as uncertain, by score intervals, when dealing with a new query and a new context. On the other hand, they illustrate the fact that it may be possible to *corroborate* such uncertain descriptions (of scores of objects) from different views, in order to build a most refined or informative approximation of the top-k result that would be obtained by looking at the actual data instead of the views.

Motivation 1: Location-aware search. Let us consider the spatial-search scenario in Figure 1.2, in which we have objects at various locations in an euclidian space (objects $o1, \ldots, o5$ in the figure, as gray dots). Each object (e.g., a Web document) is characterized by a bag of attributes. For instance, $o5$ has attributes $t1$ and $t2$, both with a single occurrence.

Now, users located at various points request the top-k objects with respect to a set of attributes. In response, objects are ranked by a combination between the distance of the object w.r.t. the seeker's location and the object's content. While the details of the spatial ranking model will be clarified in Section 3.7, let us assume in the following that the location relevance of an object contributes 30% to the score of an object. The remaining 70% represent the weight of the textual score (e.g., tf-idf measures).

Consider a new query Q in the system, asking for the top-2 items for attributes $\{t1, t2\}$ at the point marked by a white dot in the figure. Intuitively, spatial search algorithms [22], by using indices such as the R-tree [36], would proceed by incrementally increasing the search distance until enough objects are found. However, an alternative execution plan may be possible, if we assume access to cached results of previous queries (initiated at the black dots).

For example, let us assume that $v1$ gives the top-3 documents for $\{t1, t2\}$, as the ranked list $\{o5 = 1.062, o4 = 1.029, o2 = 1\}$. Also, sharing the same location, we have $v2$ and $v3$. The former gives the top-4 for $\{t1\}$ as $\{o2 = 0.946, o3 = 0.575, o5 = 0.425, o4 = 0.262\}$. The latter gives the top-4 for $\{t2\}$ as $\{o4 = 0.962, o1 = 0.437, o5 = 0.425, o2 = 0.246\}$.

Since $v1$, $v2$ and $v3$ are closer to Q than any of the objects, it would be tempting to use their lists of pre-computed results, instead of looking for the actual objects.

In particular, one may resort to using only the results of $v1$, as it is the closest to Q both spatially and textually. For that, we need first to perform a change of context, to account for the fact that objects that were close to $v1$ may be even closer to Q, as they may be farther. This will introduce uncertainty in the scores of $v1$'s result set. More precisely, knowing that the normalized distance between Q and $v1$ is 0.175, for Q's perspective, $v1$'s list should now have objects with *score intervals*, as fol-

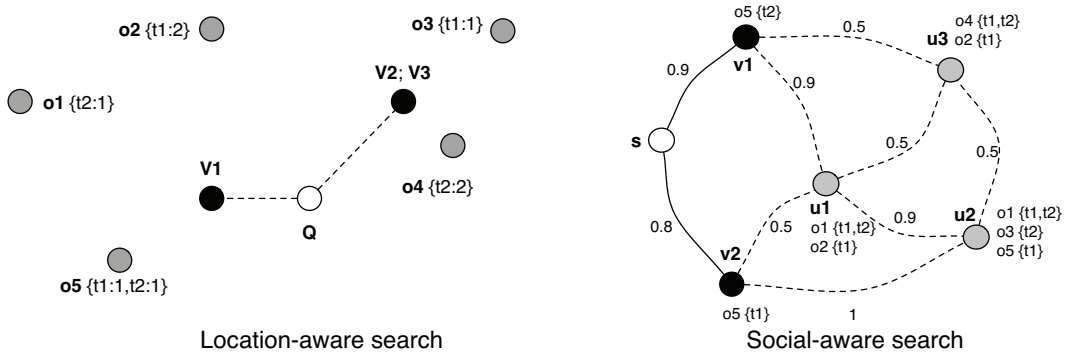


Figure 1.2.: Context-aware search scenarios.

lows: $\{o5 \in [1.062 - 0.3 \times 0.35, 1.062 + 0.3 \times 0.35], o4 \in [1.029 - 0.3 \times 0.35, 1.029 + 0.3 \times 0.35], o2 \in [1 - 0.3 \times 0.35, 1 + 0.3 \times 0.35]\}$ = $\{o5 \in [0.957, 1.167], o4 \in [0.924, 1.134], o2 \in [0.895, 1.105]\}$.[12]

We can see that $v1$'s result is not sufficient to answer Q with certainty, since any object among the three candidates may be in the top-2. Yet the solution can come from $v2$ and $v3$, albeit more distant, if we corroborate their results with the ones of $v1$. Knowing that $v2$ and $v3$ are at a normalized distance of 0.25 the transposed scores would be, for $v2$ $\{o2 \in [0.871, 1], o3 \in [0.5, 0.65], o5 \in [0.35, 0.5], o4 \in [0.187, 0.337]\}$ and for $v3$ $\{o4 \in [0.887, 1.037]$, $o1 \in [0.362, 0.512], o5 \in [0.35, 0.5], o2 \in [0.171, 0.321]\}$.

Aggregating the three result sets, after accounting for the context transposition, would allow us to identify the top-2 objects for Q as $\{o4 \in [1.074, 1.135], o2 \in [1.043, 1.106]\}$, since all other objects have scores of at most 1.043.[13] We can output this result, without having to compute the exact locations and scores of $o4$ and $o2$.

Motivation 2: Social-aware search. As a second motivating example, we consider the setting of collaborative bookmarking applications (such as Flickr or Del.icio.us). In these applications, users bookmark (or tag) objects from a common pool of objects (e.g., Web sites). Users form a social network, in which relationships are weighted (e.g., a similarity or proximity value). Such a setting is illustrated in Figure 1.2. For example, user $u1$ has tagged object $o1$ with t_1 and t_2, and $o2$ with t_1, and it is 0.9-close (or similar) to u_2.

We use the social ranking model introduced by Amer-Yahia et al. [4] and extended by Schenkel et al. [69] and ourselves in [MC12]. Intuitively, under this model, the score of an object for a given tag is proportional to the sum of the proximities of the taggers w.r.t. the seeker. The score of an object for a set of tags is then computed by

[12] 0.35 is obtained as $0.175 + 0.175$, since the query has two tags.
[13] For example, $o4$ has a minimal score obtained as $\max(0.887 + 0.187, 0.924)$.

aggregating the per-tag scores, e.g., by summation.

Let us now assume that the top-3 items for $\{t1, t2\}$ are requested by user s (the seeker). As in location-aware search, early termination algorithms [69] for social search would incrementally explore the most promising users (and their objects) until the top-k is found. This may lead to the visit of a non-negligible fraction of the network. For our query, an exploration of the network would need to go as far as $u2$ to establish a top-3 as $\{o1, o2, o5\}$.

Yet an alternative, more efficient processing approach may rely on pre-computed results. Let us assume that users $v1$ and $v2$ have such data: $v2$ has obtained the top-4 for $\{t1\}$, as $\{o1 = 1.71, o5 = 1.63, o2 = 0.5, o4 = 0.5\}$, and the one for $\{t2\}$ as $\{o1 = 1.71, o5 = 1, o2 = 0.9, o3 = 0.81\}$. $v1$ has obtained the top-5 for $\{t1, t2\}$ as $\{o1 = 3.42, o5 = 2.63, o2 = 1.35, o4 = 1, o3 = 0.81\}$.

Knowing that the distance between s and $v1$ is equal to 0.9, in manner similar to the spatial-aware search scenario, the context transposition from $v1$ to s (the formal ranking model will be described in Section 2.2) leads to the following result set for $\{t1, t2\}$: $\{o1 \in [3.07, 3.8], o5 \in [2.27, 2.81], o2 \in [1.21, 1.5], o4 \in [0.9, 1.11], o3 \in [0.72, 0.9]\}$. Similarly, knowing the distance between $v2$ and s is 0.8, transposing the context leads to the results for $\{t1\}$: $\{o1 \in [1.36, 2.13], o5 \in [1.3, 2.03], o2 \in [0.4, 0.62], o4 \in [0.4, 0.62]\}$. and $\{t2\}$: $\{o1 \in [1.36, 2.13], o5 \in [0.8, 1.25], o2 \in [0.72, 1.12], o3 \in [0.64, 1.01]\}$.

It can hence be seen that, after visiting just the neighbors $v1$ and $v2$, the search for s's query can give the top-3 objects as $\{o1 \in [3.07, 3.8], o5 \in [2.27, 2.81], o2 \in [1.21, 1.5]\}$.

The general goal of this study is to enable efficient context-aware top-k retrieval through techniques that exploit *exclusively* the views. The rationale for this is that in many practical applications, access methods may be extensively optimized for views, the size of cached results may be much less important than the one of the complete data (e.g., of the inverted lists), and view results (pre-computed for groups of attributes) may be much more informative towards finding the result for the input query. For instance, a user may go through a sequence of query reformulations, for which result caching may be highly beneficial. View results may even be bound to main-memory, in certain scenarios.

Contributions We formalize and study in this work the problem of context-aware top-k processing based on possibly uncertain precomputed results, in the form of *views* over the data.

We start by investigating top-k processing *after the context transposition* has been performed, for a given input query and its context. The problem of answering such top-k queries using *only* the information in views, inevitably, requires an adaptation to the fact that these views may now offer objects having uncertain scores. Consequently, there might exist view instances from which an exact top-k cannot be extracted with

full confidence. When this is the case, it would be unsatisfactory to simply refute the input query, or to consider alternative, more expensive execution plans (e.g., by going through the per-attribute lists). Instead, it would be preferable to provide a *most informative answer*, in terms of (i) objects G that are guaranteed to be in the top-k result, and (ii) objects P that may appear in the top-k result.

We formalize this query semantics and describe two adaptations of TA and NRA, called SR-TA and SR-NRA. They support precomputed lists with *score ranges* and the above described query semantics and are *sound* and *complete*, i.e., they output the (G, P)-answer. Intuitively, they implement the corroboration principle illustrated before, based on a *linear programming* formulation.

Given that in many applications the set of views may be very large – think of social applications in which many users may have pre-computed results – we also consider optimizations for SR-TA and SR-NRA, based on *selecting some (few) most promising views*. Obviously, with fewer views, the most informative answer (G, P) may no longer be reached, and we are in general presented a trade-off between the number of selected views – which determines the cost of the top-k algorithms SR-NRA and SR-TA – and the "quality" of the result (a distance with respect to the most informative answer given by all the views). Importantly, we also show that SR-NRA and SR-TA, when selecting views, are *complete* and *instance optimal* for an important family of view specifications. Complementing our top-k retrieval through view selection, we also show how a final refinement step allows us to reach the most informative result.

As a last level of service that can be provided to users, we then consider a sampling-based approach by which, from the most informative result, a probabilistic interpretation can also lead to a *most likely top-k* answer to the input query.

Importantly, our algorithms provide a *one-size-fits-all solution* for many search applications that are context-dependent, and we show how they can be directly applied in our two motivating applications scenarios for context-aware search. For both scenarios, we also describe the necessary step of *context transposition*, transforming scores or ranges thereof, valid in one initial context, to ranges that are valid in a new context, for ranking models combining textual and location dimensions.

Extensive experiments on both synthetic and real-world datasets illustrate the potential of our techniques – enabling high-precision retrieval and important running-time savings. More generally, they illustrate the potential of top-k query optimization based on cached results in a wide range of applications.

1.3. Inferring Signed Social Networks

An important trend in social platforms aims at exploiting the already existing user relationships, links between users (e.g., social links), in order to improve core function-

alities in the system.

This is especially the case when links can be viewed as being *signed*, indicating a *positive* or *negative* attitude; possible meanings for positive links could be trust, friendship or similarity, while negative links could stand for distrust, opposition or antagonism. In settings where explicit relationships do not exist, are sparse or are inadequate indicators of one's attitude towards fellow members of the community, it becomes thus important to uncover *implicit* user inter-connections, positive or negative links, from relevant user activities and their interactions.

Contributions This work represents a study of the interaction patterns between Wikipedia contributors and of the relationships that can be inferred from them.

We extracted the signed network from the entire revision history of the English Wikipedia, and we also present a study on a smaller scale – a collection of 563 articles from the politics domain[14]. Starting from the revision history, we investigate mechanisms by which relationships between contributors - in the form of signed directed links - can be inferred from their interactions. We take into account *edits* over commonly-authored articles, activities such as *votes* for administrator positions, the *restoring* of an article to a previous version, or the assignment of *barnstars* (a prize, acknowledging valuable contributions).

Since the English Wikipedia contains around 5 million articles and over 260 million revisions, the extraction algorithms described in our work need to be adapted to a distributed setting, such as the MapReduce paradigm [24] and its open-source implementation, Hadoop [7]. Moreover, important primitives such as graph triangle counting and machine learning algorithms need algorithms that are adapted to a distributed setting [74, 21].

The signed network we build is based on a local model for user relationships: for a given ordered pair of members of the online community - the link *generator* and the link *recipient* - it will assign a positive or negative value, whenever such a value can be inferred. This could be interpreted as subjective trust / distrust in a contributor's ability to improve the Wikipedia, and we call the set of such values in the network the "web of trust". In short, our approach aims at converting interactions into indicators of user affinity or compatibility: to give a brief intuition, deleting one's text or reverting modifications (backtracking in the version thread) would support a negative link, while surface editing text or restoring a previous version would support a positive one.

Example 3. *To exemplify, let us consider six contributors editing Wikipedia articles, having interacted on article text in the manner shown in Table 1.1. We detail in Section 4.2 how such aggregated interaction data is built (in* interaction vectors*).*

[14] http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Politics

Generator	Recipient	Interactions	Words inserted	deleted	replaced
Dodo19	Loopy	10	247	0	10
Capt_Jim	Zscout370	5	6	120	0
99.227.60.251	VolkovBot	1	0	1	0

Table 1.1.: Interaction vectors for three pairs of Wikipedia contributors.

*Looking at the first pair of contributors Dodo*19 − *Loopy, one can note that, through 10 interactions, Dodo*19 *has added 247 words near the text authored by Loopy, while only modifying 10 words. Intuitively, this hints that Dodo*19 *regards the text of Loopy as valuable. Hence, a* positive *directed link could be created from Dodo*19 *to Loopy.*

The second pair of contributors, Capt_Jim − *Zscout*370 *illustrates the opposite situation. Over 5 interactions, Capt_Jim has deleted 120 words of Zscout*370*'s while only inserting 6 words. It can be reasonably argued (see Section 4.4 for an empirical evaluation of this argument) that the first contributor generally distrusts (or dislikes) the text of the second contributor. In this case, a* negative *directed link could be created from Capt_Jim to Zscout*370*.*

A case in which no link *can be created with good confidence is illustrated by the third pair of contributors,* 99.227.60.251 − *VolkovBot. In this case, as only one interaction consisting of one deleted word took place, there is not enough evidence to decide the existence and polarity of a link.*

Besides interactions on Wikipedia text, other kinds of interactions between contributors, e.g., through votes or prizes, allowing a negative or positive interpretation, are also taken into account to support such signed links.

Our work provides valuable insight into principles underlying a signed network that is captured by social interactions. We look into whether the network over Wikipedia contributors, called hereafter WikiSigned, represents indeed a plausible configuration of link signs. First, we assess connections to social theories such as *structural balance* and *status*, which have been tested in similar online communities [52]. Second, we evaluate on WikiSigned the accuracy of a learning approach for *edge sign prediction*. This amounts to exploiting existing links - in particular *link triads* - to infer new links and could be viewed as *propagation* of signed relationships. Using machine learning techniques that have been applied in previous literature [51] on explicit signed networks, namely

- Slashdot, in which users can tag other users as *friends* or *foes*,
- Epinions, in which users can indicate whether they *trust* or *distrust* other users,
- Wikipedia adminship election, in which contributors *support* or *oppose* or the election of other contributors to positions of higher responsibility in Wikipedia,

we obtain good accuracy over the WikiSigned network (better than the one achieved in [51] over a Wikipedia adminship election network). By cross training-testing we obtain strong evidence that our network does reveal an implicit signed configuration and that these networks have similar characteristics at the local level, even though WikiSigned is inferred from interactions while the other networks are explicitly declared.

There are many opportunities that present to us for exploiting such a network at the application level, e.g., in the management tasks of contributors. We discuss one application that also impacts the readers, namely the classification of Wikipedia articles by importance and quality. The intuition here is that such article features depend on how contributors relate to one another.

A core contribution of this work is a thesis: user interactions in online social applications can provide good indicators of implicit relationships – in a richer sense than simply indicating generic relationships – and should be exploited as such.

2. Efficient Social-Aware Search

In this chapter, we detail our approach to the first problem introduced in Chapter 1, the problem of *efficient social search*. We concentrate on a kind of applications that represent a good abstraction of online social applications in general: the collaborative tagging networks. After an overview of the related work in Section 2.1, we start by formalizing, in Section 2.2, the general setting and the first contribution of this work: the principle behind the on-the-fly computation of proximities between the given seeker and relevant users in the network.

Using the on-the-fly computation of proximities enables the design of efficient, even *instance optimal* algorithms for the exclusively social case, and we present our approach for this case in Section 2.3. The general case is detailed in Section 2.4.

To achieve further efficiency, we then show how using two simple statistics on distances can drastically increase the performance of our algorithms in Section 2.5. An analysis of both memory requirements and time performance, in comparison with the state of the art (the ContextMerge algorithm [69]), in Section 2.6, shows that our algorithm has the potential to be significantly more efficient than approaches that use precomputed distances. We also present the implemented system, Taagle, with the personalized search functionalities it offers, in Section 2.7.

Finally, we validate our approaches via extensive experiments, for both efficiency and effectiveness, in Section 2.8.

2.1. Related Work

Social search The topic of search in a social setting has received increased attention lately. Studies and models of personalization of social tagging sites can be found in [77, 39, 27, 80]. Other studies have found that including social knowledge in scoring models can improve search and recommendation algorithms. In [16], personalization based on a similarity network is shown to outperform other personalization approaches and the non-personalized social search. A study on a last.fm dataset in [48] has found that incorporating social knowledge in a graph model system improves the retrieval recall of music track recommendation algorithms. Another study on a MSN Messenger network [71] has found that people that chat more are more likely to have common interests, and, moreover, people that have a friend in common are also more similar.

The architecture of a question answering search system based on human input was described in [40]. Another architecture for social data management is given in [5, 6], along with a framework for information discovery and presentation in social content sites.

Ranking resources in tagging environments Another approach to rank resources in social tagging environments is CubeLSI [12], which uses a vector space model and extends Latent Semantic Indexing to include taggers in the feature space of resources, in order to better match queries to documents. FolkRank [41] proposes a ranking model in social bookmarking sites, for recommendation and search, based on an adaptation of PageRank over the tripartite graph of users, tags and resources. It follows the intuition that a resource that is tagged with important tags by important users becomes important itself and, symmetrically, for tags and users. An alternative approach to social-aware search, using personalized PageRank, was presented in [10]. There, the same tripartite model of annotators, resources and annotations is used to compute measures of similarities between resources and queries, and to capture the social popularity of resources.

Social search algorithms Using shortest-path based approaches to infer distances between users that are not directly connected has an intuitive reason: if people can "reach" each other faster, they are more likely to be closer in term of data scope and interests. This assumption is first used in Amer-Yahia et al. [4] and the ContextMerge algorithm of [69]. More recently, shortest path distance oracles were used in a distributed setting to answer social top-k queries [9].

Social scoring models A scoring model in which both the content of the documents and the relationship between the searcher and the users contributing to the documents is considered is studied in [34]. The scoring model we use in this chapter is also used in [69, 4] in the same collaborative tagging network setting. It is revisited in [82], where a textual relevance and a social influence score are combined in the overall scoring of items, the latter being computed as the inverse of the shortest path distance between the seeker and the document publishers. Combining context-aware scores with textual scores is not exclusive to the social setting. Such a scoring model can be used in the context of top-k retrieval of spatial web objects [14], where a prestige-based relevance score is computed by combining the overall relevance of an object with its spatial distance.

2.2. General Setting

We consider a social setting in which we have a set of items (could be text documents, URLs, photos, etc) $\mathcal{I} = \{i_1, \ldots, i_m\}$, each tagged with one or more distinctive tags from a dictionary of tags $\mathcal{T} = \{t_1, t_2, \ldots, t_l\}$ by one or more users from $\mathcal{U} = \{u_1, \ldots, u_n\}$. We assume that users form an undirected weighted graph $G = (\mathcal{U}, E, \sigma)$ called the *social network*. In G, nodes represent users and σ is a function that associates to each edge $e = (u_1, u_2)$ a value in $(0, 1]$, called *the proximity* (or social) score between u_1 and u_2.

Given a seeker user s, a keyword query $Q = (t_1, ..., t_r)$ (a set of r distinct tags) and an integer value k, the top-k retrieval problem is to compute the (possibly ranked) list of the k items having the highest scores with respect to the seeker and query.

We describe next the score model for this problem.

Extending the model for social tagging systems presented in [4], we also assume the following two relations for tags:

- **tagging**: $Tagged(v, i, t)$: says that a user v tagged the item i with tag t,
- **tag proximity**: $SimTag(t_1, t_2, \lambda)$: says that tags t_1 and t_2 are similar, with similarity value $\lambda \in (0, 1)$.

We assume that a user can tag a given item with a given tag at most once. We first model for a user, item and tag triple (s, i, t) the *score* of item i for the given seeker s and tag t. This is denoted $score(i \mid s, t)$. Generally,

$$score(i \mid s, t) = h(fr(i \mid s, t)) \tag{2.2.1}$$

where $fr(i \mid s, t)$ is the *overall term frequency* of item i for seeker s and tag t, and h can be any a positive monotone function.

The overall term frequency function $fr(i \mid s, t)$ is defined as a combination of a network-dependent component and a document-dependent one, as follows:

$$fr(i \mid s, t) = \alpha \times tf(t, i) + (1 - \alpha) \times sf(i \mid s, t). \tag{2.2.2}$$

The former component, $tf(t, i)$, is the term frequency of t in i, i.e., the number of times i was tagged with t. The latter component stands for social frequency, a measure that depends on the seeker.[1]

If we consider that each user brings her own weight (proximity) to the score of an item, we can define the measure of social frequency as follows:

$$sf(i \mid s, t) = \sum_{v \in \{v \mid Tagged(v,i,t))\}} \sigma(s, v). \tag{2.2.3}$$

[1] The linear combination of Eq. (2.2.2) is one that is widely used when a local retrieval score and a global one are to be combined, e.g., in spatial search [14] or in social search [69]. However, any monotone combination of the two score components can be used in these approaches, as in ours.

Then, given a query Q as a set of tags $(t_1, \ldots, t_r)$, the overall score of i for seeker s and query Q,

$$score(i \mid s, Q) = g(score(i \mid s, t_1), \ldots, score(i \mid s, t_r)),$$

is obtained using a monotone aggregate function g over the individual scores for each tag. In this work, the aggregation function g is assumed to be a summation, $g = \sum_{t_j \in Q} score(i \mid s, t_j)$.

Extended proximity. The above scoring model takes into account only the neighborhood of the seeker (the users directly connected to her). But this can be extended to deal also with users that are indirectly connected to the seeker, following a natural interpretation that user links (e.g., similarity or trust) are (at least to some extent) transitive. We denote by σ^+ an *extended proximity*, which is to be computable from σ for any pair of users connected by a path in the network. Now, σ^+ can replace σ in the definition of social frequency we consider before (Eq. (2.2.3)), yielding an overall item scoring scheme that depends on the entire network instead of only the seeker's vicinity. We discuss shortly possible alternatives for σ^+ by means of aggregating σ values along paths in the graph. In the rest of this chapter, when we talk about proximity we refer to the extended one.

For a given seeker u, by her *proximity vector* we denote the list of users with non-zero proximity with respect to u, ordered in descending order of these proximity values.

Remark 1. In Eq. (2.2.2), the α parameter allows to tune the relative importance of the social component with respect to classic term frequency. When α is valued 1, the score becomes network-independent. On the other hand, when α is valued 0 the score depends exclusively on the social network.

Remark 2. Note that a network in which all the user pairs have a proximity score of 1 amounts to the classical document retrieval setting (i.e., the result is independent of the user asking the query).

Remark 3. Tag similarity can be integrated into Eq. (2.2.3), e.g., by setting a threshold τ s.t. if $SimTag(t, t', \lambda)$, with λ above τ, and $Tagged(v, i, t')$, we also add $\sigma(u, v)$ to $sf(i \mid u, t)$. For the sake of simplicity this is ignored here, but remains an integral part of the model.

Remark 4. Note that queries are not assumed to use only tags from $\mathcal{T}$. For any tag outside this dictionary, items will obviously have a score of 0.

2.2.1. Computing Extended Proximities

We describe in this section a key aspect of our algorithm for top-k search, namely on-the-fly computation of proximity values with respect to a seeker s. The issue here is to facilitate at any given step the retrieval of the most relevant unseen user u in the network, along with her proximity value $\sigma^+(s, u)$. This user will have the potential to

contribute the most to the partial scores of items that are still candidates for the top-k result, by Eq. (2.2.1) and (2.2.3).

We start by discussing possible candidates for σ^+, arguably the most natural ones, drawing inspiration from studies in the area of trust propagation for belief statements. We then give a wider characterization for the family of possible functions for proximity computation, to which these candidates belong.

Candidate 1(f_{mul}). Experiments on trust propagation in the Epinions network (for computing a final belief in a statement) [66] or in P2P networks show that (i) multiplying the weights on a given path between u and v, and (ii) choosing the maximum value over all the possible paths, gives the best results (measured in terms of precision and recall) for predicting beliefs. We can integrate this into our scenario, by assuming that belief refers to *tagging with a tag t*. We thus aggregate the weights on a path $p = (u_1, \ldots, u_l)$ (with a slight abuse of notation) as

$$\sigma^+(p) = \prod_i \sigma(u_i, u_{i+1}).$$

For seeker *Alice* in our running example, we gave in the previous section (Example 2) the proximity values and the ordering of the network under this candidate for σ^+.

Candidate 2(f_{min}). A possible drawback of Candidate 1 for proximity aggregation is that values may decrease quite rapidly. A σ^+ function that avoids this could be obtained by replacing multiplication over a path with minimal, as follows:

$$\sigma^+(p) = \min_i \{\sigma(u_i, u_{i+1})\}.$$

Under this σ^+ candidate, the values with respect to seeker *Alice* would be the following: $\{Bob : 0.9, Danny : 0.9, Charlie : 0.6, Frank : 0.6, Eve : 0.5, George : 0.5, Harry : 0.5, Ida : 0.25, Jim : 0.25\}$.

Candidate 3(f_{pow}). Another possible definition for σ^+ we consider relies on an aggregation that penalizes long paths, i.e., distant users, in a controllable way, as follows:

$$\sigma^+(p) = \lambda^{-\sum_i \frac{1}{\sigma(u_i, u_{i+1})}}.$$

where $\lambda \geq 1$ can be seen as a "drop parameter"; the greater its value the more rapid the decrease of proximity values. Under this candidate for σ^+, for $\lambda = 2$, the rounded values w.r.t seeker *Alice* would be $\{Bob : 0.46, Charlie : 0.31, Danny : 0.21, Eve : 0.077, Frank : 0.0525, George : 0.013, Ida : 0.003, Harry : 0.003, Jim : 0.0007\}$.

The key common feature of the candidate functions previously discussed is that they are monotonically decreasing over any path they are applied to, when σ draws values from the interval $[0, 1]$. More formally, they verify the following property:

Property 1. *Given a social network G and a path* $p = \{u_1, \ldots, u_l\}$ *in G, we have* $\sigma^+(u_1, \ldots, u_{l-1}) \geq \sigma^+(u_1, \ldots, u_l)$.

We then define σ^+ for any pair of user (s, u) who are connected in the network by taking the maximal weight over all their connecting paths. More formally, we define $\sigma^+(s, u)$ as

$$\sigma^+(s, u) = max_p\{\sigma^+(p) \mid s \overset{p}{\rightsquigarrow} u\}. \tag{2.2.4}$$

Note that when the first candidate (multiplication) is used, we obtain the same aggregation scheme as in [66], which is also employed in [69] in the context of top-k network aware search.

Example 4. *In our running example, if we use multiplication in Eq. (2.2.4), for the seeker Alice, for* $\alpha = 0$ *(hence exclusively social relevance), by Eq.(2.2.2) we obtain the following values for social frequency:* $SF_{Alice}(news) = \{D4 : 2.6, D2 : 1.01, D1 : 0.7, D6 : 0.6, D3 : 0.1, D5 : 0.05\}$ *and* $SF_{Alice}(site) = \{D4 : 1.11, D2 : 1.1, D3 : 0.9, D6 : 0.6, D1 : 0.05, D5 : 0.05\}$.

We argue next that to all aggregation definitions that satisfy Property 1 and apply Eq.(2.2.4) a greedy branch and bound approach is applicable. This will allow us to browse the network of users on the fly, at query time, visiting them in the order of their proximity with respect to the seeker.

More precisely, by generalizing Dijkstra's algorithm [26], we will maintain a max-priority queue, denoted H, whose top element $top(H)$ will be at any moment the *most relevant unvisited user*[2]. A user is *visited* when her tagged items are taken into account for the top-k result, as described in the following sections (this can occur at most once). At each step advancing in the network, the top of the queue is extracted (visited) and its unvisited neighbours (adjacent nodes) are added to the queue (if not already present) and are *relaxed* . Let $\otimes$ denote the aggregation function over a path (one that satisfies Property 1). Relaxation updates the best proximity score of these nodes, as described in Algorithm 1.

Algorithm 1: RELAX(s,u,v)

```
if σ+(s,u) ⊗ σ(u,v) > σ+(s,v) then
    σ+(s,v) = σ+(s,u) ⊗ σ(u,v)
end if
```

It can be shown by straightforward induction that this greedy approach allows us to visit the nodes of the network in decreasing order of their proximity with respect to the seeker, under any function for proximity aggregation that satisfies Property 1.

[2] Dijkstra's classic algorithm [26] computes single-source shortest paths in a weighted graph without negative edges.

We describe in the following section and in Section 2.4 how this greedy procedure for iterating over the network is used in our top-k social retrieval algorithm. Without loss of generality, in the rest of the chapter, consistent with social theories and with previous work on social top-k search, proximity will be based on Candidate 1 (multiplication).

2.3. Top-k Algorithm for the Social Case

As the main focus of this work is on the social aspects of search in tagging systems, we detail first our top-k algorithm, TOPKS, for the special case when the parameter α is 0. In this case, $fr(i \mid s,t)$ is simplified as

$$fr(i \mid s,t) = sf(i \mid s,t).$$

For each user u and tag t, we assume a precomputed projection over the *Tagged* relation for them, giving the items tagged by u with t; we call these the *user lists*. No particular order is assumed for the items appearing in a user list.

We keep a list D of top-k candidate items, sorted in descending order by their minimal possible scores (to be defined shortly). An item becomes candidate when it is met for the first time in a *Tagged* triple.

As usual, we assume that, for each tag t, we have an inverted list $IL(t)$ giving the items i tagged by it, along with their term frequencies $tf(t,i)$[3] in descending order of these frequencies. Starting from the topmost item, these lists will be consumed one item at a time, whenever the current item becomes candidate for the top-k result. By $CIL(t)$ we denote the items already consumed (as known candidates), by $top\_item(t)$ we denote the item present at the current (unconsumed) position of $IL(t)$, and we use $top\_tf(t)$ as short notation for the term frequency associated with this item.

We detail mostly the computation of social frequency, $sf(i \mid u,t)$, as it is the key parameter in the scoring function of items. Since when $\alpha = 0$ we do not use metrics that are tag-only dependent, it is not necessary to treat each tag of the query as a distinct dimension and to visit each in round-robin style (as done in the threshold algorithm or in ContextMerge). It suffices for our purposes to get at each step, for the currently visited user, all the items that were tagged by her with query terms (one user list for each term).

For each tag $t_j \in Q$, by $unseen\_users(i,t_j)$ we denote the maximal number of yet unvisited users who may have tagged item i with t_j. This is initially set to the maximal possible term frequency of t_j over all items (value that is available at the current position of the inverted list of $IL(t_j)$, as $top\_tf(t)$).

[3] In TOPKS, even though the social frequency does not depend on tf scores, we will exploit the inverted lists and the tf scores by which they are ordered, to better estimate score bounds. In particular, as detailed later, this allows us to achieve instance optimality.

Each time we visit a user u who tagged item i with t_j we can (a) update and approximation of sf, $\widetilde{sf}(i \mid s, t_j)$ (initially set to 0) by adding $\sigma^+(s, u)$ to it, and (b) decrement $unseen\_users(i, t_j)$.

When $unseen\_users(i, t_i)$ reaches 0, the social frequency value $sf(i \mid s, t_j)$ is final, i.e., $sf(i \mid s, t_j) = \widetilde{sf}(i \mid s, t_j)$ This also gives us a possible termination condition, as discussed in the following.

At any moment in the run of the algorithm, the *optimistic* score MaxScore$(i \mid s, Q)$ of an item i that has already been seen in some user list will be estimated using as social frequency for each tag t_j of the query the following value:

$$top(H) \times unseen\_users(i, t_j) + \widetilde{sf}(i \mid s, t_j).$$

Symmetrically, the *pessimistic* overall score, MinScore$(i \mid s, Q)$, is estimated by the assumption that, for each tag t_j, the current social frequency $\widetilde{sf}(i \mid s, t_j)$ will be the final one. The list of candidates D is sorted in descending order by this lowest possible score.

An upper-bound score on the yet unseen items, MaxScoreUnseen is estimated using as social frequency for each tag t_j the value $top(H) \times top\_tf(t))$.

When the maximal optimistic score of items that are already in D but not in its top-k is less than the pessimistic score of the last element in the current top-k of D (i.e., $D[k]$), the run of the algorithm can terminate, as we are guaranteed that the top-k can no longer change. (Note however that at this point the top-k items may have only partial scores and, if a ranked answer is needed, the process of visiting users should continue.)

We present the flow of TOPKS in Algorithm 2. Key differences with respect to ContextMerge's social branch are (i) the on-the-fly computation of proximity values, in lines 1-7 and 29-31 of the algorithm, and (ii) the consuming of inverted list positions, when they become candidates, in lines 20-28. For clarity, we first exemplify a TOPKS run without the latter aspect (this would correspond to a ContextMerge run).

Example 5. *Revisiting Example 1, recall that we want to compute the top-2 items for the query $Q = \{news, site\}$ from Alice's point of view. To simplify, let us assume that $score(i \mid u, t) = sf(i \mid u, t)$ and g is addition. We consider next how the algorithm described above runs.*

At the first iteration of the line 8 loop in the algorithm, we visit Bob's user lists, adding D3 to the candidate buffer. At the second iteration, we visit Danny's user lists, adding D2 and D4 to the candidate buffer. At the third iteration (Charlie's user list) we add D6 to the candidate list. D1 is added to the candidate list when the algorithm visits Frank's user lists, at iteration 4. Recall that $top\_tf(news) = 7$ and $top\_tf(site) = 5$.

The 6th iteration of the algorithm is the final one, visiting George's user lists, finding D2 tagged with news, site and D4 tagged with site. D4 and D2 are the top-2 candidates, with MinScore$(D4, Q) = 2.61$ *and* MinScore$(D2, Q) = 2.21$. *The closest candidate is D6,*

Algorithm 2: TOPKS$_{\alpha=0}$: top-k algorithm for $\alpha = 0$

Require: *seeker* s, *query* $Q = (t_1, \ldots, t_r)$

for all *users* u, *tags* $t_j \in Q$, *items* i **do**
  $\sigma^+(s,u) = -\infty$
  $\widetilde{sf}(i \mid s, t_j) = 0$
  set $IL(t_j)$ *position on first entry;* $CIL(t_j) = \emptyset$
end for
$\sigma^+(s,s) = 0$; $D = \emptyset$ (candidate items)
$H \leftarrow$ *max-priority queue of nodes* u *(sorted by* $\sigma^+(s,u)$*), initialized with* $\{s\}$
while $H \neq \emptyset$ **do**
  u=EXTRACT_MAX(H);
  for all *tags* $t_j \in Q$, *triples* $Tagged(u,i,t_j)$ **do**
    $\widetilde{sf}(i \mid s, t_j) \leftarrow \widetilde{sf}(i \mid s, t_j) + \sigma^+(s,u)$
    if $i \notin D$ **then**
      add i *to* D
      for all *tags* $t_l \in Q$ **do**
        $unseen\_users(i, t_l) \leftarrow top\_tf(t_l)$(initialization)
      end for
    end if
    $unseen\_users(i, t_j) \leftarrow unseen\_users(i, t_j) - 1$
  end for
  while $\exists t_j \in Q$ *s.t.* $i = top\_item(t_j) \in D$ **do**
    $tf(t_j, i) \leftarrow top\_tf(t_j)$($t_j$'s frequency in i is now known)
    advance $IL(t_j)$ *one position*
    $\Delta \leftarrow tf(t_j, i) - top\_tf(t_j)$ (the top_tf drop)
    for all *items* $i' \in D \setminus CIL(t_j)$ **do**
      $unseen\_users(i', t_j) \leftarrow unseen\_users(i', t_j) - \Delta$
    end for
    add i *to* $CIL(t_j)$
  end while
  for all users v s.t. $\sigma(u,v) \in E$ **do**
    RELAX(s,u,v)
    UPDATE(v,H)
  end for
  if MINSCORE$(D[k], Q) > max_{l>k}$(MAXSCORE$(D[l], Q))$ AND MINSCORE$(D[k], Q) >$ MAXSCOREUNSEEN **then**
    break
  end if
end while
return $D[1], \ldots, D[k]$

with $\textsc{MinScore}(D6, Q) = 1.2$ *and* $\textsc{MaxScore}(D6, Q) = 1.2 + 6 \times 0.1 + 4 \times 0.1 = 2.2$. *Also,* $\textsc{MaxScoreUnseen}(Q) = 7 \times 0.1 + 5 \times 0.1 = 1.2$. *Finally,* $\textsc{MaxScore}(D6, Q) < \textsc{MinScore}(D2, Q)$ *and since we have* $\textsc{MaxScoreUnseen}(Q) < \textsc{MinScore}(D2, Q)$, *the algorithm stops returning* $D4$ *and* $D2$ *as the top-*2 *items.*

We discuss next the interest of consuming of inverted list positions, when these become candidates (illustrated in Example 6). In lines 20-28, we aim at keeping to a minimum the worst-case estimation of the number of unseen taggers. More precisely, we test whether there are top-k candidates i (i.e., items already seen in user lists) for which the term frequency for some tag t_j of Q, $tf(t_j, i)$, is "within reach" as the one currently used (from $IL(t_j)$) as the basis for the optimistic (maximal) estimate of the number of yet unseen users who tagged candidate items with t_j. When such a pair (i, t_j) is found, we can do the following adjustments:

1. refine the number of unseen users who tagged i with t_j from a (possibly loose) estimate to its *exact* value; this is marked when i is added to the CIL list of t_j (line 27), and from this point on the number of unseen users will only change when new users who tagged i with t_j are found (line 18).

2. advance (at the cost of a sequential access) beyond i in the inverted list of t_j, to the next best item; this allows us to refine (at line 25) the estimates $unseen\_users(i', t_j)$ for all candidates i' for which the exact number of users who tagged with t_j is yet unknown.

(We found in the experimental evaluation (Section 2.8) that this aspect has the potential to drastically improve the cost of the search. Since tf-values in inverted lists fall quite rapidly in most practical settings, we witnessed significant cost savings, while using relatively few such list position increments.)

Example 6. *Let us now consider how the choice of advancing in the inverted lists when possible influences the number of needed iterations. At first,* $top\_tf(news) = 7$, $top\_item(news) = D4$, *and* $top\_tf(site) = 5$, $top\_item(site) = D2$.

The first iteration only introduces $D3$ *and thus we cannot advance in any of the two inverted lists. However, the discovery of* $D2$ *and* $D4$ *in step* 2 *allows us to fix their exact tf values and advance the inverted lists. The new positions are:* $top\_tf(news) = 2$, $top\_item(news) = D1$, *and* $top\_tf(site) = 1$, $top\_item(site) = D6$. $D6$*'s discovery in iteration* 3 *allows us to advance further in the inverted lists. Finally, in step* 4, *the discovery of* $D1$ *allows the algorithm to advance in the inverted lists to* $top\_tf(news) = 1$, $top\_item(news) = D5$, *and* $top\_tf(site) = 1$, $top\_item(site) = D5$ *(the only undiscovered item). This allows for some drastic score estimation refinements. We have the same top-*2 *candidates,* $D4$ *and* $D2$ *having* $\textsc{MinScore}(D4, Q) = 1.81$ *and* $\textsc{MinScore}(D2, Q) = 1.21$. *The closest item is again* $D6$ *having* $\textsc{MinScore}(D6, Q) = \textsc{MaxScore}(D6, Q) = 1.2$, *since we know that we have*

visited all users who tagged D6. $\text{MaxScoreUnseen}(Q) = 1 \times 0.3 + 1 \times 0.3 = 0.6$*, since the maximal unseen document, D6 is tagged only once with each tag.* $\text{MaxScore}(D6, Q) < \text{MinScore}(D2, Q)$ *and* $\text{MaxScoreUnseen}(Q) < \text{MinScore}(D2, Q)$ *allows us to exit the loop, two steps before the unrefined algorithm, returning the exact top-2: D4 and D2.*

We can prove the following property of our algorithm:

Property 2. *For a given seeker s,* $\text{TOPKS}_{\alpha=0}$ *visits the network in decreasing order of the* σ^+ *values with respect to s.*

As a corollary of Property 2, we have that $\text{TOPKS}_{\alpha=0}$ visit users who may be relevant for the query in the same order as ContextMerge [69]. More importantly, we prove in Section 2.3.1 that our algorithm visits as few users as possible, i.e., it is instance optimal with respect to this aspect. Moreover, the experiments show that TOPKS can drastically reduce the number of visited user lists in practice (see Section 2.8).

2.3.1. Instance Optimality

We will use the same definition of instance optimality as in [31]. For a class of algorithms **A**, a class of legal inputs (instances) **D**, $cost(\mathcal{A}, \mathcal{D})$ denotes the cost of running algorithm $\mathcal{A} \in \mathbf{A}$ on input $\mathcal{D} \in \mathbf{D}$. An algorithm $\mathcal{A}$ is said to be *instance optimal* for its class **A** over inputs **D** if for every $\mathcal{B} \in \mathbf{A}$ and every $\mathcal{D} \in \mathbf{D}$ we have $cost(\mathcal{A}, \mathcal{D}) = O(cost(\mathcal{B}, \mathcal{D}))$.

Let c_{UL} be the abstract cost of accessing the user list - a process which involves the relatively costly operations of finding the proximity value of the user and retrieving the items tagged by the user with query terms - and let $users(\mathcal{A}, \mathcal{D})$ be the number of total user lists needed for establishing the top-k for algorithm $\mathcal{A}$ on input $\mathcal{D}$. Let c_S be the abstract cost of sequentially accessing the data in IL_t, and let $seqitems(\mathcal{A}, \mathcal{D})$ be the total number of sequential accesses to IL for algorithm $\mathcal{A}$ on input $\mathcal{D}$. In practice, $c_{UL} \gg c_S$ is a reasonable assumption, hence, for two algorithms $\mathcal{A}$ and $\mathcal{B}$, we have

$$\frac{users(\mathcal{A}, \mathcal{D}) \times c_{UL} + seqitems(\mathcal{A}, \mathcal{D}) \times c_S}{users(\mathcal{B}, \mathcal{D}) \times c_{UL} + seqitems(\mathcal{B}, \mathcal{D}) \times c_S} \approx \frac{users(\mathcal{A}, \mathcal{D})}{users(\mathcal{B}, \mathcal{D})}.$$

Therefore, for a fair cost estimate in practical social search settings, a reasonable assumption is to consider

$$cost(\mathcal{A}, \mathcal{D}) = users(\mathcal{A}, \mathcal{D}).$$

Let us now define the class of "social" algorithms **S** to which both $\text{TOPKS}_{\alpha=0}$ and ContextMerge (when $\alpha = 0$) belong. These algorithms correctly return the top-k items for a given query Q and seeker s, they do not use random accesses to $IL(t)$ indexes in order to fetch a certain tf value, and they do not include in their working

buffers (e.g., candidate buffer D) items that were not yet encountered in the user lists. The last assumption could be seen as a "no wild guess" policy, by which the algorithm cannot guess that an item might be encountered in some later stages. This is a reasonable assumption in practice, as the number of items needed for computing a top-k result for a given seeker should in general be much smaller than the total number of items tagged by query terms.

The class **D** of accepted inputs consists of the inputs that respect the setting described in Section 2.2.

Theorem 1. TOPKS$_{\alpha=0}$ *is instance optimal over* ***S*** *and* ***D****, when the cost is defined as* $cost(\mathcal{A}, \mathcal{D}) = users(\mathcal{A}, \mathcal{D})$.

Proof. Since on each access to a user list, all items tagged by the respective user with any of the query terms are retrieved, the position in the proximity vector at any step in the run of the algorithm is not tag-dependent. So $cost(\mathcal{A}, \mathcal{D})$ is equal to the position p in the seeker's proximity vector at the moment of $\mathcal{A}$'s termination. Throughout the proof, we use the subscript p to denote the value of a given variable at step p in the execution of $\mathcal{A}$. We will use a proof argument similar in style to the one for NRA [31].

Let us assume that TOPKS$_{\alpha=0}$ does not stop at position p (in the proximity vector) and that there exists an algorithm $\mathcal{A} \neq$ TOPKS$_{\alpha=0}$ that does.

Since TOPKS$_{\alpha=0}$ does not stop at position p, there exists an item $r \notin \{D_p[1], \ldots, D_p[k]\}$ having $\text{MaxScore}_p(r, Q) > \text{MinScore}_p(D_p[k], Q)$, and $\text{MinScore}_p(r, Q) \leqslant \text{MinScore}_p(D_p[i], Q), \forall i \in \{1, \ldots, k\}$. If $\text{MinScore}_p(r, Q) = \text{MinScore}_p(D_p[k], Q)$ then necessarily $\text{MaxScore}_p(r, Q) \leqslant \text{MaxScore}_p(D_p[k], Q)$ (ties for pessimistic scores are broken by the optimistic ones, then arbitrarily for the optimistic scores).

In $\mathcal{D}$, we assume that at step p we have with TOPKS$_{\alpha=0}$ in the current (unconsumed) position in each of the $|Q|$ inverted lists $IL(t_j)$ an item v_j, necessarily not yet candidate. By definition, for any algorithm $\mathcal{A} \in \mathbf{S}$, for any tag t_j of the input Q, $\mathcal{A}$ is *at most as advanced* in the inverted list $IL(t_j)$ as TOPKS$_{\alpha=0}$. Without loss of generality, let us assume $\mathcal{A}$ is as advanced as TOPKS$_{\alpha=0}$.

Towards a contradiction, showing that $\mathcal{A}$ is not sound over all possible inputs, we will construct an instance $\mathcal{D}'$, which is equal to $\mathcal{D}$ up to position p. We consider the following two possible cases:

Case 1: $\mathcal{A}$ outputs r as one of the top-k items, i.e., there do not exist k items having a higher score than r.

In constructing $\mathcal{D}'$, we start from what $\mathcal{A}$ could have already read and used, including the items $v_j = top\_item_p(t_j)$ and the value $top_p(H)$ (the proximity value of the $p+1$ user).

$\mathcal{D}'$ will be such that $\text{Score}(r, Q) = \text{MinScore}_p(r, Q)$, and $\text{Score}(D_p[i], Q) = \text{MaxScore}_p(D_p[i], Q), \forall i \in \{1, \ldots, k\}$. Now, for each $D_p[i]$, if $\text{MaxScore}_p(D_p[i], Q) > \text{MinScore}_p(D_p[i], Q)$, i.e., we do not have $D_p[i]$'s final score at step p, we assume the

following in $\mathcal{D}'$. For each $t_j \in Q$ for which $tf(D_p[i], t_j)$ is unknown, we assume that we have $D_p[i]$ in $IL(t_j)$ after v_j, with $tf(D_p[i], t_j) = top\_tf_p(t_j)$. Also, for every $t_j \in Q$ we set in the proximity vector, after $p+1$, the next $x_{ij} = unseen\_users(D_p[i], t_j)$ values to $top_p(H)$, making also $D_p[i]$ present in each of these users' lists for t_j. By doing so, the exact score of each $D_p[i]$, $i \in \{1, \ldots, k\}$, is equal to the maximal possible one at step p; after $max_{i,j}(x_{ij})$ steps, all these k scores $\textsc{Score}(D_p[i], Q)$ would be computed. For item r, for each $t_j \in Q$ for which we do not have $tf(r, t_j)$, since r must come later in $IL(t_j)$ (after v_j), we can assume that $tf(r, t_j) = partial\_tf(r, t_j)$ (this makes $unseen\_users(r, t_j) = 0$). Also, for every $t_j \in Q$ for which we do know $tf(r, t_j)$, after the required $max_{i,j}(x_{ij})$ proximity values set as described previously, we set the next $unseen\_users(r, t_j)$ in the proximity vector to 0, with each of these users having tagged r with t_j. All this ensures that $\textsc{MinScore}_p(r, Q) = \textsc{Score}(r, Q)$.

We can now contradict the correctness of algorithm $\mathcal{A}$, showing that $\textsc{Score}(r, Q) < \textsc{Score}(D_p[i], Q)$ for all i.

We have the following inequalities:

$$\textsc{MinScore}_p(D_p[k], Q) \geqslant \textsc{MinScore}_p(r, Q) \quad (2.3.1)$$

$$\textsc{MinScore}_p(D_p[k], Q) \leqslant \textsc{MinScore}_p(D_p[i], Q), \forall i \quad (2.3.2)$$

$$\textsc{MinScore}_p(D_p[i], Q) \leqslant \textsc{MaxScore}_p(D_p[i], Q), \forall i \quad (2.3.3)$$

If $\textsc{MinScore}_p(r, Q) < \textsc{MinScore}_p(D_p[k], Q)$ then it follows from Eq. (2.3.1), (2.3.2), (2.3.3) that

$$\textsc{Score}(r, Q) < \textsc{Score}(D_p[i], Q), \forall i.$$

If $\textsc{MinScore}_p(r, Q) = \textsc{MinScore}_p(D_p[k], Q)$ then, for each $i \in \{1, \ldots, k\}$, if:

1. $\textsc{MinScore}_p(D_p[k], Q) = \textsc{MinScore}_p(D_p[i], Q)$: we have $\textsc{MaxScore}_p(r, Q) > \textsc{MinScore}_p(D_p[k], Q)$ and $\textsc{MaxScore}_p(r, Q) \leqslant \textsc{MaxScore}_p(D_p[i], Q)$; it follows that $\textsc{Score}(r, Q) < \textsc{Score}(D_p[i], Q)$,

2. $\textsc{MinScore}_p(r, Q) < \textsc{MinScore}_p(D_p[k], Q)$: we have $\textsc{MinScore}_p(D_p[k], Q) < \textsc{Score}(D_p[i], Q)$; it follows that $\textsc{Score}(r, Q) < \textsc{Score}(D_p[i], Q)$.

Hence, in any possible configuration, r is not in the top-k result over $\mathcal{D}'$. But since $\mathcal{D}'$ and $\mathcal{D}$ are indistinguishable by algorithm $\mathcal{A}$, which stops at step p outputting r in the result, this contradicts $\mathcal{A}$'s correctness.

Case 2: $\mathcal{A}$ does not output r as a top-k item, which means that $\mathcal{A}$ assumes that the final score of r, $\textsc{Score}(r, Q)$ is not in the top-k scores for $\mathcal{D}$.

$\mathcal{D}'$, undistinguishable from $\mathcal{D}$ up to position p, will now be such that $\textsc{Score}(r, Q) = \textsc{MaxScore}_p(r, Q)$ and $\textsc{Score}(D_p[i], Q) = \textsc{MinScore}_p(D_p[i], Q)$, for each $D_p[i] \in D_p$ s.t. $D_p[i] \neq r$.

If r's score at step p is not already the final one, i.e., $\text{MaxScore}_p(r, Q) = \text{MinScore}_p(r, Q)$, we assume the following in $\mathcal{D}'$: for each tag $t_j \in Q$ for which $tf(r, t_j)$ is yet unknown, we assume that r comes later (after v_j) in $IL(t_j)$, having $tf(r, t_j) = top\_tf_p(t_j)$. Then, for every $t_j \in Q$ we set in the proximity vector, after the $p+1$ position, the next $x_j = unseen\_users(r, t_j)$ values to $top_p(H)$, making also r present in each of these users' lists for t_j.

By this, the exact score of r is equal to the maximal possible one at step p; after $max_j(x_j)$ steps, the score $\text{Score}r, Q)$ would be computed.

Symmetrically, for each each $D_p[i] \in D_p$ s.t. $D_p[i] \neq r$, and each $t_j \in Q$ for which $tf(D_p[i], t_j)$ is yet unknown, we assume that $D_p[i]$ comes later (after v_j) in $IL(t_j)$, having $tf(D_p[i], t_j) = partial\_tf(D_p[i], t_j)$ (hence $unseen\_users(D_p[i], t_j) = 0$). Then, for every $t_j \in Q$ for which we know $tf(D_p[i], t_j)$, after the $max_j(x_j)$ values set as described previously in the seeker's proximity vector, we set the next $y_{ij} = unseen\_users(D_p[i], t_j)$ values to 0, making also $D_p[i]$ present in each of these users' lists for t_j. This construction ensures that, the exact score of each $D_p[i]$ is equal to the minimal possible one at step p; after $max_{i,j}(y_{ij})$ steps, all these scores $\text{Score}(D_p[i], Q)$ would be computed.

Since we have that

$$\text{Score}(r, Q) = \text{MaxScore}_p(r, Q) > \text{MinScore}_p(D_p[k], Q)$$

and $\text{MinScore}_p(D_p[k], Q) = \text{Score}(D_p[k], Q)$, given that for every item $D_p[l]$, $l > k$ s.t. $D_p[l] \neq r$ we have $\text{Score}(D_p[l], Q) \leqslant \text{MinScore}_p(D_p[k], Q)$, r should be among the top-k items in $\mathcal{D}'$. But since $\mathcal{D}'$ and $\mathcal{D}$ are indistinguishable by algorithm $\mathcal{A}$, which stops at step p without outputting r in the result, this contradicts $\mathcal{A}$'s correctness.

In this proof, we have ignored $\text{MaxScoreUnseen}(Q)$ in the inequalities. The unseen items can be simulated by adding one virtual item i_v to D, which does not exist and will never be encountered in user lists, with $\text{MinScore}(i_v, Q) = 0$ and $\text{MaxScore}(i_v, Q) = \text{MaxScoreUnseen}(Q)$. Then, the same proof argument applies to these items. □

2.4. Algorithm for the General Case

For the general case, in which $\alpha \in [0, 1]$, we adapt the ContextMerge [69] algorithm to include the on-the-fly processing of user proximities.

At each iteration, the algorithm can alternate, by calling ChooseBranch(), between two possible execution branches: the *social branch* (lines 8-31 of Algorithm 2) and the *textual branch*, which is a direct adaptation of NRA.

As in the exclusively social setting of the previous section, we will read term frequency scores $tf(t_j, i)$ from the inverted lists, on a per-need basis, either as in line 21 of $\text{TOPKS}_{\alpha=0}$, or when advancing on the textual branch. Initially, all unknown tf-scores are assumed to be set to 0.

The optimistic overall score MaxScore(i, Q) of an item i that is already in the candidate list D will now be computed by setting $fr(i \mid s, t)$, defined in Eq. (2.2.2), to

$$\begin{aligned} fr(i \mid s,t) &= (1-\alpha) \times top(H) \times unseen\_users(i,t) + (1-\alpha) \times \widetilde{sf}(i \mid s,t) + \\ &\quad \alpha \times max(\ tf(t,i),\ top\_tf(t)\ ). \end{aligned}$$

The last term accounts for the textual weight of the score, and uses either the exact term frequency (if known), or an upper-bound for it (the score in the current position of $IL(t)$).

Symmetrically, for the pessimistic overall score MinScore(i, Q), the frequency $fr(i \mid u, t)$ will be computed as

$$fr(i \mid s,t) = (1-\alpha) \times \widetilde{sf}(i \mid s,t) + \alpha \times max(\ tf(t,i)\ ,\ partial\_tf(t)\ ),$$

where *partial_tf* represents the count of visited users who tagged i with t_j, which is used as lower-bound for $tf(t_j, i)$ when this is not yet known.

The upper-bound for the score on the yet unseen items, MaxScoreUnseen, is estimated using as overall frequency for each tag t_j the following value:

$$fr(i \mid s,t) = \alpha \times top\_tf(t) + (1-\alpha) \times top(H) \times top\_tf(t)).$$

We present the flow of the general case algorithm in Algorithm 3. Method Initialize() amounts to lines 1-6 of TOPKS$_{\alpha=0}$, and method ProcessSocial() amounts to lines 8-31 of TOPKS$_{\alpha=0}$ (modulo the straightforward adjustment for the count *partial_tf*).

The difference between the $\alpha = 0$ case and the general case is the processing of the inverted lists (textual branch), which is done as in the NRA algorithm (see lines 7-13 of Algorithm 3). We discuss how the choice of the branch to be followed is done, by the ChooseBranch() subroutine, in Section 2.4.1.

2.4.1. Choosing Between the Social and Textual Branches

The TOPKS$_{\alpha=0}$ algorithm, in which only the social branch matters, is instance optimal (see Theorem 1), with the cost being estimated as $users(\text{TOPKS}_{\alpha=0}, \mathcal{D})$. As the NRA algorithm [31], when only the textual branch matters, TOPKS$_{\alpha=1}$ is instance optimal, with the cost being estimated as $seqitems(\text{TOPKS}_{\alpha=0}, \mathcal{D})$.

When α is not one of the extreme values, under a cost function as a combination of the two above, of the form

$$users(\text{TOPKS}_{\alpha=0}, \mathcal{D}) \times c_{UL} + seqitems(\text{TOPKS}_{\alpha=1}, \mathcal{D}) \times c_S,$$

a key role for efficiency is played by ChooseBranch().

In [69], the choice between the textual branch or the social one was done by estimating the maximum potential score of each, in round-robin manner over the query

dimensions. For a query tag t_j, the maximal contribution of the social branch would be estimated as $\text{MaxSocial}(t_j) = (1-\alpha) \times max\_tf(t_j) \times top(H)$, where $max\_tf(t_j)$ is the maximum tf for t_j (i.e., the number of taggers for the item that has been tagged the most with t_j). For the textual part, the maximal potential contribution would be estimated by setting $\text{MaxTextual}(t_j) = \alpha \times top\_tf(t_j)$. Then, if $\text{MaxSocial}(t_j) > \text{MaxTextual}(t_j)$ the social branch was chosen, otherwise the textual branch is chosen.

Algorithm 3: TOPKS: top-k algorithm for the general case

Require: seeker s, query $Q = (t_1, \ldots, t_r)$
1: Initialize()
2: **while** $H \neq \emptyset$ **do**
3: ChooseBranch()
4: **if** *social branch* **then**
5: ProcessSocial()
6: **else**
7: **for all** *tags* $t_j \in Q$, *item* $i = top\_item(t_j)$ **do**
8: **if** $i \notin D$ **then**
9: *add* i *to* D *and* $CIL(t_j)$
10: **end if**
11: $tf(t_j, i) \leftarrow top\_tf(t_j)$
12: *advance* $IL(t_j)$ *one position*
13: **end for**
14: **end if**
15: **if** $\text{MinScore}(D[k], Q) > max_{l>k}(\text{MaxScore}(d[l], Q)$ and $\text{MinScore}(D[k], Q) > \text{MaxScoreUnseen}$ **then**
16: **break**
17: **end if**
18: **end while**
19: **return** $D[1], \ldots, D[k]$

We use a different heuristics for the branch choice. At any point in the run of TOPKS, unless termination is reached, we have at least one item r with $\text{MaxScore}(r, Q) \geqslant \text{MinScore}(D[k], Q)$. We consider the item $r = D[argmax_{l>k}(\text{MaxScore}(D[l], Q)]$, which has the highest potential score, and we choose the branch that is the most likely to refine r's score (put otherwise, the branch that counts the most in the MaxScore estimation for r). The intuition behind this branch choice mechanism is that it is more likely to advance the run of the algorithm closer to termination.

For each tag $t_j \in Q$, we set $\text{MaxTextual}(t_j)$ to $\alpha \times top_tf(t_j)$ if the term frequency $tf(t_j, r)$ is not yet known, or to 0 otherwise. For the social part of the score, we set

$$\text{MaxSocial}(t_j) = (1-\alpha) \times unseen\_users(t_j, r) \times top(H).$$

Then, we follow the social branch if, for at least one of the tags, MaxSocial is greater

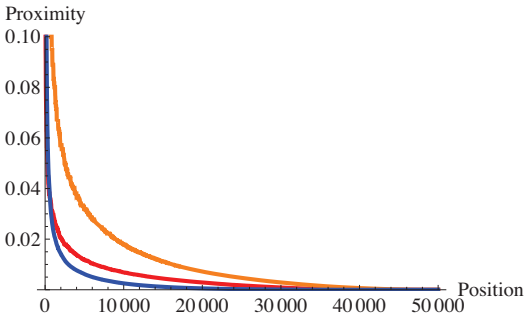


Figure 2.1.: Examples on the evolution of proximity values.

than MaxTextual.

Note that we deal with the tags of the query "in bulk", and advance simultaneously on their inverted lists when the textual branch is followed.

Remark. We have adopted so far a "disjunctive" interpretation for queries, in which items can score on each tag-dimension individually. However, our approach can be adapted in straightforward manner to a "conjunctive" interpretation: the pessimistic score should be maintained at 0 until the item's scores – at least partial ones – are known for all tags.

2.5. Efficiency by Approximation

The algorithm described in the previous section is sound and complete, and requires no prior (aggregated) knowledge on the proximity values with respect to a certain seeker (e.g., statistics); this was also the assumption in [69]'s ContextMerge algorithm. Moreover, it is instance optimal in the exclusively social setting (our main focus in this work) with respect to the number of visited users. While we improve the running time in both this setting and the general one (more on experimental results in Section 2.8), in practice, however, the search may still visit a significant part of the user network and their item lists before being able to conclude that the top-k answer can no longer change.

But if some statistics about proximity are known at query time (i.e., on how the values in a proximity vector variate from the most relevant user to the least relevant one), this may enable us to use more refined termination conditions, and thus to minimize the gap between the step at which the final top-k has been established and the actual termination of the algorithm. Indeed, the experiments we performed on Del.icio.us data showed that, in average, the last top-k change occurs much sooner, hence *there is a clear opportunity to stop the browsing of the network earlier.*

We take a first step in this direction, discussing two possible approaches for using

score estimations based on proximity statistics, which trade accuracy for efficiency (in terms of visited users). More specifically, in Algorithm 3, the MaxScore, MaxScoreUnseen and MinScore bounds have all used the safest possible values for the proximities of yet unseen users: either the top (maximum) value of the max-priority queue ($top(H)$) for the first two bounds, or its minimal possible value (zero) for the third one. In practice, however, any of these extreme configurations is rarely met. For illustration, we give in Figure 2.1 the proximity vectors for some randomly sampled users. Observe that these fall rapidly, and this may be the case in many real-world similarity or proximity networks.

Hence one possible direction for reducing the number of visited users is to precompute and materialize a high-level description (more or less complex, more or less accurate) of users' proximity vectors (of their distribution of values). This would allow us to use a tighter estimation for the remaining (unseen) users, instead of uniformly associating them the extreme score ($top(H)$ or 0). In doing so, we may obviously introduce approximations in the final result, and our approximate techniques provide a trade-off between accuracy drop on one hand and negligible memory consumption and reduced running time on the other hand.

2.5.1. Estimating Bounds using Mean and Variance

We first consider as a proximity vector description one that is very concise yet generally-applicable and effective, keeping for a given seeker two parameters: the *mean* value of the proximities in the vector and the *variance* of these values. We adopt here the simplifying assumption that the values in the seeker's vector are independent, essentially interpreting the proximity vector as a random one.

At any step in the run of the algorithm, using the mean and variance, for the remaining (yet unvisited) $unseen\_users(i,t)$ for a given item i and tag $t \in Q$, we can derive (a) lower bounds for the average of their proximity values, for MinScore estimations, or (b) upper bounds for the average of their proximity values, for MaxScore estimations. The guarantees of these bounds can be controlled (in a probabilistic sense) via a precision parameter $\delta \in (0,1]$, by which lower values lead to higher precision and 1 leads to a setting with no guarantees.

More precisely, let p be the current position in the proximity vector and let $\sigma^+_{p:}(s)$ be the vector containing the remaining (unseen) values of $\sigma^+(s)$. Knowing the overall mean and variance of the entire proximity vector $\sigma^+(s)$, and having the proximity values seen so far (denoted $\sigma^+_{0:p}(s)$), we can easily compute the average and variance of the remaining proximity values (those in $\sigma^+_{p:}(s)$).

Then, the mean and variance of the average of $unseen\_users(i,t)$ randomly chosen

proximity values from the remaining ones can be obtained as follows:[4]

$$\begin{aligned} Exp[\sigma^{+}_{p:}, unseen\_users(i,t)] &= E[\sigma^{+}_{p:}], \\ Var[\sigma^{+}_{p:}, unseen\_users(i,t)] &= \frac{Var[\sigma^{+}_{p:}]}{unseen\_users(i,t)}. \end{aligned}$$

When the input query contains more than one tag, its size $|Q|$ needs to be taken into account in the estimations. In order to avoid computational overhead, we uniformly chose a non-optimal per-tag probabilistic parameter δ' that ignores per-tag score distributions, as follows:

$$\delta' = 1 - (1-\delta)^{1/|Q|}. \tag{2.5.1}$$

EstMax(p, δ) represents, for each query tag, the upper bound of the expected value of the average of $unseen\_users(i,t)$ values drawn from $\sigma^{+}_{p:}(s)$, which holds with probability at least $1-\delta'$. Similarly, EstMin(p, δ) represents the lower bound of the expected value of the average of $unseen\_users(i,t)$ values drawn from $\sigma^{+}_{p:}(s)$, which holds with probability at least $1-\delta'$. For estimating MinScore when $i \notin CIL(t)$, the fact that we have no information about the difference between $tf(i,t)$ and $partial\_tf(t,i)$ (the users who tagged item i with t so far) means that we cannot assume that other users may have tagged i, so we keep this estimation as in the initial (exact) algorithm.

By using Chebyshev's inequality [58], these bounds can be computed as follows:

$$\begin{aligned} \text{EstMax}(p,\delta) &= E[\sigma^{+}_{p:}(s)] + \sqrt{\frac{Var[\sigma^{+}_{p:}(s)]}{unseen\_users(i,t) \times \delta'}} \\ \text{EstMin}(p,\delta) &= E[\sigma^{+}_{p:}(s)] - \sqrt{\frac{Var[\sigma^{+}_{p:}(s)]}{unseen\_users(i,t) \times \delta'}} \end{aligned}$$

We give the score estimations, changed by generalizing the proximity estimations, in Table 2.1. We present in the experimental results the effect of this approximate approach on running time, showing significant overall improvement. In our experiments, even for $\delta = 0.9$, the returned top-k answers had reasonable precision levels (around 90%).

We discuss in the next section another approach for tighter score estimates, using more detailed descriptions of proximity vectors. We conclude this section with a discussion on how these concise descriptions of proximity vectors could be maintained up-to-date in dynamic environments, in Section 2.5.3.

2.5.2. Estimating Bounds Using Histograms

The advantage of the approach described the previous section is twofold: low memory requirements and estimation bounds that are applicable for any value distribution.

[4] This is possible under independence assumptions that may not entirely hold, but turn out to be reasonable in practice (see Section 2.8).

Table 2.1.: Optimistic and pessimistic estimations of $fr(i \mid t, u)$ at step p (general case).

score	$i \in CIL(t)$	estimation
MinScore(i,t)	yes	$\alpha \times tf(i,t) + (1-\alpha) \times (sf(i \mid s,t) +$ EstMin$(p,\delta) \times unseen_users(i,t))$
	no	$\alpha \times partial\_tf(t,i) + (1-\alpha) \times sf(i \mid s,t)$
MaxScore(i,t)	yes	$\alpha \times tf(i,t) + (1-\alpha) \times (sf(i \mid s,t) +$ EstMax$(p,\delta) \times unseen_users(i,t))$
	no	$\alpha \times top\_tf(t) + (1-\alpha) \times (sf(i \mid s,t) +$ EstMax$(p,\delta) \times unseen_users(i,t))$
MaxScoreUnseen(t)		$\alpha \times top\_tf(t) +$ $(1-\alpha) \times$ EstMax$(p,\delta) \times top_tf(t)$

However, it may offer estimation bounds that are too loose in practice, and hence not reach the full potential for efficiency of approximate score bounds. To address this issue, we can imagine – as a compromise between keeping only these two statistics and keeping the entire pre-computed proximity vector – an approach in which we describe the distribution at a finer granularity, based on *histograms*.

More precisely, for a seeker s, we denote this histogram as $h(\sigma^+(s))$. It consists of b buckets, each bucket b_i, for $i \in \{1, \ldots, b\}$, containing n_i items in the interval $(low_i, high_i]$ (the 0 values are assigned to bucket b). Then, the probability that there exists a proximity value x greater than low_i, knowing the histogram $h(\sigma^+(s))$, is

$$Pr[x > low_i \mid h(\sigma^+(s))] = \sum_{j=1}^{i} n_j / n.$$

At any step p in the run of the algorithm, we maintain a partial histogram denoted as $h(\sigma^+_{p:}(s))$, obtained by removing from $h(\sigma^+(s))$ the p already encountered proximity values.

Similar to the previous approach, we can drill down the overall δ parameter to a δ' one for each query tag. Then, EstMax(p,δ) can be given by the minimal value in the partial histogram, such that the resulting estimation of MaxScore(i,t) holds with at least probability $1-\delta'$. Conversely, EstMin(p,δ) is given by the maximal value in the partial histogram, such that the resulting estimation of MinScore(i,t) holds with at least probability $1-\delta'$.

In manner similar to Eq.(2.5.1), we need to take into account the fact that a number of $unseen\_users(i,t)$ such estimated values lead to an overall approximate estimation, for both EstMin and EstMax. Therefore, each of these values is uniformly estimated using a stronger probabilistic parameter $\delta''(i,t)$, depending on $unseen\_users(i,t)$, as

follows:

$$\delta''(i,t) = 1 - (1-\delta'')^{1/unseen\_users(i,t)}.$$

Formally, having $h(\sigma^{+}_{p:}(s))$ and $\delta''(i,t)$, we estimate $\textsc{EstMax}(p,\delta)$ and $\textsc{EstMin}(p,\delta)$ as follows:

$$\begin{aligned} \textsc{EstMax}(p,\delta) = &\quad \min\{low_i \mid Pr[x > low_i \mid h(\sigma^{+}_{p:}(s))] \leq \delta''(i,t)\}, \\ \textsc{EstMin}(p,\delta) = &\quad \max\{low_i \mid Pr[x > low_i \mid h(\sigma^{+}_{p:}(s))] \geq 1-\delta''(i,t)\}. \end{aligned} \tag{2.5.2}$$

The space needed for keeping such histograms is linear in the number of users and buckets. For instance, by setting the latter using the square-root choice, the memory needed is $O(n^{\frac{3}{2}})$. Also, as a consequence of the on-the-fly computation of proximity values, we can easily update the histogram of the seeker by merging the partial, "fresh" histogram obtained in the current run (until termination) with the remaining values from the existing (pre-computed) histogram.

2.5.3. Maintaining the Description of the Proximity Vector

Since social tagging applications are highly dynamic in nature, we need to take into account the fact that the statistics we keep are likely to change quite often. While we can hope that mean, variance and even histogram descriptions are less subject to change than individual proximity values, we should still strive to maintain these statistics as fresh as possible. Recomputing them from scratch, at certain intervals, is an obvious option to consider, though one that may still be too expensive, knowing that we want to avoid keeping the $n \times n$ materialized proximity matrix, as well as naïve re-computation of mean and variance pairs.

A more suitable alternative would be to rely on approximate techniques to maintain fully dynamic all-pairs shortest path information in the network, equivalent to the Dynamic-APSP problem [68]. We give a short description of such an approach in the following.

Since our proximity metric relies on path multiplication, we can reformulate the computation of proximity values into a problem of computing shortest paths in a network with (a) the same set of vertices and edges, and (b) edge weights valued $w(u,v) = -\log \sigma(u,v)$, where $\sigma(u,v)$ is the user proximity from the original network.

A $(2+\epsilon)$-approximate algorithm was given in [11], which handles fully-dynamic updates in a graph in $\tilde{O}(e)$ (almost linear) time. It exhibits a query time of $O(\log\log\log n)$ (the query returns an estimation of the shortest distance between two nodes), without the need of keeping a distance matrix. We could directly rely on this algorithm in the transformed $-\log \sigma(u,v)$ graph. Mapping back the distances thus queried to our

setting would give us an estimation $\sigma^+_{est}(u,v)$ that verifies the inequality:

$$\sigma^+(u,v) \geqslant \sigma^+_{est}(u,v) \geqslant \sigma^+(u,v)^{2+\epsilon}.$$

For a given seeker s, we could thus compute an approximation of its proximity vector in $O(n \log\log\log n)$ time, and then compute the approximate statistics efficiently.

2.6. Scaling and Performance

We argue in this section that, in a real-world setting, our algorithm TOPKS outperforms the one from existing literature [69] both in terms of memory requirements and execution time. We discuss its practical impact in experiments in Section 2.8.

Memory requirements Let us consider, as an illustrating example, one of the most popular bookmarking applications, Del.icio.us, which currently has probably around 10^7 users. Unsurprisingly, this social network is quite sparse, with an average degree of about 100. If a similar graph configuration would be maintained when weights (the σ function) are associated to the edges of the network (e.g., based on tagging proximity or some other measure) the size of an index that would precompute the extended proximity value for each pair of connected users in the network (the σ^+ function) would be roughly of 700 terabytes (i.e., $(10^7)^2 \times 7$ bytes, considering that 3 bytes are necessary for an user Id and 4 bytes are necessary for the float value of proximity). On the other hand, the weighted graph would require memory space of roughly 7 gigabytes (as $10^7 \times 100 \times 7$ bytes), and could easily fit in the RAM space of an average commodity workstation.[5] More, existing techniques for network compression [19] might allow us to reduce the space required to store the network by a factor of $10 - 15$ while still supporting efficient updates and random access on compressed data.

The difference in memory requirements for the two alternatives becomes much more drastic when assuming a user base of the order of Facebook's social network, which currently consists of roughly 9×10^8 users (and is still growing at a fast pace). Precomputed lists for extended proximity go up to about 9 exabytes of memory space, while the network itself requires only about one terabyte.

Performance analysis We next discuss general performance aspects, which in practice may be as impacting as the memory and updatability advantages that our algorithm presents.

[5] We stress that, for the sake of generality, this is not assumed nor exploited in our algorithms, and is not accounted for in the experimental results for TOPKS (in both abstract cost and running time).

Table 2.2.: Computational costs for processing a query Q, when $\alpha = 0$.

Algorithm	Disk access		RAM access
	RA	SA	
ContextMerge	1	n	$(\|Q\| - 1) \times n$
TOPKS$_{\alpha=0}$	0	0	$O(n \lg n + e) + (\|Q\| - 1) \times n + n + e$

Let n denote the number of users and let e denote the number of edges in the network. We assume without loss of generality that the query consists of a single tag (for multiple-tag queries, all dimensions can share the results of a single σ^+ computation).

For our algorithm, let us assume that the social network resides in main memory, e.g., by means of adjacency lists: for each vertex, we have a list of its neighbors and their associated weights (we can safely assume the list comes presorted descending by weight). For one top-k query execution, we will need at most $n + e$ operations to visit the entire network (we are guaranteed to take each vertex only once). For the proximity computation we can use a Fibonacci-heap based max-priority queue, since our graph is likely to be very sparse [61]. Each insertion into the heap takes $O(1)$ amortized time, each extraction takes $O(\lg n)$ and each increase of a key (a relaxation step) takes $O(\lg n)$, for an overall queue complexity of $O(n \lg n + e)$.

ContextMerge requires no computations for proximity at query time. However, it uses disk accesses to read the precomputed proximity values: one random access to locate the seeker's list and n sequential disk accesses to read this list. (It suffices to do this just for one query term, and then keep and access a shared copy of this list in main memory.)

If we value the latency of a memory access as 1 and the one of a sequential disk access as t (usually about five orders of magnitude slower than RAM access), with minor simplifications, our algorithm has the potential to perform better than ContextMerge when the following holds: $t > \lg n + \frac{e}{n}$. So the network sparseness should verify the following inequality:

$$e < n \times (t - \lg n),$$

which is a very plausible assumption in real applications.

A summary of this comparison on execution time is given in Table 2.2. Note that in this analysis we omitted initialization costs: the overhead necessary for ContextMerge to compute σ^+ values for all user pairs and the overhead to load in main-memory the social network, for our algorithm.

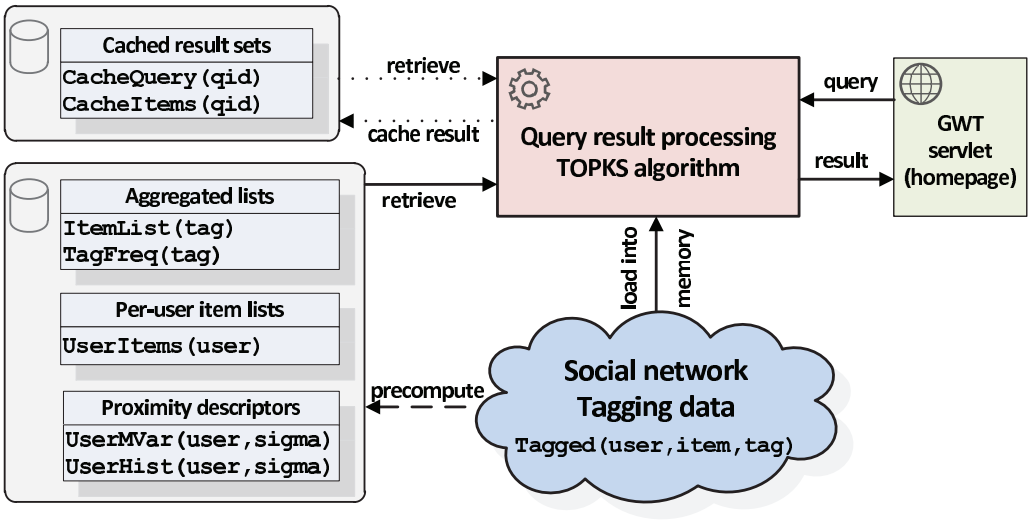


Figure 2.2.: Taagle system architecture.

2.7. System Implementation

Figure 2.2 gives an overview of the general architecture of Taagle, our system implementation of the TOPKS class of algorithms.

Architecture The entry point of the application is a GWT [32] servlet via which users formulate top-k queries, using the various options and parameters available. The queries and corresponding parameters are then sent asynchronously, via RPC calls, to the server-side application, which handles the top-k processing. The server-side application is an implementation of the TOPKS algorithm and its variants. The results are sent back to the client servlet, which displays them along with relevant meta-data (e.g., statistics).

TOPKS uses data from precomputed tables and handles the on-the-fly computation of social proximities. At initialization, it loads the social network into main-memory. The system uses pre-computed projections of the `Tagged` relation: the per-tag inverted lists in `ItemList`, total tag frequencies in `TagFreq` (used for idf values), per-user item lists; it also uses per-user materialized views: the description in `CacheQuery` and the resulting items with their score ranges in `CacheItems`. A per-user item list is only accessed when the algorithm visits the respective user. Finally, the TOPKS approximate version uses a high-level description of each per-user proximity list, stored in `UserMVar` (mean and variance) and `UserHist` (histograms).

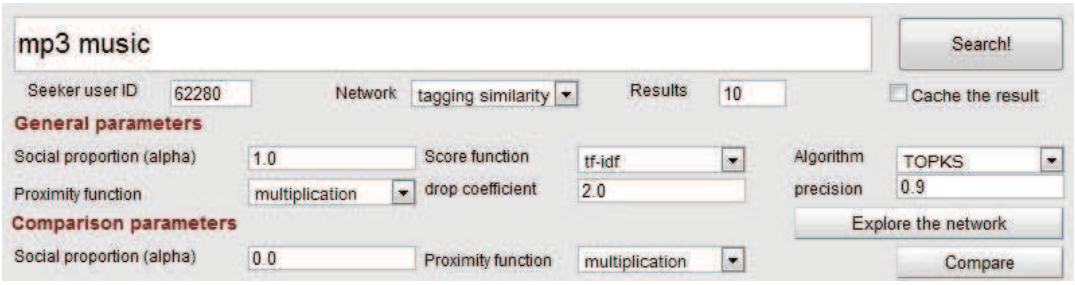


Figure 2.3.: Taagle's search interface.

User experience After logging in, Taagle visitors identify with one of the members of the network, and are able to formulate queries from that perspective (as the seeker). They are able to fully customize their search experience using the interface in Figure 2.3, having access to (i) a selection of ways to compute similarity measures (used for proximity computations) such as tag, item or item-tag similarity, (ii) various proximity aggregation functions (path multiplication, minimum, the parameterized function with drop parameter λ, or enter their own), (iii) tools for comparing the results and the behavior of the various algorithms implemented in the system (TOPKS, the exact algorithm; TOPKS/MVar and TOPKS/Hist, approximate algorithms using high-level descriptions of proximities, controlled by a precision parameter; TOPKS/Views, exact algorithm that may exploit precomputed results of other seekers).

The results window, for which a capture is given in Figure 2.4, presents the top-k items, together with an explanation of the item scores and algorithm statistics: running times, number of users and documents processed, threshold values and termination parameters, interactive plots of the social proximities that were generated and accessed during execution.

2.8. Experimental Results

Dataset and testing methodology We have performed our experiments on a publicly available Del.icio.us dataset [78], containing 80,000 users tagging 595,811 items with 198,080 tags. As this dataset does not give information regarding links between users, we have generated three similarity networks:

- *Item similarity network.* This network was constructed by computing the Dice coefficient of the common items bookmarked by any two users, resulting in a network of 49,038 users and 3,329,540 links.
- *Tag similarity network.* This network was generated by computing the Dice coefficient of the common tags used by any two users. Since this computation results

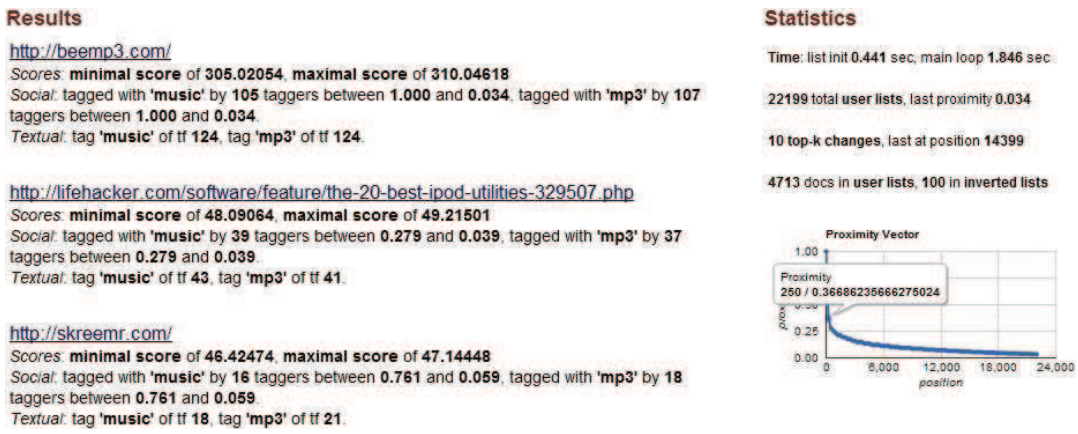


Figure 2.4.: Results page of Taagle.

in a network that is too dense, we have filtered out the users who used less than 10 distinct tags in their tagging activity. The final networks thus contains 40,319 users and 8,335,544 links.

- *Item-tag similarity network.* This network was constructed by computing the Dice coefficient of the common items and tags bookmarked by any two users, resulting in a network containing 40,353 users and 1,849,898 links.

We computed the top-10 and top-20 answers, generating a number of 20 two and three-tag semantically coherent queries, from tags that have a medium frequency (i.e., between 3,000 and 5,000 in our dataset). For each similarity network, 10 random users were also randomly chosen in the role of the seeker.

Testing was performed using two ranking functions (the h-function from our model). The first one is the standard tf-idf ranking function:

$$score(i \mid u,t) = fr(i \mid u,t) \times idf(t).$$

The second one is the BM15 ranking function used in [69]:

$$score(i \mid u,t) = \frac{(k1+1)fr(i \mid u,t)}{k1 + fr(i \mid u,t)} \times idf(t),$$

where inverse frequency $idf(t)$ is defined in standard manner as

$$idf(t) = \log \frac{|\mathcal{I}| - |\{i \mid Tagged(v,i,t)\}| + 0.5}{|\{i \mid Tagged(v,i,t)\}| + 0.5}.$$

and the aggregation function g is summation.

While these are two of the most commonly used ranking functions in IR literature, they have different properties when used in approximate approaches as the ones we

describe. More precisely, since tf-idf is a linear function, both the maximal and minimal estimates over fr scores lead to valid estimates for the overall scores. This is not necessarily the case for BM15: since it is a concave function, only the maximal overall score can be estimated. This was taken into account in the experiments.

We used a Java implementation of our algorithms, on a machine with a 2.8GHz Intel Core i7 CPU, 8GB of RAM, running Ubuntu Linux 10.04 and PostgreSQL 9.0.

As our focus is on optimizing the social branch of the top-k retrieval, we report here our results for $\alpha \in \{0, 0.1, 0.2, 0.3\}$. As [69], multiplication over the paths was chosen as the proximity aggregation function, as the best suited candidate for predicting implicit similarities.

Remark. The relevance of personalized query results is a topic that has been extensively treated ([77, 27, 48]). The relevance of social search results was also extensively evaluated in [69], over Del.icio.us data, in a setting (including ranking model) similar to ours. While it is not our focus, we give an experimental evaluation of relevance using two ground truth experiments in this section.

Efficiency results For the testing environment described previously, we report on efficiency for both exact and approximate algorithms, and on precision for the latter.

For efficiency, we report on two measures: the abstract cost of the algorithms and their wall clock running times. Abstract cost, which is the standard measure for early-termination algorithms that depend on database accesses, is computed as defined in Section 2.4.1, by choosing c_{UL}, the cost of accessing a user lists, as valued 100 (a very conservative upper-bound), and c_S, the cost of sequentially accessing an item in an inverted lists, valued 1. More formally,

$$cost(A, D) = 100 \times users(A, D) + seqitems(A, D).$$

We ignore differences in favor of TOPKS that are hard to account for, namely we do not distinguish between the user accesses by ContextMerge (which in a real setting would be to external memory) and the ones by TOPKS (which would be to main memory).

Figures 2.5, 2.6 and 2.7 present the comparison of abstract costs and running times for the BM15 and tf-idf ranking functions, for each of the three similarity networks. In each subfigure, the first pair of columns gives the abstract cost of [69]'s ContextMerge algorithm, the second pair of columns the one of TOPKS, the third pair of columns the cost of TOPKS/*MVar* (approximate approach based on mean and variance of proximities, described in Section 2.5.1) and the fourth pair of columns the cost of TOPKS/*Hist* (approximate approach based on histograms, described in Section 2.5.2). For each algorithm, the average running times were recorded, and are represented by the black line in the plots (one dot indicates the average running time between the top-10 and

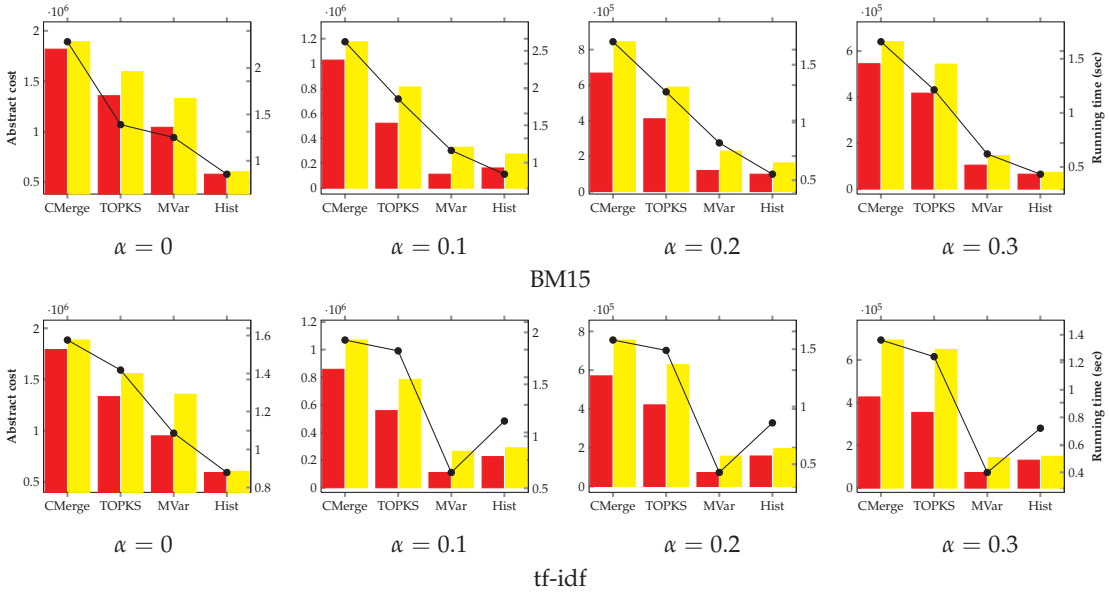


Figure 2.5.: Abstract cost and running time comparison over the tag-similarity network and the f_{mul} proximity function (red: top-10, yellow: top-20).

the top-20). One can notice there that abstract cost closely captures the actual performance of the algorithms. However, running time optimization was not the focus of the present work, and many alternatives remain to be explored in that direction (e.g., tuning the database).[6]

First, we can see that in general TOPKS drastically improves efficiency when compared to ContextMerge, in terms of both running time and abstract cost. For example, in the item-tag similarity network, when $\alpha = 0$, the running time and abstract cost are around 50% of that of ContextMerge.

Moreover, our approximate approaches lead to further improvements, which support the intuition that even limited statistics (such as mean and variance) can render the termination conditions more tight.

The abstract costs of TOPKS/*MVar* and TOPKS/*Hist* in the figure were obtained for the probabilistic threshold $\delta = 0.9$. Even though this represents a quite weak guarantee, we found that it still yields a good precision/efficiency trade-off. For a better understanding of this trade-off, we show in Figure 2.8 the impact of δ on precision. When $\alpha > 0$, visiting the per-term inverted lists in parallel to the proximity vector helps in deriving tighter score bounds for unseen items, leading to a faster termination of the approximate approaches. These tighter score bounds also help in achieving

[6]Note that we cannot compare with [4]'s approach, as it only extends classic top-k retrieval by interpreting user proximity as a binary function (0-1 proximity), by which only users who are directly connected to the seeker can influence the top-k result.

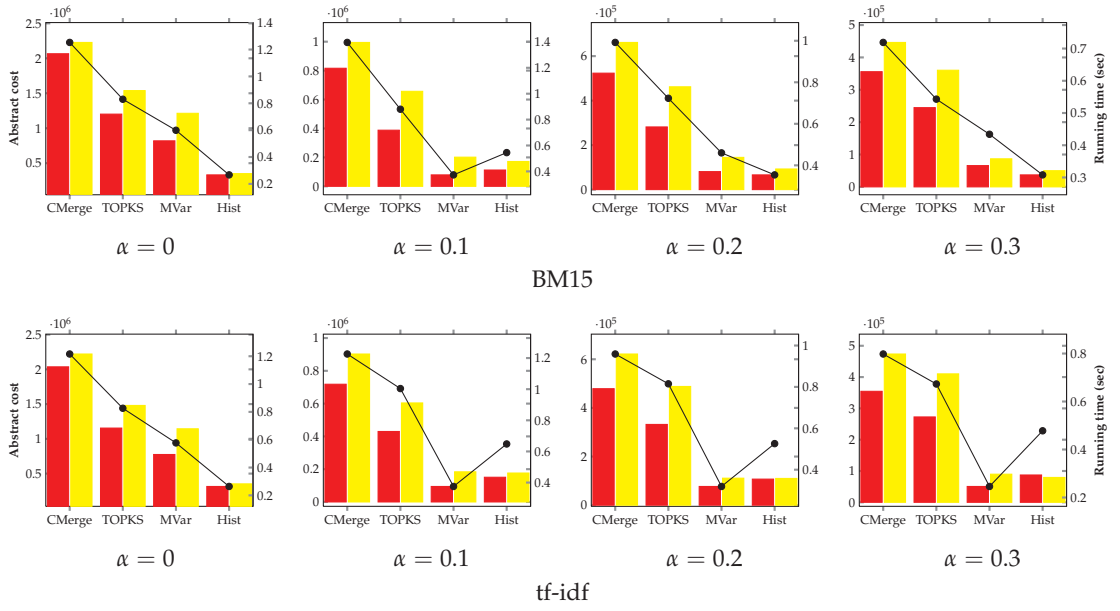


Figure 2.6.: Abstract cost and running time comparison over the item-similarity network and the f_{mul} proximity function (red: top-10, yellow: top-20).

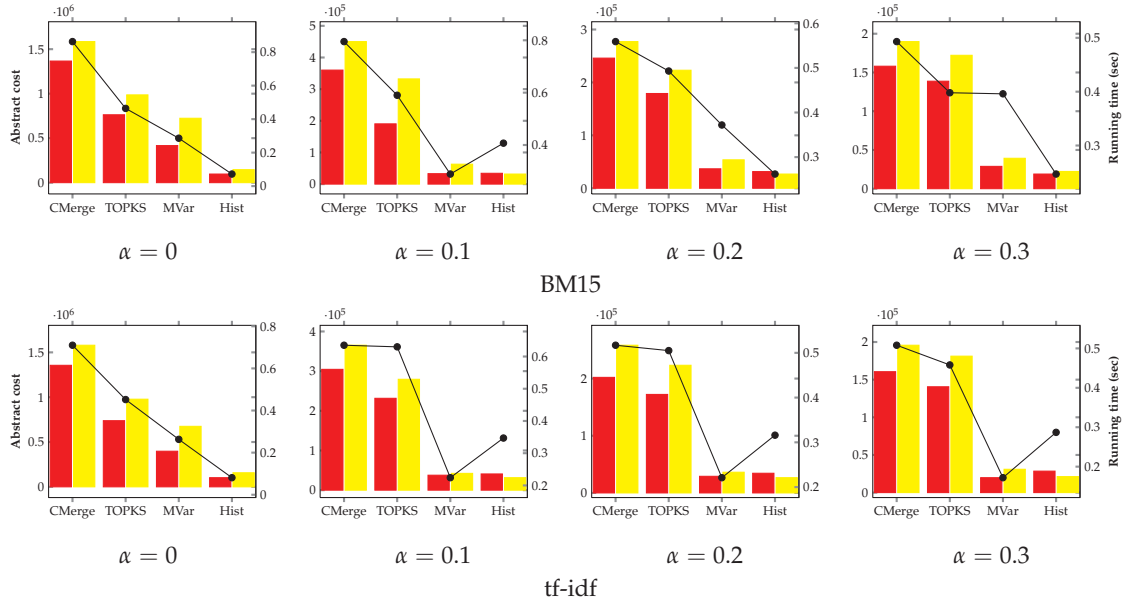


Figure 2.7.: Abstract cost and running time comparison over the item-tag similarity network and the f_{mul} proximity function (red: top-10, yellow: top-20).

Table 2.3.: Comparison between ContextMerge and TOPKS$_{\alpha=0}$.

Network	ContextMerge		TOPKS$_{\alpha=0}$	
	users	seqitems	users	seqitems
item	21878	0	15588	65
item-tag	13028	0	6898	54
tag	18718	0	15581	68

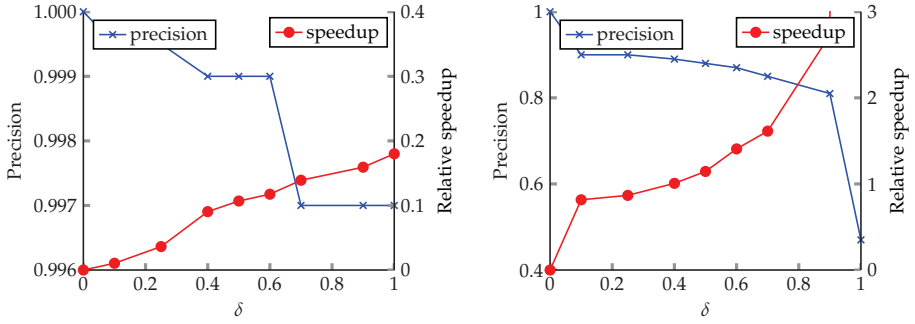


Figure 2.8.: Precision rates versus speedup relative to TOPKS, when $\alpha = 0$.

better precision levels when $\alpha > 0$, as Figure 2.9 shows.

Furthermore, our branch choice heuristic in TOPKS (in both the exact and approximate variants) brings significant improvements overall (for instance, consider the difference between the cost savings for $\alpha = 0$ and $\alpha = 0.1$, in the tag similarity network). Finding even more effective heuristics for this aspect of the algorithm remains an interesting direction for future research.

We discuss next how the instance optimality of TOPKS$_{\alpha=0}$ reflects in the performance results. Table 2.3 reports the number of visited users by ContextMerge and TOPKS$_{\alpha=0}$ (columns *users*), for the three similarity networks. One can see that TOPKS$_{\alpha=0}$ achieves good savings (in terms of visited users), while relying only on very few sequential accesses in the inverted lists (column *seqitems*).

Finally, we consider the impact of the probabilistic parameter δ on precision and speedup in the approximate algorithms. We define precision as the ratio between the size of the exact result (by TOPKS) and the number of common items returned by the respective approximate approach and TOPKS, i.e.,

$$precision = \frac{|T_{\text{TOPKS}/app} \cap T_{\text{TOPKS}}|}{|T_{\text{TOPKS}}|},$$

where $T_{\text{TOPKS}/app}$ is the set of items returned as top-k by the approximate algorithms

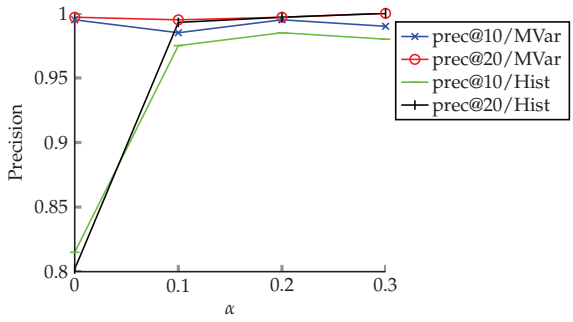


Figure 2.9.: Precision rates vs. α, when $\delta = 0.9$.

(either TOPKS/*MVar* or TOPKS/*Hist*), and T_{TOPKS} is the set of items returned by the exact algorithm.

The relative speedup is defined as

$$speedup = \frac{cost(\text{TOPKS}, D)}{cost(\text{TOPKS}/app, D)} - 1.$$

We present in Figure 2.8 the results for both approximate approaches, TOPKS/*MVar* and TOPKS/*Hist*. For TOPKS/*MVar*, one can notice that δ has a limited influence on precision (with a minimum of 0.997 for $\delta = 1$), while ensuring reasonable speedup. The speedup potential is greater when using TOPKS/*Hist* and histograms, while reasonable precision levels are obtained (for instance, precision of around 0.805 when $\delta = 0.9$, for a speedup of around 2.5). For values of $\delta > 0.9$, we notice however a rapid drop in precision. The fact that *MVar* achieves better precision than *Hist* may seem counter-intuitive, since histograms give a more detailed description of proximity vectors. This difference in precision is due to looser bounds for *MVar*, as they directly influence the termination condition of the algorithm, result in a longer run and hence to better chances of returning a more refined top-k results.

We also considered the influence of the α parameter on precision, while setting the probabilistic parameter to $\delta = 0.9$ (see Figure 2.9). We have measured both *precision*@10 (i.e., when requesting the top-10) and *precision*@20 for both TOPKS/*MVar* and TOPKS/*Hist*. We observed that the precision levels for TOPKS/*MVar* are quite stable for all values of α. For TOPKS/*Hist*, the lowest values of precision are witnessed when $\alpha = 0$, but they stabilize to high values (above 0.97) for $\alpha > 0$.

Evaluating relevance We report now on two "ground-truth" experiments we have performed to test the bookmark prediction power of the exclusively social queries.

For the first experiment, we have selected $(user, tag)$ pairs from users that have bookmarked between 5 and 10 items using tags that were used globally at least 1000

times. The objective of this experiment was to estimate the power of personalized results to predict items that are tagged using relatively popular tags.

Then, 1000 pairs were randomly selected. For each pair and for $k \in \{1, 2, 5, 7, 10\}$, we computed the following top-k result, using as query the tags corresponding to the distinct user-ids: the network-unaware top-k, and, setting the user-id as the seeker, the personalized top-k (for $\alpha = 0$) for each of the following aggregation functions: f_{mul}, f_{min}, and f_{pow} with $\lambda \in \{1.1, 2\}$. For each personalized query, the items belonging to the seeker were ignored (so as not to influence positively the precision of the results).

An item was considered as "predicted" if it appeared in the resulting top-k and was also tagged by the seeker userid with the query tags. We traced the proportion of pairs for which at least one such item has been predicted.

The results are presented in Figure 2.10. One can note that, for the item and item-tag similarity networks, personalization is considerably better at predicting bookmarked items than the "global" top-k, for all functions, except f_{min} and, to a lesser extent, f_{pow} ($\lambda = 1.1$). Moreover, the tag similarity network seems to not be such a good predictor, no matter the personalization function used, as the other two networks. This might indicate the fact that, in the case of tag similarity, one needs to go beyond simple set similarities and include more complex relationships between tags, like synonymy and polysemy.

For the second experiment, we have selected $(user, item, tag)$ triples resulting from items that have been tagged only by few people in the network (between 5 and 10). The objective of this experiment was to estimate the power of personalizing results to predict items that are unpopular, i.e., the "long tail". The tests and the measures tracked are identical to the setup of the first experiment.

The results are presented in Figure 2.11. They are similar to a good extent to those of the first experiment, with two main differences: (i) personalization fails completely in the tag similarity network, (ii) in the item and item-tag similarity networks personalization achieves considerably higher prediction performance than in the case of predicting items tagged with popular tags. This is because, in the case of long-tail items, the functions that are skewed towards the closest users, i.e., f_{mul} and $f_{pow}, \lambda = 2$, will rank higher the items belonging to the closest users.

2.9. Conclusions

We considered in this chapter top-k query answering in social bookmarking applications, proposing algorithms that have the potential to scale in real applications, in an online context where the social network, the tagging data and even the seekers' search ingredients can change at any moment. Our solutions address the main drawbacks of previous approaches. With respect to applicability and scalability, we avoid expen-

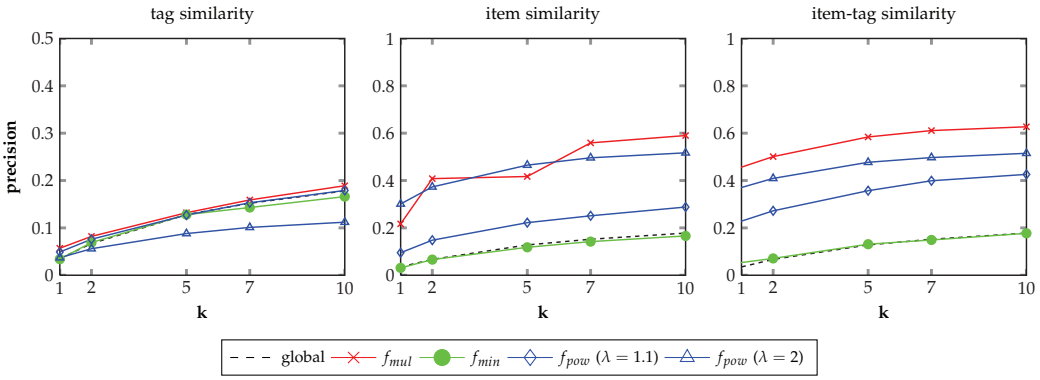


Figure 2.10.: Predicting bookmarks tagged with semi-popular tags.

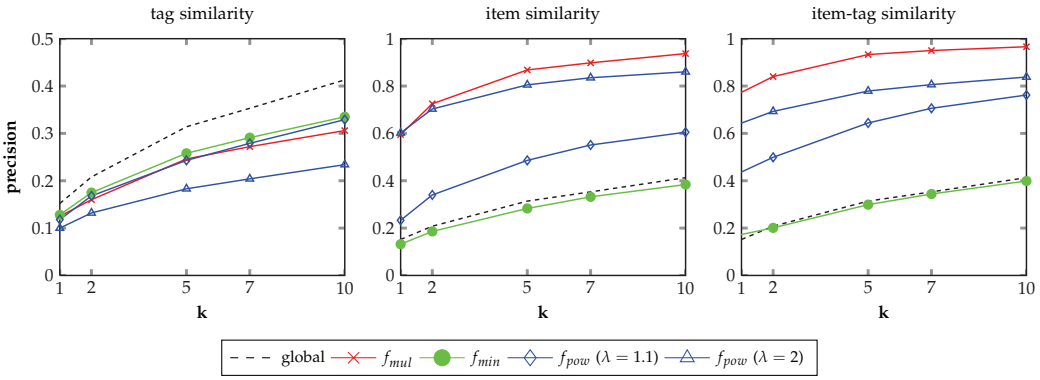


Figure 2.11.: Predicting unpopular bookmarks.

sive and hardly update-able pre-computations of proximity values, by an on-the-fly approach. We show that it is applicable to a wide family of functions for proximity computation in a social network. With respect to efficiency, we show that TOPKS is instance optimal in the exclusively social context and, via extensive experiments, that it performs significantly better than the algorithm from previous literature. We also considered widely-applicable approximate techniques, showing they have the potential to drastically reduce computation costs, while exhibiting high accuracy.

3. Context-Aware Search Using Views

In this chapter, we consider the problem of using precomputed results – that will be called *views* – for answering context-aware top-k queries.

In context-aware scenarios, unlike scenarios in which queries do not depend on a context or are computed in the same context, in order to be able to answer top-k queries originating at arbitrary users (in the case of social search) or locations (in the case of spatial search), the views – and thus the object scores contained in them – need to be *transposed* to the context of the seeker (the query initiator). Revisiting the example presented in Chapter 1, for spatial search and a query issued in *Paris*, a result precomputed in the *Versailles* context is more useful than one computed in the *Berlin* context. However, the results from the *Versailles* perspective have to be adapted to the parisian one. Therefore, even the closest views will contain after the transposition step lists of objects having non-exact (or uncertain) scores, in the form of lower and upper bounds. Consequently, a new query semantics and tailored algorithms for dealing with objects having uncertain scores are needed. In particular, a reliable approach that *selects* a subset of the views that are most promising, towards minimizing the uncertainty of the resulting top-k, is needed.

We detail these problems and a novel query semantics in Section 3.2, and present the context-aware adaptations of the threshold algorithms TA and NRA in Section 3.3. As these algorithms can only output uncertain answers – in terms of objects guaranteed to be in the top-k and objects that might belong to it – in Section 3.4 we discuss one approach to find the most probable top-k result, by using sampling.

We introduce our formulation of the view selection problem in Section 3.5, followed by the study of the instance optimality properties of the adaptations of TA and NRA, in Section 3.6.

We bridge this formal study to practical applications by detailing how to transpose the context in two important scenarios, location-aware keyword search and social-aware search, in Sections 3.7 and 3.8.

Finally, view selection and the two context-aware search scenarios are evaluated experimentally in Section 3.9.

3.1. Related Work

In this section, we overview the related work in the areas of top-k processing algorithms, top-k processing using views and location-aware keyword search. The main related work is presented in more detail after the model is described, in Section 3.2.

Data structures and algorithms for top-k query processing The efficient processing of top-k queries is an important research topic in recent IR literature, leading to advancements in both data structures and query processing algorithms. The most common data structure for top-k processing is the inverted index file (for a general survey on indexing for top-k processing see [84]). For this data structure, a key challenge is to optimize response time, mainly via index compression techniques [81, 83, 70, 85].

In terms of algorithms, among the most widely cited and used are the early termination threshold algorithms TA and NRA of [31], which provide a very important property of top-k algorithms: that of *instance optimality*. In short, instance optimal algorithms are as efficient as any other algorithm from its class – within a constant factor – over all possible inputs. Many other top-k aggregation algorithms have been proposed in the literature, both for XML [57, 17] and relational databases, for which we refer the interested reader to the excellent survey of Ilyas et al. [43] and the references therein.

Using views for top-k query processing The use of precomputed results, either as previous answers to queries [23, 42] or as cached intersection lists [49], has been identified as an important direction for efficiency. A linear programming formulation of the computation of scores over views is first introduced in [23] and extended in [49]. In [72], the authors study top-k processing when only score ranges are known, instead of exact ones, define a probabilistic ranking model based on partial orders and introduce several semantics for ranking queries, but do not deal with aggregation of uncertain scores over multiple dimensions.

Location-aware keyword search In the area of location-aware retrieval, Cong et al. [22] introduce the concept of LkT queries, for which they include in the ranking model both the distance of a document's location relative to the query point, and the textual features of the document. They propose the IR-tree index, consisting of an R-tree [36] in which each node has an inverted list of relevant documents. Other models for top-k location-aware keyword querying have been proposed, for selecting either groups of objects that collectively satisfy a query [15], or the k-best objects scored by the features in their neighborhood [67], or the top-k objects in a given query rectangle [20]. Various approaches for combining textual inverted lists and spatial indexes for keyword retrieval were also studied in [20].

3.2. Formal Setting and Problems

Context-aware score model. We assume a finite collection of objects $\mathcal{O}$ and a countable collection of attributes $\mathcal{T}$. Under a given *context parameter* $\mathcal{C}$ – an application-dependent notion – objects o are associated to certain attributes t, by an object-attribute score function $sc(o, t \mid \mathcal{C})$.

Under a context $\mathcal{C}$, a query Q consists of a set of attributes $\{t_1, \ldots, t_n\}$; its answer is given by objects $o \in \mathcal{O}$ having the highest scores $sc(o, Q \mid \mathcal{C})$, computed via a monotone aggregation function h (e.g., `sum`, `max`, `avg`) over the object-attribute scores:

$$sc(o, Q \mid \mathcal{C}) = h(sc(o, t_1 \mid \mathcal{C}), \ldots, sc(o, t_n \mid \mathcal{C})).$$

We can formalize the top-k retrieval problem as follows:

Problem 1. *Given a query $Q = \{t_1, \ldots, t_n\} \subset \mathcal{T}$, a context $\mathcal{C}$, an integer k, and a score model specification (sc, h), retrieve the k objects $o \in \mathcal{O}$ having the highest scores $sc(o, Q \mid \mathcal{C})$.*

In certain applications, the context may always be empty or may simply be ignored in the sc scores, and, when necessary, we indicate this in our notation by the '$\perp$' context. We use $sc(o, Q)$ as short notation for $sc(o, Q \mid \perp)$.

Threshold algorithms We revisit here the class of early termination top-k algorithms known as *threshold algorithms*. These algorithms, applicable in a context-agnostic setting, find the top-k objects for an input query Q by scanning sequentially (for each attribute) and in parallel (for the entire attribute set of Q), relevant per-attribute lists that are ordered descending by sc values – with inverted lists being a notable example – denoted in the following $L(t)$, as the list for attribute t. During a run, they maintain a set D of already encountered candidate objects o, bookkeeping for each candidate the following values:

1. an upper-bound on $sc(o, Q)$, the best possible score that may still be obtained for o, denoted hereafter $bsc(o, Q)$,
2. a lower-bound on $sc(o, Q)$, the worst possible score, denoted hereafter $wsc(o, Q)$,

with the objects being ordered in D by their worst scores.

At each iteration, or at certain intervals, threshold algorithms may refine these bounds and compare the worst score of the kth object in D, $wsc(D[k], Q)$, with the best possible score of either (i) objects o in D outside the top k, $bsc(o, Q)$, or (ii) not yet encountered objects, denoted $bsc(*, Q)$.

When both these best scores are not greater than the worst score of $D[k]$, the run can terminate, outputting the objects $D[1], \ldots, D[k]$ as the final top-k.

A key difference between the various threshold algorithms, and in particular between TA and NRA, resides in the way they are allowed to access the per-attribute lists. TA is allowed random accesses to lists, as soon as an object o has been encountered while sequentially accessing one list among the $|Q|$ relevant ones. These random accesses can complete the scores of objects o from a guaranteed range to an exact value. In comparison, NRA is not allowed to use random accesses in the per-attribute lists, but only sequential ones, and each object o in the final top-k may only be given a score range, $[wsc(o, Q), bsc(o, Q)]$. (Various hybrid algorithms with respect to TA and NRA are also possible and have been extensively studied in the literature.)

TA was shown to be instance optimal among algorithms that do not make "wild guesses" or probabilistic choices. Within this same class of algorithms, NRA was shown to be instance optimal for algorithms in which only *sequential* accesses are allowed.[1]

Views and precomputed results We extend the classic top-k retrieval setting of TA/NRA by assuming access to precomputed query results, called in the following *views*. Each view V is assumed to have two components: (i) a *definition*, $def(V)$, which is a pair query-context $def(V) = (Q^V, \mathcal{C}^V)$ and (ii) a set $ans(V)$ of triples (o_i, wsc_i, bsc_i), representing the *answer* to query Q^V under context $\mathcal{C}^V$. Each such triple says that object o_i has a score $sc(o_i, Q^V \mid \mathcal{C}^V)$ within the range $[wsc_i, bsc_i]$.

Since we are dealing with cached query results, all objects not appearing in $ans(V)$ – represented explicitly in $ans(V)$, to simplify presentation, by one final *wildcard* $*$ *object* – have with respect to query Q^V and context $\mathcal{C}^V$ a worst score of $wsc_* = 0$ and a best possible score of either $bsc_* = \min\{wsc_i \mid (o_i, wsc_i, bsc_i) \in ans(V)\}$, if V's result is complete, in the sense that enough objects had a non-zero score w.r.t. Q^V, or otherwise 0.

Context transposition Intuitively, when a view V and the to-be-answered query Q do not share the same context, a transposition of the exact scores or score ranges in $ans(V)$ is necessary, in order to obtain valid ranges for $sc(o_i, Q^V \mid \mathcal{C})$ from those for $sc(o_i, Q^V \mid \mathcal{C}^V)$. In particular, in the case of spatial or social search, this transformation will inevitably yield a coarser score range. We will detail the specific operation of context transposition for these two application scenarios in Section 3.7.

Exploiting views. Given an input query Q and a context $\mathcal{C}$, from a set of views $\mathcal{V}$ sharing the same context – as in $def(V) = (\ldots, \mathcal{C})$ – a first opportunity that is raised by the ability to cache results is to compute for objects $o \in \mathcal{O}$ tighter lower and upper

[1] This is an important family of algorithms for performance, given that random accesses can be orders of magnitude more costly.

$V_1(\{a\}, \perp)$			$V_2(\{c\}, \perp)$			$V_3(\{a,b\}, \perp)$			$V_4(\{b,c\}, \perp)$		
o	**ws**	**bs**	**o**	**ws**	**bs**	**o**	**ws**	**bs**	**o**	**ws**	**bs**
o3	7	8	o3	8	8	o5	16	16	o5	11	11
o5	6	7	o4	3	7	o6	10	10	o3	10	11
o6	4	4	o6	2	4	o3	10	10	o6	9	11
o7	3	5	o10	2	3	o10	6	9	o10	8	9
o9	3	4	o7	2	2	o2	6	6	o2	6	7
o10	1	3	o8	1	3	o7	6	6	o1	5	7
o2	1	2	o9	1	2	o1	5	8	o7	4	4
o1	1	1	o5	1	1	o9	4	5	o4	3	8
$*$	0	1	$*$	0	1	$*$	0	4	$*$	0	3

Table 3.1.: An example set $\mathcal{V}$ of views.

bounds over $sc(o, Q \mid \mathcal{C})$). This may be useful in threshold algorithms, as a way to refine score ranges. We formalize this task next.

Problem 2. *Given a query $Q = \{t_1, \ldots, t_n\} \subset \mathcal{T}$, a context $\mathcal{C}$, an integer k, a score model specification (sc, h) and a set of views $\mathcal{V}$ sharing the same context with Q, given an object $o \in \mathcal{O}$, compute the tightest lower and upper bounds on $sc(o, Q \mid \mathcal{C})$ from the information in $\mathcal{V}$.*

In this work, consistent with the most common ranking models for context-aware search, we will assume that the aggregation function h is summation. Under this assumption, Problem 2 could be modeled straightforwardly by the following mathematical program, whose variables are given in bold:

$$\min \sum_{t_i \in Q} \mathbf{sc}(\mathbf{o}, \mathbf{t_i} \mid \mathcal{C}) \tag{3.2.1}$$

$$\max \sum_{t_i \in Q} \mathbf{sc}(\mathbf{o}, \mathbf{t_i} \mid \mathcal{C}) \tag{3.2.2}$$

$$wsc \leq \sum_{t_j \in Q^V} \mathbf{sc}(\mathbf{o}, \mathbf{t_j} \mid \mathcal{C}),\ \forall V \in \mathcal{V}\ s.t.\ (o, wsc, bsc) \in ans(V)$$

$$\sum_{t_j \in Q^V} \mathbf{sc}(\mathbf{o}, \mathbf{t_j} \mid \mathcal{C}) \leq bsc,\ \forall V \in \mathcal{V}\ s.t.\ (o, wsc, bsc) \in ans(V)$$

$$\mathbf{sc}(\mathbf{o}, \mathbf{t_l} \mid \mathcal{C}) \geq 0, \forall t_l \in \mathcal{T}$$

Example 7. *Let us consider the views in Table 3.1. We have access to the results of four views, defined by the sets of attributes $\{a\}$, $\{c\}$, $\{a,b\}$ and $\{b,c\}$. We assume the empty context $\mathcal{C} = \perp$ for the views and for the to-be-answered query, which is $Q = \{a,b,c\}$. Considering o6, for example, we know that:*

$$
\begin{aligned}
sc(o6, \{a\}) &\geq 4 && (V_1)\\
sc(o6, \{c\}) &\geq 2 && (V_2)\\
sc(o6, \{a\}) + sc(o6, \{b\}) &\geq 10 && (V_3)\\
sc(o6, \{b\}) + sc(o6, \{c\}) &\geq 9 && (V_4)\\
sc(o6, \{a\}) &\leq 4 && (V_1)\\
sc(o6, \{c\}) &\leq 4 && (V_2)\\
sc(o6, \{a\}) + sc(o6, \{b\}) &\leq 10 && (V_3)\\
sc(o6, \{b\}) + sc(o6, \{c\}) &\leq 11 && (V_4)
\end{aligned}
$$

Then, the lower bound on $sc(o6, Q)$ *is obtained as also*

$$wsc(o6) = \min(sc(o6, \{a\}) + sc(o6, \{b\}) + sc(o6, \{c\})) = 13$$

by combining the worst scores of V_1 *and* V_4*. Similarly, the upper bound on* $sc(o6, Q)$ *is obtained as*

$$bsc(o6, Q) = \max(sc(o6, \{a\}) + sc(o6, \{b\}) + sc(o6, \{c\})) = 14$$

by combining the best scores of V_2 *and* V_3*.*

We now formulate the problem of answering input top-k queries Q using *only* the information in views, whose semantics needs to be adapted to the fact that views may offer only a partial image of the data. When an exact top-k cannot be extracted with full confidence, a *most informative result* would consist of two disjunctive, possibly-empty sets of objects from those appearing in $\mathcal{V}$'s answer:

- a set of all the objects guaranteed to be in the top-k for Q
- a set of all objects that may also be in the top-k for Q.

Problem 2 provides a way to properly define and identify objects of the former kind – the *guaranteed ones* – as the objects o_x for which

$$\min \sum_{t_i \in Q} sc(o_x, t_i \mid \mathcal{C}) \geq \max \sum_{t_i \in Q} sc(*, t_i \mid \mathcal{C}) \tag{3.2.3}$$

and at most $k - 1$ objects o_y can be found such that

$$\min \sum_{t_i \in Q} sc(o_x, t_i \mid \mathcal{C}) < \max \sum_{t_i \in Q} sc(o_y, t_i \mid \mathcal{C}). \tag{3.2.4}$$

Similarly, we can identify objects of the latter kind – the *possible ones* – as the objects o_x that are not guaranteed and for which at most $k-1$ objects o_y can be found such that

$$\min \sum_{t_i \in Q} sc(o_y, t_i \mid \mathcal{C}) > \max \sum_{t_i \in Q} sc(o_x, t_i \mid \mathcal{C}). \tag{3.2.5}$$

We formalize the *top-k retrieval problem using views* as follows.

Problem 3. *Given a query $Q = \{t_1, \ldots, t_n\} \subset \mathcal{T}$, a context $\mathcal{C}$, an integer k, and a score model specification (sc, h), given a set of views $\mathcal{V}$ sharing the same context with Q, retrieve from $\mathcal{V}$ a* most informative answer *of the form (G, P), with*

- *$G \subset \mathcal{O}$ consisting of all* guaranteed objects *(as in Eq. (3.2.3) and (3.2.4), when h is summation); they must be among those with the k highest scores for Q and $\mathcal{C}$.*
- *and $P \subset \mathcal{O}$ consisting of all* possible objects *outside G (as in Eq. (3.2.5), when h is summation); they may be among those with the k highest scores for Q and $\mathcal{C}$, i.e., there exist data instances where these appear in the top-k.*

In order to solve Problem 3, a naïve computation of upper and lower bounds for all objects o appearing in the views would suffice, but would undoubtedly be too costly in practice. Instead, we show in Section 3.3 how we can solve Problem 3 in the style of threshold algorithms, by extending NRA and TA.

Over any data instance, the exact top-k can be thought of as the set G plus the top-k' items from P, for $k' = k - |G|$. To give a *most likely result*, in a probabilistic sense, based on the G and P object sets, we discuss in Section 3.4 possible approaches for estimating the probability of possible top-k' sets from P.

Going further, even when the most promising candidate objects are considered first in SR-TA or SR-NRA, their corresponding instances of the mathematical programs in Eq. (3.2.1) and Eq. (3.2.2) may still be too expensive to compute in practice (even when we are dealing with LPs, as in Example 7): the set of views may be too large – potentially of the order $2^{|\mathcal{T}|}$ – and each view contributes one constraint in the program. In our best-effort approach, which would first *select some (few) most promising views* $\tilde{\mathcal{V}} \subset \mathcal{V}$ for the input query (Section 3.5), we are presented a trade-off between the size of the subset $\tilde{\mathcal{V}}$ – which determines the cost of the top-k algorithms SR-NRA and SR-TA – and the "quality" of the result, namely its distance with respect to the most informative answer given by all the views. We quantify the distance between the most informative result by $\tilde{\mathcal{V}}$, denoted $(\tilde{G}, \tilde{P})$, and the most informative answer (G, P) by $\mathcal{V}$ as the difference in the number of possible top-k combinations:

$$\Delta = \binom{|\tilde{P}|}{k - |\tilde{G}|} - \binom{|P|}{k - |G|}. \tag{3.2.6}$$

We also show in Section 3.5 how a final refinement step over $(\tilde{G}, \tilde{P})$, based on random accesses in the entire $\mathcal{V}$ set, allows us to reach $\Delta = 0$, i.e., the most informative result by $\mathcal{V}$.

Main related work We have already introduced TA and NRA, as reference algorithms for early-termination top-k retrieval. Techniques for top-k answering using views have been proposed in recent literature: the LPTA algorithm [23] and generalizations of the NRA and TA algorithms [49], both applicable in settings where the aggregation functions are linear combinations of the per-attribute scores. These two approaches make however significant simplifying assumptions: (i) the scores in each view are assumed exact, i.e., $wsc_i = bsc_i$, $\forall(o_i, wsc_i, bsc_i) \in ans(V)$, $\forall V \in \mathcal{V}$, and (ii) one can compute the score for the input query Q by composing (in a predefined way) the scores from a subset of the views that are selected in advance (usually the single-attribute inverted lists). In comparison, we consider also views that may only give score ranges instead of exact scores and ranks in the top-k result. In our more general setting, the exact top-k may not be obtainable will full confidence and, instead of simply refuting input queries that cannot be fully answered, we describe algorithms that can support a more general type of result from the views, in terms of guaranteed and possible objects for the top-k.

Regarding view selection, the work closest in spirit to ours is [23], whose focus is on finding the optimal top-k execution based on a selection of precomputed views, with all the per-attribute lists being assumed to be part of the view space. Their approach simulates the run of a threshold algorithm over histograms of views. The setting of [23] is fundamentally different from ours. First, any viable selection of views must output the exact, ranked top-k result, which represents a strong limitation for practical purposes. Therefore, while their focus is on optimizing the top-k computation, we are confronted with a different perspective over the view selection problem, towards minimizing the uncertainty of the result, in terms of Eq.(3.2.6)'s distance measure.

3.3. Threshold Algorithms

We start this section by presenting our adaptation of TA, called SR-TA, which can be applied when the input lists consist of objects with score ranges; SR-TA will allow us to solve Problem 3.

Each of the input lists are assumed to be available in two copies, one ordered descending by the score lower-bound and one ordered descending by the score upper-bound. SR-TA will read sequentially in round-robin manner from the former group of lists and, similar to TA, maintains a candidate set D of the objects encountered during the run. At each moment, the read heads of the latter group of lists must give objects

that are not yet in D (unseen objects), and sequential accesses are performed in SR-TA whenever necessary in order to maintain this configuration.

D is also ordered descending by the score lower-bounds. The algorithm stops when the score of any of the unseen objects – the threshold τ – cannot be greater than the one of the kth object in the candidate set D.

In our setting, the threshold τ is obtained as the solution of the following mathematical program, taking into account from each view V the score upper-bound of objects from $ans(V) - D$:

$$\begin{aligned} &\tau = \max \sum_{t_i \in Q} \mathbf{sc}(\mathbf{o}, \mathbf{t_i} \mid \mathcal{C}) \qquad (3.3.1)\\ &\sum_{t_j \in Q^V} \mathbf{sc}(\mathbf{o}, \mathbf{t_j} \mid \mathcal{C}) \leq max(bsc_i),\ \forall V \in \mathcal{V},\ o_i \notin D\ s.t. \\ &\qquad\qquad\qquad\qquad\qquad\qquad (o_i, wsc_i, bsc_i) \in ans(V) \\ &\mathbf{sc}(\mathbf{o}, \mathbf{t_l} \mid \mathcal{C}) \geq 0, \forall t_l \in \mathcal{T} \end{aligned}$$

One can note that when (i) we have only views that give answers to singleton queries, and (ii) the $wsc_i = bsc_i$ for each object o_i (i.e., the lists contain exact scores), we are in the setting of the TA family of algorithms over inverted list inputs. Relaxing condition (i), we have the setting of top-k answering using views investigated in [49, 23]. Both these settings and their corresponding algorithms can guarantee that, at termination, the exact top-k is returned.

Our more general setting, however, cannot provide such guarantees, as witnessed by the following example.

Example 8. *Let us revisit Example 7, for the top-5 query $Q = \{a, b, c\}$. We will not detail the complete run of the algorithm on this example, instead showing what happens at termination. The algorithm stops at the 6th iteration. The threshold value is either obtained by combining the best scores in $V1$ and $V4$ of the unseen (not in D) item $o1$, or by combining the best score in $V2$ of $o8$ and the best score in $V3$ of $o1$. Both result in $\tau = 8$. The worst score of the 5th item, $o7$, is also 8, enabling termination. This ensures that all the possible candidates for top-k are already present in the list D (see Table 3.2). Within this candidate list, there does not exist a combination of 5 objects that represents the top-k and, instead, we can only divide D into three sets:*

1. *the set $G = \{o3, o5, o6, o10\}$ of guaranteed result objects,*
2. *the set $P = \{o7, o4\}$ of possible result objects,*
3. *the remaining objects: $\{o2, o9\}$.*

obj	o3	o5	o6	o10	o7	o2	o9	o4
wsc	18	17	13	9	8	7	5	3
bsc	18	17	14	12	8	7	7	9

Table 3.2.: Candidates (D) at termination, for Q=$\{a, b, c\}$, k=5.

Algorithm 4 details SR-TA. Its general flow is similar to the one of TA, with the notable addition of the generalized computation of bounds and of the threshold value.

We now discuss our adaptation of NRA, called SR-NRA. Now, the exclusively sequential nature of accesses to views means that the per-view scores will only be partially filled (the random accesses in line 4 of SR-TA are no longer possible).

Algorithm 4: SR-TA$(Q, k, \mathcal{V})$

Require: query Q, size k, views $\mathcal{V}$

1: $D = \emptyset$
2: **loop**
3: **for** each view $V \in \mathcal{V}$ in turn **do**
4: $(o_i, wsc_i, bsc_i) \leftarrow$ next tuple by sequential access in V
5: read by random-accesses all other lists $V' \in \mathcal{V}$ for tuples (o_j, wsc_j, bsc_j) s.t. $o_i = o_j$
6: $wsc \leftarrow$ solution to the MP in Eq. (3.2.1) for o_i
7: $bsc \leftarrow$ solution to the MP in Eq. (3.2.2) for o_i
8: add the tuple (o_i, wsc, bsc) to D
9: **end for**
10: $\tau \leftarrow$ the solution to the MP in Eq. (3.3.1)
11: $wsc_t \leftarrow$ lower-bound score of kth candidate in D
12: **if** $\tau \leq wsc_t^{(D)}$ **then**
13: **break**
14: **end if**
15: **end loop**
16: $\{G, P\}$=PARTITION(D, k)
17: **return** G, P

At any moment in the run of SR-NRA, $seen(o, \mathcal{V}) \subseteq \mathcal{V}$ gives the views in which o has been encountered already through sequential accesses. We say that an object is *fully known* if $seen(o, \mathcal{V}) = \mathcal{V}$, and *partially known* otherwise. Then, for views $V \in seen(o, \mathcal{V})$ we keep the same constraints as in the MPs (3.2.1), (3.2.2). For each view

$V \notin seen(o, \mathcal{V})$, we adjust the corresponding constraint as

$$0 \leq \sum_{t_j \in Q^V} sc(o, t_j | \mathcal{C}) \leq \max\{bsc_i | (o_i, bsc_i, wsc_i) \in V, o_i \notin D\} \tag{3.3.2}$$

Algorithm 5: SR-NRA$(Q, k, \mathcal{V})$

Require: query Q, size k, views $\mathcal{V}$

$D = \emptyset$
for object $o \in \mathcal{O}$ **do**
  $seen(o, \mathcal{V}) = \emptyset$
end for
loop
  for each view $V \in \mathcal{V}$ in turn **do**
    $(o_i, wsc_i, bsc_i) \leftarrow$ next tuple by sequential access in V
    $seen(o_i, \mathcal{V}) \leftarrow seen(o_i, \mathcal{V}) \cup V$
    $wsc \leftarrow$ solution to the MP in Eq. (3.2.1), with Eq. (3.3.2)
    $bsc \leftarrow$ solution to the MP in Eq. (3.2.2), with Eq. (3.3.2)
    if $o_i \notin D$ **then**
      add the tuple (o_i, wsc, bsc) to D
    else
      update the existing tuple $(o_i, .., ..)$ from D
    end if
  end for
  maintain D sorted descending by wsc values
  $\tau \leftarrow$ the solution to the MP in Eq. (3.3.1)
  $bsc_{rest} = \max\{bsc_j | o_j \notin D[1..k], seen(o_j, \mathcal{V}) \neq \mathcal{V}\}$
  $wsc_t \leftarrow$ lower-bound score of kth candidate in D
  if $\tau \leq wsc_t$ and $bsc_{rest} \leq wsc_t$ **then**
    break
  end if
end loop
$\{G, P\}$=PARTITION(D, k)
return G, P

The termination conditions need to keep track, besides the threshold value, of the maximum upper-bound score of partially known objects that not in the current top-k of D, denoted bsc_{rest}. Objects that are fully known are ignored in this estimate, since their scores are fully filled and they might be candidates for P. The general flow of SR-NRA is given in Algorithm 5.

Partition for most informative result. Once the main loop of SR-TA or SR-NRA terminates, candidates D are passed as input to a sub-routine whose role is to partition it into sets G and P (line 14 in SR-TA, line 25 in SR-NRA). Algorithm 6 details this step: for each object o in D we test the conditions of Eq. (3.2.3), (3.2.4), (3.2.5).

At the termination of both SR-TA and SR-NRA, we are guaranteed that G and P are *sound and complete*, in the following sense:

Property 3. *An object o is in the output set G of* $\textsc{Partition}(D,k)$ *iff in all possible data instances o is the top-k for Q, $\mathcal{C}$.*

An object o is in the output set P of $\textsc{Partition}(D,k)$ *iff in at least one possible data instance o is in the top-k for Q, $\mathcal{C}$.*

Note that the size of G is at most k, while the one of P is at most $|\mathcal{O}|$, hence the need for completeness, maximizing $|G|$ and minimizing $|P|$.

Remark. We have only detailed here the adaptation of TA and NRA, the reference algorithms for early-termination top-k retrieval. More generally, any early-termination algorithm, as the ones following a hybrid approach between the two, can be adapted in similar manner to the setting with cached results having score ranges.

3.4. Extracting a Probable Top-k

As discussed previously, the actual (inaccessible) top-k answer for the input query could be seen as being composed of two parts: the guaranteed objects G plus a top k' over P, for $k' = k - |G|$. By definition, G and P give the most informative certain result that can be obtained from the views: there can be no deterministic way to compute a certain top-k' over the P objects, nor a way to further prune the search space towards a more refined P set.

Therefore, one can only hope to improve the quality of the result by a more detailed *probabilistic* description of the result, in which a most likely top-k could be identified from G and P. Since for each object in P we have a lower and upper bound on its exact score, let us assume a known probability density function (e.g, uniform one) for scores within the known bounds. Based on this, we can reason about the likelihood of a top-k' selection over P.

A naïve way to obtain the most likely top-k' would be the following: enumerate all possible subsets of P of size k', and compute for each the probability of being the top-k'. Each of these $\binom{|P|}{k'}$ probability values can be easily enumerated.

Example 9. *Returning to Example 8, recall $P = \{(o7,8,8), (o4,3,9)\}$, and G consists of 4 objects. To complete the requested top-5 answer, we need to estimate the top object from P, the one that is most likely to have the highest score. Denoting this top object t, we can see that $Pr[t = \{o7\}] = Pr[o7 > o4]$ and $Pr[t = \{o4\}] = Pr[o4 > o7]$. If we assume*

that each score is uniformly distributed within its known $[wsc, bsc]$ *interval, we have that* $Pr[o7 > o4] = 0.833$ *and* $Pr[o4 > o7] = 0.166$*. Overall, the most likely top-5 answer to the input query is thus* $\{o3, o5, o6, o10, o7\}$*, with a probability of* 0.833*.*

Algorithm 6: PARTITION(D, k)

Require: candidate list D, parameter k

1: $G \leftarrow \emptyset$ *the objects guaranteed to be in the top-k*
2: $P \leftarrow \emptyset$ *the objects that might enter the top-k*
3: **for** each tuple $(o, bsc, wsc) \in D$, $o \neq *$ **do**
4: $x \leftarrow |\{(o', bsc', wsc') \in D \mid o' \neq o,\ bsc' > wsc\}|$
5: $wsc_t \leftarrow$ lower-bound score of kth candidate in D
6: **if** $x \leq k$ and for $(*, wsc_*, bsc_*) \in D, bsc_* \leq wsc$ **then**
7: add o to G
8: **else if** $bsc > wsc_t$ **then**
9: add o to P
10: **end if**
11: **end for**
12: **return** G, P

A much more efficient approach than the naïve enumeration is to use techniques from research in top-k evaluation in probabilistic databases [72, 65].

Using well-studied sampling approaches as "black boxes", we describe in Algorithm 7 a tractable approach for estimating the most likely top-k' over a set of triples (o_i, wsc_i, bsc_i), under the assumption that sampling can be done in polynomial time as well. We use an encoding-decoding pair of functions that map sets of objects to numerical keys, and vice-versa: $key = encode(S)$ is the key representing the set S, and $S = decode(key)$ gives the opposite mapping.

We proceed in Algorithm 7 as follows. We first initialize a hash table T for the domain of keys (range of *encode*). For a given number of sampling rounds, at each round l we go through the objects of P and generate for each a score based on its range; we then order the objects based on these scores into a list P_l (*sample_scores* subroutine). We obtain through *encode* the key for the set consisting of the top k' objects in P_l, and we increment the value corresponding to that key in T. At termination, we return the decoding of the key having the highest count in T.

3.5. View Selection

We consider now the view selection problem, which may improve the performance of our threshold algorithms SR-NRA and SR-TA, possibly at the risk of yielding results

Algorithm 7: ESTIMATE(P, k', r)

Require: objects P, parameter k', r sampling rounds

```
initialize hash table T
for rounds l = {1, ..., r} do
    P_l ← sample_scores(P)
    key = encode({P_l[1], ..., P_l[k']})
    if key ∉ T then
        T[key] = 0
    end if
    T[key] = T[key] + 1
end for
key = argmax_i(T[i])
return decode(key)
```

that are less accurate. To address this issue, we discuss at the end of this section how results obtained through view selection can be refined to the most informative one. *Throughout this section, we remain in the setting where the query and views are assumed to have the same context.*

We argue first that view selection comes as a natural perspective in the computation of score bounds. Recall that, for a given object $o \in O$, Problem 2 could be modeled straightforwardly by the mathematical programs (3.2.1) and (3.2.2). Put otherwise, we have as the dual of the minimization problem 3.2.1 the following packing LP:

$$\max \sum_{\substack{i=1,\\ (o,wsc,..)\in ans(V_i)}}^{|\mathcal{V}|} wsc \times l_i \; s.t. \qquad \sum_{t\in Q^{V_j}} l_j \leq 1, \forall t \in Q \quad (3.5.1)$$

$$\sum_{t\in Q^{V_j}} l_j = 0, \forall t \notin Q$$

and we have as the dual the maximization problem (3.2.2) the following covering LP:

$$\min \sum_{\substack{i=1,\\ (o,..,bsc)\in ans(V_i)}}^{|\mathcal{V}|} bsc \times u_i \; s.t. \qquad \sum_{t\in Q^{V_j}} u_j \geq 1, \forall t \in Q \quad (3.5.2)$$

$$\sum_{t\in Q^{V_j}} u_j = 0, \forall t \notin Q$$

Based on the programs (3.5.1) and (3.5.2), for each object o, in order to obtain its most refined bounds, we would need to first *fractionally* select views from $\mathcal{V}$ – as opposed to

integral selection – such that the linear combinations of o's scores with the coefficients u_i and l_i are optimal. In other words, for computing the worst score or best score of each object, it would suffice to select and take into account only the views $V_i \in \mathcal{V}$ such that (i) $l_i \neq 0$, for worst scores, or (ii) $u_i \neq 0$, for best scores.[2]

Example 10. *Let us consider the views in Table 3.1, using the LPs (3.5.1) and (3.5.2) to illustrate view selection for object o6.*

For the worst score, we need to optimize

$$\max 4l_1 + 2l_2 + 10l_3 + 9l_4, \; s.t.$$
$$l_1 + l_3 \leq 1, \; l_3 + l_4 \leq 1, \; l_2 + l_4 \leq 1.$$

The optimal is reached when $l_1 = 1$, $l_2 = 0$, $l_3 = 0$ and $l_4 = 1$, i.e., relying on the worst scores of o6 from views V_1 and V_4.

For the best score, we need to optimize

$$\min 4l_1 + 4l_2 + 10l_3 + 11l_4 \; s.t.$$
$$l_1 + l_3 \geq 1, \; l_3 + l_4 \geq 1, \; l_2 + l_4 \geq 1.$$

The optimal is reached when $l_1 = 0$, $l_2 = 1$, $l_3 = 1$ and $l_4 = 0$, i.e., relying on the best scores of o6 from views V_2 and V_3.

Solving the LPs (3.5.1) and (3.5.2) for each object, as a means to select only the useful views, would obviously be as expensive as solving directly the MPs (3.2.1) and 3.2.2. Instead, it would be preferable to solve these LPs and select some most relevant views *independently of any object*, i.e., only once, before the run of the threshold algorithm. Instead of per-object *wsc* and *bsc* values, in an approximate version of the two LPs, each view V_i could be represented by two unique values, $wsc(V)$ and $bsc(V)$. Our optimization problems would then simplify as follows:

[2]Restricting the domain of the u and l values to integers would lead to an NP-hard view selection problem. More precisely, Eq. (3.5.2) would reduce to an instance of the weighted set cover problem, and Eq. (3.5.1) would reduce to an instance of the k-dimensional perfect matching problem (where $k = \max(|Q^{(V)}|), \forall V \in \mathcal{V}$). In our setting, however, the restriction to the integer domain is not necessary, and there exist tractable methods for efficiently solving the above LPs in their fractional form.

$$\max \sum_{i=1}^{|\mathcal{V}|} wsc(V_i) \times l_i \; s.t. \quad \sum_{t \in Q^{V_j}} l_j \leq 1, \forall t \in Q \quad \sum_{t \in Q^{V_j}} l_j = 0, \forall t \notin Q \tag{3.5.3}$$

$$\min \sum_{i=1}^{|\mathcal{V}|} bsc(V_i) \times u_i \; s.t. \quad \sum_{t \in Q^{V_j}} u_j \geq 1, \forall t \in Q \quad \sum_{t \in Q^{V_j}} u_j = 0, \forall t \notin Q \tag{3.5.4}$$

and this would enable us to select the "good" views in the initialization step of the top-k algorithm, those participating to the computation of the optimal, i.e., views having *non-zero* u and l coefficients.

Furthermore, for each object o encountered in the run of Algorithms SR-TA and SR-NRA, we can now replace Eq. (3.2.1) and (3.2.2) (lines 5-6 in SR-TA) by the following estimates that use only the selected views $\widetilde{\mathcal{V}}$:

$$\widetilde{wsc} = \sum_{\substack{i=1 \\ (o,wsc,..) \in ans(V_i)}}^{|\widetilde{\mathcal{V}}|} wsc \times l_i \; ; \; \widetilde{bsc} = \sum_{\substack{i=1 \\ (o,..,bsc) \in ans(V_i)}}^{|\widetilde{\mathcal{V}}'|} bsc \times u_i$$

This is possible since, by the duality property [63], we are guaranteed that the feasible solutions for Eq. (3.5.3), (3.5.4) represent safe bounds for o's scores, i.e., $\widetilde{wsc} \leq wsc$ and $\widetilde{bsc} \geq bsc$. We can similarly simplify Eq. (3.3.1), for the threshold value (for line 8 in SR-TA).

Candidates for $wsc(V)$ and $bsc(V)$ We follow the described approach – approximating view selection – in two distinct ways.

First, per-view score bounds $wsc(V)$ and $bsc(V)$ could be based solely on the view's definition Q^V, and we experimented in this work with bounds that are defined as $wsc(V) = bsc(V) = |Q^V|$. for each $V \in \mathcal{V}$. The intuition for this choice is that object scores in a view V are proportional to the number of attributes in Q^V.

Second, we consider and experiment with in Section 3.9 the natural per-view measures that are based on the views' answers: (i) the average value of scores, (ii) the maximum value of scores, and (iii) the median values of scores.

Using the attribute-query graph An important optimization that can always be performed before the actual view selection is to filter out all views that cannot influence the score of objects w.r.t. the input query. For that, we can build an undirected graph

of queries, in which we have (i) a node for the input query Q, and (ii) one node for each of the views in $\mathcal{V}$. Two nodes are connected by an edge if they have at least one attribute in common. It is then easy to see that we can filter out all the views that are not in the connected component of the input query, as their objects' scores will be useless for computing an answer to Q.

3.5.1. Retrieving (G, P) After View Selection

We now discuss how the most informative result (G, P) – that can be obtained from the complete set of views $\mathcal{V}$ – can still be retrieved by refining a result $(\tilde{G}, \tilde{P})$ obtained on a selection of views $\tilde{\mathcal{V}}$. We only need to adopt the following modifications in instances of SR-TA or SR-NRA running over a selection of views:

1. when the main loop terminates, compute the optimal bounds for all objects in $\tilde{P}$ by random-accessing their scores in all the views in $\mathcal{V}$,
2. run for a second time the partition subroutine.

It can be easily shown that, in this way, we obtain the most informative result, i.e., we reach $\Delta = 0$. Therefore, the "bulk" of the work could be done only on a selection of views and its result, potentially few candidate objects from the P set, could just be refined at the end using the complete $\mathcal{V}$. We describe in Section 3.9 the impact of this optimization on the running time of SR-TA and SR-NRA.

To summarize, we have described two variants of SR-TA and SR-NRA: without view selection, denoted SR-TA^{nosel} and SR-NRA^{nosel}, and with view selection, denoted SR-TA^{sel} and SR-NRA^{sel}. For the view selection variant, our notation convention will be to replace the *sel* superscript by a def, max or avg one, depending on the selection method being used.

3.6. Formal Guarantees

We study in this section the formal properties of our algorithms, focusing on *instance optimality*.

Let $\boldsymbol{A}$ be the class of algorithms, including SR-TA and SR-NRA, that deterministically output the sound and complete sets P and G, and do not make "wild guesses". For a given set of views $\mathcal{V}$, we denote by $D(\mathcal{V})$ the class of all instances of answers in those views, i.e., $ans(V), V \in \mathcal{V}$.

Given two algorithms $A_1 \in \boldsymbol{A}$ and $A_2 \in \boldsymbol{A}$, we write $A_1 \preceq A_2$ iff, for all sets of views $\mathcal{V}$, A_2 is guaranteed to cost at least as much as A_1 – in terms of I/O accesses (sequential, random or a linear combination of the two) – over all instances in $D(\mathcal{V})$. Conversely, we write $A_1 \not\preceq A_2$ iff there exists at least one view set $\mathcal{V}$ and an instance

in $D(\mathcal{V})$ over which A_2 costs less than A_1. We say that an algorithm $A \in \boldsymbol{A}$ is *instance optimal over* $\boldsymbol{A}$ iff $A \preceq B, \forall B \in \boldsymbol{A}$.

We first consider the question whether one of the two variants of SR-TA or SR-NRA is guaranteed to perform better that the other, for all views and answers. The answer to this question is far from obvious: on one hand, SR-TA^{sel} or SR-NRA^{sel} should use fewer views to compute the P and G sets, but they might either go too deep in the selected views or might need additional accesses in other views on the other hand, SR-TA^{nosel} or SR-NRA^{nosel} may go through views that are useless for deriving optimal bounds. We can prove the following:

Lemma 1. SR-NRA^{sel} $\not\preceq$ SR-NRA^{nosel} $\not\preceq$ SR-NRA^{sel}.

Lemma 2. SR-TA^{sel} $\not\preceq$ SR-TA^{nosel} $\not\preceq$ SR-TA^{sel}.

Proof. Recall that any algorithm only needs to use the views contained in the connected component to which Q belongs, from the attribute-query graph introduced in Section 3.5. We denote the set of views contained in this connected component, except the query Q, as $\mathcal{V}_C(Q)$. We also denote as *relevant combinations of views* the set $\mathcal{V}(Q) = \{\mathcal{V}_1(Q), \ldots, \mathcal{V}_r(Q)\}$, composed of distinct (but not necessarily disjoint) sets $V_i(Q) \subseteq \mathcal{V}_C(Q)$ of views which can be used to compute $[wsc, bsc]$ intervals for any object o.

We start by proving that the *nosel* variant of the two algorithms may perform unnecessary accesses.

Let us consider the following instance of the data and the set of views, $\mathcal{V}$. Assume that for a given query Q, the set of relevant combinations $\mathcal{V}(Q)$ contains all the views in $\mathcal{V}_C(Q)$. Furthermore, the set of relevant combinations consists of two disjoint sets $\mathcal{V}_1(Q)$ and $\mathcal{V}_2(Q)$. We also assume w.l.o.g. that $|\mathcal{V}_1(Q)| = |\mathcal{V}_2(Q)| = s$. The case where they have different sizes follows a similar reasoning, but would clutter the presentation unnecessarily.

Let us construct the following database: all the objects in the views of $\mathcal{V}_1(Q)$ are present in $\mathcal{V}_2(Q)$ at the same depths. Each such object o is constructed to ensure that $wsc_o(\mathcal{V}_1(Q)) \geq wsc_o(\mathcal{V}_2(Q))$ and $bsc_o(\mathcal{V}_1(Q)) \leq bsc_o(\mathcal{V}_2(Q))$,i.e., that the best bounds are always obtained from $\mathcal{V}_1(Q)$.

Now, let us assume that SR-TA^{nosel} or SR-NRA^{nosel} stop at depth d, encountering $a \leq sd$ distinct objects, and returning P and G as desired. They have thus performed the following number of accesses: (i) $2sd$ sequential accesses for SR-NRA^{nosel} and SR-TA^{nosel}, and (ii) $(2sd - 1)a$ random accesses for SR-TA^{nosel}. The accesses to the $\mathcal{V}_1(Q)$'s bestscore lists would be $h_1 \leq sa$, and $h_2 \leq sa$ to $\mathcal{V}_2(Q)$'s for a total of $h_1 + h_2$.

Then, algorithms SR-TA^{nosel} or SR-NRA^{nosel} that only accesses the views in $\mathcal{V}_1(Q)$ would encounter the same number a of distinct items, since $\mathcal{V}_1(Q)$ and $\mathcal{V}_2$(Q) have

the same objects at the same depths. They would also stop at the same depth d as the no selection algorithms, as they would obtain the same bounds for the a objects, and the same value for the threshold. Hence SR-TA^{nosel} and SR-NRA^{nosel} would need only sd sequential accesses. In addition, SR-TA^{nosel} would only need $(sd-1)a$ random accesses. The same reasoning holds for accesses into bestscore lists: both algorithms need only the h_1 accesses detailed above.

We turn next to proving that the *sel* variant of the two algorithms can perform unnecessary accesses.

Let us construct the following database, for a given k and query Q. The set $\mathcal{V}(Q)$ of relevant views is composed, as above, of two disjoint sets of views $\mathcal{V}_1(Q)$ and $\mathcal{V}_2(Q)$ that both can give valid bounds for Q. Again, assume w.l.o.g. $|\mathcal{V}_1(Q)| = |\mathcal{V}_2(Q)| = s$.

Then, on the first k positions in $\mathcal{V}_1(Q)$ we insert k objects s.t., for any such object o, $wsc_o(\mathcal{V}_1(Q)) = \delta$ and $bsc_o(\mathcal{V}_1(Q)) = \delta + 2$. In the following k positions, we insert objects x s.t. $wsc_x(\mathcal{V}_1(Q)) = \delta - 1$ and $bsc_x(\mathcal{V}_1(Q)) = \delta + 1$. The rest of the objects, denoted $*$ are constructed s.t. $bsc_*(\mathcal{V}_1(Q)) = \delta - 1$.

For $\mathcal{V}_2(Q)$, the same k objects are inserted s.t. $wsc_o(\mathcal{V}_2(Q)) = \delta - 1$ and $bsc_o(\mathcal{V}_2(Q)) = \delta + 1$. In the next k positions, we insert the same objects x as above, s.t. $wsc_x(\mathcal{V}_2(Q)) = \delta - 1$ and $bsc_x(\mathcal{V}_2(Q)) = \delta$. The rest of the objects $*$ are constructed as above.

We analyze now the accesses needed for the algorithms using all views. Both algorithms will stop at depth k, since the threshold value is $\tau = \delta$ and the lowest worstscore of the k items is δ. Hence, the first k objects are in G and $P = \emptyset$. SR-NRA^{nosel} would perfom $2sk$ sequential accesses in the worstscore lists and $2sk$ accesses to the bestscore lists. SR-TA^{nosel} would perfom the same number of sequential accesses as TA and also $(2s-1)k$ random accesses.

SR-NRA^{sel}, using either $\mathcal{V}_1(Q)$ or $\mathcal{V}_2(Q)$, would only be able to stop at depth $2k$, thus performing $2sk$ sequential accesses in the worstscore lists and $2sk$ accesses in bestscore lists. It would also return P containing all $2k$ objects, as the bestscores of the bottom k objects are greater than the worstcores of the top k objects, and $G = \emptyset$. Hence, a score refinement step would need to be perfomed. If we still assume that only sequential accesses are allowed, the refinement step would need another ks sequential accesses into the "opposite" set of views, to establish that $P = \emptyset$ and G is composed of the objects in the first k positions. Hence it would need ks more sequential accesses than SR-NRA^{nosel}.

For SR-TA^{nosel}, the stopping condition would also hold at depth $2k$. The same number of $4sk$ sequential accesses would be needed, and $2k(s-1)$ random accesses. For refining the scores and returning the sets P and G, another $2ks$ accesses to the other views would also be needed. Hence it would need $k(2s-1)$ more random accesses than SR-TA^{nosel}. □

Lemmas 1 and 2 tell us that neither of the two variants of SR-TA or SR-NRA can be instance optimal for all possible sets $\mathcal{V}$. However, we describe next a restricted class of views for which: (i) no refinement step is necessary after selecting a subset of the views, and (ii) SR-TA^{sel} and SR-NRA^{sel} become instance optimal.

Let $\mathbf{V}$ be the class of *sets $\mathcal{V}$ of pairwise disjoint views*, i.e., s.t. $Q^{V_i} \cap Q^{V_j} = \emptyset, \forall V_i, V_j \in \mathcal{V}, V_i \neq V_j$. We say an algorithm $A \in \boldsymbol{A}$ is *instance optimal over $\boldsymbol{A}$ and $\boldsymbol{V}$* if $A \preceq B, \forall B \in \boldsymbol{A}$ and $\forall \mathcal{V} \in \boldsymbol{V}$. We can prove the following:

Theorem 2. SR-TA^{sel} *is instance optimal over $\boldsymbol{A}$ and $\boldsymbol{V}$.* SR-NRA^{sel} *is instance optimal over $\boldsymbol{A}$ and $\boldsymbol{V}$, when only sequential accesses are allowed.*

The proofs closely follow the flow of the NRA and TA optimality proofs in [31].

Intuitively, for this class of views, the only way to obtain bounds for a query Q is the following: (i) for lower-bounds, only the views V that have $Q^V \subseteq Q$ are taken into account, while (ii) for upper-bounds all views V that verify $Q^V \cap Q \neq \emptyset$ are used. Note that this method is in effect the view selection algorithm for the class of pairwise disjoint views. Note also that the setting of [31], i.e. per-attribute lists of exact scores, is strictly subsumed by $\mathbf{V}$.

3.7. Context Transposition

We have discussed until now how queries can be answered by exploiting pre-computed results from views, with the important assumption that these share the same context with the input query. We remove now this restriction, and consider also views that may have been computed in a different context. We show how we can still answer input queries by the techniques discussed so far, by pre-processing views in order to place them in the context of the input query. We call this step the *context transposition*.

We give in this section the details on context transpositions for our two motivating application scenarios: location-aware search and social-aware search. In both applications, one view V's context $\mathcal{C}^V$ can be seen as consisting of

1. a *location* (or *start point*) $\mathcal{C}^V.l$, e.g., geo-coordinates in a multidimensional space for location-aware search, or the social identity of a seeker in social-aware search,

2. a *contextual parameter* $\mathcal{C}^V.\alpha$, which basically parameterizes the influence of the spatial or social aspect in scores.

Given an input query Q, a context $\mathcal{C}$ – with $\mathcal{C}.l$ and $\mathcal{C}.\alpha$ – and a view V with a different context (either the location or α may differ, or both), in order to be able use pre-computed results from V, we need to derive from the existing $ans(V)$ tuples new score bounds: for each $(o, wsc, bsc) \in ans(V)$ we want to obtain a new tuple

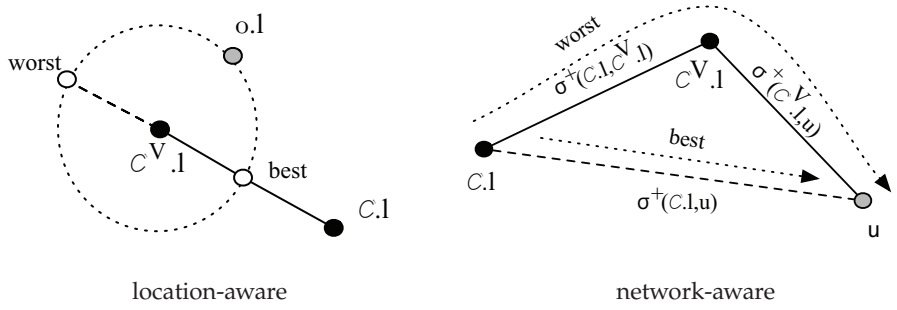


Figure 3.1.: Intuition for start point transpositions.

$(o, f_w(wsc), f_b(bsc))$. The functions f_w and f_b represent the core of the context transposition, their role being to map the worst scores and best scores of objects from $ans(V)$ to new guaranteed bounds for context $\mathcal{C}$. We detail them next for our application scenarios.

3.7.1. Location-Aware Search

In location-aware or spatial top-k querying [22], a user having a certain location is interested in the top-k objects that are relevant textually and close spatially.

We revisit here one of the most common ranking models [22, 14], in which the per-attribute score of an object is a linear combination of spatial relevance and textual relevance. Each object o consists of a bag of attributes $o.A$ and a location $o.l$. Given an input query Q, with context $\mathcal{C}$ having location and $\mathcal{C}.l$ and parameter $\mathcal{C}.\alpha$, the per-attribute score is obtained as follows:

$$sc(o, t \mid \mathcal{C}) = (1 - \mathcal{C}.\alpha)(1 - \frac{D(\mathcal{C}.l, o.l)}{maxDist}) + \mathcal{C}.\alpha \frac{TF(t, o.A)}{maxTF(t)} \tag{3.7.1}$$

where D gives the euclidean distance between Q's location (start point) and o's location, $maxDist$ is the maximal distance,
$TF(t, o.A)$ is the term frequency of t in $o.A$, and $maxTF(t)$ is a maximal term frequency of t over all objects.

We now detail how context transposition is performed. For ease of exposition, we first describe how the location component of the
context is transposed. Then, we describe the transposition for α.

Recall that we are in a setting where we are presented only with partial information, in each view V, in the form of tuples (o, wsc, bsc), and we have access neither to $D(\mathcal{C}.l, o.l)$ nor to $TF(t, o.A)$. The bounds may be tight, i.e., $wsc = bsc$, as in [22, 20], or may be loose.

Transposing the location. Given a query Q of context $\mathcal{C}$, with location $\mathcal{C}.l$, given of view V of context $\mathcal{C}^V$, with location $\mathcal{C}^V.l$, for any object $o \in ans(V)$, there are two extreme locations at which o can be situated, relative to $\mathcal{C}.l$ (see Figure 3.1), as follows:

1. on the segment between $\mathcal{C}.l$ and $\mathcal{C}^V.l$, or on the line connecting $\mathcal{C}.l$ and $\mathcal{C}^V.l$, beyond $\mathcal{C}.l$, resulting in
$$D(\mathcal{C}.l, o.l) \;=\; |D(\mathcal{C}^V.l, \mathcal{C}.l) - D(\mathcal{C}^V.l, o.l)|$$
2. on the line connecting $\mathcal{C}.l$ and $\mathcal{C}^V.l$, beyond $\mathcal{C}^V.l$, giving
$$D(\mathcal{C}.l, o.l) = D(\mathcal{C}^V.l, \mathcal{C}.l) + D(\mathcal{C}^V.l, o.l).$$

We can now derive the following new bounds for each object o from a tuple $(o, wsc, bsc) \in ans(V)$, which would be valid in a context $\mathcal{C}'$ defined by the query's location $\mathcal{C}.l$ and the view's $\mathcal{C}^V.\alpha$:

$$\begin{aligned} sc(o, Q \mid \mathcal{C}') &\geq wsc - \mathcal{C}^V.\alpha \times |Q^V| \times \frac{D(\mathcal{C}^V.l, \mathcal{C}.l)}{maxDist} = wsc' \\ sc(o, Q \mid \mathcal{C}') &\leq bsc + \mathcal{C}^V.\alpha \times |Q^V| \times \frac{D(\mathcal{C}^V.l, \mathcal{C}.l)}{maxDist} = bsc' \end{aligned} \tag{3.7.2}$$

Transposing the parameter α. We consider now the transposition for the α component, from $\mathcal{C}^V.\alpha$ to $\mathcal{C}.\alpha$, by which are obtained bounds that are valid for the input query context $\mathcal{C}$.

Let in Eq. (3.7.1)
$$x = 1 - \frac{D(\mathcal{C}.l, o.l)}{maxDist}, \;\; y = \frac{TF(t, o.A)}{maxTF(t)}.$$
From the intermediary context $\mathcal{C}'$ introduced by the transposition of location, for any tag $t \in Q^V$ and object $o \in ans(V)$, we have:

$$\begin{aligned} sc(o, t \mid \mathcal{C}') &= (1 - \mathcal{C}^V.\alpha) \times x + \mathcal{C}^V.\alpha \times y \\ sc(o, t \mid \mathcal{C}) &= (1 - \mathcal{C}.\alpha) \times x + \mathcal{C}.\alpha \times y \end{aligned}$$

This leads to
$$sc(o, t \mid \mathcal{C}) \;=\; sc(o, t \mid \mathcal{C}') - (\mathcal{C}.\alpha - \mathcal{C}^V.\alpha) \times (x - y)$$
Since $x \in [0, 1]$ and $y \in [0, 1]$, we have that
$$\max(x - y) = 1, \quad \min(x - y) = -1,$$
therefore:
$$sc(o, t \mid \mathcal{C}') - |\mathcal{C}.\alpha - \mathcal{C}^V.\alpha| \leq sc(o, t \mid \mathcal{C}) \leq sc(o, t \mid \mathcal{C}') + |\mathcal{C}.\alpha - \mathcal{C}^V.\alpha|$$

Summing over all $t \in Q^V$, replacing the l.h.s. with wsc' and the r.h.s. with bsc', we obtain:

$$sc(o, Q \mid \mathcal{C}) \geq wsc' - |Q^V| \times \mid \mathcal{C}.\alpha - \mathcal{C}^V.\alpha \mid = \mathbf{f_w}(\mathbf{wsc})$$
$$sc(o, Q \mid \mathcal{C}) \leq bsc' + |Q^V| \times \mid \mathcal{C}.\alpha - \mathcal{C}^V.\alpha \mid = \mathbf{f_b}(\mathbf{bsc}) \quad (3.7.3)$$

We present in Section 3.8 the generic algorithm that, for an input query Q and set of views, filters out the views that are certainly useless for Q, performs the necessary context transposition steps, then may select views, and finally runs one of the threshold algorithms.

3.7.2. Social-Aware Search

We consider now the social-aware setting. For illustration, we use the *collaborative bookmarking* applications – popular examples include Del.icio.us and Flickr – which represent a good abstraction for search with a social context .

For this setting, we revisit the ranking model developed in [69, 4]. Besides objects and attributes, we have a set of users $\mathcal{U} = \{u_1, \ldots, u_n\}$ who can bookmark (or tag) objects with attributes. Also, users form a social network, seen as an undirected weighted graph: a link between two users u_1 and u_2 has a weight, $\sigma(u_1, u_2) \in [0, 1]$, which could stand for proximity, similarity, affinity, etc. For pairs of users for which an explicit edge (and proximity) is not given, an extended proximity $\sigma^+(u_1, u_2) \in [0, 1]$ can be computed in the graph by aggregating (e.g., by multiplication) the weights over each path connecting u_1 and u_2, and taking the maximal aggregated score over all paths:[3]

$$\sigma^+(u_1, u_2) = \max_{p=(u_1,\ldots,u_2)} \prod_{i=0}^{k-1} \sigma(u_i, u_{i+1}).$$

A query context $\mathcal{C}$ consists now of a *seeker* $\mathcal{C}.l$ (the issuer of the query) and the parameter $\mathcal{C}.\alpha$. In manner similar to location-aware search, the per-attribute score is a linear combination between the "social location" of the seeker with respect to the taggers of an object and the classic textual score (e.g., tf/idf or BM25).

Unlike the location-aware setting, we do not deal with objects that are "located" at certain points in the search space, but with user-object-attribute tuples (u, o, t), i.e., users u who tagged o with t. The social component of the score is computed as the sum of the proximity values of taggers of o with respect to the seeker, while the textual component is the number of taggers who tagged object o with attribute t:

$$sc(o, t \mid \mathcal{C}) = (1 - \mathcal{C}.\alpha) \times \sum_{\substack{u \text{ tagged } o \\ \text{with } t}} \sigma^+(\mathcal{C}.l, u) + \mathcal{C}.\alpha \times TF(o, t) \quad (3.7.4)$$

[3] This is reminiscent of how trust or similarity can propagate, if interpreted as transitive measures.

Transposing the location. For a query Q and seekers $\mathcal{C}^V.l$ and $\mathcal{C}.l$, let u be a tagger for which we need to use $\sigma^+(\mathcal{C}.l, u)$ in score bounds. As illustrated in Figure 3.1, the path with the highest score connecting $\mathcal{C}.l$ to u may either

1. go through $\mathcal{C}^V.l$, and in that case we have:

$$\sigma^+(\mathcal{C}.l, u) = \sigma^+(\mathcal{C}.l, \mathcal{C}^V.l) \times \sigma^+(\mathcal{C}^V.l, u),$$

2. not go through $\mathcal{C}^V.l$, and in that case we have:

$$\sigma^+(\mathcal{C}.l, u) \leq \sigma^+(\mathcal{C}.l, \mathcal{C}^V.l)^{-1} \times \sigma^+(\mathcal{C}^V.l, u).$$

Now, the influence of the social component in the score of o in $ans(V)$ varies inversely with $\mathcal{C}^V.\alpha$. Therefore, the transposition that accounts for the location change should be weighted by its importance in the $sc(o, t \mid \mathcal{C}^V)$ formula, determined by $\mathcal{C}^V.\alpha$ as follows: when $\mathcal{C}^V.\alpha = 1$, the lower and upper bounds should not be affected by the location change, while when $\mathcal{C}^V.\alpha = 0$, they should be affected with weight $\sigma^+(\mathcal{C}.l, \mathcal{C}^V.l)$ and $\sigma^+(\mathcal{C}.l, \mathcal{C}^V.l)^{-1}$ respectively. We can model this by a coefficient function $c(w, \alpha)$, which applies to a weight w and value α, and is defined as

$$c(w, \alpha) = \alpha(1 - w) + w.$$

(Note that it verifies $c(w, 0) = w$ and $c(w, 1) = 1$, as needed.)

For each object o of a tuple $(o, wsc, bsc) \in ans(V)$, we can now derive the following valid bounds for a context $\mathcal{C}'$ defined by the query's seeker $\mathcal{C}.l$ and the view's parameter $\mathcal{C}^V.\alpha$.

$$\begin{aligned} sc(o, t \mid \mathcal{C}') &\geq c(\sigma^+(\mathcal{C}.l, \mathcal{C}^V.l), \mathcal{C}^V.\alpha) \times wsc && = wsc' \\ sc(o, t \mid \mathcal{C}') &\leq c(\sigma^+(\mathcal{C}.l, \mathcal{C}^V.l)^{-1}, \mathcal{C}^V.\alpha) \times bsc && = bsc' \end{aligned} \tag{3.7.5}$$

Transposing the parameter α. We consider now the transposition for the α component, from $\mathcal{C}^V.\alpha$ to $\mathcal{C}.\alpha$, yielding valid bounds for the new context, $\mathcal{C}$. Here, the transposition depends on the relationship between $\mathcal{C}.\alpha$ and $\mathcal{C}^V.\alpha$, and we obtain the following f_w and f_b:

1. if $\mathcal{C}.\alpha < \mathcal{C}^V.\alpha$, $\mathbf{f_w(wsc)} = \frac{\mathcal{C}.\mathbf{ff}}{\mathcal{C}^\mathbf{V}.\mathbf{ff}} \times \mathbf{wsc'}$, $\mathbf{f_b(bsc)} = \mathbf{bsc'}$
2. if $\mathcal{C}.\alpha > \mathcal{C}^V.\alpha$, $\mathbf{f_w(wsc)} = \mathbf{wsc'}$, $\mathbf{f_b(bsc)} = \frac{\mathcal{C}.\mathbf{ff}}{\mathcal{C}^\mathbf{V}.\mathbf{ff}} \times \mathbf{bsc'}$

We arrive at these bounds by the following steps. In Eq. (3.7.4), let

$$x = \sum_{\substack{u \text{ tagged } o \\ \text{with } t}} \sigma^+(\mathcal{C}.l, u), \quad y = TF(o, t).$$

From the intermediary context $\mathcal{C}'$ introduced by the transposition of location, for any tag $t \in Q^V$ and object $o \in ans(V)$, we have:

$$\begin{aligned} sc(o,t \mid C') &= (1 - \mathcal{C}^V.\alpha) \times x + \mathcal{C}^V.\alpha \times y \\ sc(o,t \mid C) &= (1 - \mathcal{C}.\alpha) \times x + \mathcal{C}.\alpha \times y \end{aligned}$$

Writing y in terms of $sc(o,t \mid \mathcal{C}')$ and x in the first equation we get

$$y = \frac{sc(o,t \mid \mathcal{C}') - (1 - \mathcal{C}^V.\alpha) \times x}{\mathcal{C}^V.\alpha}.$$

Then, by plugging y into the second equation, we obtain:

$$sc(o,t \mid \mathcal{C}) = (1 - \frac{\mathcal{C}.\alpha}{\mathcal{C}^V.\alpha}) \times x + \frac{\mathcal{C}.\alpha}{\mathcal{C}^V.\alpha} \times sc(o,t \mid \mathcal{C}').$$

Since we know that $x \leq sc(o,t \mid \mathcal{C}')$ (due to the fact that $x \leq y$, by definition), we can derive the following bounds:

$$\begin{aligned} sc(o,t \mid \mathcal{C}) &\leq \frac{\mathcal{C}.\alpha}{\mathcal{C}^V.\alpha} sc(o,t \mid \mathcal{C}') \;\; if \; \mathcal{C}.\alpha > \mathcal{C}^V.\alpha \\ sc(o,t \mid \mathcal{C}) &\geq \frac{\mathcal{C}.\alpha}{\mathcal{C}^V.\alpha} sc(o,t \mid \mathcal{C}') \;\; if \; \mathcal{C}.\alpha < \mathcal{C}^V.\alpha \end{aligned}$$

By subtraction, we also have that:

$$sc(o,t \mid \mathcal{C}) = sc(o,t \mid \mathcal{C}') + (\mathcal{C}.\alpha - \mathcal{C}^V.\alpha)(y - x)$$

Unlike the location-aware setting, we do not have a bound on the difference between y and x, but we know that $(y - x) \geq 0$, and therefore we can obtain:

$$\begin{aligned} sc(o,t \mid \mathcal{C}) &\geq sc(o,t \mid \mathcal{C}') \;\; if \; \mathcal{C}.\alpha > \mathcal{C}^V.\alpha \\ sc(o,t \mid \mathcal{C}) &\leq sc(o,t \mid \mathcal{C}') \;\; if \; \mathcal{C}.\alpha < \mathcal{C}^V.\alpha \end{aligned}$$

Putting everything together, over all $t \in Q^V$, we obtain the f_w and f_b transposition functions given previously.

3.8. Putting It All Together

We put together the techniques discussed so far into a generic algorithm for top-k answering using views with uncertain scores (Algorithm 8). It starts by transposing the various contexts $\mathcal{C}^V$ to the one of the input query, and selects some views. Either SR-TA or SR-NRA can then be used to find an answer $(\tilde{G}, \tilde{P})$, in terms of guaranteed-possible objects for the top-k. We can then use the sampling procedure (Algorithm 7) to find a most likely top-k answer.

Algorithm 8: ProcessQueryViews$(\mathcal{V}, Q, \mathcal{C}, k, r)$

Require: query Q, views $\mathcal{V}$, context $\mathcal{C}$, top k required, r rounds

1: **for** $V \in \mathcal{V}$ **do**
2: transpose the context $\mathcal{C}^V$ to $\mathcal{C}$
3: **end for**
4: $\tilde{\mathcal{V}} \leftarrow$ view selection on $\mathcal{V}$ for Q
5: $\{G, P\} \leftarrow$ SR-TA$(Q, k, \tilde{\mathcal{V}})$ or SR-NRA$(Q, k, \tilde{\mathcal{V}})$
6: $E =$ Estimate$(P, k - |G|, r)$
7: **return** $G \cup E$

In the experiments, we give a detailed comparison between this approach and (i) an existing early-termination algorithm for spatial-aware search, the IR-Tree of [22], and (ii) the early-termination algorithm ContextMerge of [69], for social-aware search.

Remark 1. An advantage of the bounds derived in Eq. (3.7.3) and (3.7.5) is that they do not depend on the individual objects: in scores, we only add-substract or multiply-divide with some object-independent values. This gives us the opportunity to use certain implementation "tricks": instead of recomputing the score bounds of all object in the $ans(V)$ lists, we can use these score changes as

1. additive (location-aware search) or multiplicative (social-search) coefficients directly in per-view statistics for view selection,
2. coefficients for the linear programs in Eq. (3.2.1) and (3.2.2), during the run of SR-TA or SR-NRA.

Remark 2. While we focused in the previous two sections on reference ranking models for context-aware search from previous literature, the same reasoning can be adapted to more involved ranking models, as long as they are monotonic relative to both the textual and the context-aware components of scores.

3.9. Experiments

We performed our experiments on a single core of a i7-860 2.8GHz machine equipped with 8GB of RAM. We implemented our algorithms in Java, and we used this implementation for our tests on synthetic data and social data. We also implemented them in C++, for a more reliable comparison with IR-Tree, for spatial data.

Context-agnostic setting with complete views Our first series of tests, over synthetic data, concerns a setting in which the input queries and the views share the same context (i.e., context plays no role and is ignored in the computation). We generated

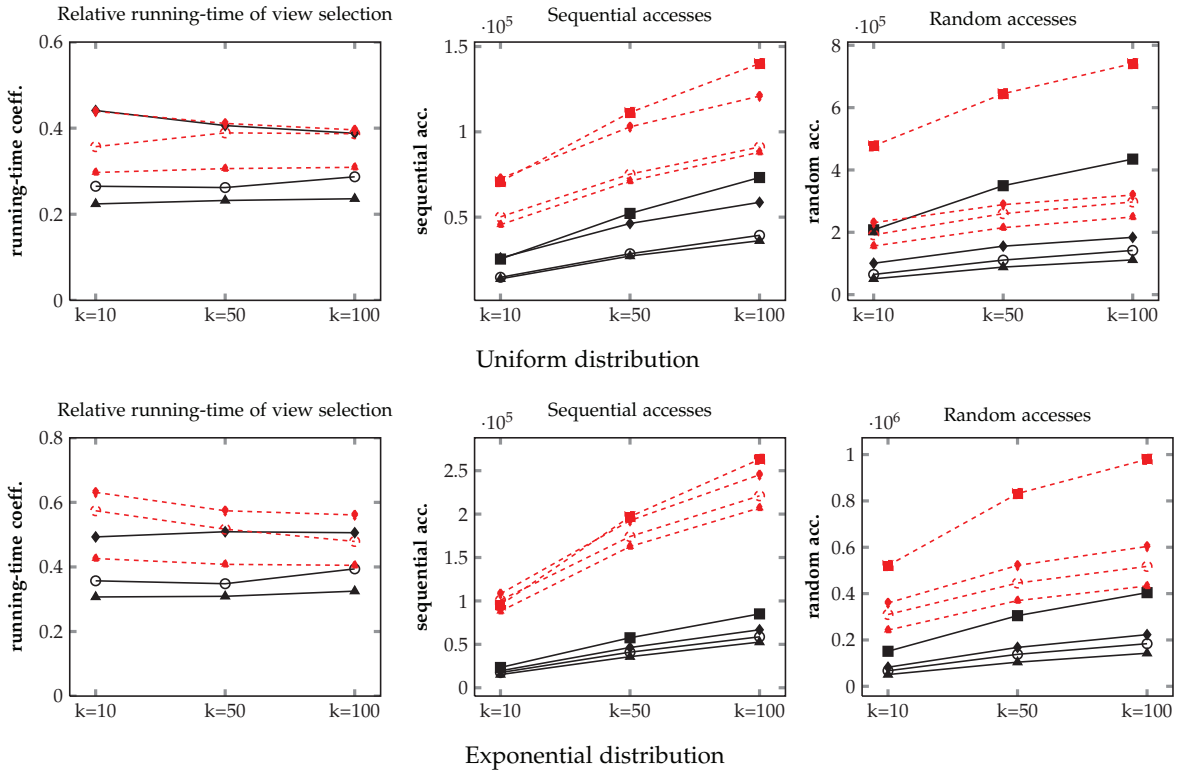


Figure 3.2.: Performance comparison between SR-TA variants over synthetic data with uniform and exponential distribution.

exact scores in the range $[0, 100]$ for 100,000 objects and 10 attributes, with exponential or uniform distributions. Then, we generated all possible combinations of 2 and 3 attributes, each representing one view. For each of the views, we computed the exact (aggregated) scores over *all* objects; the views are *complete* in that sense. We then made these lists uncertain by replacing each exact value by a score range, using the gaussian distribution with mean equal to the exact value and standard deviation (std, in short) equal to either 5, 10 or 20. Over the sets of views obtained in this way, we used 100 randomly-generated input queries consisting of 5 distinct attributes.

We compare in Figure 3.2 the SR-TA variants over the two data distributions, for the std values 5 and 10 (to avoid clutter, the plots for std 20 are not given). We have recorded (i) the relative running-time of the algorithms that use view selection w.r.t. the algorithm using all the views – three selection criteria per two std values, for six plot lines, (ii) the number of sequential accesses by all four variants – with the two std values, for eight plot lines, and (iii) the number of random accesses by all four variants – with the two std values, for eight plot lines.

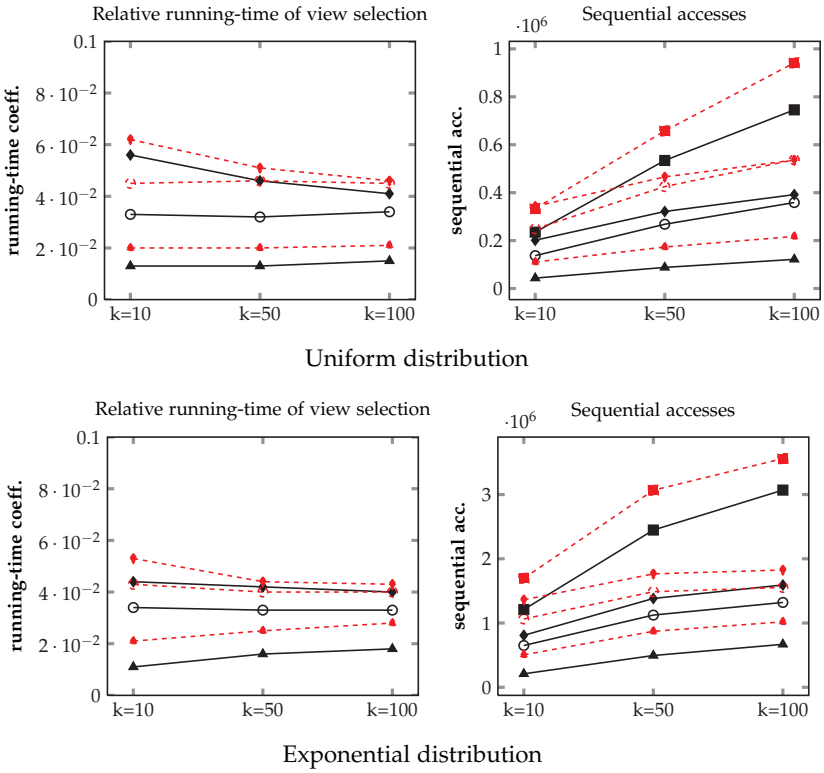


Figure 3.3.: Performance comparison between SR-NRA variants over synthetic data with uniform and exponential distribution.

Sel. + Dist.	Rel. running-time			Min. precision			$\lvert P \rvert$		
	10	50	100	10	50	100	10	50	100
avg + uni	0.576	0.676	0.712	0.57	0.69	0.72	10	36	64
def + uni	0.350	0.446	0.544	0.57	0.69	0.72	10	36	64
max + uni	0.296	0.395	0.446	0.57	0.69	0.72	10	36	64
avg + exp	0.732	1.128	1.287	0.60	0.63	0.64	10	46	86
def + exp	0.531	0.771	1.003	0.60	0.63	0.64	10	46	86
max + exp	0.456	0.684	0.827	0.60	0.63	0.64	10	46	86

Table 3.3.: Comparison between SR-TA and TA (exact scores), for uniform and exponential distributions, for std 5.

One can note that the algorithms with view selection achieve significant savings in terms of both running-time and I/O accesses. The algorithm based on max-statistics, SR-TA^{max}, achieves better performance than the one based on view definitions, SR-TA^{def}, which in turn does better than the one based on average-statistics, SR-TA^{avg}. Furthermore, we can observe that the relative running-time of these algorithms does not depend on the value of k, and the influence of the interval coarseness (by standard deviation) is more important in the exponential distribution. One can also note a "clustering" effect, by standard deviation, in the case of sequential-access measures; this is likely due to the fact that top-k processing on noisier data needs to go deeper in the views to reach termination.

The same measurements were performed for the SR-NRA variants (see Figure 3.3), in which random-accesses are not allowed. We can observe similar behavior to the SR-TA variants, in terms of running-time. However, we have a much lower relative running-time for SR-NRA^{sel}, less than 0.1 of that of SR-NRA^{nosel}. This is due to the fact that the overhead induced by solving the LPs for score bounds is much more notable in this case (in the case of SR-TA, it was dominated by the cost of the random accesses).

We also compared the performance of SR-TA, over score ranges with low noise (std of 5), with the one of Fagin's TA over the exact per-attribute inverted lists. We trace two measures: the relative running-time and the minimum precision. The latter is computed as $|G|/k$, i.e., the ratio between the size of the guaranteed set and the required k. The results are presented in Table 3.3. One can note that SR-TA^{sel} can have a running-time that is a low fraction of that of TA (as low as 0.296, with a precision@10 of 0.577). This is mainly due to the fact that, although inexact, we have aggregated scores pertaining to 2 or 3 query terms, while the noise levels are rather low. While using exact lists of aggregated data for top-k processing would certainly improve effi-

Sel.	Std	Overhead	$\lvert\mathbf{G}\rvert - \lvert\tilde{\mathbf{G}}\rvert$	$\lvert\mathbf{P}\rvert - \lvert\tilde{\mathbf{P}}\rvert$	Δ
avg	5	0.031	38	-208	1.96×10^{70}
avg	10	0.033	35	-734	4.14×10^{129}
avg	20	0.119	15	-4828	2.65×10^{212}
def	5	0.040	37	-206	2.76×10^{69}
def	10	0.038	34	-727	1.96×10^{129}
def	20	0.138	15	-4749	5.93×10^{211}
max	5	0.041	35	-179	5.54×10^{64}
max	10	0.041	33	-575	6.38×10^{119}
max	20	0.117	15	-3592	7.96×10^{200}

Table 3.4.: Running-time overhead and Δ difference, for SR-TA^{sel} with final refinement versus SR-TA^{sel} without refinement , for k=100 and exponential distribution.

ciency, as shown in [49], our experiments show that even relatively noisy aggregated data can lead to improvements, with reasonable precision.

Finally, we give in Table 3.4 the overhead of the refinement step discussed in Section 3.5, which uses random-accessing to refine a result ($\tilde{G}$, $\tilde{P}$) to the most informative one, (G,P). Overhead is measured as the ratio between the running-time of the base algorithm and the one of the refined algorithm. We also report on the Δ measure. Note that, while the number of possible combinations that are "avoided" increases exponentially with the standard deviation, the overhead of additional I/O accesses is small (range 3%-13%).

Location-aware search The dataset used in this setting is the PolyBot one, provided by the authors of [20]. It consists of 6,115,264 objects (documents) and their coordinates in a 2D space, and a total of 1,876 attributes (terms). We have generated 20 views defined by 2-term queries at 5 different locations, varying the size of their *ans* lists (500, 1000 and 2000 entries). We used 10 to-be-answered queries at 5 locations (different to the ones of views) and we varied $k \in \{10, 20\}$ and $\alpha \in \{0.7, 0.8, 0.9\}$. For the α values, we used values close to those indicated by the authors of [22].

The algorithm we use as baseline in our evaluation is our implementation of the IR-tree of [22]. It is based on R-tree indices [36], whose nodes are enriched with inverted lists consisting of the documents located inside the rectangle defined by the node. The algorithm maintains a priority queue, containing either objects and their scores, or tree nodes and the maximum scores in their inverted list. The algorithm alternates between visiting nodes and adding objects to the candidate list. It stops when k objects have been retrieved. Our implementation of this algorithm achieves

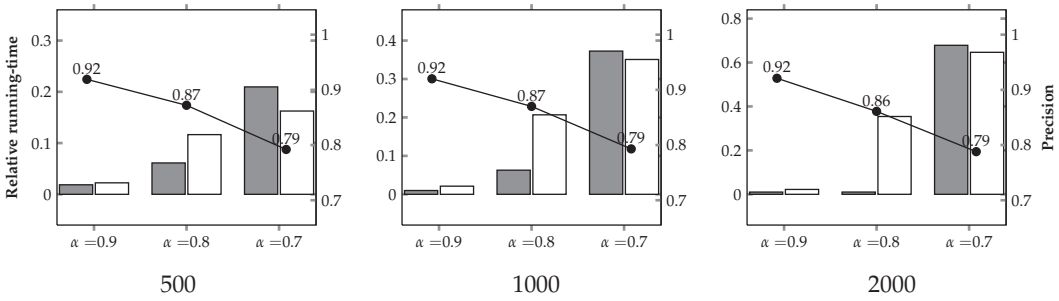


Figure 3.4.: Location-aware search: performance and precision of SR-TA^{sel} versus exact early-termination algorithm (IR-TREE), for various α values and list sizes (grey=top-10, white=top-20).

very similar running-time to the one reported in [22].

We present in Figure 3.4 the results for relative running-time and precision. The relative running-time is computed as the ratio between the running-time of SR-TA and the one of IR-TREE. Precision is computed as the percentage of top-k items returned by SR-TA that also appear in the output of IR-TREE. Here, we used the sampling method from Section 3.4 to obtain the most likely top-k from the (G,P) answer, through 1,000 rounds of uniform sampling.

One can note that, for high values of α and low values of k, the response time of SR-TA is significantly lower than that of the IR-TREE (in practice, of the order of milliseconds), with reasonably high precision levels (between 0.86 and 0.92). This is because the top-k answer is based on a large set G of guaranteed objects, which reduces the overhead of the sampling procedure. When the uncertainty introduced by coarser score ranges in views leads to larger sets P instead, the sampling procedure is more costly, but overall the running-time remains a small fraction of the one of the IR-TREE, with a precision around 0.8.

Social-aware search For this application scenario, we used the publicly-available Delicious bookmarking data of [78]. We extracted a random subset of it, containing 80,000 users, their tagging behavior on 595,811 objects (items) with 198,080 attributes (tags). For assigning weights to links between users, we generated three similarity networks, by computing the Dice coefficients of either (i) common tags in a tag similarity network, (ii) common items in an item similarity network or (iii) common item-tag pairs in an item-tag similarity network.

For each of the three similarity networks, we randomly chose 5 seekers for our tests. Then, a number of 10 users were randomly chosen, among those having a link with weight of at most 0.66 to any of the 5 seekers (to ensure that no view is too "useful", having too strong an influence on the running-time and precision). For each of these

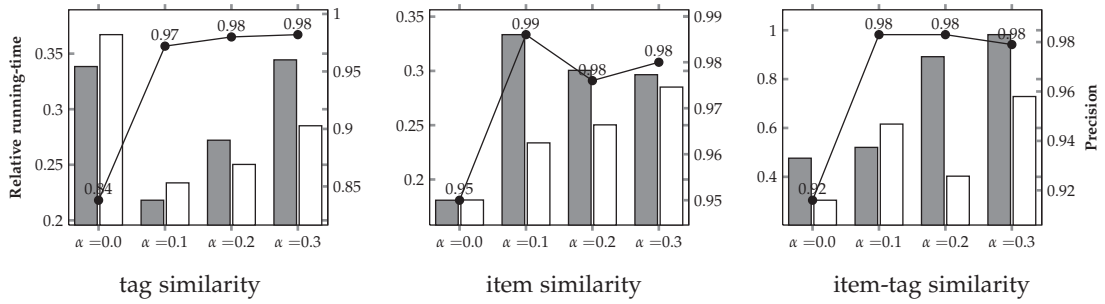


Figure 3.5.: Social-aware search: performance and precision of SR-TA^{sel} versus exact early-termination algorithm (ContextMerge), in three similarity networks (grey=top-10, white=top-20).

users and for $\alpha \in \{0.0, 0.1, 0.2, 0.3\}$, we generated 40 views of 1 and 2-tag queries, each containing 500 entries.

The tests were made on a set of 10 3-tag queries for each of the 5 seekers, varying $\alpha \in \{0.0, 0.1, 0.2, 0.3\}$ and $k \in \{10, 20\}$.

The baseline algorithm we used for the performance comparison is a direct adaptation of the ContextMerge algorithm of [69]. In short, depending on the value of α, ContextMerge alternates between per-attribute inverted lists of objects and an inverted list containing users ordered descending by their proximity relative to the seeker. When the algorithm visits a user, her relevant objects – those that were tagged by her with attributes appearing in the input query – are retrieved and added to the candidate list. In manner similar to NRA, the algorithm keeps a threshold value representing the maximal possible score of objects, based on the maximal scores from the inverted lists and the proximity value of not yet visited users. The termination condition is very similar to that of NRA.

Similar to the location-aware search, we present in Figure 3.5 the results in terms of relative running-time and precision. One can note that the running-time is still a low fraction of the one of the exact algorithm, while the precision levels are considerably higher than in the case of location-aware search. As expected, the lowest precision levels are obtained when the search relies exclusively on the social component of the score. This is due to the fact that the bounds computed by Eq. (3.7.5) yield coarser score ranges when $\alpha = 0$, which are source of more uncertainty in the scores and the top-k result. Moreover, due to the skew in proximity values in the network, even when α has low non-zero values, the textual component has a strong influence in scores, and thus leads to significant improvements in the top-k estimates (the most likely result).

3.10. Conclusions

We formalized and studied in this chapter the problem of context-aware top-k processing based on uncertain precomputed results, in the form of *views* over the data. This problem is motivated on one hand by search applications in which query results depend on a context, and any result caching or pre-computation mechanism needs to perform certain transformations – what we call a *context transposition* – in order to answer new queries, which may pertain to new contexts. On the other hand, even in context-agnostic search scenarios, some of the most common threshold algorithms, such as NRA, for more efficiency, may output results with score ranges instead of exact scores.

We introduced the query semantics needed for dealing with objects of uncertain scores and describe two algorithms, SR-TA and SR-NRA that support this semantics and are *sound* and *complete*, i.e., they output what we call *the most informative result*: (i) all the guaranteed objects, and (ii) all and only the objects that may appear in the top-k in some data instance. We also consider optimizations for SR-TA and SR-NRA, based on selecting some (few) most promising views, instead of using the entire, potentially very large, set of views. From the most informative result, a probabilistic interpretation can also lead to a *most likely top-k* answer to the input query. Extensive experiments, on both synthetic and real-world data, for spatial and social search, illustrate the potential of our techniques – enabling high-precision retrieval and important running-time savings. More generally, they illustrate the potential of top-k query optimization based on cached results in a wide range of applications.

Importantly, our algorithms provide a *one-size-fits-all solution* for many search applications that are context-dependent, with the only application-dependent aspect being the context transposition.

4. Inferring Signed Networks

We present in this chapter our study on the interaction patterns between contributors in Wikipedia, focusing on the inference and evaluation of an implicit *signed network*.

The first part of this chapter details the general methodology for extracting the signed network and presents its evaluation on a small subset of the Wikipedia articles. We start by detailing how one can extract interactions from the revision history of Wikipedia articles and other community interactions in Section 4.2. The resulting interaction vectors can then be transformed into positive and negative links, using the methodology described in Section 4.3. We motivate our assumptions about the interpretation assigned to each type of interaction by empirically evaluating them in Section 4.4.

There are two possible avenues for evaluating the inferred signed network: by studying the global properties of the network and assessing if they are similar to the properties of explicit signed network (i.e., Epinions, Slashdot), and by measuring the benefits of using the information present in a signed network at the application level. We show that our network exhibits similar characteristics as the explicit networks, both for global properties and edge sign prediction accuracy, in Section 4.5. At the application level, we show in Section 4.6 how using the information about the signed network of users can improve the prediction of two important measures of articles: quality and importance.

The second part of this chapter presents the extraction of the signed network for the entire Wikipedia article corpus in Section 4.7.

4.1. Related Work

Signed networks The study of signed networks (or "webs of trust") in the context of online social networks is a relatively recent research development. Among the first to mention "trust" in an online context, we have studies concentrating on the Semantic Web [66], P2P networks [47], and web spam detection [37], and mainly dealing with a *global* notion of trust. This work is concerned with a *local* notion of trust in a network, in which nodes (representing users, resources, etc.) establish *negative* or *positive* links with other nodes. Examples of such *explicit* signed networks already exist on the Web, and their structure and properties were recently studied: for the Epinions review site [59] and Slashdot [50]. In Wikipedia, Brandes et al. [13] deal with interactions

between contributors of Wikipedia articles, using the concept of an "edit network" to measure the degree of polarization in articles. Leskovec et al. [52] study the properties of Epinions, Slashdot and the Wikipedia election graph, through the lens of two social theories, *balance* [38] and *status*.

Several papers deal with edge sign prediction – an extension of the *link prediction* problem [53] – having an existing signed network as input. Guha et al. [35] use a "trust propagation" model based on four atomic operators: trust transitivity, trust coupling, co-citation and transpose trust. Leskovec et al. [51] use a logistic regression model for link prediction, based on a feature vector consisting of the types of directed triads a link is involved in. More recently, Chiang et al. [18] go beyond triangles, and find that using longer cycles can improve edge sign prediction accuracy.

User reputation in Wikipedia There have been several approaches to measure the worthiness of contributors to Wikipedia. In [2, 1], a contributor reputation system and a measure of trustworthiness of text are derived based on their interactions over Wikipedia content. Another paper that experiments with reputation systems using the editing interactions between contributors is [44].

To the best of our knowledge, this is the first study on inferring an *implicit* signed network directly from user interactions. The work that is closest in spirit to ours uses a semi-supervised approach and existing links to build a predictor of trust-distrust from interactions in Epinions [54]. More recently, a study on the positive and negative votes in Wikipedia [64] and the co-editorship pattern was shown to increase the precision of detecting controversial articles.

4.2. Extracting Interactions from Wikipedia

The main context of interaction in Wikipedia is the collaborative editing of text on articles. However, the community itself is not restricted to such interactions. In order both to keep a minimum editorial standard and to limit the actions of low-quality contributors in the community, the contributors of Wikipedia participate in high-level

Figure 4.1.: The interaction vector (from a generator to a recipient).

interactions that are not directly related to the editing of articles. Project pages, user pages, administrator elections, etc., can serve the purpose of raising the quality level of Wikipedia articles.

We can thus separate the interactions into two main categories: *interactions on article content (text)* and the *community interactions*. We present the extraction methodology for each of them next.

Interactions on article content For measuring the interactions on article content we have extracted the following measures: the amount of words inserted, deleted and replaced in the text of the article and the number of article revisions (i.e., versions) that have been restored or reverted/discarded.

For establishing these interaction metrics we have extracted the full revision history of a corpus of 563 articles from the Politics domain of the English Wikipedia, composed of 910,209 total revisions and a total of 197,798 unique contributors (we have not filtered the anonymous contributors or the Wikipedia bots in this extraction).

The revision of a given article A at time t can be seen as a triple:

$$R_A^t = (auth_A^t, txt_A^t, comm_A^t)$$

composed of the author (or the contributor) who issued the changes on the article, the text resulted from the modification and the comment used by the author to describe the modification.

A contributor $auth_A^t$ has two actions at her disposal: she can either *edit* the text of an article or *revert* the text to a previous version of it. We consider these two actions as independent and mutually exclusive (i.e., the author cannot, at time t, both edit and revert the article).

In order to quantify the interactions between authors, we establish, for each revision, the ownership at word level based on the text difference between two consecutive revisions of an article. This is represented as a list O^t of triples of the form $(owner, \Delta_{start}, \Delta_{end})$ for each revision R^t, consisting of the owner id and the span of her ownership (encoded as deltas in words from the start of the document). This list is created using a text diff algorithm that outputs a list of the operations needed to reach txt_t from txt_{t-1}. These operations represent the amount of text (in words) that $auth^t$ has either deleted, inserted, replaced or kept unchanged. Following this, we establish the new ownership list, as follows:

1. for text inserted and replaced, the new author is $auth^t$ and the deltas are the new positions resulted from the text difference algorithm,

2. for deleted text, the previous author and its positions are removed from O^t and the remaining offsets are updated to account for the missing text.

The interaction thus formed is between the author of the current revision and the owners of the text in the previous revision, as follows:

$$text^t_{auth^t,a} = (ins^t_{auth^t,a}, del^t_{auth^t,a}, rep^t_{auth^t,b})$$

where the components represent a count of the words in each interaction, and $a \in O^t$.

By parsing the text of $comm^t$ we can have an indication of the revision R^t being in fact a reversion to a past revision R^{t-x}. When this is the case, we can form new interactions between contributors. First, a *restore* interaction is defined as the interaction between $auth^t$ and $auth^{t-x}$. Then, the ownership list is reverted to O^{t-x}. Finally, we establish *revert* interactions between $auth^t$ and the authors $\{auth^{t-x-1}, ..., auth^{t-1}\}$. For each possible pair of contributors a and b we represent their interaction at time t in this dimension, as follows:

$$rev\_res^t_{a,b} = (revert^t_{a,b}, restore^t_{a,b})$$

where the components encode how many times each type of interaction has occurred.

In our case, the total number of content interactions (textual and reverts-restores) that were established using this model was 30,670,861.

Community interactions Using the list of unique contributors, resulted from extracting the article history, we can further crawl the pages of Wikipedia that are not articles (in some sense, the metadata of Wikipedia) to retrieve the contributor user pages. We can thus establish if they have participated in the Wikipedia "Requests for Adminship" elections (RFAs), either as voters or as candidates. This list of community interactions is by no means exhaustive, as contributors can participate in a variety of other interactions. An important one represents the debates between the contributors, present on the talk pages attached to the articles. One could also exploit such interactions (e.g., by means of natural language processing and sentiment detection). This goes beyond the scope of this work, but we may follow this direction in future research.

By crawling the pages for RFAs (filtering out the pages for which the candidate is not in our contributor list), we can track the votes cast by the contributors from our list, votes that can be either positive or negative. This election interaction will be represented as follows:

$$election_{a,b} = (vote^+_{a,b}, vote^-_{a,b})$$

where a is the voter and b is the candidate for adminship.

Finally, by crawling the user pages of all the contributors in our list, we can retrieve the Wikipedia *barnstars*. Barnstars are prizes that users can give to each other for perceived valuable contributions and are usually present on the receiver's page. We have thus retrieved the user profile pages of all our contributors and extracted this information, resulting in the $barnstars_{a,b}$ measure which denotes the number of barnstars given by contributor a to contributor b.

Aggregating the interactions For representing the global interaction between a pair of contributors, we have used an aggregation over the interactions. The aggregation was performed by summing the interactions on article content, as follows:

$$text_{a,b} = (\sum_t ins_{a,b}, \sum_t del_{a,b}, \sum_t rep_{a,b})$$

$$rev\_res_{a,b} = (\sum_t revert^t_{a,b}, \sum_t restore^t_{a,b})$$

(Note that we do not need to aggregate the content interactions, as by definition they give the number of times the respective interaction occurred.)

This aggregation yielded a total number of 17,262,082 interactions.

4.3. Building The Signed Network

The four types of interactions presented previously (edits, reverts-restores, election votes and barnstars) can be viewed as an interaction vector from a generator to a recipient. This vector will form the basis for inferring signed edges between users. We describe next how these are further organized and then interpreted as positive or negative units.

Our approach is the following: we give each of the atomic interactions previously identified (text insert, delete and replace; reverts and restores; votes cast and barnstars) a positive or negative interpretation. For instance, for edits on text, we interpret inserts as positive while replacements and deletions of text are seen as negative. Then, the restores of a revision are interpreted as positive interactions, while conversely the reverts of a revision are negative ones. The votes cast in an election are recorded accordingly as positive or negative interactions, while the presence of barnstars is seen as a positive interaction.

Figure 4.1 summarizes the components of this interaction vector and the sign interpretation of each (positive or negative).

Note that these vectors denote *directed interactions*, from a generator to a recipient, and the presence of interactions in one direction does not necessary imply that interactions in the other direction exist.

Then, for deciding a final link sign, for a given pair (a, b) of contributors, we used the following straightforward heuristic. Each atomic interaction votes with its weight (or its magnitude) by the positive or negative interpretation of the higher-level interaction.

For determining the vote of the textual interactions, we have used Kendall's τ coefficient as follows:

$$\tau_{text} = \frac{ins_{a,b} - (del_{a,b} + rep_{a,b})}{ins_{a,b} + del_{a,b} + rep_{a,b}}$$

network	nodes	edges	positive	negative
WikiSigned k=2	138,592	740,397	87.9%	12.1%
WikiSigned k=3	131,544	590,505	86.2%	13.8%
WikiSigned k=4	126,559	497,196	84.5%	15.5%
WikiSigned k=5	123,070	439,644	83.0%	17.0%
textual inter. k=2	73,723	568,488	96.2%	3.8%
non-text inter.	90,188	178,354	60.5%	39.5%

Table 4.1.: WikiSigned features for varying k values.

giving us a measure within the $[-1, 1]$ interval. In order to better control the link formation for textual interactions, we have used a threshold (both positive and negative) on the τ_{text} coefficient for deciding the vote of the textual interaction. We also recorded the size of the textual interaction, $size_{a,b}$ representing the number of different revisions over which the two contributors interacted. We used a parameter k which acts as a threshold on $size_{a,b}$ and regulates when the vote of textual interactions is taken into account. (Note that if one would only be interested in the sign of the interaction on text, computing the difference between the number of words inserted and the number of words replaced and deleted would suffice.)

Reverts and restores vote for the sign of $rev\_res_{a,b}$ and adminship votes for the sign of $election_{a,b}$. The barnstars can only vote positively or be absent from the vote.

Finally, these votes are aggregated into a link sign from a generator to a receiver, by the sign of the sum of the votes of each interaction type.

In our experiments, we have used a threshold value of 0.5 on τ_{text}, a threshold of 10 for the minimum number of words interacted upon and we variated k. We chose as our most representative network the one given by $k = 2$ (the median value for the number of interactions in our entire corpus is 1, meaning that over half of the contributor pairs interacted on text only once).

The WikiSigned network obtained in this way has 138,592 nodes and 740,397 edges, of which 87.9% are positive (a link proportion that is very similar to the ones of the existing signed networks). Please note that our mined election network (which can be seen as an explicit signed network) could not have skewed the results, as the total number of election interactions extracted represents less that 10% of the links of WikiSigned.

We present in Table 4.1 network for $k = 2$ and, for comparison, the ones for other values of k (3, 4 and 5). Also, to better understand the provenance of WikiSigned links, we describe the networks obtained when ignoring the textual interactions or when, instead, using only these interactions (for a value of $k = 2$).

4.4. A Taxonomy of Delete and Replace Interactions in Wikipedia

We turn now our attention to the study of delete and replace interactions in Wikipedia, in order to evaluate the correctness of the assumption that they support a negative interpretation of the interactions in which they appear.

We start by establishing a taxonomy of the possible reasons that may lead to a contributor deleting or replacing the text of another contributor. Depending on the goal of such interactions, we can differentiate the following classes:

1. **Vandalism and anti-vandalism**:
   - *vandalism*: this kind of interaction usually occurs when contributors replace either the entire text of an article with something unrelated to the article, or replace sentences with nonsensical or out-of-context facts,
   - *anti-vandalism*: representing the replacement of vandalised content with new content, without reverting the article to a previous revision,
2. **Deletion of content in an article**:
   - *deletion of entire sentences/paragraphs*: deletions of significant parts of an article, deemed irrelevant by the deleting contributor,
   - *deletion of single words*: i.e., the deletion parts of sentences to improve readability;
3. **Replacement of content**:
   - *reformulations*: used for changing the meaning of a sentence or sequence of words, without deleting it entirely,
   - *updating content*: for cases in which content becomes obsolete, e.g., team rosters for sports teams, episode lists for TV shows, discography lists for artists;
   - *grammar/spelling correction*: small changes that do not change the meaning of a sentence;
4. **Meta-changes**
   - *layout and style changes*: usually, changing the formatting of the article, e.g., putting links to other articles, changing date formats, changing header formats,
   - *tag removal or update*: for instance, removing a protected or redirect tag for an article, adding or changing article categories.

Class	Subclass	Number	True. neg.	Prop.
vandalism	vandal	10	10	1.000
	anti-vandal	7	7	1.000
	total	17	17	1.000
text change	reformulation	19	19	1.000
	update contents	19	16	0.842
	grammar/spelling	8	4	0.500
	total	46	39	0.847
text delete	whole sentence/paragraph	20	20	1.000
	single words	8	7	0.875
	total	28	27	0.964
meta change	style/layout	4	1	0.250
	article tags	5	2	0.400
	total	9	3	0.333
total		**100**	**86**	**0.86**

Table 4.2.: Classification of delete and replace text interactions.

We then performed a manual validation over a random sample of delete and replace interaction. We select, via sampling without replacement, 100 such interactions. We then assign each interaction to a class and subclass, from the ones presented above, and establish if the interaction allows a negative interpretation, by looking at the difference between the texts of the two contributors in the sampled interaction. We call an interaction that allows a negative interpretation a *true negative* interaction. The results of the manual evaluation are presented in Table 4.2.

One can see that the overall ratio of true negative interactions is 0.86, having a corresponding 0.95 confidence interval [79] within $[0.778, 0.914]$. This suggests that it is indeed very likely that such an interaction would allow a negative interpretation. Furthermore, it can be seen that the "meta-change" class has a lower than average true negative rate (indeed, it is more likely for it to contain false negatives than true negatives), as do interactions representing grammar or spelling corrections. This indicates that a reliable way to detect and remove such interactions when constructing the signed network may improve significantly the true negative rate.

triad	count	P(+)	lrn	bal	stat
t_1	2,513,952	0.98	0.1646		
t_2	125,765	0.88	-0.1589		
t_3	2,581,081	0.96	0.0197		
t_4	81,353	0.85	-0.0300		X
t_5	130,225	0.52	-0.3062		
t_6	44,530	0.32	-0.4268	X	
t_7	77,673	0.44	-0.4093		
t_8	39,642	0.34	-0.1849	X	
t_9	3,705,565	0.96	0.0186		
t_{10}	81,629	0.75	-0.2683		
t_{11}	387,386	0.89	-0.0546	X	
t_{12}	48,940	0.71	0.0575	X	
t_{13}	147,869	0.87	0.0201	X	
t_{14}	112,412	0.63	-0.2011	X	
t_{15}	60,768	0.79	0.0817	X	
t_{16}	33,920	0.38	-0.1388	X	X

Table 4.3.: Statistics on triads. The X symbol marks a contradiction with theory.

4.5. Empirical Validation

We present in this section our analysis of the WikiSigned network, testing mainly whether its structural properties are consistent with a signed network. For that, we rely on social theories on the formation of links between individuals, which have been tested in similar online communities, and on comparison with explicit networks. First, at the global level, we studied the properties of WikiSigned in relation to the theories of structural balance and status. Then, at the local level, we studied how accurate an edge sign prediction can be performed on WikiSigned. Finally, we considered the indegree and outdegree distributions of contributors and look into how well they fit into a power-law distribution.

Global properties of WikiSigned We first analyzed the global properties of WikiSigned, checking whether overall it represents a plausible configuration of link signs. For that, we study the role of "link triads" in our signed network. We used a methodology that has already been employed on explicit networks, in [52, 51], allowing us to compare the properties of our network with the existing ones.

A triad represents the composition between the link from A to B and the possible

	Epinions	Slashdot	Elections	WikiSigned
Epinions	0.926	0.905	0.787	0.765
Slashdot	0.929	0.806	0.792	0.716
Elections	0.922	0.895	0.814	0.775
WikiSigned	0.882	0.839	0.755	0.852

Table 4.4.: Predictive accuracy in training on the row data and testing on the column data.

links to a third party node X. Depending on the direction and sign of the link connecting A, and B, with X, there are sixteen such types of triads.[1] Some of the triads are representations of well-known social scenarios: t_6 is a representation of "the enemy of my enemy", t_1 of "trust transitivity", t_9 is a triad in which X points positively to both A and B. Figure 2 illustrates these triads.

We looked at the distribution of link triads and the proportion of positive $A - B$ links in each type of triad. We found that both measures are very similar with the ones reported in [52] (see columns *count* and *P(+)* in Table 4.3).

Next, we studied the configuration of our network in comparison with two social theories, *status* and *balance*, theories that aim to define and predict the formation of links between individuals. *Structural balance theory* posits that triads which are "balanced" (i.e., have either one or three positive link signs, in an undirected sense) are more prevalent in real-world networks that the other types of triads [38]. *Status* posits that a directed negative link between A and B means that A regards B as having lower "status", while a positive link mean that A regards B as having higher "status" [35]. As such, for the network to have the same properties as the ones predicted by balance theory, in triads t_1, t_3, t_6, t_8, t_9, t_{11}, t_{14} and t_{16} the $A - B$ link should be positive, while it should be negative in the rest of the triad types. For a network to be in line with status, triads t_1, t_4, t_{13}, t_{16} should have a positive $A - B$ link and triads t_6, t_7, t_{10}, t_{11} should have a negative one.[2]

Link prediction in signed networks has been studied in [51], by training a link prediction model using logistic regression learning on a feature vector consisting of the total number of triads of each type that the link participates in. We have used the same methodology for training the model on WikiSigned, with 10-fold cross-validation and a balanced set of negative and positive links with a minimum link embeddedness of

[1] As in [52], we encode them by a summation starting at 1 and adding 8 for the $A - X$ link if it is pointing backwards then 4 if the link is negative; for the $X - B$ link we add 2 if the link is backward and 1 if the link is negative. This gives us the id of the triad.

[2] Status predicts link signs only for these triads.

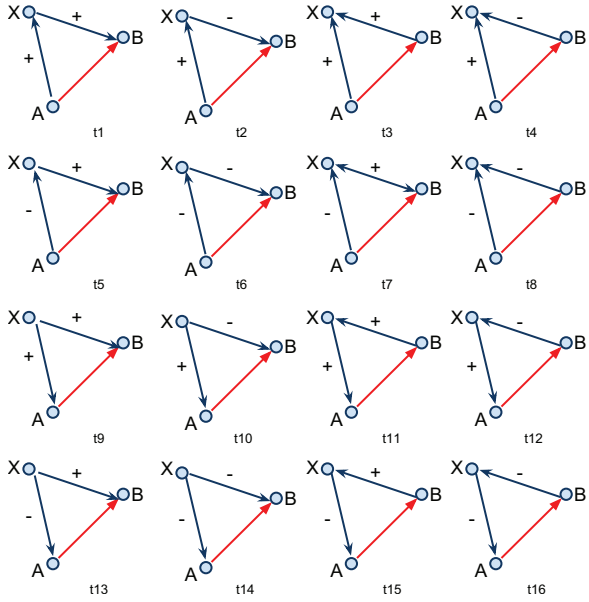


Figure 4.2.: The concept of directed link triad.

25 (i.e., the total number of triads in which each link participates). Since the positive links represent a large majority of the link signs, naively predicting all link signs as positive would have an accuracy of 0.879 (i.e., the proportion of the positive links in the network). To avoid this bias, we have randomly selected as our training set 5000 edges for each link sign, via reservoir sampling.

The signs of the coefficients of the trained model are an indication of the influence that each triad type has on the final link sign. Hence, we can compare these signs with the predictions of the two social theories. We perform this comparison on WikiSigned, counting the contradictions with these theories, i.e., the differences between the sign of the learned coefficients for each triads and the prediction of the two social theories. We find that at a global level our interaction-based network is more consistent with the theory of status (two contradictions with the theory, in t_4 and t_{16}) similar to what has been observed in [52] on the Wikipedia election network (see columns *lrn*, *bal* and *stat* in Table 4.3; *X* marks a contradiction with the social theory in the column).

Local properties of WikiSigned For the local properties analysis of WikiSigned, using the same link prediction model, we tested the accuracy of predicting link signs. The predictive accuracy thus obtained was of 0.852 with an AUC of 0.924.

Furthermore, to better understand how WikiSigned relates structurally to the explicit networks, we have also applied this learning methodology over the three explicit networks considered in [51], asking the following question: how well a predictor

learned on one network performs when applied on another network (see Table 4.4). First, one can notice that our results that use and apply to explicit networks are almost identical to the ones reported in [52]. WikiSigned performs better than the election network, in that prediction on itself is worse than self-prediction over Epinions and Slashdot, while learning the predictor on WikiSigned and applying it on both Epinions and Slashdot yields good prediction rates, the inverse performing slightly worse. All this indicates that these networks have quite similar characteristics at the local level, even though our network is inferred from interactions while the other three are explicitly declared by users.

Fitting into power-law distributions Previous work has found that the distribution of outdegrees and indegrees of nodes in online social networks generally follows a power-law curve [61]. We measured a similar aspect, defining an *absolute degree* of a node as the difference between the positive and negative links pointing to or from that node. For inlinks or outlinks only, we have the related concepts of *absolute indegree* and *absolute outdegree*.

We have plotted the complementary cumulative distribution functions (CCDF) on a log-log scale for the two explicit networks of Epinions and Slashdot and for WikiSigned. The results are presented in Figure 4.3.

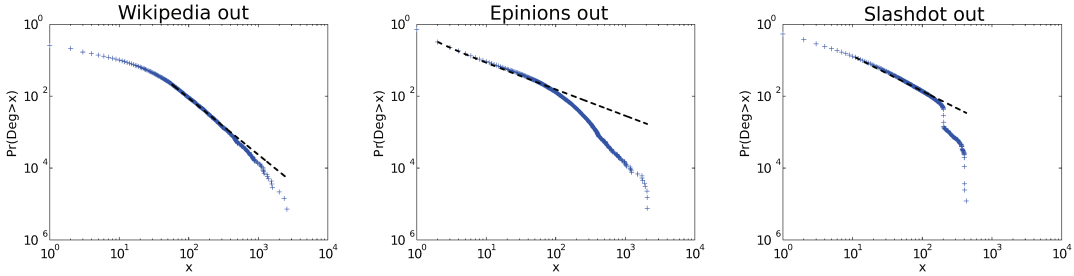
One can see that all the tested networks exhibit power-law characteristics, with various exponents. In the case of WikiSigned, one can see that the distribution follows closely the ideal power-law fit (the dotted line), while in real networks, for big absolute indegrees and outdegrees the curve is below the ideal fit. In our view, this could be due to the fact that the link inference for WikiSigned is based on voting, without including potentially complex link formation conditions that may occur in explicit networks (for instance, the formation of links using the knowledge about other links).

4.6. Exploiting WikiSigned at the Application Level

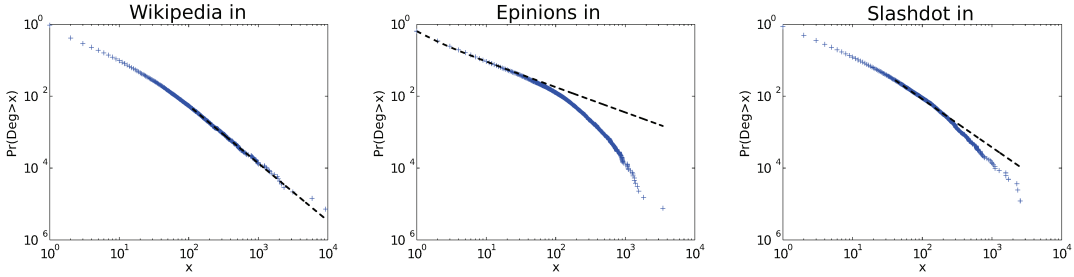
We also investigated the usefulness of having the signed network in applications, by considering how link structure can be exploited in the classification of articles. There are two article features that are explicit on the homepage of the Wikipedia Politics project[3]: the article quality and the article importance (or priority). In our dataset, we have articles that span the top 5 article qualities (Featured Articles, Great Articles, A-class Articles, B-class articles and C-class articles) and all the importances (Top, High, Mid, Low).

For our experiments, we have separated the article qualities and importances into two classes (top-tier and bottom-tier). For the article importance, we have considered

[3]http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Politics



Absolute outdegree: WikiSigned, Epinions and Slashdot.



Absolute indegree: WikiSigned, Epinions and Slashdot.

Figure 4.3.: Absolute indegree and outdegree power-law fits and CCDFs for different signed networks.

the Top, High, Mid as the top tier and the Low importance as the bottom tier, randomly sampling 150 articles for each. As the A-class of articles contains only 8 articles, we have excluded this class for the training, and we have randomly sampled 50 articles from each remaining class. Furthermore, we have categorized as top-tier the FA and GA articles and the B and C-class articles as bottom-tier. This resulted in two equally balanced datasets: 100 for each article quality tier, and 150 for each article importance tier.

We have used the following set of features for each article: the number of *authors*; three features (total, positive and negative) for each of the following: *outgoing links* (links from the authors towards other contributors), *incoming links* (the links from other contributors towards the authors) and *inside links* (links from authors to authors); and the following information about the contributors in the article: the number of *incoming* total positive and negative links (in the entire networks) for the contributors of the article, how many of them have more positive links than negative and vice-versa. The same information is also extracted for *outgoing* links, giving us a total of 18 features for our article prediction model.

We report the predictive accuracy we obtained via logistic regression in Table 4.5. Following the intuition that more important articles have a larger participation and thus more links, we tested the predictive power of these two values (*contributors* and

type	Importance	Quality
contributors	0.691	0.518
contribs.+links	0.743	0.835
contribs.+ soc links	0.749	0.895
contribs.+ soc links + rep.	**0.756**	**0.935**

Table 4.5.: Predictive rates for article importance and article quality.

contribs.+links) alone. We found that, while using knowledge about positive or negative links in separation does not provide better accuracy, their combination yields better results (*contribs.+soc. links*). This suggests that the characteristics of an article are not defined solely by the number of contributors, but also by their relationships with other Wikipedia contributors. When we also introduce the information about the contributors (*contribs.+ soc links + rep.*), we see further improvement, especially in the case of quality, which seems to support the intuition that the quality of an article is determined by the "quality" of its contributors.

4.7. Extracting WikiSigned from the Complete History of Wikipedia

We present in this section the extension of the approach previously described, to the entire article corpus of the English Wikipedia, discussing in particular the scalability challenges of such an analysis at an unprecedented scale.

The English Wikipedia contains around 27 million pages, each having an average of 19 revisions[4]. Obviously, this amount of revisions cannot be feasibly processed on a single machine, sequentially. Hence, for processing this amount of data, we have to use distributed computing frameworks, such as MapReduce. Figure 4.4 presents the sequential chain tasked with building WikiSigned from a list of articles. It is composed of the following jobs:

1. **Extracting the text interactions.** The first job in the MapReduce chain is the extraction of text interactions, using the methodology presented in Section 4.2. Starting from an input consisting of a list of articles, each mapper is tasked with extracting the entire revision history of an article, via API calls. Then, via the previously described procedure, interactions between contributor pairs are computed for each revision. The mapper then presents two inputs to the reducers:

[4] `http://en.wikipedia.org/wiki/Special:Statistics`

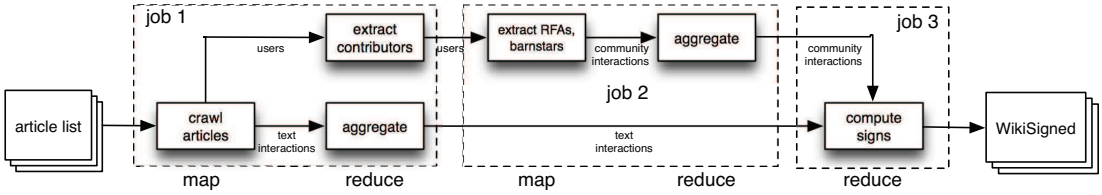


Figure 4.4.: MapReduce chain for extracting WikiSigned.

one containing contributor pairs and their interaction on a revision, and a list of contributors detected by the mapper.

One reducer is then tasked with the aggregation of the interactions for each user pair, while the second generates a list of unique contributors to be used in the next job of the chain.

2. **Extracting the barnstars and election data.** The second job in the chain takes as input the contributor list, and then parses the HTTP sources of election and contributor profile pages [5], thus establishing the community interaction vector. As in the case of the text interactions, the reducer aggregates, via summation, the interactions of unique contributor pairs.

3. **Computing WikiSigned.** Finally, the text and community interactions are given as an input to the final job, consisting of a reducer that implements the methodology described in Section 4.3, applying the described thresholds and parameters to output the final WikiSigned network.

We have extracted the English Wikipedia corpus starting from the article list publicly available on Wikipedia, using a MapReduce cluster on the Amazon Web Services platform and our own cluster of 32 machines. Starting from a list of 5,263,187 articles, we have extracted a number of 261,663,028 revisions, resulting in 8,003,495,285 interaction vectors. For these vectors, WikiSigned networks were built for various thresholds, and we present in Table 4.6 the properties of the resulting WikiSigned networks, for $k \in \{2, \ldots, 5\}$. One can note that the signed networks are around two orders of magnitude bigger than the ones built from the Politics dataset.

For the prediction of edge signs, the single-machine algorithms we have used for the smaller scale networks are no longer usable for such network sizes. To deal with this, we have modified the distributed triangle counting algorithm for undirected graphs, presented in [74], to handle directed signed edges. For the logistic regression algorithm, we have implemented the one presented in [21].

[5] User profile, barnstar and election data are not available via API calls, so they have to be extracted separately.

k	nodes	edges	positive	negative
2	13,820,607	99,357,735	88.2%	11.8%
3	12,446,260	72,327,442	84.9%	15.1%
4	11,538,482	57,255,302	81.6%	18.4%
5	10,933,199	48,578,600	78.7%	21.3%

Table 4.6.: WikiSigned properties for the entire corpus of English articles.

While the full-scale evaluation of this network is under way, we can already report that, even for this network size, the logistic regression model for edge sign prediction still maintains a reasonable precision of **0.789**.

4.8. Conclusion

We have presented in this chapter our approach to extract an implicit signed network of Wikipedia contributors, by considering the various interactions they are involved in. We have shown that the implicit signed network extracted from a subset of Wikipedia articles, named WikiSigned, constitutes a plausible signed network – it has similar properties as explicit signed networks on the Web. We have also studied the link prediction capabilities of WikiSigned, by cross training and testing with the explicit signed networks. Furthermore, we have shown that the presence of a signed network can significantly contribute to the accuracy of predicting important article features, such as quality and importance.

Finally, we have taken first steps towards the extraction and validation of the signed network for the editors of the entire Wikipedia corpus, establishing the processing chain for computing a network of such unprecedented size and showing that the edge sign prediction accuracy remains at a reasonable level (only 5% less accurate that the small-scale network).

5. Research Perspectives

In this chapter, we discuss some of the most promising research directions stemming from the research performed during this thesis, for the three main problems studied in this thesis: social search, context-aware search using views, and signed social networks.

5.1. Social Search

Other similarity functions for computing user proximities A next step in this research is to exploit more involved social functions for the computation of the seeker-dependent proximities (or similarities). Random-walk like approaches are another natural candidate for this type of functions, as they provide a principled definition for the notion of proximity/similarity between two users.

One research avenue is to also include in the model measures from the literature, like Personalized PageRank [46] and SimRank [45]. Adapting such measures to our setting represents a significant challenge, stemming from the fact that top-k algorithms exploit the access to already sorted lists of scores to achieve early termination. A shortest-path algorithm, as used in our approach to social search, allows the adaptation of such algorithms, as it generates sorted lists of proximities as needed. It is not clear whether the two measures cited (or similar ones) could be adapted to fit such a requirement. Exact computations are probably unfeasible, but fast approximate algorithms (as in [8], for instance) seem more feasible.

Community search in a social-aware interpretation We are also interested in the search scenario in which one wishes to compute the representative social-aware top-k for communities of users. In this case, an aggregation of the social-aware top-k results for the members of the targeted community needs to be computed, a problem related to the well-known rank aggregation problem [28, 30, 29]. The main challenge to applying rank aggregation in the social-aware context is that one is not presented with full rankings, as their computation can be expensive (it may, for instance, involve visiting a large portion of the social network). Instead, rank aggregation has to be computed in a dynamic, online manner. Inevitably, the aggregated top-k would be an approximation, preferably with controllable parameters, of the aggregation of the full rankings.

5.2. Context-Aware Search Using Views

Optimal context-aware top-k algorithms We have identified a *sufficient* class of views for which the instance optimality properties of the threshold algorithms hold even in the case of context-aware views. It is, however, not clear whether this class of views is also *necessary* for instance optimality. Hence, a thorough study of optimal algorithms in the context of the uncertainty induced by the views would be interesting to pursue. Instance optimality, as defined by Fagin et al. [31] might be hard to achieve in this context. However, it would be interesting to see if relaxing the optimality definition, to allow for non-constant factors, would lead to reasonable guarantees.

5.3. Signed Networks

Applications for signed networks While detecting the existence of an implicit signed network, be it a trust or a similarity/dissimilarity network, is an interesting result in itself, the bigger challenge lies in finding applications that can fully use the power of such networks. Examples include recommendation systems, content appraisal or even search. In order to support such applications, a first step is to establish a principled approach to the problems of ranking and classifying nodes or data in a signed graph. Two directions seem promising:

1. *Ranking nodes in signed networks.* This direction requires a revisit of current algorithms for ranking nodes, like PageRank or TrustRank, and several approaches that have been proposed lately for this problem (see [75, 60]) may constitute a starting point.

2. *Clustering nodes in signed networks.* This direction strongly resembles the problem of Correlation-Clustering [62], an optimization problem known to be NP-complete. While the study of this problem is quite mature, with several approximation algorithms proposed in recent years [25, 3], the majority of these algorithms would not scale to Web-sized (or Wikipedia-sized) graphs. Moreover, it is still not clear whether the direct application of such algorithms to signed networks would reflect real-world social communities.

Social search in signed networks A natural way of bridging our studies on social search, on one hand, and signed networks, on the other hand, is to consider social search in a setting where relationships may be signed. In this case, relevance of results becomes a key issue, as it remains to be clarified how negative edges can guide the search towards more relevant results. Adapting graph algorithms to the signed network setting, and more generally to social settings, going beyond link/no-link models, raises many research challenges.

Appendix A.

Other Collaborations

S-Cores: Degeneracy Based Evaluation of Trust in Signed Directed Graphs [CGM+13] While many algorithms for mining social networks have been proposed in the past, the concept of degeneracy in the context of community evaluation in signed networks has been ignored so far in the literature. This paper introduces the S-core, a novel extension of the k-core structure for modelling degeneracy in signed graphs.

We define various network metrics based on S-cores, which can be used for the evaluation of robustness and other trust-based measures. Additionally, we define reciprocity in the context of signed networks.

We present an extensive experimental evaluation, on both existing signed networks in Web applications and signed networks inferred from interactions among editors of the Wikipedia. We evaluate trust in different thematic domains of Wikipedia, by looking at their editors' S-core features. Our experimental results indicate that S-cores can be a robust tool for evaluating trust in signed networks.

This work is the result of a collaboration with C. Giatsidis, M. Vazirigiannis of LIX, École Polytechnique and D. Thilikos of National and Kapodestrian University of Athens.

Search Behaviour on Photo Sharing Platforms [MOA+13] The behaviour, goals, and intentions of users while searching for images in large scale online collections are not well understood. There has been some work on the analysis of image search logs, yet the insights that they can provide are limited, due to the fact that one does not have access to all the user actions after querying.

We study user behaviour while searching for photos in a large photo-sharing platform, characterise the types of behaviour when searching in such a platform, and provide insights into user behavior during image search. In particular, we show how user click behaviour is influenced by the type of the query, going beyond the clicking on result images. We show how classes of users of the platform display significantly different characteristics, both in click and querying behaviour. Finally, we provide an analysis on how users behave when they reformulate their queries.

This work is the result of collaborating with the Social Media Engagement team of Yahoo! Research Barcelona, during a summer research internship.

Self References

[CGM+13] Bogdan Cautis, Christos Giatsidis, Silviu Maniu, Dimitrios M. Thilikos, and Michalis Vazirigiannis. S-Cores: degeneracy based evaluation of trust in signed directed graphs. *Submitted for publication in WSDM*. 2013.

[MAC11] Silviu Maniu, Talel Abdessalem, and Bogdan Cautis. Casting a web of trust over Wikipedia: an interaction-based approach. In *WWW*, 2011.

[MC12] Silviu Maniu and Bogdan Cautis. Taagle: efficient, personalized search in collaborative tagging networks. In *SIGMOD*, 2012.

[MC13] Silviu Maniu and Bogdan Cautis. Context-aware top-k processing using views. *Preliminary version published in BDA 2012. Submitted for publication in ICDE*. 2013.

[MCA11] Silviu Maniu, Bogdan Cautis, and Talel Abdessalem. Building a signed network from interactions in Wikipedia. In *DBSocial*, 2011.

[MCA12] Silviu Maniu, Bogdan Cautis, and Talel Abdessalem. Enabling network-aware search in online social bookmarking applications. *Preliminary version published in BDA 2011. Submitted for publication in IEEE TKDE*. 2012.

[MOA+13] Silviu Maniu, Neil O'Hare, Luca Maria Aiello, Luca Chiarandini, and Alejandro Jaimes. Search behaviour on photo sharing platforms. *Submitted for publication in WSDM*. 2013.

External References

[1] B. T. Adler, K. Chatterjee, L. de Alfaro, M. Faella, I. Pye, and V. Raman. Assigning trust to Wikipedia content. In *WikiSym*, 2008.

[2] B. T. Adler and L. de Alfaro. A content-driven reputation system for the Wikipedia. In *WWW*, 2007.

[3] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 2008.

[4] S. Amer-Yahia, M. Benedikt, L. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. In *VLDB*, 2008.

[5] S. Amer-Yahia, J. Huang, and C. Yu. Building community-centric information exploration applications on social content sites. In *SIGMOD*, 2009.

[6] S. Amer-Yahia, L. Lakshmanan, and C. Yu. Socialscope: Enabling information discovery on social content sites. In *CIDR*, 2009.

[7] Apache. Apache Hadoop. `http://hadoop.apache.org`.

[8] B. Bahmani, A. Chowdhury, and A. Goel. Fast incremental and personalized PageRank. In *PVLDB*, 2010.

[9] B. Bahmani and A. Goel. Partitioned multi-indexing: Bringing order to social search. In *WWW*, 2012.

[10] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *WWW*, 2007.

[11] A. Bernstein. Fully dynamic $(2 + \epsilon)$ approximate all-pairs shortest paths with fast query and close to linear update time. In *FOCS*, 2009.

[12] B. Bi, S. Lee, B. Kao, and R. Cheng. An effective and efficient method for searching resources in social tagging systems. In *ICDE*, 2011.

[13] U. Brandes, P. Kenis, J. Lerner, and D. van Raaij. Network analysis of collaboration structure in Wikipedia. In *WWW*, 2009.

[14] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. In *PVLDB*, 2010.

[15] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, 2011.

[16] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'el, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *CIKM*, 2009.

[17] L. J. Chen and Y. Papakonstantinou. Supporting top-k keyword search in XML databases. In *ICDE*, 2010.

[18] K.-Y. Chiang, N. Natarajan, A. Tewari, and I. S. Dhillon. Exploiting longer cycles for link prediction in signed networks. In *CIKM*, 2011.

[19] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On compressing social networks. In *KDD*, 2009.

[20] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel. Text vs. space: Efficient geo-search query processing. In *CIKM*, 2011.

[21] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-Reduce for machine learning on multicore. In *NIPS*, 2006.

[22] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. In *VLDB*, 2009.

[23] G. Das, D. Gunopulos, N. Koudas, and D. Tsirogiannis. Answering top-k queries using views. In *VLDB*, 2006.

[24] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI*, 2004.

[25] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 2006.

[26] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959.

[27] Z. Dou, R. Song, and J. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW07*.

[28] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, 2001.

[29] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *PODS*, 2004.

[30] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *SIGMOD*, 2003.

[31] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[32] Google. Google web toolkit. `http://developers.google.com/web-toolkit/`.

[33] Google. An update to Google social search. `http://www.googleblog.com/2011/02/update-to-google-social-search.html`, 2011.

[34] L. Gou, X. L. Zhang, H.-H. Chen, J.-H. Kim, and C. L. Giles. Social network document ranking. In *JCDL*, 2010.

[35] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW*, 2004.

[36] A. Guttman. R-tree: a dynamic index structure for spatial searching. In *SIGMOD*, 1984.

[37] Z. Gyöngi, H. G. Molina, and J. Pedersen. Combating web spam with TrustRank. In *VLDB*, 2004.

[38] F. Heider. Attitudes and cognitive organization. *Journal of Psychology*, 1946.

[39] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *WSDM*, 2008.

[40] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *WWW*, 2010.

[41] A. Hotho, R. JÃd'schke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *ESWC*, pages 111–114, 2006.

[42] V. Hristidis and Y. Papakonstantinou. Algorithms and applications for answering ranked queries using ranked views. In *VLDBJ*, 2004.

[43] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. In *ACM Comp. Surv.*, 2008.

[44] S. Javanmardi, C. Lopes, and P. Baldi. Modeling user reputation in wikis. *Stat. Anal. Data Min.*, 2010.

[45] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *KDD*, 2002.

[46] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, 2003.

[47] S. D. Kamvar, M. T. Schlosser, and H. G. Molina. The eigentrust algorithm for reputation management in p2p netwokrs. In *WWW*, 2003.

[48] I. Konstas, V. Stathopoulos, and J. Jose. On social networks and collaborative recommendation. In *SIGIR09*.

[49] R. Kumar, K. Punera, T. Suel, and S. Vassilvitskii. Top-k aggregation using intersections of ranked inputs. In *WSDM*, 2009.

[50] J. Kunegis, A. Lommatzsch, and C. Bauckhage. The Slashdot Zoo: mining a social network with negative edges. In *WWW*, 2009.

[51] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *WWW*, 2010.

[52] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *CHI*, 2010.

[53] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. In *JASIST*, 2007.

[54] H. Liu, E. Lim, H. W. Lauw, M. Le, A. Sun, J. Srivastava, and Y. Kim. Predicting trusts among users of online communities: an Epinions case study. In *EC*, 2008.

[55] S. Maniu and B. Cautis. Taagle: Efficient, personalized search in collaborative tagging networks. In *SIGMOD*, 2012.

[56] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[57] A. Marian, S. Amer-Yahia, N. Koudas, and D. Srivastava. Adaptive processing of top-k queries in XML. In *ICDE*, 2005.

[58] A. A. Markov. On certain applications of algebraic continued fractions. In *PhD Thesis, St. Petersburg*, 1884.

[59] P. Massa and P. Avesani. Controversial users demand local trust metrics: an experimental study on epinions.com. In *AAAI*, 2005.

[60] A. Mishra and A. Bhattacharya. Finding the bias and prestige of nodes in networks based on trust scores. In *WWW*, 2011.

[61] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, 2007.

[62] B. Nikhil, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 2004.

[63] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publications, 1998.

[64] H. S. Rad, A. Makazhanov, D. Rafiei, and D. Barbosa. Leveraging editor collaboration patterns in Wikipedia. In *HT*, 2012.

[65] C. Ré, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[66] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *ISWC*, 2003.

[67] J. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Norvag. Efficient processing of top-k spatial preference queries. In *VLDB*, 2010.

[68] L. Roddity and U. Zwick. On dynamic shortest paths problems. In *Algorithmica*, 2010.

[69] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, and G. Weikum. Efficient top-k querying over social-tagging networks. In *SIGIR*, 2008.

[70] F. Scholer, H. Williams, J. Yiannis, and J. Zobel. Compression of inverted indexes for fast query evaluation. In *SIGIR*, 2002.

[71] P. Singla and M. Richardson. Yes, there is a correlation - from social networks to personal behavior on the web. In *WWW*, 2008.

[72] M. A. Soliman, I. F. Ilyas, and S. Ben-David. Supporting ranking queries on uncertain and incomplete data. In *VLDBJ*, 2010.

[73] B. Suh, G. Convertino, E. H. Chi, and P. Pirolli. The singularity is not near: Slowing growth of Wikipedia. In *WikiSym*, 2009.

[74] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *WWW*, 2011.

[75] V. A. Traag, Y. E. Nesterov, and P. V. Dooren. Exponential ranking: taking into account negative links. In *SocInfo*, 2010.

[76] Twitter. Twitter turns six. `http://blog.twitter.com/2012/03/twitter-turns-six.html`, 2012.

[77] J. Wang, M. Clements, J. Yang, A. P. de Vries, and M. J. T. Reinders. Personalization of tagging systems. *Inf. Process. Manage.*, 2010.

[78] R. Wetzker, C. Zimmermann, and C. Bauckhage. Analyzing social bookmarking items: A del.icio.us cookbook. In *ECAI Mining Social Data Workshop*, 2008.

[79] E. B. Wilson. Probable inference, the law of succession, and statistical inference. In *J. American Statistical Association*, 1927.

[80] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. In *SIGIR*, 2008.

[81] H. Yan, S. Ding, and T. Suel. Inverted index compression and query processing with optimized document ordering. In *WWW*, 2009.

[82] P. Yin, W.-C. Lee, and K. C. Lee. On top-k social web search. In *CIKM*, 2010.

[83] J. Zhang, X. Long, and T. Suel. Performance of compressed inverted list caching in search engines. In *WWW*, 2008.

[84] J. Zobel and A. Moffat. Inverted files for text search engines. In *ACM Comp. Surv.*, 2006.

[85] M. Zukowski, S. Heman, N. Nes, and P. Boncz. Super-scalar RAM-CPU cache compression. In *ICDE*, 2006.