Adaptive Methods for User-Centric Information Access Applications

Paul Lagrée

NNT: 2017SACLS341

Thèse de doctorat de l'

Université Paris-Saclay

Préparée à l'

Université Paris-Sud

École Doctorale — 580
Sciences et Technologies de l'Information et de la Communication (STIC)
Laboratoire de Recherche en Informatique (LRI)
Spécialité Informatique

Présentée par

M. Paul Lagrée

Intitulée

Méthodes Adaptatives pour les Applications d'Accès à l'Information Centrées sur l'Utilisateur

Thèse présentée et soutenue à Orsay, le 12 octobre 2017

Composition du jury :

Fabian	Suchanek	Professeur	Télécom ParisTech, U. Paris-Saclay	Président
Ludovic	Denoyer	Professeur	UPMC U. Paris 06, LIP6	Rapporteur
Pierre	Senellart	Professeur	École Normale Supérieure	Rapporteur
Aurélien	Garivier	Professeur	U. Paul Sabatier, IMT	Examinateur
Stratis	Ioannidis	Prof. Assistant	Northeastern University	Examinateur
Themis	Palpanas	Professeur	U. Paris Descartes, LIPADE	Examinateur
Olivier	Cappé	Dir. de Recherche	CNRS, U. Paris-Saclay, LIMSI	Co-directeur
Bogdan	Cautis	Professeur	U. Paris-Sud, U. Paris-Saclay, LRI	Directeur

9 décembre 2017 — version 1.2

SUPERVISORS:
Professor Bogdan Cautis
Professor Olivier Cappé

LOCATION:
2, Rue Madeleine Brès, 75013 — Paris, France

SYNTHÈSE EN FRANÇAIS

Au cours des deux dernières décennies, la gestion des données est devenue un sujet centré sur l'utilisateur : les utilisateurs se sont progressivement transformés de simples consommateurs de contenu en producteurs et juges du contenu. Nous avons assisté à l'émergence d'une pléthore de systèmes, en particulier sur le Web, sur lesquels les utilisateurs contribuent, accèdent à l'information, évaluent et interagissent dans des environnements complexes, explicitement ou implicitement. Lorsque les utilisateurs interagissent sur ces systèmes, ils laissent de nombreuses empreintes que nous nous proposons d'exploiter pour développer de meilleures applications d'accès à l'information. Nous étudions une famille de techniques centrées sur les utilisateurs, qui tirent parti des nombreux types de rétroaction pour adapter et améliorer les services offerts aux utilisateurs. Nous nous concentrons sur des applications telles que la recommandation et le marketing d'influence dans lesquelles les utilisateurs génèrent des feedbacks réguliers (par ex. des clics, des *likes*, etc.) et nous les intégrons dans nos algorithmes afin de fournir des services fortement contextualisés aux utilisateurs. Il est important de noter que, dans les applications considérées, nos algorithmes doivent faire face à l'incertitude concernant soit l'utilisateur à qui ils proposent le contenu, soit le contenu lui-même, et parfois les deux à la fois.

La première partie de cette thèse est consacrée à une approche de recherche à la carte sur les médias sociaux. Le problème consiste à récupérer un ensemble de k résultats de recherche dans un environnement social sous la contrainte que la requête peut être incomplète (par exemple, si le dernier terme est un préfixe). Ce problème est abordé à travers le prisme de la recommandation. Chaque fois que l'utilisateur met à jour sa requête, le système met à jour l'ensemble des résultats de recherche en conséquence, afin d'améliorer l'expérience utilisateur sur la plate-forme sociale. Nous adoptons une interprétation de la pertinence de l'information qui tient compte du réseau, selon laquelle l'information produite par les utilisateurs qui sont plus proches de l'utilisateur qui fait une demande est jugée plus pertinente. Ce modèle de recherche soulève des défis pour l'efficacité et l'efficience de la recherche en ligne.

Dans la deuxième partie de la thèse, nous étudions une version générique de la maximisation de l'influence, dans laquelle nous voulons maximiser l'influence des campagnes d'information ou de marketing en sélectionnant de manière adaptative les utilisateurs initiant la propagation de l'information parmi un petit sous-ensemble de la population. Notre approche ne fait aucune hypothèse sur le modèle de diffusion sous-jacent ni même sur la structure du réseau de diffusion. Notre méthode a d'importantes applications dans le marketing d'influence qui vise à s'appuyer sur les influenceurs de réseaux sociaux pour promouvoir des produits ou des idées.

Enfin, nous abordons le problème bien connu du démarrage à froid auquel sont confrontés les systèmes de recommandation par une approche adaptative. Si aucune information n'est disponible concernant l'appréciation d'un article, le système de recommandation doit recueillir des signaux (clics, etc.) afin d'estimer la valeur de l'article. Cependant, afin de minimiser les mauvaises recommandations faites aux utilisateurs, le système ne doit pas recueillir ces signaux de façon négligente. Nous introduisons un algorithme dynamique qui vise à alterner intelligemment les recommandations visant à accumuler de l'information

et celles s'appuyant sur les données déjà recueillies. Notre approche via les bandits multi-bras se propose d'exploiter les informations disponibles concernant le biais d'affichage sous le modèle de clic dit *position-based*.

ABSTRACT

In the last two decades, data management has become a user-centric subject: users have gradually transformed themselves from simple content consumers into producers and judges of the content. We have seen the emergence of a plethora of systems, especially on the Web, on which users contribute, access information, evaluate, and interact in complex environments, either explicitly or implicitly. When users interact on these systems, they let numerous footprints which we propose to exploit so as to develop better applications for information access. We study a family of techniques centered on users, which take advantage of the many types of feedback to *adapt* and improve services provided to users. We focus on applications like recommendation and influencer marketing in which users generate *discrete* feedback (e.g. clicks, "likes", reposts, etc.) and we incorporate them in our algorithms in order to deliver strongly contextualized services. Importantly, in the applications considered in this dissertation, our algorithms have to face uncertainty regarding either the user it proposes content to or the content itself, and sometimes both.

The first part of this dissertation is dedicated to an approach for as-you-type search on social media. The problem consists in retrieving a set of k search results in a social-aware environment under the constraint that the query may be incomplete (e.g., if the last term is a prefix). This problem is addressed through the prism of adaptive contextual recommendation. Every time the user updates his / her query, the system updates the set of search results accordingly, so as to improve the user experience on the social platform. We adopt a "network-aware" interpretation of information relevance, by which information produced by users who are closer to the user issuing a request is considered more relevant. This query model raises challenges for effectiveness and efficiency in online search.

In the second part of the dissertation, we study a generic version of influence maximization, in which we want to maximize the influence of information or marketing campaigns (e.g., on social media) by adaptively selecting "spread seeds" from a small subset of the population. Influencer marketing is a straightforward application of this, in which the focus of a campaign is placed on precise key individuals – the influencers – who are typically able to reach millions of consumers through their blog or social platform's personal page. This represents an unprecedented tool for online marketing that we propose to improve using an adaptive approach. Notably, we make no assumptions on the underlying diffusion model and we work in a setting where neither a diffusion network nor historical activation data are available. We choose to address this task using an approach similar to that of multi-armed bandits.

Finally, we propose to address the cold start problem: a well-known issue faced by recommender systems when new items are introduced to the pool of items they recommend from. If no information is available regarding the user appreciation of an item, the recommender system needs to gather feedback – e.g., clicks / non-clicks – so as to estimate the value of the item. However, in order to minimize "bad" recommendations and to maintain the best user experience possible, a well-designed system should not collect feedback

carelessly. We introduce a dynamic algorithm that aims to intelligently achieve the balance between "bad" recommendations – which are necessary to gather more information so as to have a better understanding of user evaluations of items – and "good" recommendations. Our multi-armed bandit approach proposes to exploit available information regarding the display bias under the so-called position-based click model.

CONTENTS

1

INTRODUCTION

In this introductory chapter, we present the objectives, the motivations and the challenges of this dissertation. We focus on the *information access* aspect of the ALICIA project, and, in particular, on adaptive recommendation in user-centric environments. We provide a summary of the three main contributions of this dissertation, namely, (i) a method to adaptively propose items matching a search need of a user as *he/she is typing his/her query* on social media, (ii) an approach to *adaptively* select users from a subpopulation of influencers who have access to a diffusion media (e.g. a social network) in order to maximize the impact of a diffusion campaign (be it for marketing, politics, etc.), (iii) an algorithm to sequentially select which items to display in cold-start recommendation scenarios where the display of the page exhibits strong position bias.

The research of this thesis took place at the Laboratoire de Recherche en Informatique (LRI) under the supervision of Bogdan Cautis and Olivier Cappé. I also had an office at Télécom ParisTech where part of the research was done. During the 3 years of the Ph.D., I was funded by the French research project "Adaptive Learning for Intelligent Crowdsourcing and Information Access" (ALICIA)[1].

Contents

1.1 OBJECTIVES

In the last two decades, data management has gradually become a user-centric subject: users are transforming themselves from simple content consumers into producers and judges of the content. We have seen the emergence of a plethora of systems, especially on the Web, on which users contribute, access information, evaluate, and interact in complex environments, either explicitly or implicitly.

The rise of user-centric applications has deeply transformed the original Web from a static system to a gigantic dynamic and consistantly evolving medium of communication and information. As long ago as 1999, Darcy DiNucci [33], an information architecture consultant, introduces the term "Web 2.0" to refer to this new generation of the Web that thoroughly opposes to the static Web 1.0. She writes:

1 Grant ANR-13-CORD-0020 provided by the French research agency.

> The Web we know now, which loads into a browser window in essentially static screenfuls, is only an embryo of the Web to come. The first glimmerings of Web 2.0 are beginning to appear, and we are just starting to see how that embryo might develop. The Web will be understood not as screenfuls of text and graphics but as a transport mechanism, the ether through which interactivity happens.

When users *interact* on Web 2.0 systems, they let footprints which can be exploited to develop better applications for information access. In this dissertation, we study a family of techniques centered on users, which take advantage of the many types of feedback to *adapt* and improve services provided to them. For example, in the context of search and recommendation, user profiles can help to better personalize the displayed content.

Interestingly, the term Web 2.0 was popularized only 5 years after DiNucci's paper, at the first Web 2.0 Conference in 2004, and thus, Tim O'Reilly, the creator of the event, is now often (wrongly) credited for the expression. Around these years, many firms which have become (or were) Web 2.0 figureheads were launched: LinkedIn and MySpace in 2003, Facebook and Flickr in 2004, Twitter in 2006, etc. Many research challenges appeared alongside the rise of Web 2.0, in particular with the democratization of social networks:

- large-scale *data management*: the enormous quantity of data produced by users on Web 2.0 applications transfigures the Web. For example, Facebook scaled from a single server running on a laptop to an estimated number of 180,000 servers distributed on several data centers around the globe in 2012 [92]. The specific needs of companies such as Google or Facebook to store and access these Big Data led to the development of alternatives to relational databases, namely, the *Not Only SQL* (NoSQL) databases, that accept to compromise a feature (e.g. *consistency*) in favor of others (e.g. *availability*, *speed*);

- large-scale *data analysis*: not only do the enormous quantity of data require to scale up storage capacities, processing and analizing data is another crucial facet in the value creation process. The programming model MapReduce [30] is maybe the most famous step forward in processing Big Data on large clusters, notably through the open source implementation Hadoop [122]. More recently, Spark [124] introduced resilient distributed datasets (RDDs) to improve efficiency in applications that need to access a working dataset multiple times in a loop – e.g., iterative machine learning algorithms. In parallel, thanks to the new capabilities brought by large datasets and the progress of computing hardware, we have seen a tremendous development of machine learning since the beginning of the century and, in particular, that of deep learning for computer vision tasks since 2012;

In this dissertation, we study and propose models and *adaptive* algorithms for user-centric applications for information access. We focus on applications like recommendation and influencer marketing in which users generate feedback and we incorporate them in our algorithms in order to deliver strongly contextualized services.

More precisely, in every scenario we consider in this dissertation, users produce feedback, whether consciously or unconsciously, while they are using the application. For example, in our applications, users will provide feedback when they type characters in the search bar of a social network, when they "like" a post on a social platform, when they click on an ad, etc. All these simple actions generate an enormous amount of complex data

that we propose to exploit for building adaptive algorithms. Importantly, we assume that users interact in discrete time steps with the application: when a user requests a service from the application, the algorithm uses all the historical feedback gathered heretofore and provides content accordingly.

In every application considered in the following, the algorithm faces *uncertainty* regarding either the user it proposes content to or the content itself, and often both. For example, in the as-you-type application, the user's intent is unknown and the algorithm needs to make recommendations without being able to guarantee they satisfy the user. Conversely, in the application to multiple-item recommendation, the algorithm faces the cold-start problem, and thus, needs to take decisions without complete knowledge of the items it chooses from. Specifically, the algorithm needs to deal with the crucial explore-exploit tradeoff as it needs to choose between recommending seemingly best items ("exploiting") and cumulating more feedback to improve the items' estimation ("exploring").

1.2 SUMMARY OF THE CONTRIBUTIONS

This dissertation proposes to incorporate the many types of feedback, including complex networked data, from user-centric applications in adaptive algorithms in order to improve user satisfaction. During the period of the thesis, we introduced algorithms for three different applications which are summarized in the following.

AS-YOU-TYPE SOCIAL-AWARE SEARCH (CHAPTER 2). The first part of this dissertation is dedicated to an approach for as-you-type search on social media. More precisely, the problem consists in retrieving a set of k search results in a social-aware environment under the constraint that the query may be incomplete (e.g., if the last term is a prefix). This problem is addressed through the prism of adaptive contextual recommendation. Every time the user updates his / her query, the system updates the set of search results accordingly, so as to improve the user experience on the social platform. We adopt a "network-aware" interpretation of information relevance, by which information produced by users who are closer to the user issuing a request is considered more relevant. This query model raises challenges for effectiveness and efficiency in online search.

Contributions. We describe TOPKS-ASYT, a memory-efficient and incremental prefix-based retrieval algorithm, which also exhibits an anytime behavior, allowing to output an answer within any chosen running-time limit, a major concern in real-time applications such as as-you-type systems. The algorithm borrows ideas to Maniu and Cautis [88] and introduces the following novelties to deal with the as-you-type paradigm:

1. We introduce CT-IL, a completion trie over the set of tags – the keywords on which we search – which allows the algorithm to access the inverted lists efficiently. This new index structure is a combination of tries and inverted lists and is a key component of TOPKS-ASYT to read in sorted order of relevance the possible keyword completions and the items for which they occur.

2. A key difference between as-you-type search and its counterpart for fully specified queries is that the system must *adapt* as the user continues modifying his/her query. Said differently, the system must adjust the set of search results every time the user issuing the query makes a modification. We propose an *incremental* version of TOPKS-ASYT that relies on previous computations in the current session

to provide subsequent answers efficiently, instead of starting a computation from scratch every time the user completes further the query.

3. We characterize the computational *complexity* for the main data structures and algorithmic steps of our method.

4. Answers, albeit approximate, must be ready to be outputted at any time, and in particular after any given time lapse. We refer to this feature as the *anytime output* behavior of TOPKS-ASYT.

5. We introduce and evaluate experimentally a novel feature, denoted *supernodes*, which consists in clustering users in groups of chosen size in order to speed up graph exploration. The goal is to improve the precision of TOPKS-ASYT when the time allocated to serve responses is greatly constrained.

We evaluate our approach through extensive experiments for several applications and search scenarios: we consider searching for posts in micro-blogging (Twitter and Tumblr), for businesses (Yelp), as well as for movies (Amazon) based on reviews.

The work has been presented at the ACM CIKM conference in 2015 [69] and an extended version has been published in the ACM TIST journal in 2017 [70].

ADAPTIVE INFLUENCE MAXIMIZATION WITH PERSISTENCE (CHAPTER 3). Influence maximization is the problem of finding influent users / nodes in a graph so as to maximize the spread of information. It has many applications in advertising and marketing on social networks. In the second part of this dissertation, we study a generic version of influence maximization, in which we want to maximize influence campaigns by *adaptively* selecting "spread seeds" from a set of *candidates*, a small subset of the node population. Influencer marketing [15] is a straightforward application of this, in which the focus of a campaign is placed on precise key individuals – the candidates – who are typically able to reach millions of consumers through their blog or social platform's personal page. This represents an unprecedented tool for online marketing that we propose to improve using an adaptive approach. Importantly, we make the hypothesis that, in a given campaign, previously activated nodes remain "persistently" active throughout, and thus, do not yield further rewards. The rationale is that users who were already activated in the ongoing campaign – e.g., have adopted the product or endorsed a political candidate – remain activated / commited to the cause, and thus will not be accounted for more than once in the objective function. Notably, we make no assumptions on the underlying diffusion model and we work in a setting where neither a diffusion network nor historical activation data are available. We call this problem *online influence maximization with persistence* (OIMP) and choose to address this task using an approach similar to that of multi-armed bandits.

Contributions.

1. We propose to *estimate* the candidates' missing mass – the expected number of nodes that can still be reached from a given seed candidate – by the well-known Good-Turing estimator. We justify its strength to rapidly estimate the desired value which proves to be key in designing an efficient algorithm for short campaigns of tens to hundreds steps, which is typical in the considered scenarios.

2. We describe a novel algorithm, GT-UCB, relying on upper confidence bounds on the missing mass. In order to derive the confidence intervals, we take inspiration

from an approach introduced by Bubeck et al. [17] for rapidly discovering elements of interest from a population, by making sequential requests to so-called *experts* introduced, in which we need to deal with two important changes: (i) after selecting a candidate, every node connected to that candidate is sampled leading to potentially very large feedback that we need to control using the variance to obtain reasonable bounds, (ii) in contrast to [17], the number of times nodes have been activated – a key statistic for the Good-Turing estimator –are independent, which simplifies the derivation of the confidence intervals.

3. We provide an *analysis* of the waiting time – the round at which the missing mass of each candidate is smaller than a certain proportion of the initial missing mass – of GT-UCB, by comparing it to the waiting time of an oracle policy that knows beforehand the objective and the sampled spreads.

4. We conduct *experiments* to show that our approach leads to high-quality spreads on both simulated and real datasets, even though it makes almost no assumptions on the diffusion medium. It is orders of magnitude faster than state-of-the-art influence maximization methods, making it possible to deal with large-scale online scenarios.

This work required the development of a software package coded in C++ available at `https://github.com/plagree/oim` that implements state-of-the-art influence maximization methods as well as OIMP algorithms. This work is currently under review for publication.

ADAPTIVE RECOMMENDATION WITH POSITION BIAS (CHAPTER 4). The *cold start* problem is a well-known issue faced by recommender systems when new items are introduced to the pool of items they recommend from. If no information is available regarding the user appreciation of an item, a recommender system needs to gather feedback – e.g., clicks / non-clicks – so as to estimate the value of an item. However, in order to minimize "bad" recommendations and to maintain the best user experience possible, a well-designed system should not collect feedback carelessly. The third part of the dissertation proposes to solve the recommendation cold-start problem using a multiple-item bandit approach. We introduce a dynamic algorithm that aims to intelligently achieve the balance between "bad" recommendations – which are necessary to gather more information so as to have a better understanding of user evaluations of items – and "good" recommendations. This situation is a typical illustration of the explore-exploit dilemma that bandit algorithms try to answer. Our approach proposes to exploit available information regarding the display bias under the so-called position-based click model introduced by Craswell et al. [28]. Importantly, a major concern in this context is that the system receives *ambiguous* feedback from users we recommend items to. For example, when several ads are recommended to a user on a web page, much of the content may have been simply ignored by the user if he / she has not scrolled down until the bottom of the page.

Contributions.

1. We discuss how the position-based model differs from the cascade model and other variants of click models considered in several previous works on multiple-play bandits. Importantly, previous bandit approaches restrained to click models that view users as people scrolling lists of items in a deterministic way. Consequently, the resulting feedback are unambiguous and the algorithms are quite straightforward.

Conversely, in the position-based model, the feedback received by the learning agent are the product of two *independent* Bernoulli-distributed random variables standing for the *examination* of the positions and the *attraction* of the displayed items, leading to *censored* observations.

2. We provide a novel *regret lower bound* for the position-based model using a proof scheme that is interesting on its own, and which can be generalized to various settings.

3. We introduce *two computationally efficient algorithms*: (i) PBM-UCB consists in sorting optimistic indices in drecreasing order, using confidence bounds based on Hoeffding's inequality (ii) PBM-PIE is an adapted version of PIE(l) – initially introduced by [27] for the cascade model – based on Chernoff's inequality. The derived confidence intervals are tighter than those of PBM-UCB, and thus, PBM-PIE requires less exploration. In practice, this leads to significantly better performances for Bernoulli-distributed rewards with low probability of success. We provide a theoretical analysis of the regret incurred by these two algorithms, which is asymptotically optimal in the case of the latter.

4. We conduct a series of *experiments* on two types of datasets. First, we compare our strategies to state-of-the-art methods in the learning to rank bandit literature with a simple synthetic problem to validate the theoretical results. Then, we evaluate the algorithms in a search advertising scenario using a real dataset provided for KDD Cup 2012 track 2 involving session logs of a search engine called soso.com.

This work has been presented at the NIPS conference in 2016 [71] and is a shared work (equal contribution) with the Ph.D. student Claire Vernade. A preliminary version of this work was presented at the Workshop on Online Advertising Systems at ICML 2016 [116].

APPENDIX 1.A INTRODUCTION TO MULTI-ARMED BANDITS

In this dissertation, we study applications in which users produce feedback and we try to incorporate these signals in order to improve user experience. Specifically, we aim to propose *adaptive* and *dynamic* algorithms in user-centric situations: every time a user connects to the application, the approach provides a personalized service (e.g., a recommendation, a query answer, an ad, etc.) relying on all past observations. These kinds of applications perfectly fit the multi-armed bandit framework which are sequential learning methods that simultaneouly attempt to acquire new knowledge on available options (called "exploration") and optimize choices based on previous decisions (called "exploitation"). In this appendix, we give a short introduction to multi-armed bandits to give the reader basic knowledge about these methods. They will prove to be key when designing adaptive algorithms for influence maximization in Chapter 3 and recommendation in the position-based model in Chapter 4. For a longer introduction to the bandit literature, we refer the reader to the survey of Bubeck and Cesa-Bianchi [16].

A stochastic multi-armed bandit model is a collection of K distributions (called *arms* in the bandit literature) $\nu := (\nu_1, \dots, \nu_K)$, where each ν_k is a probability distribution that is *unknown* to a learning agent. The agent can interact with the model by choosing an arm $I(t) \in [K] := \{1, \dots, K\}$ at each discrete step $t \geq 1$. Then, it observes a variable $X(t)$

sampled from the distribution associated to the chosen arm which can be used to improve the estimation of the unknown mean of this arm.

For the sake of simplicity, we restrain this introduction to multi-armed bandits to the *stochastic* model with Bernoulli distributed random variables: the model parameters are the arm expectations $\theta := (\theta_1, \dots, \theta_K)$, which lie in $\Theta = (0,1)^K$ and the optimal (unknown) arm k^* has expectation $\theta^* := \max_{k \in [K]} \theta_k = \theta_{k^*}$ The objective of the learner is to construct a policy (also called algorithm or strategy) π that maximizes the expected sum of rewards, or equivalently, minimize the expected *regret* defined as follows for a horizon T:

$$\mathbb{E}[R(T)] := T\theta^* - \mathbb{E}\left[\sum_{t=1}^{T} X(t)\right].$$

Intuitively, regret corresponds to the difference between selecting at every step the optimal (unknown) arm k^*, and the actual policy π. Denoting $N_k(T) := \sum_{t=1}^{T} \mathbb{1}\{I(t) = k\}$ the number of times arm k has been chosen up to time T, regret can be rewritten

$$\mathbb{E}[R(t)] = \sum_{k \neq k^*} (\theta^* - \theta_k)\mathbb{E}[N_k(T)] = \sum_{k \neq k^*} \Delta_k \mathbb{E}[N_k(T)],$$

where $\Delta_k := \theta^* - \theta_k$ is the expected *gap to optimality*.

The seminal paper of Lai and Robbins [72] provides a lower bound on the expected regret of *uniformly efficient* strategies defined as follows:

for any $\theta \in \Theta$ such that there is a unique optimal arm, for all $\alpha \in (0,1]$, $R(T) = o(T^\alpha)$.

This suggests that there exist algorithms with sub-polynomial regret. In the case of Bernoulli distributed arms, the lower bound can be stated as follows.

Theorem 1.1 (Lower bound [72]). *For any uniformly efficient algorithm, we have*

$$\liminf_{T \to \infty} \frac{\mathbb{E}[R(T)]}{\log(T)} \geq \sum_{k \neq k^*} \frac{\theta^* - \theta_k}{d(\theta_k, \theta^*)},$$

where $d(p,q) := p \log \frac{p}{q} + (1-p)\frac{1-p}{1-q}$ *is the Kullback-Leibler divergence between two Bernoulli distributions.*

The main difficulty of bandit problems consists in dealing with the famous so-called *explore-exploit* dilemma: the agent wants to maximize its future rewards based on the historical data gathered heretofore, but also needs to maintain exploration as the stochastic nature of feedback may be misleading. Two different points of view on the stochastic multi-armed bandits have been considered in the literature to judiciously allocate specific steps to exploration. The first bandit algorithm adopted the Bayesian framework and was proposed as soon as 1933 by Thompson [111]. In short, the algorithm maintains a posterior distribution on every arm's parameter. At each step t, a value is sampled from each of these distributions and the chosen arm $I(t)$ is set to the arm whose sample is the largest. Formally, after t steps, and given some prior distribution $p(\theta)$ on the unknown parameters, the posterior is given by

$$p(\theta \mid \mathcal{H}_t) \propto \prod_{s=1}^{t} p\left(X(s) \mid I(s), \theta\right) p(\theta),$$

where $\mathcal{H}_t$ denotes the sigma field $\sigma(I(1), X_{I(1)}, \ldots, I(t-1), X_{I(t-1)})$. The procedure, now referred to as THOMPSON SAMPLING, is detailed in Algorithm 1 in the case of Bernoulli distributed rewards. Note that in the case of Bernoulli distributed rewards, the prior is set to a Beta distribution – the Beta distribution is the conjugate prior of the Bernoulli distribution [32] – so as to easily update the posterior using Bayes' rule.

Algorithm 1: THOMPSON SAMPLING

Data: α, β prior parameters of Beta distribution
Initialization: $S_k = 0, F_k = 0, \forall k \in [K]$;
for $t = 1, \ldots, T$ **do**
  for $k = 1, \ldots, K$ **do**
    Sample $\tilde{\theta}_k$ according to Beta$(S_k + \alpha, F_k + \beta)$;
  end
  Choose arm $I(t) \leftarrow \arg\max_k \tilde{\theta}_k$ and observe reward $X(t)$;
  if $X(t) = 1$ **then**
    $S_k \leftarrow S_k + 1$;
  else
    $F_k \leftarrow F_k + 1$;
  end
end

Importantly, THOMPSON SAMPLING has been shown asymptotically optimal only very recently and simultaneously by Kaufmann et al. [64] and by Agrawal and Goyal [2].

The second perspective to tackle bandit problems follows the frequentist point of view and is generally referred to as "optimistic". Instead of maintaining a distribution on the unknown parameters, an index is computed for each arm k based on past observations. The index can be seen as a statistic well defined so as to control the balance between exploration and exploitation. Agrawal [1] is the first to introduce a *UCB-like* (for Upper Confidence Bound) algorithm in which indices are decomposed in the sample mean and an extra exploration term. At every step, the algorithm chooses the arm whose index is the largest. The rationale is that, if the selected arm is the optimal one, the agent is satisfied, whereas the selection of a suboptimal arm is useful in better estimating the unknown parameter, thus reducing the exploration term in subsequent steps. Many studies have focused on improving the exploration term in order to explore suboptimal arms the minimum possible[2].

Auer et al. [6] introduced the UCB1 algorithm that derives the exploration term from the Hoeffding concentration inequality.

At each step t, UCB1 computes an index $b_k^{\mathrm{UCB}}(t)$ for each arm k whose exploration term is derived from Hoeffding's concentration bound. Auer et al. also provide the first *finite-time* analysis and obtain a logarithmic dominant term which is in line with Lai and Robbins' lower bound.

2 Note however that every suboptimal arm needs to be selected by the bandit algorithm a logarithmic number of times. This is a direct consequence of Lai and Robbins' lower bound in Theorem 1.1.

Algorithm 2: Optimistic algorithm – UCB1

1 **Initialization:** first K rounds, play each arm once;
2 $N_k(K+1) \leftarrow K$ for all k;
3 **for** $t = K+1, \dots, T$ **do**
4 **for** $k = 1, \dots, K$ **do**
5 $b_k^{\text{UCB}}(t) \leftarrow \hat{\theta}_k(t) + \sqrt{\frac{2\log(t)}{N_k(t)}}$;
6 **end**
7 Choose arm $I(t) = \arg\max_k b_k^{\text{UCB}}$ and observe reward $X(t)$;
8 **for** $k = 1, \dots, K$ **do**
9 **if** $k = I(t)$ **then**
10 $N_k(t+1) \leftarrow N_k(t) + 1$;
11 **else**
12 $N_k(t+1) \leftarrow N_k(t)$;
13 **end**
14 **end**
15 **end**

Several improvements on UCB1 were made until Garivier and Cappé [40] introduced KL-UCB, the first asymptotically optimal UCB-like algorithm. Specifically, at each step t, KL-UCB computes the index

$$b_k^{\text{KL-UCB}}(t) := \sup_{q \in [\hat{\theta}_k(t), 1)} \left\{ q \;\middle|\; N_k(t) d(\hat{\theta}_k(t), q) \leq \log(t) \right\}$$

for each arm k. Then, it selects the arm with largest KL-UCB index in line 8 of Algorithm 2.

Interestingly, THOMPSON SAMPLING approaches generally lead to good empirical performances, but many studies prefer using the frequentist point of view for mainly two reasons: (i) the posterior distribution may be complicated to derive – by that, understand it may be a "non-standard" distribution –, and thus, difficult to sample from (we will see an example of this in Chapter 4), (ii) in the Bayesian framework, the analysis is typically much harder than its frequentist counterpart.

Multi-armed bandits are still a very active research area, even though the canonical bandit setting has been solved under both Bayesian (THOMPSON SAMPLING) and frequentist (KL-UCB) perspectives. Variations of the classic setting continue to interest many researchers. In this dissertation, we will study in Chapter 4 a variation of the *multiple-play* – several arms are selected at each step – bandit in the framework of online recommendation. In Chapter 3, we rely on the UCB approach to derive an algorithm which adaptively selects influential nodes so as to maximize the expected spread of diffusion.

PRODUCTIONS

- **Publications**
  - Paul Lagrée, Bogdan Cautis, and Hossein Vahabi. "A Network-Aware Approach for Searching As-You-Type in Social Media." In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM). 2015.
  - Claire Vernade, Paul Lagrée, and Olivier Cappé. "Online Inference for Multiple-Item Display Under the Position-Based Model." In: Workshop on Online Advertising Systems. ICML. 2016.
  - Paul Lagrée, Claire Vernade, and Olivier Cappé. "Multiple-Play Bandits in the Position-Based Model." In: Advances in Neural Information Processing Systems 29 (NIPS). Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 1597–1605.
  - Paul Lagrée, Bogdan Cautis, and Hossein Vahabi. "As-You-Type Social Aware Search." In: ACM Transactions on Intelligent System and Techology (TIST). 2017.
  - Paul Lagrée, Olivier Cappé, Bogdan Cautis, and Silviu Maniu. "Effective Large-Scale Online Influence Maximization", In: IEEE 16th International Conference on Data Mining (ICDM). 2017.
  - Paul Lagrée, Olivier Cappé, Bogdan Cautis, and Silviu Maniu. "Algorithms for Online Influencer Marketing", submitted for publication. 2017.
- **Software**
  - `oim`: C++ implementation of online influence maximization with persistence algorithms, and, in particular, GT-UCB, as described in "Effective Large-Scale Online Influence Maximization". The code is available at `https://github.com/plagree/oim`.

2

AS-YOU-TYPE SOCIAL RECOMMENDATION

The first part of this dissertation is dedicated to an approach for as-you-type search on social media, that is data published by users who are interconnected through a social network. More precisely, the problem consists in retrieving a set of k search results, i.e., performing a search with a given prefix, and showing the top ranked results. Interestingly, items are displayed while the user issuing a query is still completing it. In this respect, our approach makes information retrieval meet recommendation systems. We adopt a "network-aware" interpretation of information relevance, by which information produced by users who are closer to the user issuing a request is considered more relevant. This query model raises new challenges for effectiveness and efficiency in online search, even when the intent of the user is fully specified, as a complete query given as input in one keystroke.

We describe TOPKS-ASYT, a memory-efficient prefix-based retrieval algorithm, which also exhibits an anytime behavior, allowing to output the most likely answer within any chosen running-time limit. Furthermore, we propose an incremental version of TOPKS-ASYT that relies on the adaptive aspect of as-you-type search to speed-up computations. At the end of this chapter, we evaluate our approach through extensive experiments for several applications and search scenarios. We consider searching for posts in micro-blogging (Twitter and Tumblr), for businesses (Yelp), as well as for movies (Amazon) based on reviews.

The work has been presented at the conference CIKM in 2015 [69] and an extended version has been published in the ACM TIST journal in 2017 [70].

Contents

2.1 INTRODUCTION

Web search is the main tool used today to access the enormous quantity of information available on the Web, and in particular in the social media. Starting from simple text-based search ranking algorithm, it is now an interdisciplinary topic involving data mining, machine learning, knowledge management, just to mention a few. Significant improvements have been done on how to answer keyword queries on the Web in the most effective way (e.g., by exploiting the Web structure, user and contextual models, user feedback, semantics, etc). However, answering information needs in social media applications (such as Tumblr, Twitter, or Facebook) often requires a significant departure from socially-agnostic approaches, which generally assume that the data being queried is decoupled from the users querying it.

While progress has been made in recent years to support this novel, social and *network-aware* query paradigm – especially towards efficiency and scalability – more remains to be done in order to address information needs in real applications. In particular, providing the most *accurate* answers while the user is typing her query, *almost instantaneously*, can be extremely beneficial, in order to enhance the user experience and to guide the retrieval process.

Figure 2.1 shows an example of as-you-type search in Tumblr. A user is typing a query as-you-type *"as yo"*. In the first part of the results (section "Search"), candidates are selected among queries within the query log and correspond to prefix based *query auto-completion* (such as *"as you are"*). In the second part (section "Blogs"), search results are presented for the partial query *"as yo"* (search result such as the blog *"love everybody"*). This suggestion framework is referred to as *as-you-type search* and is the focus of this work.

In this chapter, we extend as-you-type search – a functionality by now supported in most search applications, including Web search – to social data. In particular we extend existing algorithms for top-k retrieval (where k is the number of results returned, typically $k = 10$) over social data. Our solution, called TOPKS-ASYT (for TOP-k Social-aware search AS-You-Type), builds on the generic network-aware search approach of [88, 105] that we briefly recall in the following section.

We consider a generic setting common to a plethora of social applications, where users produce unstructured content (keywords) in relation to items, an activity we simply refer

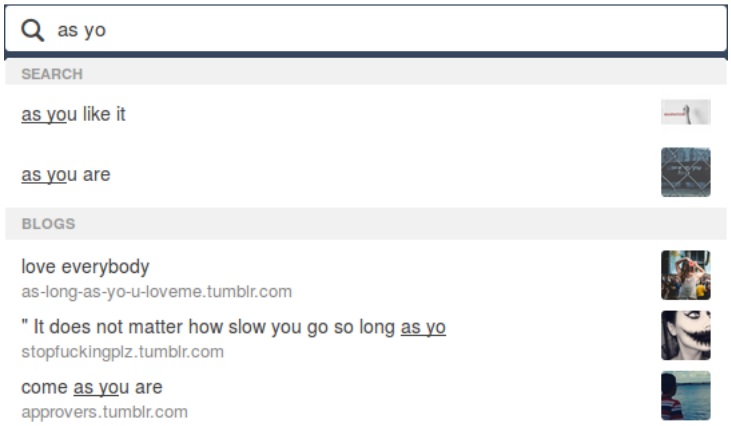


Figure 2.1: An as-you-type search example ("Search": autocompletion, "Blogs": as-you-type results).

to as *social tagging*. More precisely, our core application data can be modelled as follows: (i) users form a social network, which may represent relationships such as similarity, friendship, following, etc, (ii) items from a public pool of items (e.g., posts, tweets, videos, URLs, news, or even users) are "tagged" by users with keywords, through various interactions and data publishing scenarios, and (iii) users search for some k most relevant items by keywords. We devise a novel index structure for TOPKS-ASYT denoted CT-IL which is a combination of tries and inverted lists. While basic trie structures have been used in as-you-type search scenarios in the literature (e.g., see [78] and the references therein), ranked access over inverted lists requires an approach that performs ranked completion more efficiently. Therefore, we rely on a trie structure tailored for the problem at hand, offering a good space-time tradeoff, namely the *completion trie* of [54], which is an adaptation of the well-known Patricia trie using priority queues. This data structure is used as access layer over the inverted lists, allowing us to read in sorted order of relevance the possible keyword completions and the items for which they occur. Importantly, we use the completion trie also as a key internal component of our algorithm, in order to speed-up the incremental computation of results.

In this as-you-type search setting, it is necessary to serve in a short (fixed) lapse of time, after each keystroke and in social-aware manner, top-k results matching the query in its current form, i.e., the terms $t_1, \ldots, t_{r-1}$, and all possible completions of the term t_r. This must be ensured independently of the execution configuration, data features, or scale. This is why we ensure that our algorithms have also an *anytime* behaviour, being able to output the most likely result based on all the preliminary information obtained until a given time limit for the TOPKS-ASYT run is reached.

2.2 SOCIAL-AWARE SEARCH BACKGROUND

We adopt a well-known generic model of social relevance for information, previously considered among others in [87, 88, 105, 123]. In short, the social bias in scores reflects the social proximity of the producers of content with respect to the seeker (the user issuing a search query), where proximity is obtained by some aggregation of shortest paths (in the social space) from the seeker towards relevant pieces of information. We depict in Fig-

ure 2.2 a social network and the tagging activity of its users. In the following, this running example will be used to illustrate definitions and algorithms.

2.2.1 *Notations and context*

In this chapter, we consider a social setting, in which we have a set of items (e.g. text documents, blog posts, tweets, URLs, photos, videos, etc) that are tagged by users from a social network. We formally state the setting in the following.

Context *We assume a set of items* $\mathcal{I} = \{i_1, \ldots, i_m\}$*, where each item is tagged with one or more distinct tags from a tagging vocabulary* $\mathcal{T} = \{t_1, t_2, \ldots, t_l\}$*, by users from* $\mathcal{U} = \{u_1, \ldots, u_n\}$*. We denote our set of unique triples by* $\text{Tagged}(v, i, t)$*, each such triple saying that a user* v *tagged the item* i *with tag* t*.*

Note that Tagged encodes many-to-many relationships: in particular, any given item can be tagged by multiple users, and any given user can tag multiple items. We also assume that a user will tag a given item with a given tag at most once.

While social media applications can adopt for their explicit social links either the directed graph model (e.g. , Twitter or Tumblr) or the undirected one (e.g., Yelp or Facebook), we assume in the following that users form a social *similarity* network, modeled for our purposes as an undirected weighted graph $G = (\mathcal{U}, E, \sigma)$, where nodes are users and the σ function associates to each edge $e = (u_1, u_2)$ a value in $(0, 1]$, called *the proximity* (social) score between u_1 and u_2. Proximity may come either from explicit social signals (e.g., friendship links, follower/followee links), or from implicit social signals (e.g., tagging similarity), or from combinations thereof.

To illustrate, one σ instantiation, i.e., similarity measure, we rely on in our experiments is the *Dice's coefficient*: given two users u and v, we compare their friends (respectively vocabulary, items) to compute a local social (respectively tag, item) similarity. For example, denoting by N_u and N_v the set of users connected to u and v, the Dice's social coefficient is computed as follows:

$$\sigma^{Dice}(u, v) = \frac{2|N_u \cup N_v|}{|N_u| + |N_v|}. \tag{2.1}$$

Other similarities such as the Jaccard index or SimRank can be used to build the social similarity network.

2.2.2 *Top-k retrieval algorithms*

In modern search engines, queries point to thousands (or millions) relevant matching documents. The recall is no longer an interesting metric to measure the effectiveness of the engine. Instead, top-k algorithms focus on retrieving the k most relevant documents, trading recall for speedup of several orders of magnitude. The rationale is that most users will only read the documents displayed on the first pages. Classic top-k retrieval algorithms are *early-termination* algorithms and exploit the textual similarity. They rely on pre-computed inverted lists which are data structures containing exact scores for each term in the entire dataset. For example, in Figure 2.2, $IL(\texttt{grunge}) = (i_2 : 3, i_1 : 2, i_4 : 1, i_5 : 1, i_6 : 1)$ is the inverted list of the word `grunge`, where this means for example that item i_2 has been tagged 3 times with the tag `grunge`.

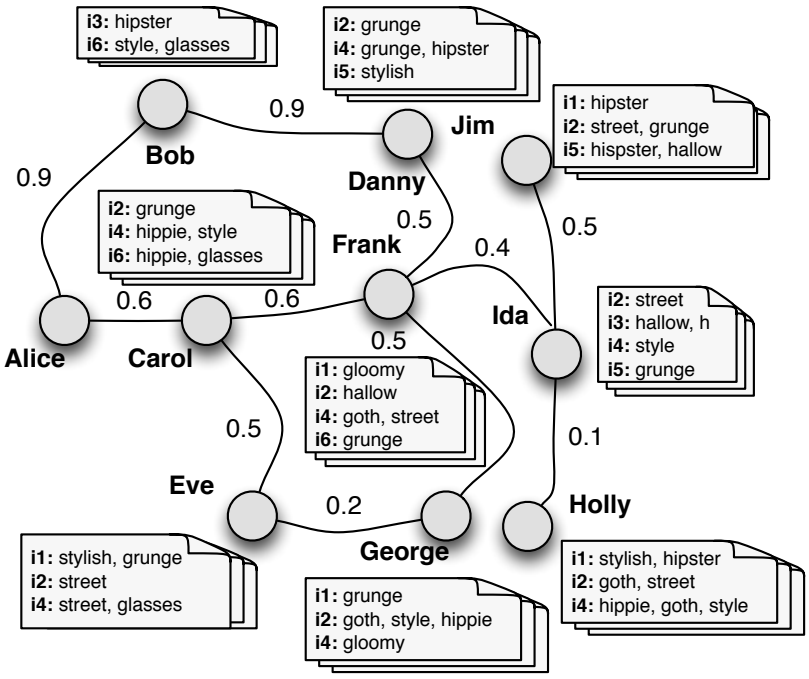


Figure 2.2: Running example: social proximity and tagging.

NO RANDOM ACCESS (NRA): Two famous top-k algorithms allowing for early termination are the Threshold Algorithm (TA) and the No Random Access algorithm (NRA) [38]. The former alternates sequential accesses with random accesses to the inverted lists to compute the *exact* scores of the items discovered. We refer the reader to [38] for a detailed description of the algorithm. Conversely, the NRA only performs sorted accesses to the inverted lists, maintaining lower and upper bounds of every item found sequentially and an upper bound of the unobserved items. When the k-th item's lower bound is larger than the upper bounds of unobserved items and the $k+1$-th candidate item, the NRA terminates. Note that the algorithm may not report the exact scores of the top-k items since it relies on bounds to avoid reading the entire inverted lists.

Example 2.1 *In our running example from Fig. 2.2, let us assume we require the top-2 item for the query* $Q = (\texttt{style}, \texttt{glasses})$. *The corresponding inverted lists are respectively* $IL(\texttt{glasses}) = (i_6 : 2, i_4 : 1)$ *and* $IL(\texttt{style}) = (i_4 : 3, i_2 : 1, i_6 : 1)$.

NRA executes the following steps: at the first access to the inverted lists, the top item of each list is added to the candidates. We have:

$$\textsc{MinScore}(i_6) = 2, \textsc{MaxScore}(i_6) = 5, \textsc{MinScore}(i_4) = 3, \text{ and } \textsc{MaxScore}(i_4) = 5.$$

In addition, the MaxScore *for unobserved items is also* 5, *which prevents us from stopping the algorithm at this step.*

At the second sequential access, the scores of the candidates items become

$$\textsc{MinScore}(i_4) = 3, \textsc{maxScore}(i_4) = 3,$$
$$\textsc{MinScore}(i_6) = 2, \textsc{maxScore}(i_6) = 3,$$
$$\textsc{MinScore}(i_2) = 1, \textsc{MaxScore}(i_2) = 2,$$

and the MaxScore of the unobserved items is now equal to 2. *NRA stops and returns items* i_4 *and* i_6. *Indeed, NRA reached its termination condition as the maximal scores of item* i_2 *and of unobserved items cannot exceed the minimal scores of the items of the answer.*

The NRA algorithm is a key ingredient in social-aware top-k algorithms such as TOPKS from Maniu and Cautis [88], as well as in the *as-you-type* extension TOPKS-ASYT.

2.2.3 *Social-aware search*

Top-k retrieval algorithms have been adapted to network-aware query models for social applications, following the idea of biasing results by social relevance [88, 105, 123] and even time freshness [81].

Person search represents another facet of "social search", as the task of finding highly relevant persons for a given seeker and keywords. Usually, the approach used in this type of application is to identify the most relevant users, and then to filter them by the query keywords [7, 97]. In this area, [29] describes the main aspects of the Unicorn system for search over the Facebook graph, including a typeahead feature for user search. One key difference w.r.t. [88, 105, 123] is that the Unicorn system does not bias results by social proximity in the retrieval phase. Instead, it searches for items in the user environment – her friends, her friend-of-friends, etc. – and returns documents matching the query. Then, the resulting set of items is scored by another procedure that takes into account several factors such as the *query*, the *user* who issued the query or data associated to the *documents*. The authors propose to use a *forward index* that maps documents ids to some metadata to retrieve efficiently these data.

2.2.4 *Query auto-completion and as-you-type search*

As-you-type search (also known as type-ahead search) and query auto-completion are two of the most important features in search engines today, and belong to the broader area of *instant search* (see [115] for a recent tutorial on the topic). They can be seen as facets of the same paradigm: providing accurate feedback to queries on-the-fly, i.e., as they are being typed (possibly with each keystroke). In as-you-type search, feedback comes in the form of the most relevant answers for the query typed so far, allowing some terms (usually, the last one in the query sequence) to be prefix-matched. In query auto-completion, a list of the most relevant query candidates is to be shown for selection, possibly with results for them. We discuss each of these two directions separately. Also, instant search shares many challenges with *exploratory search*, for settings dealing with under-specified, undirected, and even interactive search tasks (see [3] and the references therein).

The problem we study here, namely top-k as-you-type search for multiple keywords, has been considered recently in [78], which mainly differs from our work in the absence of social dimension in data. There, the authors consider various adaptations of TA/NRA top-k algorithms of [38], even in the presence of minor typing errors in the query terms (fuzzy search), based on standard tries. A similar fuzzy interpretation for full-text search was followed in [57], yet not in a top-k setting. The techniques of [77] rely on precomputed materialization of top-k results, for values of k known in advance. In [9, 10], the goal is to find all the query completions leading to results as well as listing these results, based on inverted list and suffix array adaptations; however, the search requires a full

computation and then ranking of the results. For structured data instead of full text, type-ahead search has been considered in [39] (XML) and in [76] (relational data). Finally, [126] studies location-aware as-you-type search by providing location-biased answers, instead of socially-biased ones.

Query auto-completion is the second main direction for instant response to queries in the typing, by which some top query completions are presented to the user (see for example [18, 106, 107, 125] and the references therein). This is done either by following a predictive approach, or by pre-computing completion candidates and storing them in trie structures. Probably the best known example today is the one of Google's instant search, which provides both query predictions (in the search box) and results for the top prediction. In [37], the authors discuss in depth various systems choices involving index partitioning or caching, for query auto-completion under typo-tolerant and word-order tolerant assumptions. Query suggestion goes one step further by proposing alternative queries, which are not necessarily completions of the input one (see for instance [58, 112]).

Several space-efficient trie data structures for ranked (top-k) completion have been studied recently in [54], offering various space-time tradeoffs, and we rely in our work on one of them, namely the completion trie. In the same spirit, data structures for the more general problem of substring matching for top-k retrieval have been considered in [53].

2.2.5 *Social and textual relevance framework*

The general (not as-you-type) keyword search can be formulated as follows:

Problem 2.1 (Social-aware search). *Given a seeker s, a keyword query $Q = (t_1, \ldots, t_r)$ (a set of r distinct terms/keywords) and a result size k, the top-k keyword search problem is to compute the (possibly ranked) list of the k items having the highest scores with respect to s and the query Q.*

We describe hereafter the model ingredients on which we rely to score query results in the social media context. Note that this relevance framework was initially introduced by [88]

We model by $\text{score}(i \mid s, t)$, for a seeker s, an item i, and a tag t, the relevance of that item for the given seeker and query term t. Generally, we assume

$$\text{score}(i \mid s, t) = h(\text{fr}(i \mid s, t)), \tag{2.2}$$

where $\text{fr}(i \mid s, t)$ is the *frequency* of item i for seeker s and tag t, and h is a positive monotone function (e.g., could be based on inverse term frequency, BM25, etc).

Given a query $Q = (t_1, \ldots, t_r)$, the overall score of i for seeker s and Q is simply obtained by summing the per-tag scores:

$$\text{score}(i \mid s, Q) = \sum_{t_j \in Q} \text{score}(i \mid s, t_j). \tag{2.3}$$

Note that this naturally corresponds to an OR semantics, where items that do not necessarily match all the query tags may still be selected (for an AND one, each term's score should be non-empty).

SOCIAL RELEVANCE MODEL. In an exclusively social interpretation, we can explicitate the $\text{fr}(i \mid s, t)$ measure by the *social frequency* for seeker s, item i, and one tag t, denoted $\text{sf}(i \mid s, t)$. This measure adapts the classic term frequency (tf) measure to account for the seeker and its social proximity to relevant taggers. We consider that each tagger brings her own weight (proximity) to an item's score, and we define social frequency as follows:

$$\text{sf}(i \mid s, t) = \sum_{v \in \{v \mid \text{Tagged}(v,i,t))\}} \sigma(s, v). \tag{2.4}$$

Note that, under the frequency definition of Eq. (2.2), we would follow a ranking approach by which information that may match the query terms but does not score on the social dimension (i.e., is disconnected from the seeker) is deemed entirely irrelevant.

NETWORK-AWARE RELEVANCE MODEL. A more generic relevance model, which does not solely depend on social proximity but is *network-aware*, is one that takes into account textual relevance scores as well. For this, we denote by $\text{tf}(t, i)$ the *term frequency* of t in i, i.e., the number of times i was tagged with t, and $IL(t)$ is the inverted list of items for term t, ordered by term frequency.

The frequency score $\text{fr}(i \mid s, t)$ is defined as a linear combination of the previously described social relevance and the textual score, with $\alpha \in [0, 1]$, as follows:

$$\text{fr}(i \mid s, t) = \alpha \times \text{tf}(t, i) + (1 - \alpha) \times \text{sf}(i \mid s, t). \tag{2.5}$$

This formula combines a global popularity of the item with one among people close to the seeker. Note that Eq. 2.5 will be a key ingredient to design our network-aware relevance model in as-you type scenarios.

REMARK. Interestingly, this model of triples for social data is a simple abstraction for quite diverse types of social media. Consider Tumblr [19]: one broadcasts posts to followers and rebroadcasts incoming posts; when doing so, the re-post is often tagged with chosen tags or short descriptions (hashtags). We can thus see a post and all its re-posted instances as representing one informational item, which may be tagged with various tags by the users broadcasting it. Text appearing in a blog post can also be interpreted as tags, provided either by the original author or by those who modified it during subsequent re-posts; it can also be exploited to uncover implicit tags, based on the co-occurrence of tags and keywords in text. Furthermore, a post that is clicked-on in response to a Tumblr search query can be seen as being effectively tagged (relevant) for that query's terms. All this data has obviously a social nature: e.g., besides existing follower/followee links, one can even use similarity-based links as social proximity indicators.

Example 2.2 *Getting back to our running example in Fig. 2.2, for seeker Alice, we have, for* $\alpha = 0.2$, $\text{tf}(\texttt{glasses}, i_6) = 2$, *and*

$$\text{sf}(i_6 \mid \textit{Alice}, \texttt{glasses}) = \sigma(\textit{Alice}, \textit{Bob}) + \sigma(\textit{Alice}, \textit{Carol}) = 0.9 + 0.6 = 1.5,$$
$$\text{fr}(i_6 \mid \textit{Alice}, \texttt{glasses}) = 0.8 \times 1.5 + 0.2 \times 2.$$

EXTENDED PROXIMITY. The model described so far takes into account only the immediate neighbourhood of the seeker (the users it connects to explicitly). In order to broaden the scope of the query and go beyond one's vicinity in the social network, we also account

for users that are indirectly connected to the seeker, following a natural interpretation that user links and the query relevance they induce are (at least to some extent) transitive. To this end, we denote by σ^+ the resulting measure of *extended proximity*, which is to be computed from σ for any pair of users connected by at least one path in the network. Now, σ^+ can replace σ in the definition of social frequency in Eq. (2.4).

For example, one natural way of obtaining extended proximity scores is by (i) multiplying the weights on a given path between the two users, and (ii) choosing the maximum value over all the possible paths. Another possible definition for σ^+ can rely on an aggregation that penalizes long paths, via an *exponential decay* factor, in the style of the Katz social proximity [62]. More generally, any aggregation function that is monotonically non-increasing over a path, can be used here. Under this monotonicity assumption, one can browse the network of users *on-the-fly* (at query time) and "sequentially", i.e., visiting them in the order of their proximity with the seeker.

Example 2.3 *In Fig. 2.2, for seeker Alice, when extended proximity between two users is defined as the maximal product of scores over paths linking them, the users ranked by proximity w.r.t. Alice are in order Bob* : 0.9, *Danny* : 0.81, *Carol* : 0.6, *Frank* : 0.4, *Eve* : 0.3, *George* : 0.2, *Ida* : 0.16, *Jim* : 0.07, *Holly* : 0.01.

Hereafter, when we talk about proximity, we refer to the extended one, and, for a given seeker s, the *proximity vector* of s is the list of users with non-zero proximity with respect to it, ordered decreasingly by proximity values (we stress that this vector is not necessarily known in advance).

2.3 THE AS-YOU-TYPE APPROACH

We now describe our solution TOPKS-ASYT to the social-aware as-you-type search problem. Note that TOPKS-ASYT builds on [88, 105] and deals with three systemic changes:

1. *Prefix matching:* answers must be computed following a query interpretation by which the last term in the query sequence can match tag / keyword prefixes.

2. *Incremental computation:* answers must be computed *incrementally*, instead of starting a computation from scratch. For a query representing a sequence of terms $Q = (t_1, \ldots, t_r)$, we can follow an approach that exploits what has already been computed in the query session so far, i.e., for the query $Q' = (t_1, \ldots, t_{r-1}, t'_r)$, with t'_r being a one character shorter prefix of the term t_r.

3. *Anytime output:* answers, albeit approximate, must be ready to be outputted at any time, and in particular after any given time lapse (e.g., $50 - 100ms$ is generally accepted as a reasonable latency for as-you-type search).

2.3.1 *The as-you-type search problem*

With respect to the general keyword search problem formulated before, we consider in this work a specialized and potentially more useful level of search service for practical purposes, in which queries are being answered *as they are typed*. Instead of assuming that the query terms are given all at once, a more realistic assumption is that input queries are sequences of terms $Q = (t_1, \ldots, t_r)$, in which all terms but the last are to be matched

exactly, whereas the last term t_r is to be interpreted as a tag potentially still in the writing, hence matched as a tag prefix.

We extend the query model in order to deal with tag prefixes p by defining an item's score for p as the maximal one over all possible completions of p:

$$\mathrm{sf}(i \mid s, p) = \max_{t \in \{p\text{'s completions}\}} \mathrm{sf}(i \mid s, t) \tag{2.6}$$

$$\mathrm{tf}(p, i) = \max_{t \in \{p\text{'s completions}\}} \mathrm{tf}(t, i) \tag{2.7}$$

Note that when we compute the importance of an item, we might consider two different tag completions, for the social contribution and for the popularity one.

Example 2.4 *In Fig. 2.2, if Alice's query is* `hipster g`*, as* `g` *matches the tags* `gloomy`*,* `glasses`*,* `goth` *and* `grunge`*, we have* $\mathrm{sf}(i_4 \mid Alice, \texttt{g})$ *as*

$$\begin{aligned} & \max_{t \in \{g \text{ completions}\}} \mathrm{sf}(i_4 \mid Alice, t) \\ = \; & \max[\mathrm{sf}(i_4 \mid Alice, \texttt{gloomy}), \mathrm{sf}(i_4 \mid Alice, \texttt{glasses}), \mathrm{sf}(i_4 \mid Alice, \texttt{grunge}), \mathrm{sf}(i_4 \mid Alice, \texttt{goth})] \\ = \; & \max[0.2, 0.3, 0.81, 0.41] = 0.81. \end{aligned}$$

In the social-aware retrieval setting, when social proximity determines relevance, the data exploration must jointly consider the network (starting from the seeker and visiting users in descending proximity order), the per-user/personal tagging spaces, and all available socially-agnostic index structures such as inverted lists. It is thus important for efficiency to explore the social network by order of relevance/proximity to the seeker, as to access all the necessary index structures, in a *sequential manner* as much as possible. We favor such an approach here, instead of an *incomplete* "one dimension at a time" one, which would first rely on one dimension to identify a set of candidate items, and then use the scores for the other dimension to re-rank or filter out some of the candidates.

2.3.2 *Non-incremental algorithm*

We first describe the TOPKS-ASYT approach for exclusively social relevance ($\alpha = 0$ in Eq. 2.5) and without incremental computation, namely when the full sequence of terms is given in one keystroke, with the last term possibly a prefix, as $Q = (t_1, \ldots, t_r)$. We follow an early-termination approach that is "user-at-a-time": its main loop step visits a new user and the items that were tagged by her with query terms. Algorithm 3 gives the flow of TOPKS-ASYT.

MAIN INPUTS. For each user u and tag t, we assume a precomputed selection over the Tagged relation, giving the items tagged by u with t; we call these the *personal spaces* (in short, p-spaces). No particular order is assumed for the items appearing in a user list.

We also 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 $\mathrm{tf}(t, i)$[1], and by which they are sorted in descending order. The lists can be seen as unpersonalized indexes. A completion trie over the set of tags represents the access layer to these lists. As in Patricia tries, a node can represent more than one character, and the scores corresponding to the heads of the

1 Even when $\alpha = 0$, although social frequency does not depend directly on tf scores, we exploit the inverted lists and the tf scores by which they are ordered, to better estimate score bounds.

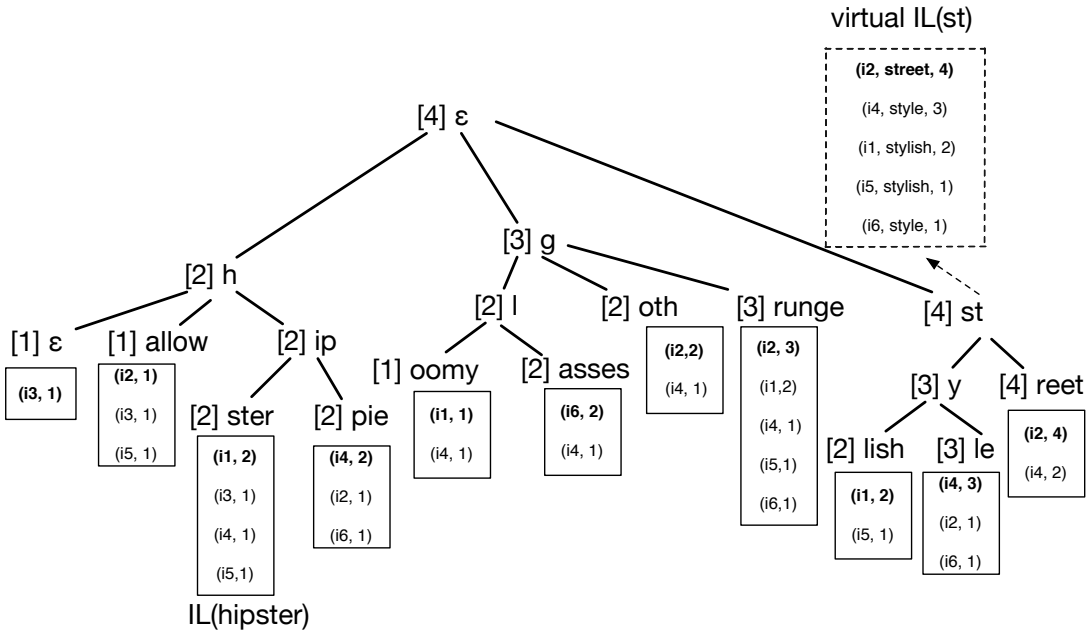


Figure 2.3: The CT-IL index.

lists are used for ranked completion: each leaf has the score of the current entry in the corresponding inverted list, and each internal node has the maximal score over its children (see example below). This index structure is denoted hereafter the CT-IL index.

Example 2.5 (CT-IL index). *We give in Figure 2.3 an illustration of the main components of CT-IL, for our running example. Each of the tags has below it the inverted list (the one of the* `hipster` *tag is explicitly indicated). The cursor positions in the lists are in bold. By storing the maximal score at each node (in brackets in Figure 2.3), the best (scoring) completions of a given prefix can be found by using a priority queue, which is initialized with the highest node matching that prefix. With each pop operation, either we get a completion of the prefix, or we advance towards one, and we insert in the queue the children of the popped node.*

For comparison, we also illustrate in Figure 2.4 the CT-IL index that would allow us to process efficiently Alice's top-k queries, without the need to resort to accesses in social network and p-spaces. Obviously, building such an index for each potential seeker would not be feasible.

While leaf nodes in the trie correspond to concrete inverted lists, we can see each internal node of the trie and the corresponding keyword prefix as described by a "virtual inverted list", i.e., the ranked union of all inverted lists below that node. As defined in Eq. (2.6-2.7), for such a union, for an item appearing in entries of several of the unioned lists, we keep only the highest-scoring entry. In particular, for the query term t_r, by $IL(t_r)$ we refer to the virtual inverted list corresponding to this prefix. There is a notable difference between the concrete inverted lists and the virtual ones: in the former, entries can be seen (stored) as pairs $(item, score)$ (the tag is implied); in the latter, entries must be of the form $(item, tag, score)$, as different tags (completions) may appear in such a list.

For each $t \in \{t_1, \ldots, t_r\}$, we denote by top_item$(t)$ the item present at the current (unconsumed) position of $IL(t)$, we use top_tf(t) as short notation for the term frequency

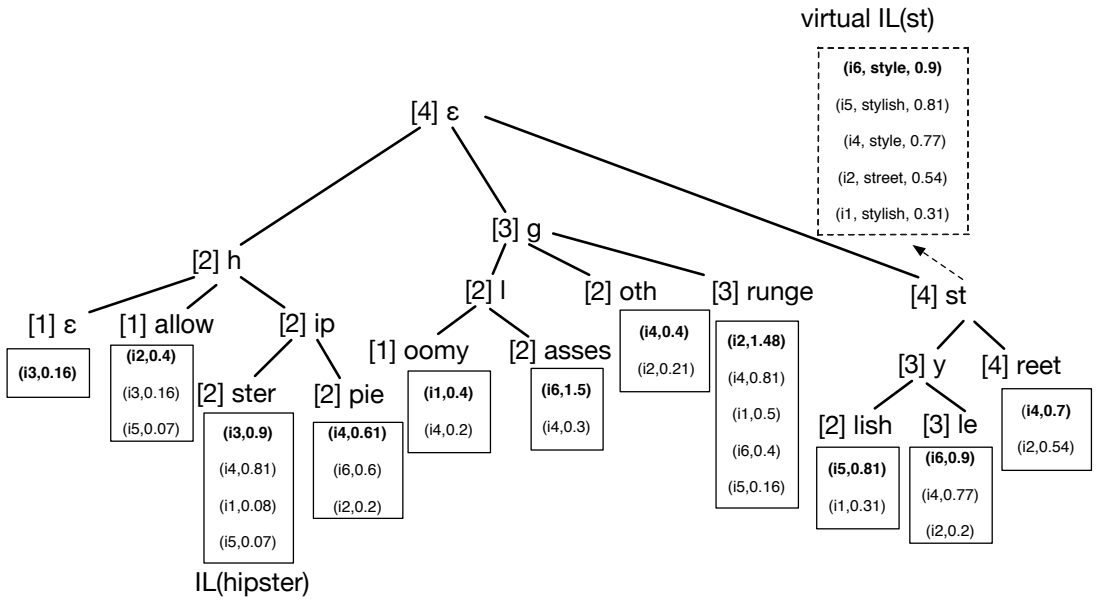


Figure 2.4: Alice's personalized CT-IL index.

associated with this item, and, for $IL(t_r)$, we also denote by top_tag(t_r) the t_r completion in the current entry.

Example 2.6 (Virtual lists). *The virtual inverted list for the prefix* `st` *is given in Fig. 2.3. The* top_tag(`st`) *is* `street`*, for* top_item(`st`) *being* i_2*, for its entry scored* 4 *dominates the one scored only* 2*, hence with a* top_tf(`st`) *of* 4*. A similar one, for the "personalized" CT-IL index for seeker Alice is given in Fig. 2.4.*

CANDIDATE BUFFERS. For each tag $t \in \{t_1, \ldots, t_{r-1}\}$, we keep a list D_t of candidate items i, along with a sound score range: a lower bound and an upper bound for $\mathrm{sf}(i \mid s, t)$ (to be explained hereafter). Similarly, in the case of t_r, for each completion t of t_r already encountered during the query execution in p-spaces (i.e., by triples (u, i, t) read in some u's p-space), we record in a D_t list the candidate items and their score ranges. Candidates in these D-buffers are sorted in descending order by their score lower bounds.

An item becomes candidate and is included in D-buffers only when it is first met in a Tagged triple matching a query term.

For uniformity of treatment, a special item $*$ denotes all the yet unseen items, and it implicitly appears in each of the D-lists; note that, in a given D_t buffer, $*$ represents items which are not yet candidates, but also candidate items which may already be candidates but appear only in other D-buffers (for tags other than t).

MAIN ALGORITHMIC COMPONENTS. When accessing the CT-IL index, inverted list entries are consumed in some $IL(t)$ only when the items they refer to are candidates (they appear in at least one buffer $D_{t'}$, where t' may be any term in the query or completion of the prefix)[2]. We keep in lists called CIL_t (for *consumed IL entries*) the items read (referred to

2 The rationale is that our algorithm does not make any "wild guesses", avoiding reads that may prove to be irrelevant and thus leading to suboptimal performance.

Algorithm 3: TOPKS-ASYT (non-incremental, for $\alpha = 0$)

Input: seeker s, query $Q = (t_1, \ldots, t_r)$
forall *users* u **do**
    $\sigma^+(s,u) = -\infty$;
end
forall *tags* $t \in \{t_1, \ldots, t_{r-1}\}$ **do**
    $\mathrm{sf}(i \mid s,t) = 0$; $D_t = \emptyset$, $CIL_t = \emptyset$;
    Set $IL(t)$ position on first entry;
end
Set $IL(t_r)$ position on first entry, $\sigma^+(s,s) = 0$, $C = \emptyset$ (t_r completions);
$H \leftarrow$ priority queue on users; init. $\{s\}$, computed on-the-fly;
while $H \neq \emptyset$ **do**
    u = EXTRACT_MAX(H);
    PROCESS_P_SPACE(u);
    PROCESS_CT-IL;
    if *termination condition* **then**
        break;
    end
end
return top-k items;

as candidates) in the inverted lists (virtual or concrete), for t being either in $\{t_1, \ldots, t_{r-1}\}$ or a completion of t_r for which a triple $(item, t, score)$ was read in the virtual list of t_r. We also keep by the set C all t_r completions encountered so far in p-spaces. Note that t_r completions encountered in p-spaces may not necessarily coincide with those encountered in $IL(t_r)$.

For each t being either in $\{t_1, \ldots, t_{r-1}\}$ or a completion of t_r already in C, we denote by unseen_users(i,t) the maximal number of yet unvisited users who may have tagged item i with tag t. This number is initially set to the maximal possible term frequency of t over all items. unseen_users(i,t) then reflects at any moment during the run of the algorithm the difference between the number of taggers of i with t already visited and one of the following values:

- $\mathrm{tf}(t,i)$, if this term frequency has been read already by accessing CT-IL, or otherwise
- top_tf(t), if $t \in \{t_1, \ldots, t_{r-1}\}$, or
- top_tf(t_r), if t is instead a completion of t_r.

For every candidate i of a candidate list D_t, we accumulate in $\mathrm{sf}(i \mid s,t)$ the social score (initially 0).

Each time we visit a user u having a triple (u,i,t) in her p-space (Algorithm 4), we can

1. update $\mathrm{sf}(i \mid s,t)$ by adding $\sigma^+(s,u)$ to it, and
2. decrement unseen_users(i,t); when this reaches 0, the social frequency $\mathrm{sf}(i \mid s,t)$ is exact.

The maximal proximity score of yet to be visited users is denoted max_proximity. With this proximity bound, a sound score range for candidates i in D_t buffers is computed and maintained as

- a score upper bound (maximal score) MaxScore$(i \mid s, t)$, by max_proximity $\times$ unseen_users(i, t)+ sf$(i \mid s, t)$.
- a score lower bound (minimal score), MinScore$(i \mid s, t)$, by assuming that the current social frequency sf$(i \mid s, t)$ is the final one (put otherwise, all remaining taggers u of i with t, which are yet to be encountered, have $\sigma^+(s, u) = 0$).

Consuming the inverted list entries (Algorithm 5) in CT-IL, whenever top items become candidates, allows us to keep as accurate as possible the worst-case estimation on the number of unseen taggers. When such a tuple $(i, t, score)$ is accessed, we can do adjustments on score estimates:

1. if $i \in D_t$, we can mark the number of unseen taggers of i with t as no longer an estimate but an *exact* value; from this point on, the number of unseen users will only change whenever new users who tagged i with t are visited,
2. by advancing to the next best item in $IL(t)$, for $t \in \{t_1, \ldots, t_{r-1}\}$, we can refine the unseen_users(i', t) estimates for all candidate items i' for which the exact number of users who tagged them with t is yet unknown,
3. by advancing to the next best item in $IL(t_r)$, with some $t = \text{top\_tag}(t_r)$ completion of t_r, if $t \in C$, we can refine the estimates unseen_users(i', t) for all candidate items $i' \in D_t$ for which the exact number of users who tagged them with t is yet unknown.

Termination condition. From the per-tag D_t buffers, we can infer lower bounds on the global score w.r.t. Q for a candidate item (as defined in Eq. (2.3)) by summing up its score lower bounds from $D_{t_1}, \ldots, D_{t_{r-1}}$ and its maximal score lower bound across all D_t lists, for completions t of t_r. Similarly, we can infer an upper bound on the global score w.r.t. Q by summing up score upper bounds from $D_{t_1}, \ldots, D_{t_{r-1}}$ and the maximal upper bound across all D_t lists, for completions t.

After sorting the candidate items (the wildcard item included) by their global score lower bounds, TOPKS-ASYT can terminate whenever (i) the wildcard item is not among the top-k ones, and (ii) the score upper bounds of items not among the top-k ones are less than the score lower bound of the kth item in this ordering (we know that the top-k can no longer change).

As in [88], it can be shown that TOPKS-ASYT visits users who may be relevant for the query in decreasing proximity order and, importantly, that it visits as few users as possible (it is *instance optimal* for this aspect, in the case of exclusively social relevance).

Example 2.7 *Revisiting our running example in Fig. 2.2, let us assume Alice requires the top-2 items for the query $Q = (\texttt{style}, \texttt{gl})$ where $\texttt{gl}$ is a prefix and for $\alpha = 0$. The first data access steps of TOPKS-ASYT are as follows: at the first execution of the main loop step, we visit Bob, get his p-space, adding i_6 both to the $D_{\texttt{style}}$ buffer and to a $D_{\texttt{glasses}}$ one. There may be at most two other taggers of i_6 with* `style` *(*unseen_users$(i_6, \texttt{style})$*), and at most one other tagger of i_6 with* `glasses` *(*unseen_users$(i_6, \texttt{glasses})$*). No reading is done in $IL(\texttt{style})$, as its current entry gives the non-candidate item i_4, but we can advance with one pop in the virtual list of the* `gl` *prefix, for candidate item i_6. This clarifies that there is* exactly *one other tagger with* `glasses` *for i_6. After this read in the virtual list of* `gl`*, we have* top_item$(\texttt{gl}) = i_1$

Algorithm 4: Subroutine PROCESS_P_SPACE(u)

forall $t \in \{t_1, \ldots, t_{r-1}\}$, *triples* Tagged$(u, i, t)$ **do**
  if $i \notin D_t$ **then**
    Add i to D_t, $\mathrm{sf}(i \mid s, t) \leftarrow 0$;
    $\mathrm{unseen\_users}(i, t) \leftarrow \mathrm{top\_tf}(t)$;
  end
  $\mathrm{unseen\_users}(i, t) \leftarrow \mathrm{unseen\_users}(i, t) - 1$;
  $\mathrm{sf}(i \mid s, t) \leftarrow \mathrm{sf}(i \mid s, t) + \sigma^+(s, u)$;
end
forall *tags* t *completions of* t_r, *triples* Tagged(u, i, t) **do**
  if $t \notin C$ **then**
    Add t to C, $D_t = \emptyset$;
  end
  if $i \notin D_t$ **then**
    Add i to D_t, $\mathrm{sf}(i \mid s, t) \leftarrow 0$;
    $\mathrm{unseen\_users}(i, t) \leftarrow \mathrm{top\_tf}(t)$;
  end
  $\mathrm{unseen\_users}(i, t) \leftarrow \mathrm{unseen\_users}(i, t) - 1$;
  $\mathrm{sf}(i \mid s, t) \leftarrow \mathrm{sf}(i \mid s, t) + \sigma^+(s, u)$;
end

(if we assume that items are also ordered by their ids). At this point max_proximity *is* 0.81. *Therefore, we have*

$$\begin{aligned}
&\textsc{MaxScore}(i_6 \mid \mathit{Alice}, \texttt{style}) = 0.81 \times 2 + 0.9,\\
&\textsc{MinScore}(i_6 \mid \mathit{Alice}, \texttt{style}) = 0.9,\\
&\textsc{MaxScore}(i_6 \mid \mathit{Alice}, \texttt{glasses}) = 0.81 \times 1 + 0.9,\\
&\textsc{MinScore}(i_6 \mid \mathit{Alice}, \texttt{glasses}) = 0.9.
\end{aligned}$$

We thus have that $\mathrm{score}(i_6 | \mathit{Alice}, Q)$ *is between* 1.8 *and* 4.23.

At the second execution of the main loop step, we visit Danny, whose p-space does not contain relevant items for Q. *But a side-effect of this step is that* max_proximity *becomes* 0.6, *affecting the upper bound scores above:* $\mathrm{score}(i_6 \mid \mathit{Alice}, Q)$ *can now be estimated between* 1.8 *and* 3.6.

At the third execution of the main loop step, we visit Carol, and find the relevant p-space entries for i_4 *(with tag* `style`*) and* i_6 *(with tag* `glasses`*). Now* max_proximity *becomes* 0.4. *Also, we can advance with one pop in the inverted list of* `style`. *This clarifies that there are* exactly *2 other taggers with* `style` *on* i_4, *and now we have* $\mathrm{top\_item}(\texttt{gl}) = i_1$ *and* $\mathrm{top\_item}(\texttt{style}) = i_2$. *This makes* $\mathrm{score}(i_6 \mid \mathit{Alice}, Q)$ *to be known precisely at* 2.4, $\mathrm{score}(i_4 \mid \mathit{Alice}, Q)$ *to be estimated between* 0.6 *and* $0.6 + 3 \times 0.4 = 1.8$, *and* $\mathrm{score}(* \mid \mathit{Alice}, Q)$ *is at most* 0.8. *At this point the algorithm has reached the termination condition.*

2.3.3 *Adaptations for the network-aware case*

We sketch in this section the necessary extensions to Algorithm 3 for arbitrary values of α, hence for any textual-social relevance balance. When $\alpha \in [0, 1]$, at each iteration, the

Algorithm 5: Subroutine PROCESS_CT-IL

```
while ∃t ∈ Q s.t. i = top_item(t) ∈ ⋃_x D_x do
    if t ≠ t_r then
        tf(t, i) ← top_tf(t) (t's frequency in i is now known);
        Advance IL(t) one position;
        Δ ← tf(t, i) − top_tf(t) (the top_tf drop);
        Add i to CIL_t;
        forall items i' ∈ D_t \ CIL_t do
            unseen_users(i', t) ← unseen_users(i', t) − Δ;
        end
    end
    if t = t_r then
        t' ← top_tag(t_r) (some t_r completion t');
        tf(t', i) ← top_tf(t_r) (t''s frequency in i known);
        Advance IL(t_r) one position;
        Δ ← tf(t', i) − top_tf(t_r) (the top_tf drop);
        Add i to CIL_t' or set CIL_t' to {i} if previously empty;
        forall t'' ∈ C and items i' ∈ D_t'' \ CIL_t'' do
            unseen_users(i', t'') ← unseen_users(i', t'') − Δ;
        end
    end
end
```

algorithm can alternate between two possible execution branches: the *social branch* (the one detailed in Algorithm 3) and a *textual branch*, which is a direct adaptation of NRA over the CT-IL structure, reading in parallel in all the query term lists (concrete or virtual). Now, items can become candidates even without being encountered in p-spaces, when read in inverted lists during an execution of the textual branch. As before, each read from CT-IL is associated with updates on score estimates such as unseen_users. For a given item i and tag t, the maximal possible fr-score can be obtained by adding to the previously seen maximal possible sf-score (weighted now by $1-\alpha$) the maximal possible value of $\mathrm{tf}(t,i)$; the latter may be known (if read in CT-IL), or estimated as $\mathrm{top\_tf}(t)$ otherwise. Symmetrically, the minimal possible value for $\mathrm{tf}(t,i)$ is used for lower bounds; if not known, this can be estimated as the number of visited users who tagged i with t.

The choice between the two possible execution branches can rely on heuristics which estimate their utility w.r.t approaching the final result. We mention the existing works of Maniu and Cautis [88] and Schenkel et al. [105] who propose to guide this choice either by estimating the maximum potential score of each branch, or by choosing the branch that is the most likely to refine the score of the item outside the current top-k which has the highest estimated score (a choice that is likely to advance the run of the algorithm closer to termination).

2.3.4 *Adaptations for incremental computation*

We now describe the extension to perform the as-you-type computation *incrementally*, as follows:

1. when a new keyword is initiated (i.e., t_r is one character), we take the following steps in order:
   a) purge all D_t buffers for $t \in C$, except for $D_{t_{r-1}}$ (t_{r-1} is no longer a potential prefix, but a complete term),
   b) reinitialize C to the empty set,
   c) purge all CIL_t buffers for $t \notin \{t_1, \ldots, t_{r-1}\}$,
   d) reinitialize the network exploration (the queue H) to start from the seeker, in order to visit again p-spaces looking for triples for the new prefix, t_r. (This amounts to the following changes in Algorithm 3: among its initialization steps (1-12), steps (4-8) are removed, and steps for points (a) and (c) above are added.)
2. when the current t_r is augmented with one additional character (so t_r is at least two characters long), we take the following steps in order:
   a) purge D_t buffers for $t \in C$ s.t. t is not a t_r completion
   b) remove from C all ts which aren't completions for t_r,
   c) purge all CIL_t buffers for $t \notin \{t_1, \ldots t_{r-1}\} \cup C$,
   d) resume the network exploration. (This amounts to the following changes in Algorithm 3: among its initialization steps (1-12), steps (4-8) and (10-12) are removed, and steps for points (a), (b), and (c) above are added.)

Note that, in the latter case, we can efficiently do the filtering operations by relying on a simple trie structure for the C set and use it as the index for directly accessing the other data structures (D-lists and CIL-lists).

2.3.5 *Complexity analysis*

Recall that Tagged(v, i, t) denotes the set of unique triples and consider a query $Q = (t_1, \ldots, t_r)$. Let f denote the average fan-out in the CT-IL index, d_r the average depth of the trie subtree rooted at the node corresponding to t_r (models the size of t_r completions), and p the average p-space size. Finally, remember that users in $\mathcal{U}$ are interconnected in the similarity graph through edges E.

CT-IL is a space-efficient trie structure for sorted access, as a node can represent a sequence of characters. Thus, the memory space to store the trie is reduced compared to the trie index of [78]. Given an item i and a tag t, there corresponds a unique entry $(i, \mathrm{tf}(i,t))$ in $IL(t)$. In total, there are as many entries in inverted lists as *unique* pairs (i, t), leading to a total space for the inverted lists at the leaves of CT-IL of $O(|\{\text{unique } (i,t)\}|)$. The number of inverted lists corresponds to the number of distinct tags in the vocabulary, $|\mathcal{T}|$. For example, in the case of our Yelp dataset, there are 177,286 such lists and a simple computation reveals that in average each would have approximately 70 entries.

For each user u, we store a p-space index containing all pairs (i, t) of u. Thus, each triple is indexed in p-spaces exactly once. The shortest-path computations for exploring the social graph by order of proximity is straightforwardly $O(|E|+|\mathcal{U}|\log|\mathcal{U}|)$.

In the run of TOPKS-ASYT, we visit one user at a time in the social graph and, in the worst-case, we may visit the entire network, unless an event like the termination condition,

a keystroke, or (most likely) the time limit occurs.[3] While entries in inverted lists are read at most once (see Algorithm 5), the situation is different for p-spaces, as they may need to be explored once for each new word that is added to the query (see step (1)(d) from the previous section), leading to $O(r)$ network explorations.

A one-character search (i.e., expansion of t_r) initially costs $O(f)$ in CT-IL, and is followed by a sequence of variable length of sorted accesses in the trie and in the social graph; their actual number depends on the value of α and on the overlap between p-spaces and CT-IL. Individually, the former accesses have a direct cost of $O(d_r \log d_r)$. However, compared with the compact-trie of [54], this direct cost we incur is roughly double (albeit reduced), since our leaves are not necessarily single strings, but lists thereof, and thus a sorted access in a priority queue most often will translate in a score update instead of a normal pop operation.

Just like the NRA algorithm of [38], whose complexity is quadratic in the size of its buffers, the bookkeeping steps are more expensive complexity-wise because score intervals are maintained throughout the computation, so we cannot have bounded buffers for our candidates. Whenever the p-space of some user u is visited (Algorithm 4), for a given completed tag $t \in Q$ used by u, the cost of the updates to be done on the buffer D_t is $O(p|D_t|)$; an additional cost of $O(p \sum_t |D_t|)$, for t denoting t_r completions, corresponds to the tag t_r still in the typing. Regarding accesses to CT-IL by Algorithm 5, the cost is of the order $O(p(|D_t|^2+|CIL_t||D_t|))$ for the completed tags t, and $O(p \sum_t (|D_t|^2+|CIL_t||D_t|))$ for the completions t of t_r. Overall, in the most important case for our study – exclusively social, i.e., $\alpha = 0$, for one prefix query, i.e., $r = 1$ – the worst-case time complexity of our algorithm is $O(|E|+|\mathcal{U}|\log|\mathcal{U}|+d_r|\mathcal{U}|p \sum_t (|D_t|^2+|CIL_t||D_t|)$, for the completions t of t_r.

Compared to the non-incremental version, the algorithm avoids to restart the graph exploration from the seeker s and simply continues from the currently visited node. As described in Section 2.3.4, the pruning of all unnecessary data structures D_t, CIL_t, and C – for t denoting here the previous completions that do not match newly typed letter, can be done efficiently in $O(f)$ by using a trie for the C-set, which can act as the vocabulary index leading to the D_t and CIL_t buffers.

2.3.6 *Supernodes*

When visiting a user node, we need to explore its p-space – its tag contributions – by routine PROCESS_P_SPACE (Algorithm 4). This can be costly overall if p-spaces are saved on disk, since many p-spaces may be loaded in main memory. In the case of time-limited queries, when a budget is imposed (e.g., in terms of random disk accesses) and results must be returned before budget expiration, loading p-spaces from disk becomes therefore a core issue. In this section, we discuss a way to make p-space exploration go deeper in the graph, under access budget constraints.

Most sequences of users visited by TOPKS-ASYT are unique to each seeker. Thus, unless each possible sequence was materialised and cached on disk, p-spaces must be loaded one at a time. To tackle this issue, we propose to cluster users into *supernodes* and apply TOPKS-ASYT on the graph reduced to supernodes. Instead of loading one p-space at a time, several p-spaces included in the same supernode can be loaded jointly, with the tradeoff of some limited "off-track" exploration.

3 It is highly likely in practice that typing latency precludes most often a computation until termination conditions are reached.

	Twitter	Amazon	Tumblr	Yelp
Number of unique users	458,117	130,098	612,425	29,293
Number of unique items	1.6M	252,891	1.4M	18,149
Number of unique tags	550,157	91,352	2.3M	177,286
Number of triples	13.9M	24.7M	11.3M	30.3M
Average number of tags per item	8.4	53.8	7.9	685.7
Average tag length	13.1	6.9	13.0	6.5

Table 2.1: Statistics on the datasets we used in our experiments.

To build N supernodes, we first select N random users in the graph. Each user will correspond to the centroid of a supernode. Every remaining vertex u is then assigned to the supernode whose centroid is the closest to u. This method has the advantage of producing supernodes of relatively balanced sizes, which is exactly the purpose of clustering users into supernodes. Obviously, if the cluster sizes were unbalanced, that would make TOPKS-ASYT perform considerably worse when having to load many small supernodes. (Indeed, in preliminary experiments, state-of-the-art community detection assigned most users to few supernodes, letting most other supernode cardinalities far under the average number of users per cluster; this is why we followed a different user grouping.)

2.4 EXPERIMENTS

We evaluate in this section the effectiveness, scalability, and efficiency of TOPKS-ASYT. We used a Java implementation of our algorithms, on a low-end Intel Core i7 Linux machine with 16GB of RAM. We performed our experiments in an all-in-memory setting, for datasets of medium size (10-30 millions of tagging triples). We describe first the applications and datasets we used for evaluation.

2.4.1 *Datasets*

We used several popular social media platforms, namely Twitter, Tumblr, Yelp, and Amazon, from which we built sets of (user, item, tag) triples. Table 2.1 reports some statistics about each dataset.

TWITTER. We used a collection of tweets extracted during Aug. 2012. As described in Section 4.3.1, we see each tweet and its re-tweet instances as one item, and the authors of the tweets/re-tweets as its taggers. We include both the text and the hashtags as tags.

AMAZON MOVIES. We used a publicly available SNAP dataset of around 35 million movie reviews, spanning a period of 18 years up to March 2013. In this social media scenario, in order to build the user-item-tag triples, we simply considered the movie as the item, the author of the review as the tagger, and the keywords appearing in the review as the tags.

TUMBLR. We extracted a collection of Tumblr posts from Oct.-Nov. 2014, following the same interpretation on posts, taggers, and tags as in Twitter. Among the 6 different types of posts within Tumblr, we selected only the *default* type, which can contain text plus images. Moreover, in the case of Tumblr, we were able to access the follower-followee network and thus we extracted the induced follower-followee network for the selected taggers.

YELP. Lastly, we considered a publicly available Yelp dataset, containing reviews for businesses and the induced follower-followee network[4]. In this case, in order to build the triples, we considered the business (e.g., restaurant) as the item, the author of the review as the tagger, and the keywords appearing in the review as the tags.

For Twitter and Tumblr, to enrich the set of keywords associated to an item, we also expand each tag by the at most 5 most common keywords associated with it by a given user, i.e., by the tag-keyword co-occurrence. Finally, from the resulting sets of triples, we removed those corresponding to (i) items that were not tagged by at least two users, or (ii) users who did not tag at least two items.

To complete the data setting for our algorithm, we then constructed the user-to-user weighted networks that are exploited in the social-aware search. For this, we first used the underlying social network (when available). Specifically, for each user pair in Tumblr or Yelp, we computed the Dice coefficient corresponding to the common neighbors in the social network. To also study situations when such a network may not be available (as for Twitter and Amazon), exploiting a thematic proximity instead of a social one, we built two other kinds of user similarity networks, based on the Dice coefficient over either (i) the item-tag pairs of the two users, or (ii) the tags of the two users. We considered the filtering of "noise" links, weighted below a given cut-off threshold. Among the resulting ten networks, the Amazon tag similarity one was discarded due to poor connectivity coupled with high density and thus a less discriminative nature; we therefore report next on nine different network configurations.

2.4.2 *Experimental results: effectiveness*

We present in this section the results we obtained in our experiments for effectiveness, or "prediction power", with the purpose of validating the underlying as-you-type query model and the feasibility of our approach. In this framework, for all the data configurations we considered for effectiveness purposes, we imposed wall-clock time thresholds of $50ms$ per keystroke, which we see as appropriate for an interactive search experience.

To measure effectiveness, we followed an assumption used in recent literature, e.g. in [88, 96], namely that a user is likely to find his items – belonging to him or re-published by him – more interesting than random items from other users. For testing effectiveness, we randomly select triples (u, i, t) from each dataset. For each selected triple, we consider u as the seeker and t as the keyword issued by this user. The aim is to "get back" item i through search. The as-you-type scenario is simulated by considering that the user issues t one letter at a time. Note that an item may be retrieved back only if at least one user connected to the seeker tagged it. We picked randomly 800 such triples (we denote this selection as the set D), for tags having at least three letters. For each individual measurement,

4 http://www.yelp.com/dataset_challenge

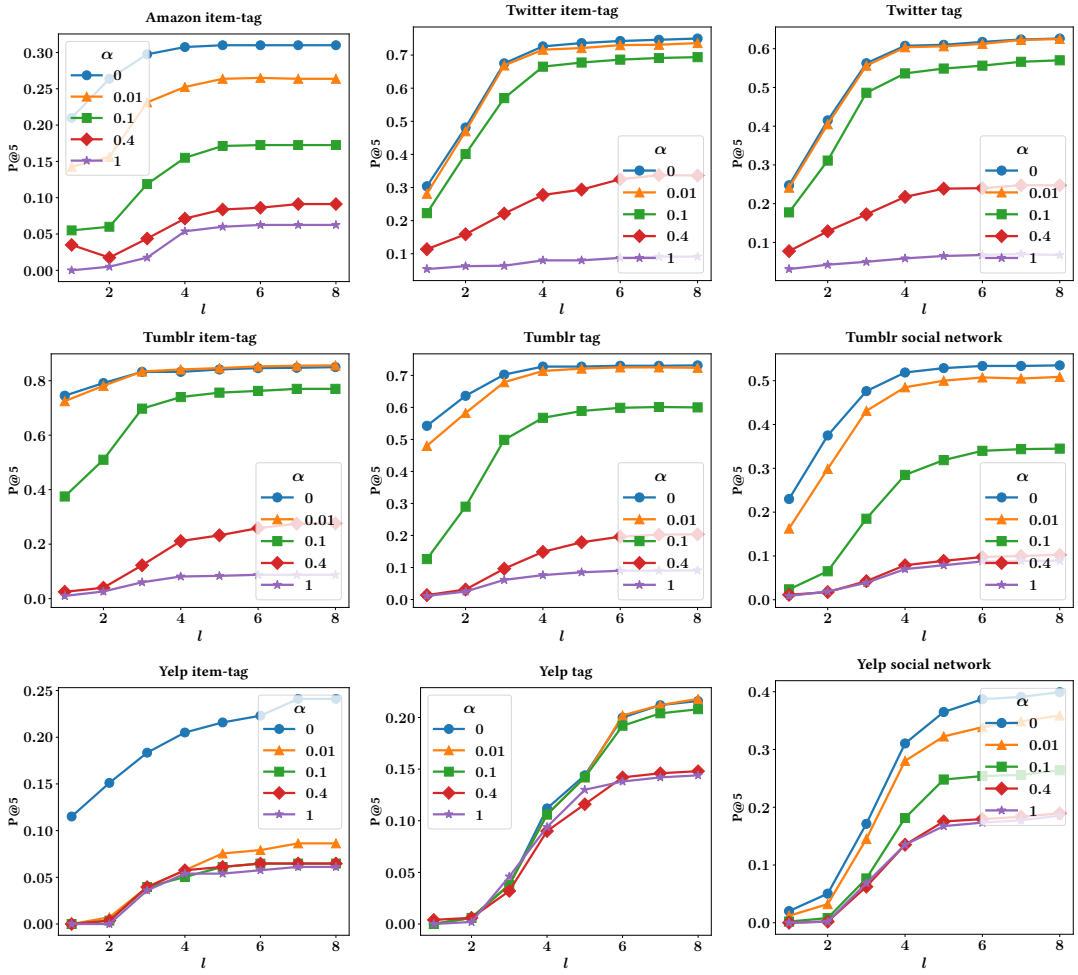


Figure 2.5: Impact of α on precision.

we gave as input a triple *(user, item, tag)* to be tested (after removing it from the dataset), and then we observed the ranking of *item* when *user* issues a query that is a prefix of *tag*.

Note that we tested effectiveness using single-word search for Twitter, Tumblr, and Amazon. On the contrary, for Yelp, due to its distinct features of having many triples per user, we did two-word search: given a query $q = (w_1, w_2)$, we first filtered items tagged by w_1, we then processed the remaining triples with query w_2 in the same manner as we did for Twitter and Tumblr.

We define the precision $P@k$ for our selected set D as

$$P@k = \frac{\#\{triple \mid ranking < k, triple \in D\}}{\#D}$$

Since $P@k$ can be seen as a function of the main parameters of our system, one goal was to understand how it is influenced by them. We describe now the different parameters we took into account.

- l, length of the prefix in the query (number of characters).
- θ, the threshold used to filter similarity links keeping only those having a score above.

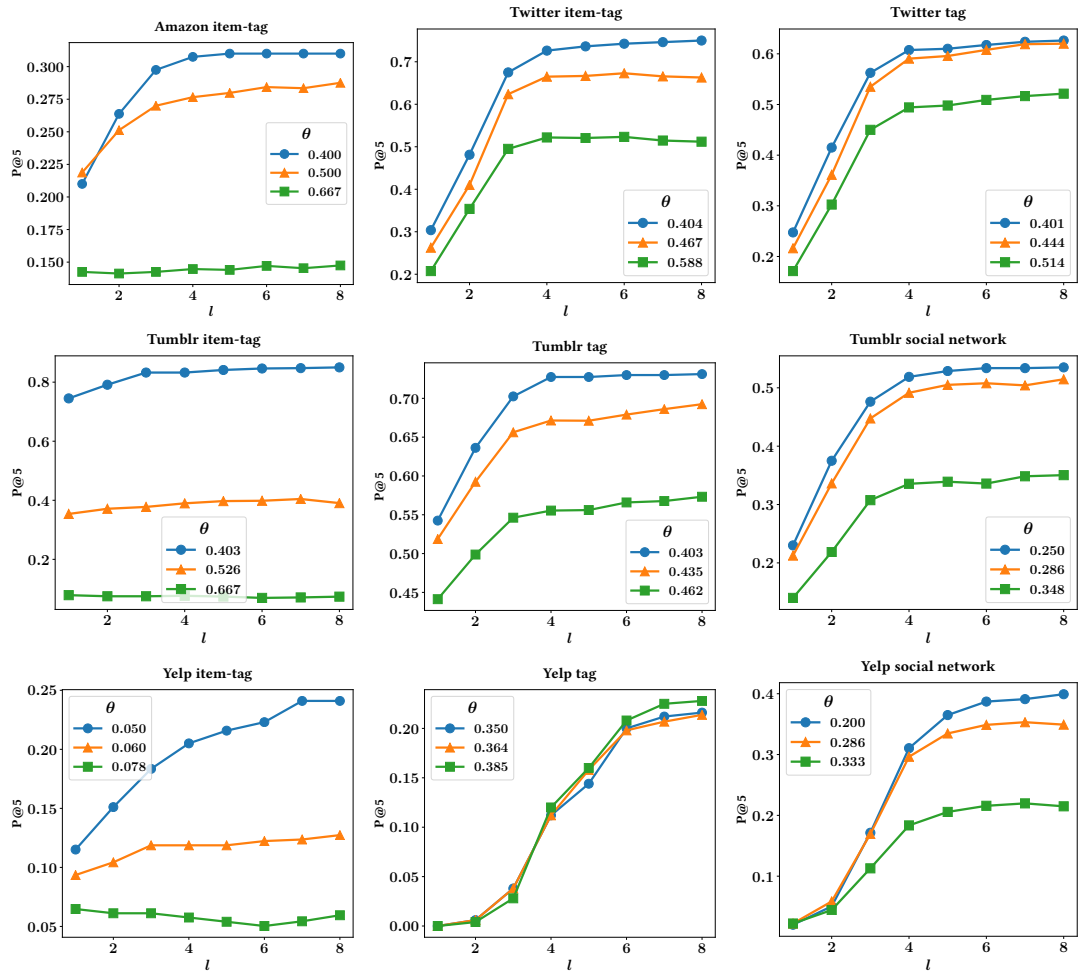


Figure 2.6: Impact of θ on precision.

- α, the social bias ($\alpha = 0$ for exclusively social score, $\alpha = 1$ for exclusively textual score).
- $\eta_i(u)$, the number of items tagged by user u, a user activeness indicator (for simplicity, hereafter referred to as η_i).
- $\eta_u(i)$, number of users who tagged item i, an item popularity indicator (η_u).

We present next the results we obtained for this experiment. In each figure, parameters whose impact is not monitored are set to the following default values: $\alpha = 0$ (fully social bias), θ is assigned the lowest value of the tested dataset, η_i and η_u are associated to active users and popular items ($\eta_i \geq 3$ and $\eta_u \geq 10$).

IMPACT OF α. As shown in Figure 2.5, α can have a major impact on precision. With a fully social bias ($\alpha = 0$), we obtained the best results for the four datasets and all the available similarity networks. Moreover, typing new characters to complete the prefix increases the precision. However, the evolution for $\alpha = 0$ can be quite slow, with the Tumblr or Yelp item-tag similarity network for witness. In this case, one likely reason is that these networks are quite rich in information, and the neighbors of the seeker are very likely to have the searched item, with the right tag, due to the way this network was built. This can

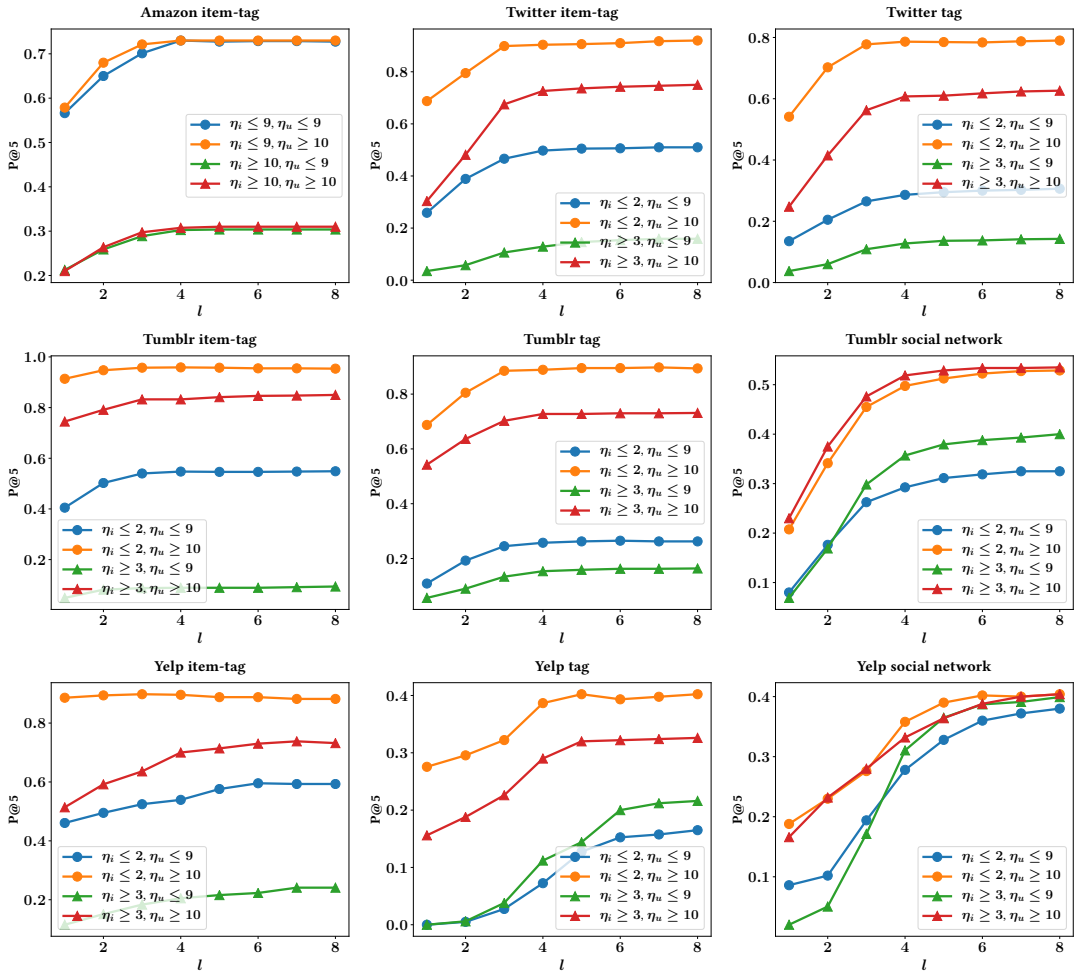


Figure 2.7: Precision for various types of users and items.

also explain why the precision for the item-tag networks is higher in the case of Tumblr than those for tag and social similarity networks. The precision for the social similarity network is the lowest for Tumblr, while in the case of the Yelp dataset the best results are obtained using the social network. Recall that the tag and item-tag networks were built based on the same content we were testing on, whereas the social similarity network only uses the links between users to infer distances between them. Yelp and Amazon exhibit lower precision levels overall, unsurprisingly, since they are denser datasets (more triples per user).

Interestingly, we obtain good precisions levels with such networks of similarity in social links (the highest in the case of Yelp) supporting our claim for social bias. For example, in the case of Tumblr, we can reach $P@5$ of around 0.82 for the item-tag similarity network, 0.7 for the tag one, and still 0.5 for the social one. This indicates that we can indeed find relevant information using a content-agnostic network using TOPKS-ASYT. Importantly, it also indicates that we can always search with the same social similarity network, even when the content evolves rather rapidly, with the same precision guarantees.

Finally, we observe that the evolution curve for small values of α, as new characters are added, varies depending on the similarity network. In Tumblr for example, the precision for low values of α does not increase much using the item-tag similarity network. The

items were found very close to the seeker and a few characters already give the final score in most cases. Very likely, the average number of items per user is too low to make the length of the prefix have an impact (most probably, users close to the seeker would not have several items tagged with the same prefix, even if this prefix is short). On the contrary, with the social similarity network, items with a tag matching the prefix are more likely to be diverse around the seeker. The distribution of the searched item in the network should thus be less concentrated around the seeker. Therefore, the number of result candidates with a high score for short prefixes is larger, and increasing l has more impact on the precision. Whereas an item-tag network tends to do so by definition, this can be seen as a clear consequence of the social bias that motivated our work.

IMPACT OF θ. In Figure 2.6, we illustrate the impact of θ on the quality of results. We mention that the two highest θ values lead to 33% and 66% cuts on the total number of edges obtained with the lowest θ value. Unsurprisingly, removing connections between users decreases the precision. When using the similarity network filtered by the lowest θ value, the seeker is almost always connected to the network's largest connected component, and we can visit many users to retrieve back the targeted item. With higher θ values, the connectivity for certain seekers we tested with is broken, making some of the tested items unreachable.

IMPACT OF POPULARITY / ACTIVENESS. We show in Figure 2.7 the effects of item popularity and user activity. For all similarity networks, the precision is better for popular items (high η_u). This is to be expected, as a popular item is more likely to be found when visiting the graph, as it is expected that it will score high since it has many taggers. Along with item popularity, we can observe that user activeness has a different effect in both content-based and the social similarity networks. Active users yield a better precision score when similarity comes from social links, whereas it is the opposite with content-based similarity networks. Reasonably, retrieving back an item for a non-active seeker in a content-based network is easier since his similarity with neighbors is stronger (Dice coefficient computed on less content).

2.4.3 *Experimental results: effectiveness with multiple words*

We describe next an additional experimental evaluation for effectiveness, focusing on the case of multiple word queries, in the densest dataset (Yelp). When dealing with multi-word queries, the score of the prefix can have a highly disproportionate weight compared to other terms in the query.

As we take the maximal score over all the possible completions (Eq.(2.6), (2.7)), the score for short prefixes is likely to be very high and render irrelevant the preceding terms. For example, if the query is composed of two terms t_1 and t_2, since t_2 is interpreted as a prefix, its length can influence the expectated score (for any value of α) as follows: say t_2 is a prefix of length 2, it is very likely to be the root of many possible completions; thus the expected value of the maximal score over all completions will likely be much larger than the score of t_1. Furthermore, note that short prefixes bring little information about a seeker's intent.

The top row of Figure 2.8 shows the values for precision for multi-word queries in Yelp, *without correcting the score of the last term.* The first four letters ($l = 1, \ldots, 4$) correspond

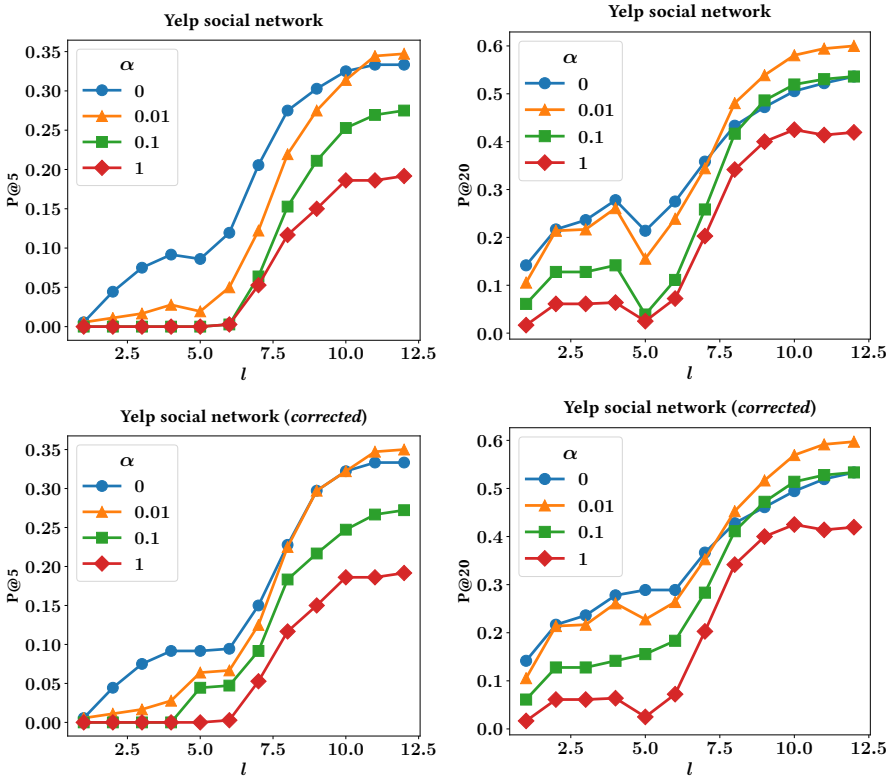


Figure 2.8: Precision for multiple-word queries without (top row) and with (bottom row) correction.

to the last letters of the first word. The following characters ($l \geq 5$) correspond to the next word. As expected, due the effect described above, we can see drops in the precision when starting a new word ($l = 5$) for any value of α. The precision loss is particularly observable when measuring $P@20$. This motivated the following model adjustment.

To make the score of a small prefix comparable to those of the preceding terms, we propose to re-scale it by a data-dependent constant. Specifically, for each prefix length ($l \geq 1$), we compute a normalizer value that maximized the precision through cross-validation. For example, we computed the parameter N_1 (i.e., the normalizer of prefixes of length 1) to optimize the precision of queries of the form $q = (t_1, p)$, where p is a prefix of length 1. Proceeding similarly for other prefix lengths, in Yelp we obtained constants $N_1 = 10^3$, $N_2 = 200$, $N_3 = 50$, $N_4 = 20$, $N_5 = 8$ and $N_6 = 2$.

In Figure 2.8 bottom row, with the normalized scores, we can observe that the drop of precision seen in Figure 2.8 has almost disappeared (see the case of $P@20$, where we had significant drops when starting a new term, but the correction with N_1 now preserves precision).

2.4.4 *Experimental results: effectiveness with SimRank proximity scores*

We conducted similar experiments for effectiveness using, instead of the neighborhood-based Dice proximity extended to shortest paths, the well-known path-based proximity model SimRank. For space reasons and to avoid repetition, we highlight the results over the densest dataset (Yelp), for comparison with all the initial plots for effectiveness. This

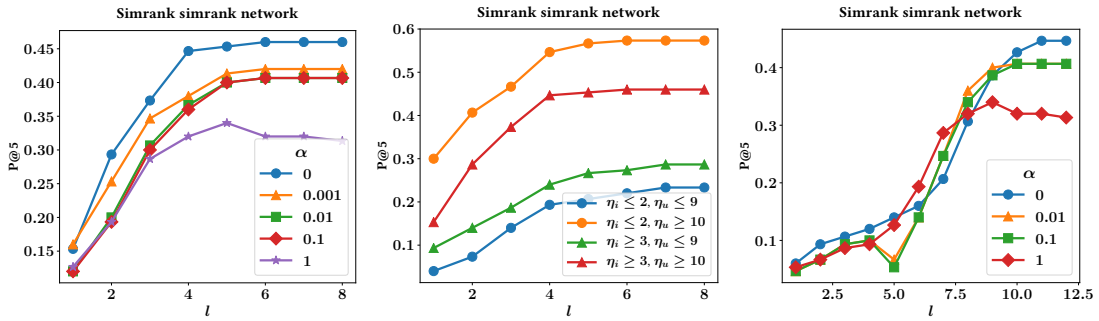


Figure 2.9: Impact of α (left), type of users / items (middle), multiple words with correction (right) with SimRank proximity (for comparison with Figures 2.5, 2.7, and 2.8 bottom row respectively).

allowed us to observe how the chosen similarity (local/single-path or global) impacts results. The SimRank model, introduced by [56], gives a recursive definition of the similarity between users u and v as follows:

$$\sigma^{SimRank}(u,v) = \frac{c}{|N_u||N_v|} \sum_{u' \in N_u, v' \in N_v} \sigma^{SimRank}(u',v') \tag{2.8}$$

for some decay factor $c \in [0,1]$. (A similar definition can be given for directed graphs.) The rationale is that "two objects are considered to be similar if they are referenced by similar objects". Since this definition is recursive, the SimRank score between two users depends on the whole graph.

In Figure 2.9, we used SimRank similarity computed on the social network instead of the Dice's coefficient used before. On the left figure, we display the impact of α on precision and observe the best results using a fully social bias. Interestingly, we have a slight improvements using SimRank similarity as it reaches a precision P@5 of 0.45 after typing 5 letters, which is to compare to the value 0.35 observed with Dice's coefficient. This supports the use of path-based similarity measures that encompass the general relationships between nodes.

On the right figure, we observe the impact of user activeness and item popularity on precision. Once again, results are better for popular items. Note that SimRank has no cut-off threshold preventing us from experimenting the impact of θ.

2.4.5 *Experimental results: efficiency and scalability*

We now turn our attention to the efficiency and scalability aspects of our solutions. *NDCG@k* In Figure 2.10 top row, we display the evolution of NDCG@20 vs. time, for the densest dataset (Yelp), for different α values (where α is normalized to have similar social and textual scores in average). The NDCG is computed w.r.t. the exact top-k that would be obtained running the algorithm on the entire graph. Formally, the *normalized discounted cumulative gain* is defined as follows:

Definition 2.1 (NDCG@k). *Let* $D_{ORACLE} = (o_1, \ldots, o_k)$ *be the top-k that would be obtained running the algorithm on the entire graph, sorted by decreasing score lower bounds and* $D_{TOPKS\text{-}ASYT} = (i_1, \ldots, ß_k)$ *the top-k obtained by TOPKS-ASYT. Finally, let* rel(i) *denote*

the relevance of item i (in our experiments, it is set to $k+1$ minus the position of i in the oracle topk-k D_{oracle}). The normalized discounted cumulative gain *accumulated at position k is:*

$$NDCG@k = \frac{DCG_k}{IDCG_k}, \textit{where } DCG_k = \sum_{l=1}^{k} \frac{\text{rel}(i_l)}{\log_2(l+1)} \textit{ and } IDCG_k = \sum_{l=1}^{k} \frac{\text{rel}(o_l)}{\log_2(l+1)}.$$

This measure is an important indicator for the feasibility of social-aware as-you-type search, illustrating the accuracy levels reached under "typing latency", even when the termination conditions are not met. In Fig. 2.10, we fixed the prefix length size to $l = 4$. The left plot is when a user searches with a random tag (not necessarily used by her previously), while the right plot follows the same selection methodology as in Section 2.4.2. Importantly, with α corresponding to exclusively social or textual relevance, we reach the exact top-k faster than when combining these two contributions ($\alpha = 0.5$). Note also that this trend holds even when the user searches with random tags.

In Figure 2.10 bottom row, we show the evolution of NDCG@20 vs. time in Yelp, for different prefix lengths (the left plot is for random tags). Results shows that with lower l values we need more time to identify the right top-k. The reason is that shorter prefixes can have many potential (matching) items, hence the item discrimination process evolves more slowly. For these prefix lengths, we only mention here that we also analyzed the evolution of NDCG@20 when visiting a *fixed number number of users*, observing similar behavior. As expected, the more users we visit the higher NDCG we reach and for longer prefixes it is necessary to visit more users. E.g., when $l = 6$, after visiting 500 users, we reach an NDCG of 0.8 while for $l = 2$ the NDCG after 500 visits is 0.9.

Finally, in the experiment illustrated in Figure 2.11 we observed the time to reach the exact top-k for different dataset sizes. For that, we sorted Yelp triples chronologically and partitioned them into five consecutive (20%) chunks. For each dataset we perform searches using prefixes of $l = 2, 3, 4, 5$. While the time to reach the exact top-k increases with bigger datasets and shorter prefixes, the algorithm scales adequately when l is more than 2. For instance, for $l = 3$, the time to reach the result over the complete dataset is just twice the time when considering only 20% of this dataset.

2.4.6 *Experimental results: incremental versus non-incremental TOPKS-ASYT*

We now analyze the impact of the incremental computation. In Figure 2.12, we display the time to reach the exact top-k for both TOPKS-ASYT and its incremental counterpart. For that, we compare the two algorithms on sequences of consecutive prefixes, e.g. *sou*, *sour*, *sourc*, and *source*. Let p_t and p_{t+1} be two consecutive queries differing by a single character. Whereas TOPKS-ASYT starts a new query for each new letter, the incremental version calculates the answer for p_{t+1} relying on computations for p_t. Obviously, the time to reach the exact top-k for the first prefix p_1 is the same (the same algorithm is run). For $l = 4$, the time to reach the result is already slightly smaller for the incremental version of the algorithm. We emphasize that the first part of the incremental algorithm, which consists in filtering the previous candidate list explains the small improvement. For longer prefixes ($l = 5, 6$), the candidate list is shorter and the incremental algorithm takes full advantage of previous computations (speed increase from $\times 2$ for $l = 5$ up to $\times 4$ for $l = 6$).

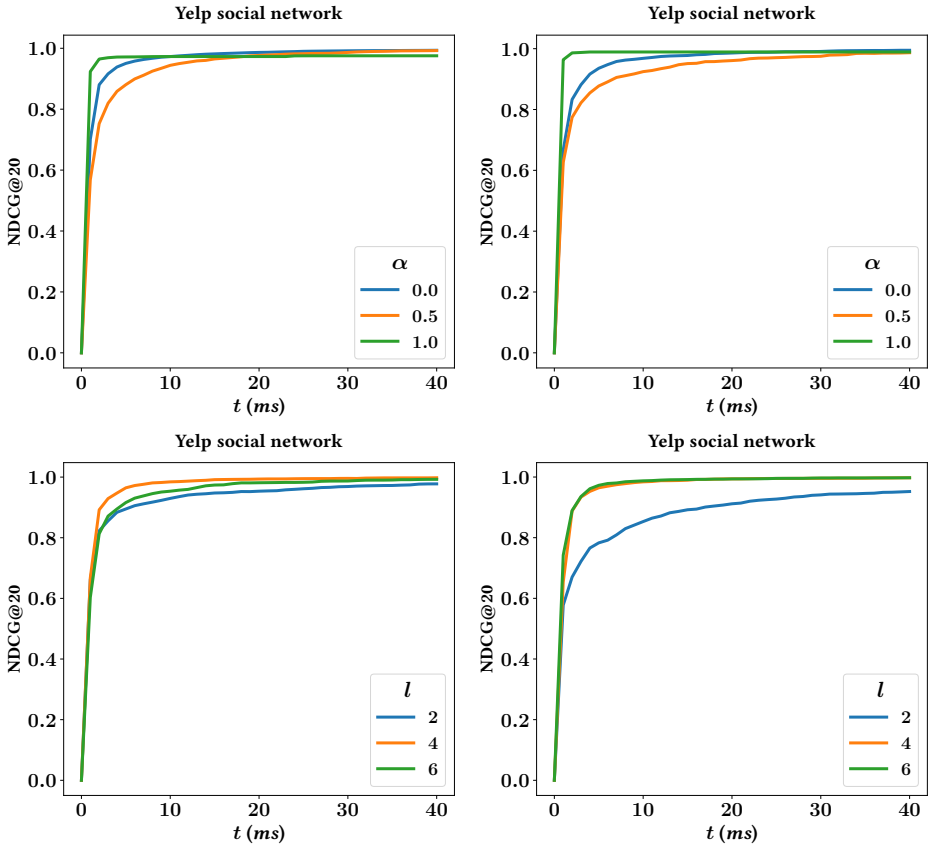


Figure 2.10: Impact of α (top) or l (bottom) on NDCG vs time for random search (left) or personal search (right).

2.4.7 *Experimental results: TOPKS-ASYT versus state-of-the-art baseline methods*

We compare TOPKS-ASYT with three different baselines methods. The first two methods respectively build on the state-of-the-art social top-k search TOPKS algorithm from [88] and the type-ahead textual search algorithm NRA(HEAP) of [78][5]. The third method relies on the online Yelp Search API with query-autocompletion.

TOPKS-M: SOCIAL TOP-k BASELINE. We first compare TOPKS-ASYT to TOPKS-M (for TOPKS-Merge), a baseline method that follows a natural idea relying on the social top-k state-of-the-art (such as algorithm TOPKS from [88]), but does not benefit from CT-IL (i.e., does not benefit from prefix-based retrieval). The approach of TOPKS-M works in two stages, as follows: first, we load the inverted lists of all the possible completions of the final term given in the query and merge them in a unique list. As a result, this step may be very costly for short prefixes. Once the first step is completed, we can directly apply algorithm TOPKS, using prefixes as complete words with their own inverted list.

In Figure 2.13 top row, we show the NDCG@20 of TOPKS-ASYT and TOPKS for various budgets (50, 100, 200, 400 – each value corresponds to a color intensity, from lighter to darker) and various α. A budget B corresponds to the maximal number of *significant disk accesses* we allow the algorithm to do to answer a query. In our interpretation, a significant

5 This method was implemented and made available by the authors, as part of an instant-search engine called SRCH2; its source code is available at https://github.com/SRCH2/srch2-ngn.

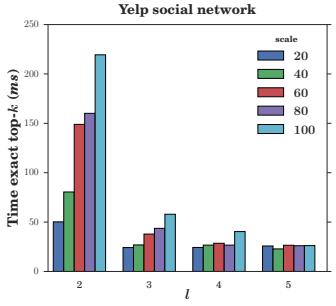


Figure 2.11: Time to exact top-k for different dataset sizes.

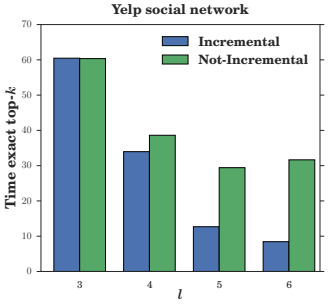


Figure 2.12: Incremental vs non-incremental version.

disk access can be either a p-space exploration (visit of the next user in the algorithm) or the loading of an inverted list. Our experiments show that the second is more costly ($\times 12$ in average), thus, we count a budget consumption of 12 for a disk access corresponding to an entire inverted list. Similarly to Section 2.4.5, the NDCG is computed w.r.t. the exact top-k that would be obtained by running the algorithm on the entire similarity graph without budget restrictions.

For α ranging from 0 to 1, we observe a similar behavior, when comparing TOPKS-ASYT to the baseline method. Results show that the NDCG of TOPKS-M is much smaller than the one of TOPKS-ASYT, even for relatively important budgets (e.g., $B = 400$). The cost of merging inverted lists before applying TOPKS prevents the algorithm from providing high-quality answers fast. For example, for short prefixes ($l = 3, 4$), too many completions are possible and thus the baseline loads too many inverted lists compared to the budget. Even for a budget $B = 400$, TOPKS cannot catch up with the precision of TOPKS-ASYT for a prefix length $l = 3$.

TOPKS-2D: TEXTUAL SEARCH AS-YOU-TYPE BASELINE. We now compare TOPKS-ASYT to a *dimension-at-a-time* approach, denoted TOPKS-2D (for TOPKS-2-Dimensions), which processes social and textual contributions separately. First, we retrieve documents matching the query on the textual dimension using the NRA(HEAP) type-ahead baseline from [78]. That is, we read inverted lists and build the candidate list, ignoring the social contribution (we do not use the social graph and no p-space is explored). In a second stage, we then explore the similarity graph to obtain the social contribution in the final score.

In Figure 2.13 middle row, for the same budget values and color code as before, we show the NDCG@20 of TOPKS-ASYT and TOPKS-2D, for various values of α ($\alpha = 1$ is not considered, as it is a case where TOPKS-ASYT and the baseline are virtually the same). We can see that small values of α highly favor TOPKS-ASYT: NRA(HEAP) spends useless budget on inverted lists, since it runs without knowledge of the social scores. On the contrary, TOPKS-ASYT benefits from simultaneous social and textual score computations to avoid using unnecessary inverted lists. When α increases, the textual contribution becomes more significant and the baseline method becomes more competitive, especially for longer prefixes that do not have many possible completions.

AUTOCOMPLETION+TOPKS BASELINE. We complete our performance comparison with AUTOCOMPLETE+TOPKS, a baseline method that relies on the Yelp Search API for

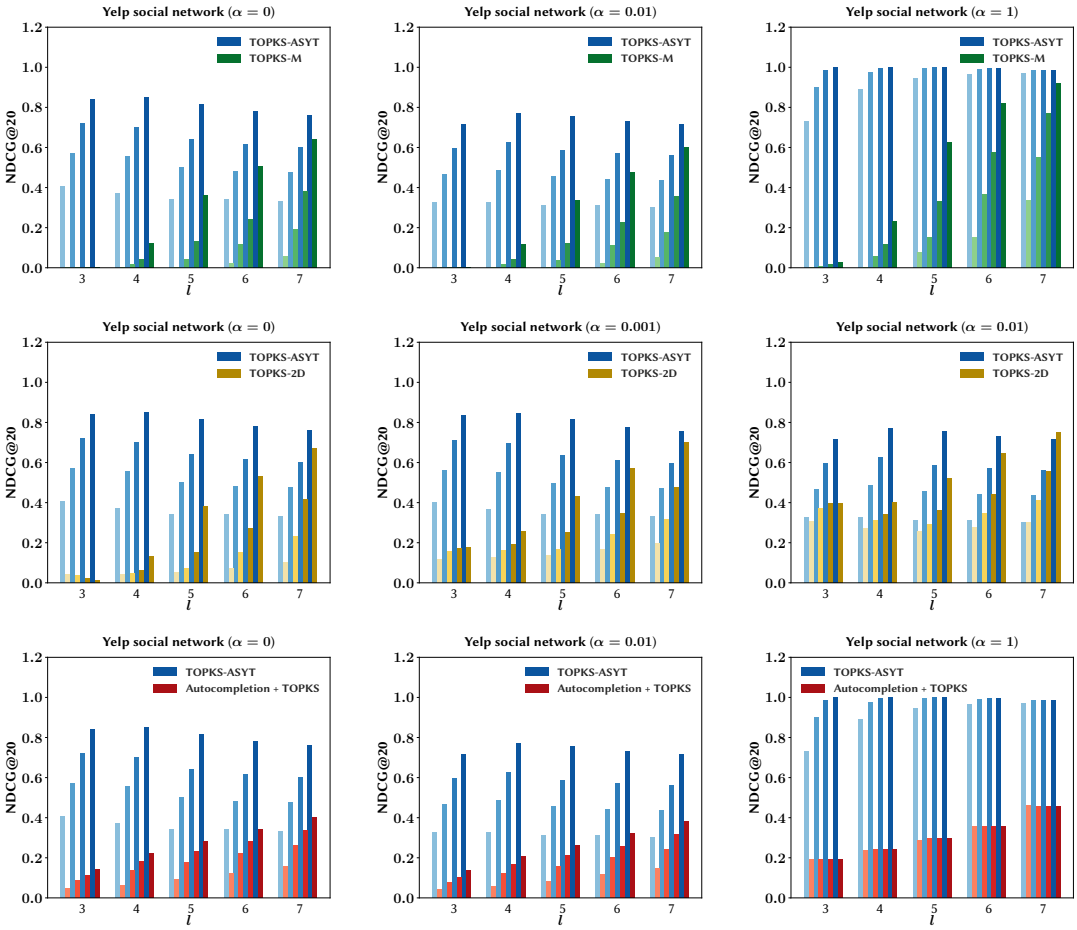


Figure 2.13: TOPKS-ASYT vs. TOPKS-M (top), TOPKS-2D (middle) or Autocompletion + TOPKS (bottom) baselines with fixed budget.

query-autocompletion[6]. This baseline method proceeds as follows: we obtain a set of queries that are predicted by Yelp to complete the current query (prefix) the seeker is typing, without using any social information (the service is not "personalized"). We then use these queries to get the set of top-k results over our data, by simply running for them the aforementioned state-of-the-art network-aware top-k algorithm TOPKS from [88]. We give TOPKS-ASYT and the baseline the same budgets as in previous experiments and, to avoid any potential evaluation bias in our favor, any costs from the Yelp API autocompletion step are ignored.

In Figure 2.13 bottom row, we show the NDCG@20 of TOPKS-ASYT and AUTOCOMPLETE+TOPKS using the same display convention as before. We can see that for all values of α, the NDCG of AUTOCOMPLETION+TOPKS is significantly smaller than the one of TOPKS-ASYT. As the API does not use social information to construct autocompletions, the final top-k is likely affected by the general query trend and this should explain the NDCG for low values of α. Interestingly, we can however observe a similar behavior even for high values of α (textual score).

6 https://github.com/Yelp/yelp-api-v3/blob/master/docs/api-references/autocomplete.md

2.4.8 *Experimental results: supernodes*

In Figure 2.14, we show the impact of the supernode materialization feature, for supernodes of average size $d = 6$ and $d = 30$. For three different budgets ($B = 10, 30, 50$), we run TOPKS-ASYT with the original similarity network and the supernode-reduced graph. Similarly to the previous section, the budget corresponds to the number of significant disk accesses we allow our algorithm to do until it outputs a top-k result. For budget $B = 50$, supernodes do not increase the NDCG, in particular for short prefixes. Small prefixes have many completions and thus are very common. This means that most of the NDCG contribution is obtained with few visited nodes. When the budget given to TOPKS-ASYT is smaller, supernodes improve the ranking quality. For instance, with budget $B = 10$, very few nodes can be visited by TOPKS-ASYT and the supernodes become a key feature. With supernodes of 6 users (resp. 30), the algorithm aggregates p-spaces of up to 60 people (resp. 300), whereas it would visit at most 10 neighbors using the initial similarity network.

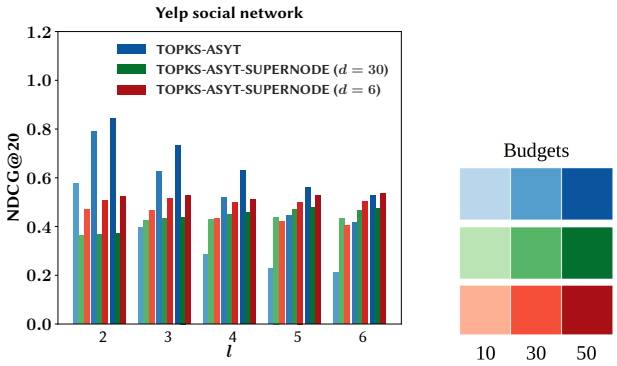


Figure 2.14: Impact of supernode sizes on NDCG for several prefix lengths.

MAIN-MEMORY VS. SECONDARY MEMORY CONSIDERATIONS. We emphasize here that we performed our experiments in an all-in-memory setting, for datasets of medium size (tens of millions of tagging triples), in which the advantages of our approach may not be entirely observed. In practice, in real, large-scale applications such as Tumblr, one can no longer assume a direct and inexpensive access to p-spaces and inverted lists, even though some data dimensions such as the user network and the top levels of CT-IL – e.g., the trie layer and possibly prefixes of the inverted lists – could still reside in main memory. In practice, with each visited user, the search might require a random access for her personal space, hence the interest for the sequential, user-at-a-time approach. Even when p-spaces may reside on disk, we experimentally observed that we can reach good precision levels by retrieving a small number of them (e.g., less than 100); depending on disk latency, serving results in, for example, under 100*ms* seems within reach. One way to further alleviate such costs may be to cluster users having similar proximity vectors, and choose the layout of p-spaces on disk based on such clusters; this is an approach we intend to evaluate in the future, at larger scale.

2.5 CONCLUSION

In this chapter, we study as-you-type query search in social media through the prism of an adaptive user-centric problem. Our aim was to retrieve the top-k ranked results, under a network-aware query model by which information produced by users who are closer to the seeker can be given more weight. We formalize this problem and we describe the GT-UCB algorithm to solve it. Our solution is based on a novel trie data structure, Index, allowing ranked access over inverted lists. In several application scenarios, we perform extensive experiments for efficiency, effectiveness, and scalability, validating our techniques and the underlying query model. As a measure of efficiency, since as-accurate-as-possible answers must be provided while the query is being typed, we investigate how precision evolves with time and, in particular, under what circumstances acceptable precision levels are met within reasonable as-you-type latency (e.g., less than 100ms). Also, as a measure of effectiveness, we analyze thoroughly the "prediction power" of the results produced by GT-UCB.

3

ADAPTIVE INFLUENCE MAXIMIZATION

In this chapter, we propose a method to maximize the spread of influence in a (potentially *unknown*) social network over multiple consecutive rounds. More precisely, we study the problem of sequentially selecting spread seeds in a graph under the hypothesis that previously activated nodes can still transfer information, but do not yield further rewards. Importantly, we make no assumptions on the underlying diffusion model. We call this problem *online influence maximization with persistence*. We describe a novel algorithm, GT-UCB, relying on upper confidence bounds on the so-called *missing mass*, that is, the expected number of nodes that can still be reached from a given seed. We show that our approach leads to high-quality spreads on both simulated and real datasets, despite being model-free. Moreover, it is orders of magnitude faster than state-of-the-art methods, making it possible to deal with very large graphs.

The first part of this chapter is devoted to the presentation of the problem and an overview of existing works in influence maximization, in both *offline* and *online* scenarios. Our method is described in the second part of this chapter. Finally, we present extensions for *semi-bandit* influence maximization and *time-varying* IM. This work required the development of a software package coded in C++, available at `https://github.com/plagree/oim`.

Contents

3.1 INTRODUCTION

Advertising based on word-of-mouth diffusion in social media has become very important in the digital marketing landscape. Nowadays, social value and social influence are arguably the hottest concepts in the area of Web advertising and most companies that advertise in the Web space must have a "social" strategy. For example, on widely used platforms such as Facebook or Twitter, promoted posts are interleaved with normal posts on user feeds. Users interact with these posts by actions such as "likes" (adoption), "shares" or "reposts" (network diffusion). This represents an unprecedented tool in advertising, be it with a commercial intent or not, as products, news, ideas, movies, political manifests, tweets, etc, can propagate easily to a large audience [119, 120].

Motivated by the need for effective viral marketing strategies, *influence estimation* and *influence maximization* (IM) have become important research problems, at the intersection of data mining and social sciences [36]. In short, IM is the problem of selecting a set of nodes from a given diffusion graph, maximizing the expected spread under an underlying diffusion model. This problem was introduced in 2003 by the seminal work of Kempe et al. [65], through two stochastic, discrete-time diffusion models, *Linear Threshold* (LT) and *Independent Cascade* (IC). These models rely on diffusion graphs whose edges are weighted by a score of influence. They show that selecting the set of nodes maximizing the expected spread is NP-hard for both models, and they propose a greedy algorithm that takes advantage of the submodularity property of the influence spread, but does not scale to large graphs. A rich literature followed, focusing on computationally efficient and scalable algorithms to solve IM. We discuss this in more details in Section 3.2.

Importantly, all the IM studies discussed in the benchmarking study of Arora et al. [5] have as starting point a specific diffusion model (IC or LT), whose graph topology and parameters – basically the edge weights – are known. In order to *infer* the diffusion parameters or the underlying graph structure, or both, [34, 42–44, 46, 102] propose *offline, model-specific methods*, which rely on observed information cascades[1].

There are however many situations where it is unreasonable to assume the existence of relevant historical data in the form of cascades. For such settings, *online approaches*, which can learn the underlying diffusion parameters *while running diffusion campaigns*, have been proposed. Bridging IM and inference, this is done by balancing between exploration steps (of yet uncertain model aspects) and exploitation ones (of the best solution so far), by *multi-armed bandit* techniques, where an agent interacts with the network to infer influence probabilities [23, 113, 114, 121].

Nevertheless, all these studies on inferring diffusion networks, whether offline or online, rely on parametric diffusion models, i.e., assume that the actual diffusion dynamics

1 In short, information cascades are time-ordered sequences of records indicating when a specific user was activated or adopted a specific item.

are well captured by such a model (e.g., IC). This maintains significant limitations for practical purposes. First, the more complex the model, the harder to learn in large networks, especially in campaigns that have a relatively short timespan, making model inference and parameter estimation very challenging within a small horizon (typically tens or hundreds of spreads). Second, it is commonly agreed that the aforementioned diffusion models represent elegant yet coarse interpretations of a reality that is much more complex and often hard to observe fully. For examples of insights into this complex reality, the *topical* or *non-topical* nature of an influence campaign, the *popularity* of the piece of information being diffused, or its specific *topic* were all shown to have a significant impact on hashtag diffusions in Twitter [34, 48, 100].

OUR CONTRIBUTION. Aiming to address such limitations, we propose in this work a *large-scale approach for online and adaptive IM*, in which the underlying assumptions for the diffusion processes are kept to a minimal (if, in fact, hardly any). We argue that it can represent a versatile tool in many practical scenarios. More precisely, we focus on social media diffusion scenarios in which influence campaigns consist of multiple *consecutive trials* (or *rounds*) spreading the same type of information from an arbitrary domain (be it a product, idea, post, hashtag, etc).[2] The goal of each campaign is to reach (or *activate*) as many distinct users as possible, the objective function being the total spread. In our setting – as in, arguably, the real-world – the campaign selects from a set of *spread seed candidates*, a small subset of a potentially large and unknown population. At each round, the learning agent picks among the candidates those from which a new diffusion process is initiated in the network, gathers some feedback on the activations, and adapts the subsequent steps of the campaign; the agent may "re-seed" certain nodes (we may want to ask a particular node to initiate spreads several times, e.g., if it has a strong *converting impact*). This perspective on influence campaigns naturally imposes a certain notion of *persistence*, which is given the following interpretation: users that were already activated in the ongoing campaign – e.g., have adopted the product or endorsed the political candidate – remain activated throughout that campaign, and thus will not be accounted for more than once in the objective function.

We call this problem *online influence maximization with persistence* (in short, OIMP). Our solution for it follows the multi-armed bandit idea initially employed in Lei et al. [73], but we adopt instead a *diffusion-independent perspective*, whose only input are the spread seed candidates, while the population and underlying diffusion network – which may actually be the superposition of several networks – remain unknown. In our bandit approach, the parameters to be estimated are the values of the candidates – how good is a specific candidate –, as opposed to the diffusion edge probabilities of a known graph as in [73]. Furthermore, we assume that different campaigns are independent, and make the model's feedback more realistic: after each trial, the agent only gathers the set of activated nodes. The rationale is that oftentimes, for a given "viral" item, we can track in applications only *when* it was adopted by various users, but not *why*.

The multi-armed bandit algorithm we propose, called GT-UCB, relies on a famous statistical tool known as the *Good-Turing estimator*, first developed during WWII to crack the Enigma machine, and later published by Good in a study on species discovery [45]. Our approach is inspired by the work of Bubeck et al. [17], who proposed the use of the Good-

2 Repeated exposure, known also as the "effective frequency", is a crucial concept in marketing strategies, online or offline.

Turing estimator in a context where the learning agent needs to sequentially select experts that only sample one of their potential nodes at each trial. In contrast, in OIMP, when a candidate is selected, it may have a potentially large spread and may activate many nodes at once. Our solution follows the well-known *optimism in the face of uncertainty* principle from the bandit literature, by deriving an upper confidence bound on the estimator for the remaining potential for spreading information of each candidate, and by choosing in a principled manner between explore and exploit steps.

Through our approach, we show that efficient and effective influence maximization can be done in a highly uncertain or under-specified social environment, along with formal guarantees on the achieved spread.

3.2 INFLUENCE MAXIMIZATION OVERVIEW

In this section, we give a general overview of the existing works in offline and online influence maximization.

3.2.1 *Influence discrete-time propagation models*

We consider a graph $G = (V, E)$, where V is the set of nodes, and E the set of directed edges connecting pairs of nodes. The diffusion of information or influence in the graph is a *discrete* process for which, at each time step t, a node is either *active* or *inactive*. Several propagation models have been proposed in the literature. They all have in common an initial set of nodes $I \subseteq V$ – the seed set – that contains the initial activated nodes before the spread of influence in the graph. Denoting by the random variable $S(I)$ the spread initiated by the seed set I, the *influence spread* function is the expected size of the spread: $\sigma(I) := \mathbb{E}[|S(I)|]$.

We now describe two of the most commonly studied propagation models, namely, the independent cascade (IC) and the linear threshold (LT) models.

INDEPENDENT CASCADE. The independent cascade model was initially formalized by Kempe et al. [65] based on particle physics models [82] and marketing studies [41]. The core idea of the IC model is to consider that an edge gets activated – diffuse the information – independently of any other edge. The IC model assumes the existence of a probability parameter $p(u, v) \in [0, 1]$ on every edge $(u, v) \in E$.

Definition 3.1 (Independent cascade). *Given a graph $G = (V, E)$ with activation probability function p, the IC model is a discrete stochastic process starting from an initial set of* active *seeds $I \subseteq V$ and which proceeds as follows. When a node $u \in V$ becomes active, it tries to activate every unactive neighbor v* once*, succeeding with probability $p(u, v)$ independently of the past activations. The process stops when no further activation is possible.*

Example 3.1 *We give an example of an IC diffusion process in Fig. 3.1. Initially, at $t = 0$, a single node is seeded in Fig. 3.1a. In the example, it corresponds to the first use of the hashtag #falcon9 on a social platform. At time $t = 1$, the seed successfully activates its one neighbor that can be activated with probability 0.2 but fails to activate its two other neighbors. Note that failed activations are represented by red crosses. At time $t = 2$, the node that has just been activated at $t = 1$ activates its 2 unactive neighbors (Fig. 3.1c). Finally, at time $t = 3$, there is*

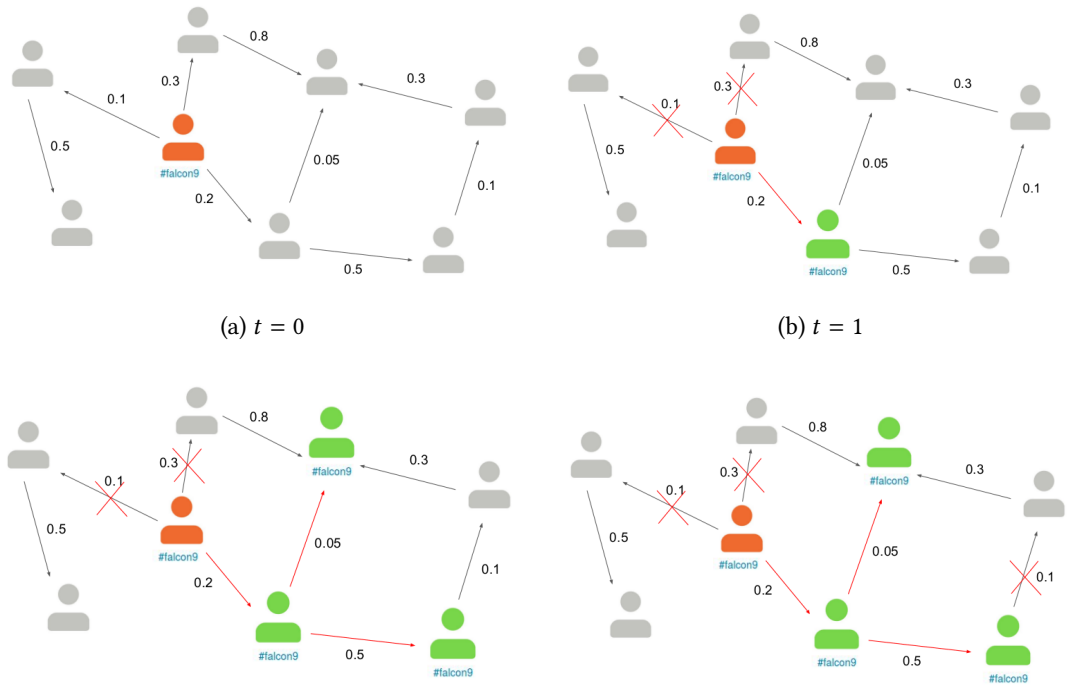


Figure 3.1: An example of the diffusion process under the IC model. The seed node is represented in orange while green nodes denote the nodes activated via the diffusion. The edges whose activations succeeded are depicted in red whereas failed edge activations are denoted by a red cross.

a single unactive neighbor and its activation fails. After $t = 3$, there is no new activated node so the diffusion process stops (Fig. 3.1d).

LINEAR THRESHOLD. The IC model is suitable for modeling situations where a single exposure to an *active* neighbor suffices to activate a node. However, it fails to incorporate the cumulative effect that often arises when several neighbors are activated. For example, when buying a new product, people may need positive feedback from several of their friends before taking the plunge. This idea was initially introduced in social sciences by Granovetter [49] and Schelling [104] who based their approach on *node-specific* thresholds. Later, Kempe et al. [65] introduced the Linear Threshold model for which nodes tend to activate monotonically with the number of their activated neighbors.

Definition 3.2 (Linear threshold). *Given a graph $G = (V, E)$ with weight function w, and such that for each node $v \in V$, $\sum_{u \text{ neighbor of } v} w(u, v) \leq 1$, the LT model is a discrete process starting from an initial set of* active *seeds $I \subseteq V$ proceeding as follows. Each node v chooses a* random *threshold $\theta_v \in [0, 1]$. Then, at every time step t, all nodes that were active at $t - 1$ remain active and each unactive node v at $t - 1$ becomes active if the total weight of its active neighbors is at least θ_v, that is:*

$$\sum_{u \text{ active neighbor of } v} w(u, v) \geq \theta_v.$$

Intuitively, the threshold θ_v represents how easily the node v is influenced by its neighbors.

SUBMODULARITY. The IC and LT models share an important property, which will turn out to be important when designing efficient algorithms in the next section, namely, the *submodularity* of their expected number of influenced nodes.

Definition 3.3 (Submodularity). *Let Ω be a finite set. A set function $f : 2^{\Omega} \to \mathbb{R}$ is* submodular *if for every $A, B \subseteq \Omega$ with $A \subseteq B$ and every $X \notin B$,*

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B).$$

Intuitively, this means that when a new element is added to a set A, f increases even more as A is small. This is sometimes referred to as the *diminishing marginal return* property. In their seminal paper, Kempe et al. [65] show that σ is submodular in both IC and LT models.

Theorem 3.1 (Theorems 2.2 and 2.5 [65]). *Under both independent cascade (IC) and linear threshold (LT) models, the influence spread function σ is submodular.*

3.2.2 *Influence maximization*

The traditional problem of influence maximization (IM) is to select a set of seed nodes $I \subseteq V$, under a cardinality constraint $|I| = L$, such that the expected *spread* of an influence cascade starting from I (or the expected number of activated nodes) is maximized. Formally:

Problem 3.1 (Influence maximization). *Denoting by the random variable $S(I)$ the spread initiated by the seed set I, IM aims to solve the following optimization problem:*

$$\arg\max_{I \subseteq V,\, |I| = L} \sigma(I) = \mathbb{E}[|S(I)|].$$

The statement of Problem 3.1 makes no assumption on the underlying diffusion model. In the following, we will focus on influence maximization under the independent cascade and linear threshold models.

COMPLEXITY OF IM. Given a diffusion model – the IC or LT model – there are two computational tasks in order to solve the IM problem:

1. computing the value of $\sigma(\{u\})$ for every user $u \in V$ – this task is sometimes called the *influence computation*,
2. finding the optimal set I maximizing the expected spread as described in Problem 3.1.

The following theorem states that the computation of σ is #P-hard under both the IC and the LT propagation models. The complexity class #P is the set of counting problems associated to problems in NP: an NP problem tries to answer a *decision* problem (yes-no question), whereas a #P problem looks for the number of solutions to the problem. Similarly to decision problems, we can define completeness and hardness for *counting*

problems: (i) a problem is #P-complete if it is in class #P and any other problem from #P can be reduced to it in polynomial time, (ii) a problem is #P-hard if it can be reduced from a #P-complete problem with a polynomial transformation.

Theorem 3.2 (Theorem 1 [117] and Theorem 1 [22]). *Given a graph $G = (V, E)$ and an initial set of seeds I (possibly a singleton), computing the influence spread $\sigma(I)$ is #P-hard under both the IC and LT models.*

This theorem essentially says that solving the first task (the influence computation) is hard. It turns out that the selection of the optimal set is also a difficulty a task. This can easily be seen because the IC model contains the set cover problem as a special case and the LT model contains the vertex cover problem as a special case which are both NP-complete.

Theorem 3.3 (Theorems 2.4 and 2.7 [65]). *Given a graph $G = (V, E)$, solving the influence maximization problem is NP-hard under both IC and LT models.*

At this point, we know that the problem to solve is very challenging for two reasons. There has been much research focusing on solving the first task, namely, the influence computation. They all try to estimate the expected influence from every node in the graph. We will describe several of the proposed ideas in the following section.

Interestingly, all IM methods share the same approximation method to solve the second task. The selection of the optimal set is done by the famous GREEDY algorithm for monotone and submodular functions, which allows to return answers that cannot deviate too much from the optimal set.

FINDING THE OPTIMAL SET. In Algorithm 6, we give the generic GREEDY algorithm used to select a set that greedily increases the value of the monotone and submodular function f.

Algorithm 6: GREEDY algorithm

Data: size of the returned set L, monotone and submodular set function f
1 **Initialization:** $S = \emptyset$;
2 **for** $i = 1, \ldots, L$ **do**
3 Choose $u = \arg\max_{v \in V \setminus S} f(S \cup \{v\})$;
4 $S = S \cup \{u\}$;
5 **end**
Result: S

Algorithm 6 can be easily adapted to the IM problem. The submodular function f is simply replaced in Line 3 by an *estimation* $\hat{\sigma}$ of the influence function σ. The main strength of the GREEDY algorithm is that it cannot provide too bad approximations because of the submodularity of the function to optimize. More specifically, the algorithm provides a $1 - 1/e$-approximation to the optimal set S^*, which roughly corresponds to a 63%-approximation. Note that this result – summarized in Theorem 3.4 – is very conservative, but, in practice, the approximation given by the GREEDY algorithm is usually much better for IM problems [80].

Theorem 3.4 (Greedy algorithm [93]). *Let f be a monotone and submodular set function and $S^* = \arg\max_{|S|=L} f(S)$ be the set of size L maximizing f. We denote by $\hat{S}$ the set computed by the* Greedy *procedure from Algorithm 6. We have that*

$$f(\hat{S}) \geq \left(1 - \frac{1}{e}\right) f(S^*).$$

Proof *We denote $S^* := \{s_1^*, \ldots, s_L^*\}$ and let $\hat{S}_i := \{\hat{s}_1, \ldots, \hat{s}_i\}$ be the set chosen by the* Greedy *algorithm at step $i \geq 1$. By definition, $\hat{S}_0 := \emptyset$.*

Then, for $l \in \{0, \ldots, L-1\}$, one has

$$\begin{aligned}
f(S^*) &\leq f(S^* \cup \hat{S}_l) && (f \text{ is monotone}) \\
&= f(S^* \cup \hat{S}_l) - f(\{s_1^*, \ldots, s_{l-1}^*\} \cup \hat{S}_l) + f(\{s_1^*, \ldots, s_{l-1}^*\} \cup \hat{S}_l) \\
&= \cdots \\
&= f(\hat{S}_l) + \sum_{i=1}^{L} f(\{s_1^*, \ldots, s_j^*\} \cup \hat{S}_l) - f(\{s_1^*, \ldots, s_{j-1}^*\} \cup \hat{S}_l) \\
&\leq f(\hat{S}_l) + \sum_{i=1}^{L} f(\hat{S}_l \cup \{s_i^*\}) - f(\hat{S}_l) && (f \text{ is submodular}) \\
&\leq f(\hat{S}_l) + \sum_{i=1}^{L} f(\hat{S}_{l+1}) - f(\hat{S}_l) && (\text{by greedyness of the Algorithm}) \\
&= f(\hat{S}_l) + L(f(\hat{S}_{l+1}) - f(\hat{S}_l)).
\end{aligned}$$

Let $\Delta_l := f(S^*) - f(\hat{S}_l)$. We obtain that

$$\Delta_{l+1} \leq (1 - 1/L)\Delta_l,$$

and thus, by induction, $\Delta_L \leq (1 - 1/L)^L \Delta_0$. By definition, $f(\emptyset) = 0$, and thus, $\Delta_0 = f(S^*)$. Reorganizing terms concludes the proof as $(1 - (1 - 1/L)^L) \geq 1 - 1/e$. □

3.2.3 *Efficient influence computation*

A plethora of algorithms have been proposed to solve the influence computation task of the IM problem, under specific diffusion models. These algorithms can be viewed as *full-information* and *offline* approaches: they choose all the seeds at once, in one step, and they have the complete diffusion configuration, i.e., the graph topology and the influence probabilities. Here, we review a few existing algorithms to solve the IM problem under the IC and LT models.

In their original work, Kempe et al. [65] propose MC-Greedy, a simple Monte Carlo (MC) method to estimate the influence function σ. Given a set of seed nodes S, we simulate R random cascades and average the number of influenced nodes. Not surprisingly, the parameter R is critical in the confidence of the estimation of $\sigma(S)$. An MC simulation is performed before choosing each new node. More precisely, given a set $\hat{S}_k$ of $k < L$ chosen nodes, Greedy selects the node v that gives the largest marginal improvement. Thus, for every $u \in V$, R simulations are performed to estimate $\sigma(\hat{S}_k \cup \{u\})$. Kempe et al. [65] propose to perform the diffusion process 10,000 times but they do not give any

theoretical guarantee. The time complexity of the algorithm is $O(LnRm)$ where n is the number of nodes and m is the number of edges. Indeed, there are L rounds, one for each new node added to the current set. Furthermore, to select each new node, the algorithm needs to go through all the nodes that have not been selected yet, and, for each of these nodes, R simulations that can potentially traverse all edges are performed. In summary, we easily see that MC-Greedy does not perform efficiently in large-scale scenarios.

Significant progress has thus been made since the foundational work of [65], for proposing methods to estimate the influence function σ that scale to large networks with formal guarantees on the approximation. In their benchmarking study, Arora et al. [5] classify the different approaches into several categories: (i) the most straightforward methods rely on *explicit MC simulations* such as MC-Greedy [65], CELF [74] and CELF++ [47], (ii) in 2014, a new line of research focusing on *Reverse Reachable sets* of nodes to estimate the node influence led to algorithms such as RIS [12], TIM [110] and SSA [94], (iii) finally, *snapshots* methods, like PMC [95], generate several instances G_i of the diffusion graph and estimate node influence from these samples. In the following, we explain the main ideas of these 3 categories. The reader interested in the details of the algorithms will refer to the associated papers.

EXPLICIT MC SIMULATIONS. CELF [74] and CELF++ [47] proceed similarly to MC-Greedy, but use the submodularity of σ to reduce the number of simulations required to estimate the node influences. The marginal gain of a node at a given iteration is smaller than the marginal gain computed at the previous iteration. This allows CELF to prune Monte Carlo simulations at subsequent rounds and speed up greatly the spread computation.

REVERSE REACHABLE SETS. Borgs et al. [12] introduced a new approach for influence computation, namely, the Reverse Influence Sampling (RIS), which led to a renew of interest in the IM area.

Definition 3.4 (Reverse Reachable (RR) set). *Given a graph $G = (V, E)$, a* Reverse Reachable *set R is generated by (1) selecting a node $v \in V$ at random (2) generating a random spread – the RR set – from v on the graph where every directed edge from G is reversed.*

Note that the nodes in R correspond to nodes that can reach v in a sample graph. This definition proves to be key in designing efficent IM algorithm. If we generate a set of RR sets, influential nodes will likely appear in many of them. We formally state this intuition in Lemma 3.1.

Lemma 3.1 (Observation 3.2 in [12]). *Given $G = (V, E)$ and an RR set R generated from G, we have for every $I \subseteq V$,*

$$\sigma(I) = n\mathbb{P}(I \cap R \neq \emptyset).$$

Proof *Remember that the random variable $S(\{s\})$ denotes the spread initiated by the seed node $s \in V$. The random variable $S^T(\{u\})$ denotes the same in the* reverse *graph, that is, it corresponds to the set of users which influenced u in a spread.*

$$\begin{aligned}
\sigma(I) &= \sum_{u \in V} \mathbb{P}\left(\exists s \in I \text{ such that } u \in S(\{s\})\right) \\
&= \sum_{u \in V} \mathbb{P}\left(\exists s \in I \text{ such that } s \in S^T(\{u\})\right) \\
&= n\mathbb{P}\left(u \text{ chosen at random in } V, \exists s \in I \text{ such that } s \in S^T(\{u\})\right) \\
&= n\mathbb{P}\left(S \cap R \neq \emptyset\right).
\end{aligned}$$

□

Lemma 3.1 essentially says that the expected influence of a set of seeds I is proportional to the probability that it intersects a random RR set R. Thus, to find a set I that maximizes the influence function σ, the problem can be cast to selecting the set that intersects as many RR sets as possible. The number of RR sets generated is key in controlling the error in the approximation of the influence function. Given an approximation $\epsilon > 0$, TIM [110], IMM [109], SSA [94] and SSA-Fix[55] successively reduced the number of required sampled RR sets, but the core of the algorithm remains the same.

SNAPSHOTS. The snapshots methods are similar to explicit MC methods but work at the graph level as opposed to the node level. For example, PMC [95] – method designed specifically for the IC model – generates R instances of the graph G_i (called snapshots): for each edge $(u, v) \in E$, it keeps the edge with probability $p(u, v)$. Then, it estimates the influence of nodes by aggregating all the snapshots. Similarly to the MC methods, the number of simulations R controls the error of the estimation. However, the main advantage of this approach is that the simulations are shared between all nodes, making this method much faster than MC-Greedy for instance. Furthermore, PMC proposes an effective pruning strategy to avoid unnecessary computations.

Arora et al. [5] discuss the pros and cons of the best known techniques for IM. In particular, the authors highlight that the *Weighted Cascade* (WC) instance of IC, where the weights associated to a node's incoming edges must sum to one, leads to poor performance for otherwise rather fast IC algorithms. They conclude that PMC [95] is the state-of-the-art method to efficiently solve the IC optimization problem, while TIM+ [110] and IMM [109] – later improved by [94] with SSA – are the best current algorithms for WC and LT models. In a response paper, Lu et al. [84] examine the work and the experimental methodology of [5] and refute several claims formulated by Arora et al.

Among the "myths" raised by [5], we experimentally verified that the methods relying on the sampling of RR sets can be surprisingly slow under the WC instance of IC. Consequently, we always used PMC in our numerical expirements under the IC model. Conversely, similarly to [84], we observed that IMM is not slower than TIM – we actually obtain noticeable speed gains using IMM –, which refutes a claim formulated by [5]. Finally, while implementing IM algorithms, we noticed that authors use different pseudo-random number generators whose efficiency varies significantly from one to another. This is in line with a remark formulated by [55] which reveals that the claimed speed improvements of SSA are overestimated by [94] due to flaws in the expiremental setup.

OTHER RELATED WORKS. Besides the already discussed offline methods for inferring the diffusion network and its parameters, we mention here that a first *offline and model-free method* for inferring the diffusion network from existing cascades has been proposed recently in [101]. We have in common with this work the goal to devise generic, non-parametric methods, yet in a online IM framework.

Other methods have been devised to handle the prevalent uncertainty in diffusion media, e.g., when replacing edge probability scores with ranges thereof, by solving an IM problem whose *robust* outcome should provide some effectiveness guarantees w.r.t. all possible instantiations of the uncertain model [24, 52].

Methods for IM that take into account more detailed information, such as topical categories, have been considered in the literature [8, 34, 118]. Interestingly, [100] experimentally validates the intuition that different kinds of information spread differently in social networks, by relying on two complementary properties, namely *stickiness* and *persistence*. The former can be seen as a measure of how viral the piece of information is, passing from one individual to the next. The latter can be seen as an indicator of the extent to which repeated exposures to that piece of information impact its adoption, and it was shown to characterize *complex contagions*, of controversial information (e.g., from politics).

Finally, we also mention here some work to identify the best spreaders in social platforms using a graph topology viewpoint. Kitsak et al. [66] show that the spreading influence potential of a node can be explained by indicators such as the *density* and the *cohesiveness*. More recently, Malliaros et al. [86] found that the *truss* number is an even better indicator.

3.2.4 *Online influence maximization*

In the *online* case, during a sequence of N (what we call hereafter the *budget*) consecutive trials, L seed nodes are selected at each trial, and *feedback* on the achieved spread from these seeds is collected. In the literature, we can distinguish two kinds of *online* influence maximization: the *semi-bandit* IM and the online IM with *persistence* (OIMP).

SEMI-BANDIT IM. Chronologically, the semi-bandit IM problem was introduced after the OIMP, but, as it shares many aspects with the classic semi-bandit literature, it is somewhat easier to apprehend.

Problem 3.2 (Semi-bandit IM). *Given a graph $G = (V, E)$, a budget of N trials, and a number $1 \leq L \leq n$ of nodes to be activated at each trial, the objective of the* semi-bandit influence maximization *problem is to solve the following problem:*

$$\underset{I_n \subseteq V, |I_n|=L, 1 \leq n \leq N}{\arg\max} \mathbb{E}\left[\sum_{n=1}^{N} |S(I_n)|\right].$$

In the formulation of Problem 3.2, two key aspects are deliberately left unspecified as they vary in the problems considered in the literature:

1. *diffusion model*: Semi-bandit studies such as [23, 114, 121] assume that the underlying diffusion model is the independent cascade model. Similarly to our work, Vaswani et al. [113] propose a diffusion-independent framework using pairwise reachability parameterization.

2. *feedback*: We can distinguish two types of feedback in the literature. The *node-level* feedback assumes that the learning agent only observes the activated nodes, but does not know which edge has been successfully activated and which has not. This adds another challenge when trying to estimate diffusion probabilities. Conversely, *edge-level* feedback assumes that the learner observes edges' failed and successful activations, facilitating parameter estimation.

The learning agent sequentially selects seeds from which diffusion processes are initiated in the network; the obtained feedback is used to update the agent's knowledge of the model. By interacting with the network, the agent tries to infer influence probabilities. Not surprisingly, the problem is very challenging: there is an unknown parameter on every edge, which can be dramatic as social networks have typically up to billions of users and trillions of connections nowadays.

To make the learning problem more tractable, Wen et al. [121] make a linear assumption on the unknown parameters. More precisely, they assume that each edge $(u, v) \in E$ has a feature vector $x_{u,v} \in \mathbb{R}^d$ where the dimension d is typically very small compared to n. These feature vectors are *known* to the learner – they can be constructed using the characteristics of the two connected users. The authors assume that there exists a *unique unknown* feature vector $\theta^* \in \mathbb{R}^d$ such that, for every edge $(u, v) \in E$, $p(u, v) \approx x_{u,v}^T \theta^*$. From there, the problem can be solved with a variation of the LinUCB algorithm [79].

In a recent paper, Vaswani et al. [113] propose a diffusion-independent approach to solve the semi-bandit IM problem. They assume that, for each *pair* of users (u, v), the probability $p^*_{u,v}$ that u activates v is given by the scalar product between a *known* feature vector $x_u \in \mathbb{R}^d$ – this vector is associated to the source –and an *unknown* vector associated to the target $\theta^*_v \in \mathbb{R}^d$. Interestingly, the authors propose to set the known features to the d lowest eigenvectors of the Laplacian graph. In this setting, there are $|V| \times d$ unknown parameters, making this method still unsuitable in large-scale scenarios.

ONLINE IM WITH PERSISTENCE. The semi-bandit IM framework is suitable for scenarios where the marketing firm wants to make (many) consecutive campaigns on a static graph, and, importantly, where the goal is to convert as many users as possible at each spread, regardless of the activated users. In particular, activating the same users at each step is not considered an issue in the semi-bandit IM setting. However, in many real scenarios, these requirements are not satisfied. For example, nowadays, when a company releases a new product, it often makes a marketing campaign on social platforms in order to give the product visibility. The firm aims at reaching as many *distinct* people as possible, and may ask influential users – e.g., influential people contractually bound by a sponsorship to the company – to post about the product. Every time an influential user posts something on the social platform, he / she initiates a spread that leads to the activation of a subset of the population. Importantly, the company's ultimate goal is to maximize the total spread of the campaign, that is, the number of distinct users activated over the consecutive steps.

In the IM with persistence setting, influence campaigns typically consist of multiple consecutive trials spreading the *same type of information* (e.g., a product, an idea, etc).

Problem 3.3 (Online influence maximization with persistence (OIMP)). *Given a graph $G = (V, E)$, a budget of N trials, and a number $1 \leq L \leq n$ of nodes to be activated at*

each trial, the objective of the online influence maximization with persistence *problem is to solve the following problem:*

$$\underset{I_n \subseteq V, |I_n|=L, 1 \leq n \leq N}{\arg\max} \mathbb{E}\left|\bigcup_{n=1}^{N} S(I_n)\right|.$$

A key difference w.r.t. semi-bandit influence maximization studies such as [23, 113, 114, 121] is that these look for a *constant* optimal set of seeds, while the difficulty with OIMP is that the seemingly best action at a given trial depends on the activations of the previous trials (and thus the learning agent's past decisions).

The OIMP problem was first introduced by Lei et al. [73]. They propose a method for the IC model where they try to estimate the diffusion graph (the graph with its diffusion probabilities). More precisely, at each step, adopting an explore-exploit strategy, they maintain an estimation (or a confidence bound) of the probabilities of all edges in the graph. The estimated graph is called the *uncertain influence graph* and is given to any IM algorithm to select a set of seed nodes. Finally, a spread is initated from this set and *edge-level* feedback are used to improve probability estimations in subsequent steps.

We also mention that adaptive strategies have been studied in situations where all parameters are known – that is, situations were no learning task is necessary. The applications that rely on adaptive influence maximization tasks are generally time-critical, and thus, to overcome these challenges, Salha et al. [103] recently introduced a realistic *myopic* feedback model together with an approximated algorithm.

3.3 ONLINE INFLUENCE MAXIMIZATION VIA CANDIDATES

3.3.1 *Setting*

The goal of the *online influence maximization with persistence via candidates*, the problem we proposed in this thesis, is to successively select (or *activate*) a number of spread seed nodes from a known population of candidates, in order to *reach* (or *spread* to) as many other nodes as possible. In this section, we formally define this problem.

3.3.1.1 *Influence maximization via candidates*

The short timespan of campaigns makes parameter estimation very challenging within small horizons. In other cases, the knowledge of the topology – or even the existence – of a graph is too strong an assumption. In contrast to Lei et al. [73], we do not try to estimate edge probabilities in some graph, but, instead, we assume the existence of a subpopulation of users – referred to as the *spread seed candidates* (in short, *candidates*), in the following – who are the only access to the medium of diffusion. Formally, we let $[K] := \{1, \ldots, K\}$ be a set of candidates for selection; each candidate is connected to an unknown and potentially large base (the candidate's *support*) of basic nodes, each with an unknown activation probability. For illustration, we give in Figure 3.2 an example of this setting, with 3 candidates connected to 4, 5, and 4 basic nodes, respectively.

Now, the problem boils down to estimating the value or spread potential of the K candidates, which is typically much smaller than the number of parameters of the diffusion model. The medium over which diffusion operates may remain a general diffusion graph,

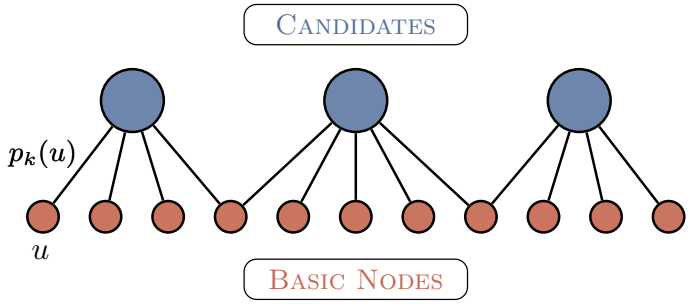


Figure 3.2: Three candidates with associated activation probabilities $p_k(u)$.

just like in the literature, but we make no assumption on that: the diffusion may happen in a completely unknown environment. Finally, note that by choosing $K = |V|$ candidates, the classic IM problem can be seen as a special instance of our setting.

We complete the formal setting by assuming the existence of K sets $A_k \subseteq V$ of basic nodes such that each candidate $k \in [K]$ is connected to each node in A_k. We denote $p_k(u)$ the probability for candidate k to activate the child node $u \in A_k$.

In this context, the diffusion process can be abstracted as follows.

Definition 3.5 (Influence process). *When a candidate $k \in [K]$ is selected, each basic node $u \in A_k$ is* sampled *for activation, according to its probability $p_k(u)$. The* feedback *(or* spread*) for k's selection consists of all the activated nodes, while the associated* reward *consists of the* newly activated *ones.*

REMARK. Limiting the IM method to working with a small subset of the entire set of nodes may seem overly restrictive, but it allows to rapidly estimate nodes' values. As a motivating example, take marketing firms that may not have knowledge of the entire diffusion graph, only having access to a few influential people that can diffuse information (the candidates in our setting). Moreover, despite the fact that we model the reach of every candidate by 1-hop links to the to-be-influenced nodes, these edges are just an abstraction of the activation probability, and may represent in reality longer paths in an underlying unknown real influence graph G.

3.3.1.2 *Online influence maximization with persistence via candidates*

We are now ready to formally state our online influence maximization with persistence via candidates:

Problem 3.4 (OIMP via candidates). *Given a set of candidates $[K] := \{1, \dots, K\}$, a* budget *of N trials, and a number $1 \leq L \leq K$ of candidates to be activated at each trial, the objective of the online influence maximization with persistence (OIMP) via candidates is to solve the following optimization problem:*

$$\underset{I_n \subseteq [K], |I_n| = L, \forall 1 \leqslant n \leqslant N}{\arg\max} \mathbb{E} \left| \bigcup_{1 \leqslant n \leqslant N} S(I_n) \right|.$$

As noticed in [73], the offline IM can be seen as a special instance of the online one, where the budget is $N = 1$. Note that, in contrast to persistence-free online influence maximization – considered, e.g., in [114, 121] – the performance criterion used in OIMP

displays the so-called *diminishing returns property*: the expected number of nodes activated by successive selections of a given seed is decreasing, due to the fact that nodes that have already been activated are discounted. We refer to the expected number of nodes remaining to activate as the *potential* or *missing mass* of a seed. The diminishing returns property implies that there is no static best set of seeds to be selected, but that the algorithm must follow an adaptive policy, which can detect that the remaining potential of a seed is small and switch to another seed that has been less exploited. Our solution to this problem has to overcome challenges on two fronts: (1) it needs to estimate the potential of nodes at each round, without knowing the diffusion model nor the activation probabilities, and (2) it needs to identify the currently best seeds, according to their estimated potential.

Other approaches for the online IM problem rely on estimating diffusion parameters [73, 114, 121] – generally, a distribution over the influence probability of each edge in the graph. However, the assumption that one can estimate accurately the diffusion parameters – and notably the diffusion probabilities – may be overly ambitious, especially in cases where the number of allowed trials (the budget) is rather limited. A limited trial setting is arguably more in line with real-world campaigns: take as example political or marketing campaigns, which only last for a few weeks.

In our approach, we work with parameters on *nodes*, instead of edges. More specifically, these parameters represent the potentials of remaining spreads from each of the candidate seed nodes. We stress that these potentials can evolve as the campaign proceeds. In this way, we can go around the dependencies on specific diffusion models, and furthermore, we can *remove entirely the dependency on a detailed graph topology*. Finally, note that even though OIMP is not a typical bandit problem – the potentials evolve as the campaign progresses –, it bears several similarities with multi-armed bandits. Indeed, an agent needs to *sequentially* choose arms (called *candidates* in this chapter) whose potentials are *unknown*, and thus, it needs to explore concurrently with the exploitation of received feedback. Furthermore, we rely on the *optimism in face of uncertainty* framework to construct our UCB-like algorithm.

3.3.2 *The GT-UCB algorithm*

In this section, we describe our UCB-like algorithm, which relies on the Good-Turing estimator to sequentially select the seeds to activate at each round, from the available candidates.

3.3.2.1 *Missing mass and Good-Turing estimator*

Given the K candidates, the OIMP problem boils down to the following: *How should we select a candidate at each step?* More precisely, a good algorithm for OIMP should aim at selecting the candidate k with the largest potential for influencing its children A_k. However, the true potential value of a candidate is *a priori* unknown to the decision maker.

We now describe our approach to estimate this value, using the concept of *missing mass*.

In the following, we index trials by t when referring to the time of the algorithm, and we index trials by n when referring to the number of selections of the candidate. For example, the t-th spread initiated by the algorithm is noted $S(t)$ whereas the n-th spread of candidate k is noted $S_{k,n}$.

Definition 3.6 (Missing mass $R_k(t)$). *Consider a candidate $k \in [K]$ connected to A_k basic nodes. Let $S(1), \ldots, S(t)$ be the set of nodes that were activated during the first t trials by the seeded candidates. The* missing mass $R_k(t)$ *is the expected number of* new *nodes that would be activated upon starting a $t+1$-th cascade from k:*

$$R_k(t) := \sum_{u \in A_k} \mathbb{1}\left\{u \notin \bigcup_{i=1}^{t} S_k(t)\right\} p_k(u),$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function.

Definition 3.6 provides a formal way to obtain the remaining potential of a candidate k at a given time. The optimal policy would simply select the candidate with the largest missing mass at each time step. The difficulty is, however, that the probabilities $p_k(u)$ are unknown. Hence, we have to design a *missing mass estimator* $\hat{R}_k(t)$ instead. It is important to stress that the missing mass is a random quantity, because of the dependency on the spreads $S_k(t), \ldots, S_k(t)$. Furthermore, due to the diminishing returns property, the sequence $(S_{k,n})_{n \geq 1}$ is stochastically decreasing.

Following ideas from [17], we now introduce a version of the Good-Turing statistic originally introduced by [45], tailored to our problem of rapidly estimating the missing mass. Denoting by $n_k(t)$ the number of times candidate k has been selected after t trials, we let $S_1, \ldots, S_{n_k(t)}$ be the $n_k(t)$ cascades sampled independently from candidate k. We denote by $U_k(u,t)$ the binary function whose value is 1 if node u has been activated *exactly* once by candidate k – such occurrences are called *hapaxes* in linguistics – and $Z_k(u,t)$ the binary function whose value is 1 if node u has never been activated by candidate k. The idea of the Good-Turing estimator is to estimate the missing mass as the proportion of hapaxes in the $n_k(t)$ sampled cascades, as follows:

$$\hat{R}_k(t) := \frac{1}{n_k(t)} \sum_{u \in A_k} U_k(u,t) \prod_{l \neq k} Z_l(u,t).$$

Albeit simple, this estimator turns out to be quite effective in practice. If a candidate is connected to a combination of both nodes having high activation probabilities and nodes having low activation probabilities, then successive traces sampled from this candidate will result in multiple activations of the high-probability nodes and few of the low-probability ones. Hence, after observing a few spreads, the candidate's potential will be low, a fact that will be captured by the low proportion of hapaxes. In contrast, estimators that try to estimate each activation probability independently will require a much larger number of trials to properly estimate the candidate's potential.

To verify this assumption in reality, we conduct an analysis of the empirical activation probabilities from a Twitter dataset. Specifically, we used a collection of tweets and re-tweets gathered via crawling in August 2012. For each original tweet, we find all corresponding retweets, and, for each user, we compute the empirical probability of a retweet occurring – this, in our case, is a proxy measure for influence probability. Specifically, for every user v "influenced" by u, i.e., v retweeted at least one original tweet from u – we compute the estimated diffusion probability: $p_{u,v} = |u\text{'s tweets retweeted by } v| \,/ |\text{tweets by } u|$. In Fig. 3.3 (left), we show the survival function of resulting empirical probabilities in a log-log plot. We can see that most probabilities are small – the 9th decile has value 0.045.

In Fig. 3.3 (right), we simulated the activation probabilities of a set of 50 nodes whose activation probabilities are chosen randomly from the Twitter empirical probabilities. Most

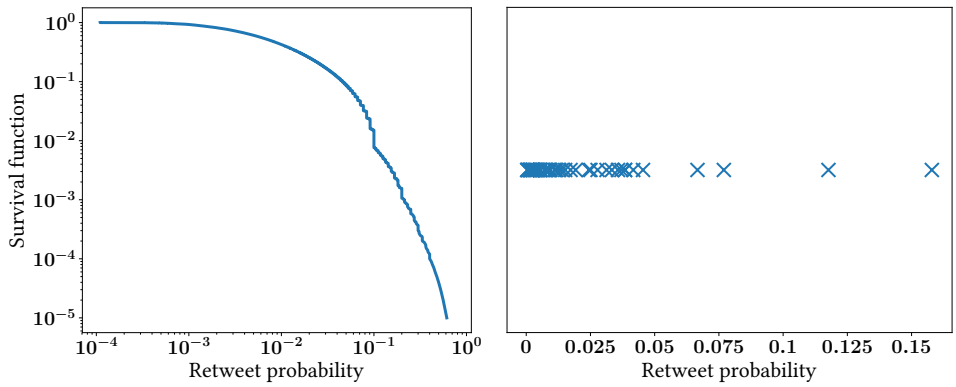


Figure 3.3: (left) Twitter empirical retweet probabilities (right) Sample of 50 empirical retweet probabilities

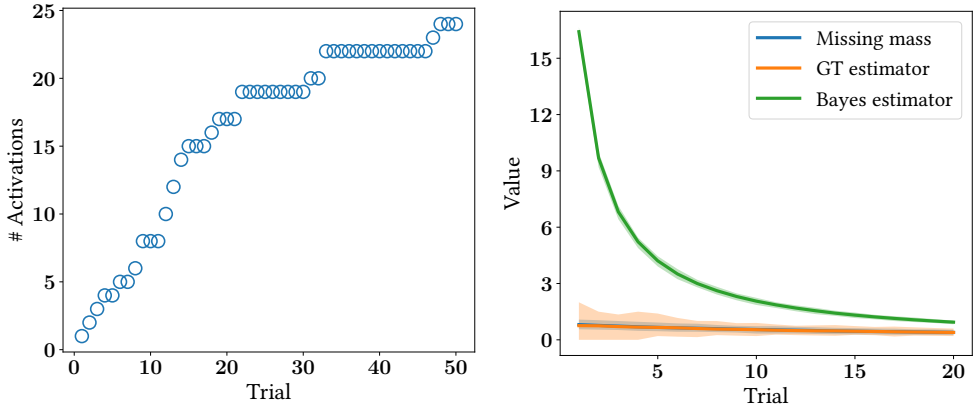


Figure 3.4: (left) Influence spread against number of rounds (right) Bayesian estimator against Good-Turing estimator.

of the sampled values are low, except a few relatively high ones. Using this sample as the activation probabilities of an hypothetical candidate node, we observe on Fig. 3.4 (left) the cumulative influence spread. The curve first shows a steep increase until approximately 20 rounds, where users with high probabilities of conversion have already been activated, while remaining ones are difficult to activate.

In Fig. 3.4 (right), we compare the Good-Turing estimator to a Bayesian estimator that maintains a posterior (through a Beta distribution) on the unknown activation probabilities, updating the posterior after each trial, similarly to [73]. In the Bayesian approach, the missing mass can be estimated by summing over the means of the posterior distributions corresponding to nodes that have not been activated so far. On Fig. 3.4 (right), the curves are averaged over 200 runs, and the shaded regions correspond to the 95% quantiles. Clearly, the Good-Turing estimator is much faster than its Bayesian counterpart, in estimating the actual missing mass. Varying the number of nodes – here equal to 50 –, shows that the time needed for the Bayesian estimator to provide a reliable estimate of the missing mass is proportional to the number of nodes, whereas it grows only sub-linearly for the Good-Turing estimator.

REMARK. While bearing similarities with the traditional missing mass concept, we highlight one fundamental difference of our problem w.r.t. the one studied in [17], which impacts both the algorithmic solution and the analysis. Since at each step, after selecting a candidate, *every* node connected to that candidate is sampled, the algorithm receives a larger feedback than in [17], whose feedback is in $[0, 1]$. However, on the contrary to [17], the hapaxes of a candidate $(U_k(u, t))_{u \in A_k}$ are independent. Interestingly, the quantity $\lambda_k := \sum_{u \in A_k} p(u)$, which corresponds to the expected number of basic nodes a candidate activates or re-activates in a cascade, will prove to be a crucial ingredient for our problem.

3.3.2.2 *Upper confidence bounds*

Following principles from the bandit literature, the GT-UCB algorithm relies on *optimism in the face of uncertainty*. At each step (trial) t, the algorithm selects the highest upper-confidence bound on the missing mass – denoted by $b_k(t)$ – and activates (plays) the corresponding candidate k. This algorithm achieves robustness against the stochastic nature of the cascades, by ensuring that candidates who "underperformed" with respect to their potential in previous trials may still be selected later on. Consequently, GT-UCB aims to maintain a degree of *exploration* of candidates, in addition to the *exploitation* of the best candidates as per the feedback gathered so far.

Algorithm 7: GT-UCB ($L = 1$)

Data: Set of candidates $[K]$, time budget N
1 **Initialization:** play each candidate $k \in [K]$ once, observe the spread $S_{k,1}$, set $n_k = 1$;
2 For each $k \in [K]$: update the reward $W = W \cup S_{k,1}$;
3 **for** $t = K + 1, \ldots, N$ **do**
4 Compute $b_k(t)$ for every candidate k;
5 Choose $k(t) = \arg\max_{k \in [K]} b_k(t)$;
6 Play candidate $k(t)$ and observe spread $S(t)$;
7 Update cumulative reward: $W = W \cup S(t)$;
8 Update statistics of candidate $k(t)$: $n_{k(t)}(t + 1) = n_{k(t)}(t) + 1$ and $S_{k,n_k(t)} = S(t)$.;
9 **end**
Result: W

Algorithm 7 presents the main components of GT-UCB for the case $L = 1$, that is, when a single candidate is chosen at each step.

The algorithm starts by activating each candidate $k \in [K]$ once, in order to initialize its Good-Turing estimator. The main loop of GT-UCB occurs at lines 3-9. Let $S(t)$ be the observed spread at trial t, and let $S_{k,s}$ be the result of the s-th diffusion initiated at candidate k. At every step $t > K$, we recompute for each candidate $k \in [K]$ its index $b_k(t)$, representing the upper confidence bound on the expected reward in the next trial. The computation of this index uses the previous samples $S_{k,1}, \ldots, S_{k,n_k(t)}$ and the number of times each candidate k has been activated up to trial t, $n_k(t)$. Based on the result of Theorem 3.5 below

– whose statement and proof are delayed to Section 3.4 –, the upper confidence bound is set as:

$$b_k(t) = \hat{R}_k(t) + \left(1 + \sqrt{2}\right) \sqrt{\frac{\hat{\lambda}_k(t) \log(4t)}{n_k(t)}} + \frac{\log(4t)}{3n_k(t)}, \tag{3.1}$$

where $\hat{R}_k(t)$ is the Good-Turing estimator and $\hat{\lambda}_k(t) := \sum_{s=1}^{n_k(t)} \frac{|S_{k,s}|}{n_k(t)}$ is an estimator for the expected spread from candidate k.

Then, in line 5, GT-UCB selects the candidate $k(t)$ with the largest index, and initiates a cascade from this node. The feedback $S(t)$ is observed and is used to update the cumulative reward set W. Note that $S(t)$ provides only the Ids of the nodes that were activated, with no information on *how* this diffusion happened in the hidden diffusion medium. Finally, statistics associated to the chosen candidate $k(t)$ are updated.

3.3.2.3 *Extensions for the case $L > 1$*

Algorithm 7 can be easily adapted to select $L > 1$ candidates at each round. Instead of choosing the candidate maximizing the Good-Turing UCB in line 5, we can select those having the L largest indices. Note that $k(t)$ then becomes a *set* of L candidates. A diffusion is initiated from the associated nodes and, at termination, all activations are observed. Similarly to [113], the algorithm requires feedback to include the candidate responsible for the activation of each node, in order to update the corresponding statistics accordingly.

If the underlying environment is a graph G whose topology is known, we propose a simple heuristic to assign activated nodes to selected candidates, by a breadth-first approach, as follows: for an activated node $u \in S(t)$, we assign this node to the selected candidate reachable from u by the shortest *live path* in G, where a live path corresponds to a sequence of activated nodes from $S(t)$.

3.4 ANALYSIS

In this section, we justify the upper confidence bound used by GT-UCB in Eq. 3.1 and provide a theoretical analysis of our algorithm.

3.4.1 *Confidence interval for the missing mass*

In the following, to simplify the analysis and to allow for a comparison with the oracle strategy, we assume that the candidates have *non intersecting support*. This means that each candidate's missing mass and corresponding Good-Turing estimator does not dependent on other candidates. Hence, for notational efficiency, we also omit the subscript denoting the candidate k. After selecting the candidates n times, the Good-Turing estimator is simply written $\hat{R}_n = \sum_{u \in A} \frac{U_n(u)}{n}$. We note that the non-interescting assumption is for theoretical purposes only – our experiments are done with candidates which can have intersecting supports.

The classic Good-Turing estimator is known to be slightly biased (see Theorem 1 in [90] for example). We show in Lemma 3.2 that our missing mass estimator adds an additional factor $\lambda = \sum_{u \in A} p(u)$ to this bias:

Lemma 3.2 *The bias of the missing mass estimator is*

$$\mathbb{E}[R_n] - \mathbb{E}[\hat{R}_n] \in \left[-\frac{\lambda}{n}, 0\right].$$

Proof

$$\begin{aligned}\mathbb{E}[R_n] - \mathbb{E}[\hat{R}_n] &= \sum_{u \in A} \left[p(u)(1-p(u))^n - \frac{n}{n} p(u)(1-p(u))^{n-1}\right] \\ &= -\frac{1}{n} \sum_{u \in A} p(u) \times np(u)(1-p(u))^{n-1} \\ &= -\frac{1}{n} \mathbb{E}\left[\sum_{u \in A} p(u) U_n(u)\right] \in \left[-\frac{\sum_{u \in A} p(u)}{n}, 0\right]\end{aligned}$$

□

Since λ is typically very small compared to $|A|$, in expectation, the estimation should be relatively accurate. However, in order to understand what may happen in the worst-case, we need to characterize the deviation of the Good-Turing estimator:

Theorem 3.5 *With probability at least* $1 - \delta$, *for* $\lambda = \sum_{u \in A} p(u)$ *and* $\beta_n := \left(1 + \sqrt{2}\right) \times \sqrt{\frac{\lambda \log(4/\delta)}{n}} + \frac{1}{3n} \log \frac{4}{\delta}$, *the following holds:*

$$-\beta_n - \frac{\lambda}{n} \leq R_n - \hat{R}_n \leq \beta_n.$$

Note that the additional term appearing in the left deviation corresponds to the bias of our estimator, which leads to a non-symmetrical interval.

Proof *We prove the confidence interval in three steps:*

1. *Good-Turing estimator deviation,*
2. *missing mass deviation,*
3. *combination of previous two inequalities for the final confidence interval.*

The samples of different nodes are assumed independent. *This is a simplification with respect to the classic missing mass concentration results, which rely on negative association [89, 90]. On the other hand, since we may activate several nodes at once, we need original concentration arguments to control the increments of both* $\hat{R}_n$ *and* R_n.

*(*1*)* ***Good-Turing deviations.*** *Let* $X_n(u) := \frac{U_n(u)}{n}$. *We have that*

$$\begin{aligned}v &:= \sum_{u \in A} \mathbb{E}[X_n(u)^2] = \frac{1}{n^2} \sum_{u \in A} \mathbb{E}[U_n(u)] \\ &= \frac{1}{n^2} \sum_{u \in A} np(u)(1-p(u))^{n-1} \leq \frac{\lambda}{n}.\end{aligned}$$

Moreover, clearly the following holds: $X_n(u) \leq \frac{1}{n}$.

Applying Bennett's inequality (Theorems 2.9, 2.10 in [14]) to the independent random variables $\{X_n(u)\}_{u\in A}$ yields

$$\mathbb{P}\left(\hat{R}_n - \mathbb{E}[\hat{R}_n] \geq \sqrt{\frac{2\lambda \log(1/\delta)}{n}} + \frac{\log(1/\delta)}{3n}\right) \leq \delta. \tag{3.2}$$

The same inequality can be derived for left deviations.

*(2) **Missing mass deviations.** Remember that $Z_n(u)$ denotes the indicator equal to 1 if u has never been activated up to trial n. We can rewrite the missing mass as $R_n = \sum_{u\in A} Z_n(u)p(u)$. Let $Y_n(u) = p(u)(Z_n(u) - \mathbb{E}[Z_n(u)])$ and $q(u) = \mathbb{P}(Z_n(u) = 1) = (1-p(u))^n$. For some $t > 0$, we have next that*

$$\begin{aligned}
\mathbb{P}(R_n - \mathbb{E}[R_n] \geq \epsilon) &\leq e^{-t\epsilon} \prod_{u\in A} \mathbb{E}\left[e^{tY_n(u)}\right] \\
&= e^{t\epsilon} \prod_{u\in A} \left(q(u)e^{tp(u)(1-q(u))} + (1-q(u))e^{-tp(u)q(u)}\right) \\
&\leq e^{-t\epsilon} \prod_{u\in A} \exp(p(u)t^2/(4n)) \\
&= \exp\left(-t\epsilon + t^2/(4n)\lambda\right).
\end{aligned}$$

The first inequality is well-known in exponential concentration bounds and relies on Markov's inequality. The second inequality follows from [11] (Lemma 3.5)

Then, choosing $t = \frac{2n\epsilon}{\lambda}$, we obtain

$$\mathbb{P}\left(R_n - \mathbb{E}[R_n] \geq \sqrt{\frac{\lambda \log(1/\delta)}{n}}\right) \leq \delta. \tag{3.3}$$

We can proceed similarly to obtain the left deviation.

***(3) Putting it all together.** We combine Lemma 3.2 with Eq. (3.2), (3.3), to obtain the final result. Note that δ is replaced by $\frac{\delta}{4}$ to ensure that both the left and right bounds for the Good-Turing estimator and the missing mass are verified.* □

3.4.2 *Theoretical Guarantees*

We now provide an analysis of the *waiting time* (defined below) of GT-UCB, by comparing it to the waiting time of an oracle policy, following ideas from [17]. Let $R_k(t)$ be the missing mass of candidate k at trial number t. This differs from $R_{k,n}$, which is the missing mass of candidate k once *it* has been played n times.

Definition 3.7 (Waiting time). *Let $\lambda_k = \sum_{u\in A_k} p(u)$ denote the expected number of activations obtained by the first call to candidate k. For $\alpha \in (0,1)$, the* waiting time $T_{UCB}(\alpha)$ *of GT-UCB represents the round at which the missing mass of* each *candidate k is smaller than $\alpha\lambda_k$. Formally,*

$$T_{UCB}(\alpha) := \min\{t : \forall k \in [K], R_k(t) \leq \alpha\lambda_k\}.$$

The above definition can be applied to any strategy for candidate selection and, in particular, to an oracle one that knows beforehand the α value that is targeted, the sampled

spreads $(S_{k,s})_{k\in[K],s\geq 1}$, and the individual activation probabilities $p_k(u), u \in A_k$. A policy having access to all these aspects will perform the fewest possible activations on each candidate. We denote by $T^*(\alpha)$ the waiting time of the oracle policy. We are now ready to state the main theoretical property of the GT-UCB algorithm.

Theorem 3.6 (Waiting time). *Let* $\lambda^{min} := \min_{k\in[K]} \lambda_k$ *and let* $\lambda^{max} := \max_{k\in[K]} \lambda_k$. *Assuming that* $\lambda^{min} \geq 13$, *for any* $\alpha \in \left[\frac{13}{\lambda^{min}}, 1\right]$, *if we define* $\tau^* := T^*\left(\alpha - \frac{13}{\lambda^{min}}\right)$, *with probability at least* $1 - \frac{2K}{\lambda^{max}}$ *the following holds:*

$$T_{UCB}(\alpha) \leq \tau^* + K\lambda^{max} \log(4\tau^* + 11K\lambda^{max}) + 2K.$$

The proof of this result is given in Section 3.A. Unsurprisingly, Theorem 3.6 says that GT-UCB must perform slightly more activations of the candidates than the oracle policy. With high probability – assuming that the best candidate has an initial missing mass that is much larger than the number of candidates – the waiting time of GT-UCB is comparable to $T^*(\alpha')$, up to factor that is only logarithmic in the waiting time of the oracle strategy. α' is smaller than α – hence $T^*(\alpha')$ is larger than $T^*(\alpha)$– by an offset that is inversely proportional to the initial missing mass of the worst candidate. This essentially says that, if we deal with large graphs, and if the candidates trigger reasonably large spreads, our algorithm is competitive with the oracle.

3.5 EXPERIMENTS

We conducted experiments on two types of datasets:

1. two graphs, widely-used in the IM literature, and
2. a crawled dataset from Twitter, consisting of tweets occurring during August 2012.

All methods are implemented in C++[3] and simulations are done on an Ubuntu 16.04 machine with an Intel Xeon 2.4GHz CPU 20 cores and 98GB of RAM.

3.5.1 *Extracting candidates from graphs*

GT-UCB does not make any assumptions about the topology of the nodes influenced by the candidates. Indeed, in many settings it may be more natural to assume that the set of candidates is given and that the activations at each trial can be observed, while the topology of the underlying graph G remain unknown. In other settings, we may start from an existing social network G, in which case we need to extract a set of K representative candidates from it. Ideally, we should choose candidates that have little intersection in their "scopes of influence" to avoid useless seed selections. While this may be interpreted and performed differently, from one application to another, we discuss next some of the most natural heuristics for selecting candidates which we use in our experiments.

MaxDegree. This method selects the K nodes with the highest out-degrees in G. Note that by this criterion we may select candidates with overlapping influence scopes.

Greedy MaxCover. This strategy follows the well-known greedy approximation algorithm for selecting a cover of the graph G. Specifically, the algorithm executes the following steps K times:

3 The code is available at https://github.com/plagree/oim

1. Select the node with highest out-degree
2. Remove all out-neighbors of the selected node

To limit intersections among candidate scopes even more, nodes reachable by more than 1 hops may be removed at step (2).

DivRank [91]. DIVRANK is a PAGERANK-like method relying on reinforced random walks, with the goal of producing diverse high-ranking nodes, while maintaining the rich-gets-richer paradigm. We adapted the original DIVRANK procedure by inverting the edge directions. In doing so, we get influential nodes instead of prestigious ones. By selecting the K highest scoring nodes as candidates, the diversity is naturally induced by the reinforcement of random walks. This ensures that the candidates are fairly scattered in the graph and should have limited impact on each other.

IM approximated algorithms. The fourth method we tested in our experiments assigns uniformly at random a propagation probability to each edge of G, assuming the IC model. Then, a state-of-the-art IM algorithm – PMC in our experiments – is executed on G to get the set of K candidates having the highest potential spread.

3.5.2 *Graph datasets*

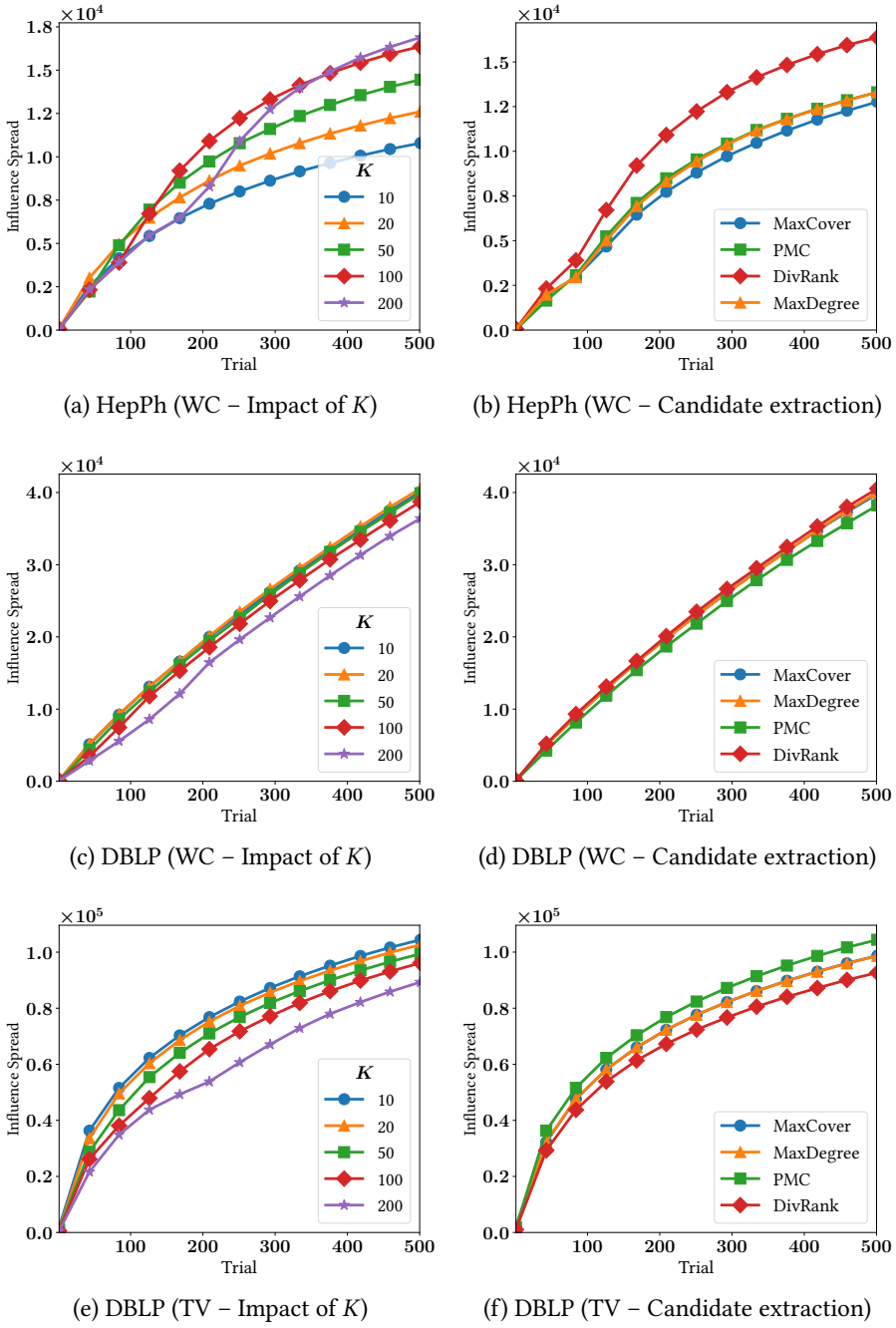
Similarly to [73], we tested our algorithm on HepPh and DBLP, two publicly available collaboration networks, where undirected edges are drawn between authors which have collaborated on at least one paper. HepPh is a citation graph, where a directed edge is established when an author cited at least one paper of another author. The datasets are summarized in Table 3.1. We emphasize that we kept the datasets relatively small to allow for comparison with computation-heavy baselines, even though GT-UCB easily scales to large data, as will be illustrated in Section 3.5.3.

Table 3.1: Summary of the datasets.

Dataset	HepPh	DBLP	Twitter
# of nodes	$34.5K$	$317K$	$11.6M$
# of edges	$422K$	$2.1M$	$38.4M$

Diffusion models. In the work closest to ours, Lei et al. [73] compared their solution on the Weighted Cascade instance of IC, where the influence probabilities on incoming edges sum up to 1. More precisely, every edge (u, v) has weight $1/d_v$ where d_v is the in-degree of node v. In this experimental study, and to illustrate that our approach is diffusion-independent, we added two other diffusion scenarios to the set of experiments. First, we included the tri-valency model (TV), which associates randomly a probability from $\{0.1, 0.01, 0.001\}$ to every edge and follows the IC propagation model. We also conducted experiments under the Linear Threshold (LT) model, where the edge probabilities are set like in the WC case and where thresholds on nodes are sampled uniformly from $[0, 1]$.

Baselines. We compare GT-UCB to several baselines. RANDOM chooses a random candidate at each round. MAXDEGREE selects the node with the largest degree at each step i, where the degree does not include previously activated nodes. Finally, EG corresponds to



(a) HepPh (WC – Impact of K)

(b) HepPh (WC – Candidate extraction)

(c) DBLP (WC – Impact of K)

(d) DBLP (WC – Candidate extraction)

(e) DBLP (TV – Impact of K)

(f) DBLP (TV – Candidate extraction)

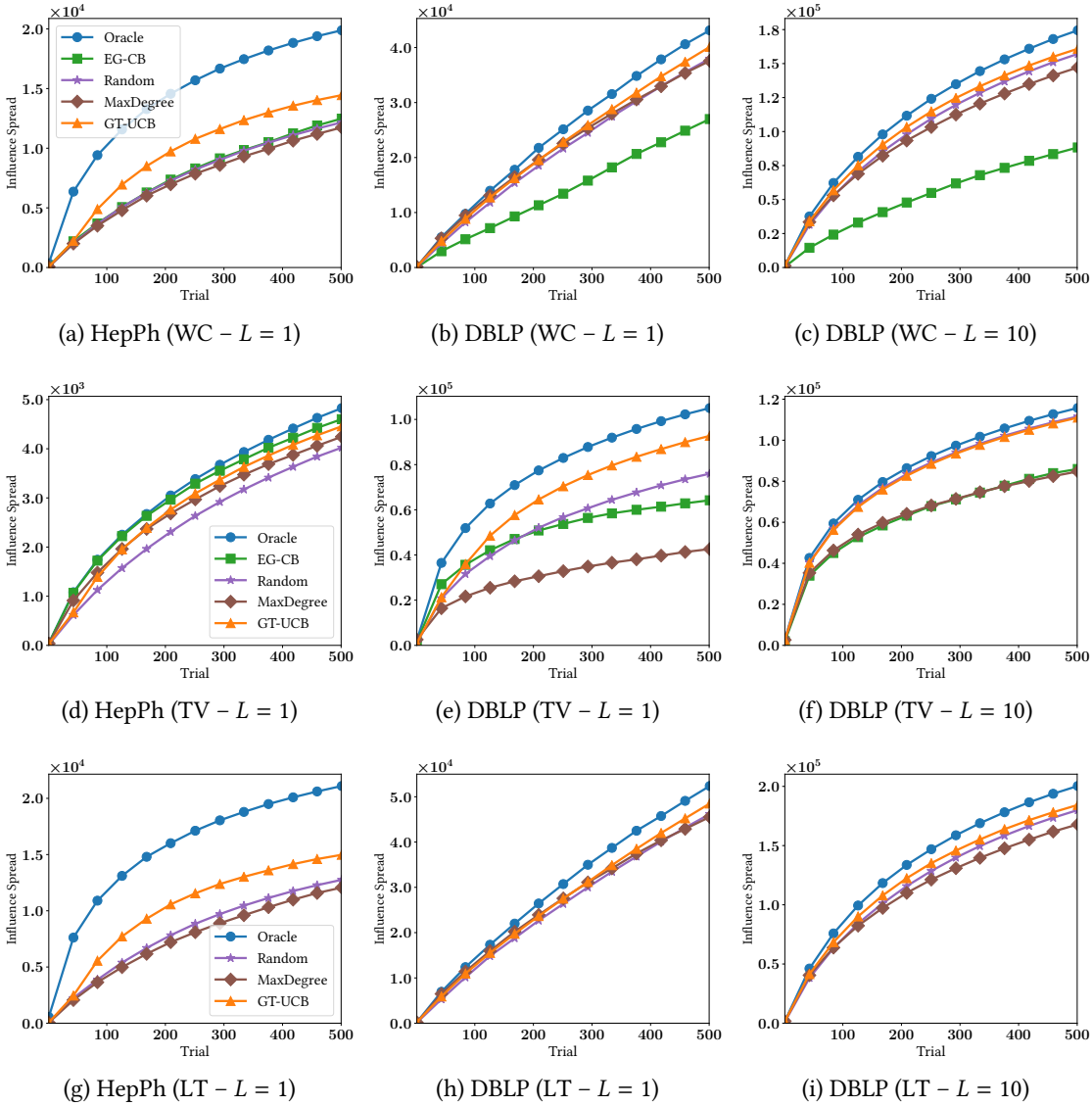
Figure 3.5: Impact of K and the candidate extraction criterion on influence spread.

the confidence-bound explore-exploit method with exponentiated gradient update from [73]; it is the state-of-the-art method for the OIMP problem (code provided by the authors). We use this last baseline on WC and TV weighted graphs and tune parameters in accordance to the results of their experiments: Maximum Likelihood Estimation is adopted for graph update and edge priors are set to Beta(1, 20). Note that EG learns parameters for the IC model, and hence is not applicable for LT. These baselines are compared to an ORACLE that knows beforehand the diffusion model together with probabilities. At each round, it runs an IM approximated algorithm – PMC for IC propagation, SSA for LT. Note that pre-

viously activated nodes are not counted when estimating the value of a node with PMC or SSA, thus, making ORACLE an adaptive strategy.

All experiments are done by fixing the trial horizon $N = 500$, a setting that is in line with many real-world marketing campaigns, which are fairly short and do not aim to reach the entire population.

GT-UCB parameters. We first analyze the effects of the different possible settings for GT-UCB. We show in Fig. 3.5b and 3.5d the impact of the candidate extraction criterion on HepPh and DBLP under the WC model. On the HepPh network, DIVRANK clearly leads to larger influence spreads. On DBLP under WC model, however, the extraction method has little impact on resulting spreads. In Fig. 3.5f, DIVRANK is the extraction method which performs the worst. In summary, the spread is slightly affected by the extraction criterion, and different datasets lead to different optimal extraction routines. We note that GT-UCB performs consistently as long as the method leads to candidates that are well spread over



(a) HepPh (WC – $L = 1$) (b) DBLP (WC – $L = 1$) (c) DBLP (WC – $L = 10$)

(d) HepPh (TV – $L = 1$) (e) DBLP (TV – $L = 1$) (f) DBLP (TV – $L = 10$)

(g) HepPh (LT – $L = 1$) (h) DBLP (LT – $L = 1$) (i) DBLP (LT – $L = 10$)

Figure 3.6: Growth of spreads against the number of rounds.

the graph. In the following, for each graph, we used DivRank as candidate extraction criterion as it performs the best in most configurations.

In Fig. 3.5a, 3.5c and 3.5e, we measure the impact of the number of candidates K on the influence spread. We can observe that, on DBLP, a small number of candidates is sufficient to yield high-quality results. If too many candidates (relative to the budget) are selected (e.g. $K = 200$), the initialization step required by GT-UCB is too long relative to the full budget, and hence GT-UCB does not reach its optimal spread – some candidates still have a large missing mass at the end. On the other hand, a larger amount of candidates leads to greater influence spreads on HepPh: this network is relatively small ($34.5K$ nodes), and thus half of the nodes are already activated after 400 trials. By having more candidates, we are able to access parts of the network that would not be accessible otherwise.

GT-UCB vs. baselines. In Fig. 3.6, we show the growth of the spread for GT-UCB and baselines. For each experiment, GT-UCB uses $K = 50$ if $L = 1$ and $K = 100$ if $L = 10$. First, we can see that MaxDegree is a strong baseline in many cases, especially for WC and LT. GT-UCB results in good quality spreads across every combination of network and diffusion model. Interestingly, on the smaller graph HepPh, we observe an increase in the slope of spread after initialization, particularly visible at $t = 50$ with WC and LT. This corresponds to the step when GT-UCB starts to select candidates maximizing $b_k(t)$ in the main loop. It shows that our strategy adapts well to the previous activations, and chooses good candidates at each iteration. Interestingly, Random performs surprisingly well in many cases, especially under TV weight assignment. However, when certain candidates are significantly better than others, it cannot adapt to select the best candidate unlike GT-UCB. EG performs well on HepPh, especially under TV weight assignment. However, it fails to provide competitive cumulative spreads on DBLP. We believe that EG tries to estimate too many parameters for a horizon $T = 500$. After reaching this time step, less than 10% of all nodes for WC, and 20% for TV, are activated. This implies that we have small confidence regarding many edge probability estimations, as most nodes are located in parts of the graph that have never been explored.

We evaluate the execution time of the different algorithms in Fig. 3.7. As expected, GT-UCB largely outperforms EG (and Oracle). The two baselines require the execution of an approximated IM algorithm at each round. In line with [5], we observed that SSA has prohibitive computational cost when incoming edge weights do not sum up to 1, which is the case with both WC and TV. Thus, both Oracle and EG run PMC on all our experiments with IC propagation. GT-UCB is several orders of magnitude faster: it concentrates most its running time on extracting candidates, while statistic updates and UCB computations are negligible.

3.5.3 *Experiments on Twitter*

We conclude the experimental section with an evaluation of GT-UCB on the Twitter data, introduced as a motivating example in Section 3.3.2. The interest of this experiment is to observe actual spreads, instead of simulated ones, and data which does not have an explicit influence graph attached.

From the retweeting logs, for each *active* user u – a user who posted more than 10 tweets – we select users having retweeted at least one of u's tweets. By doing so, we obtain the set of potentially influenceable users associated to active users. We then apply the greedy algorithm to select the users maximizing the corresponding set cover. These are the can-

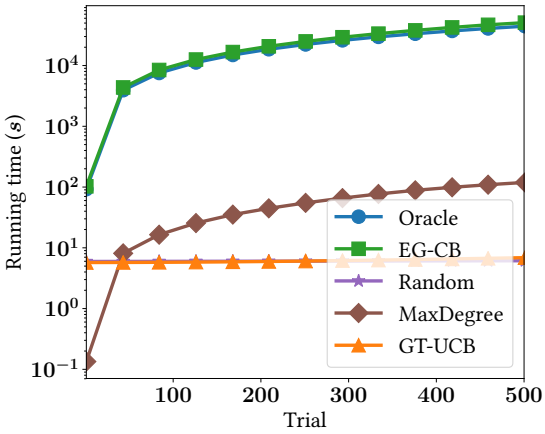


Figure 3.7: DBLP (WC) – Execution time.

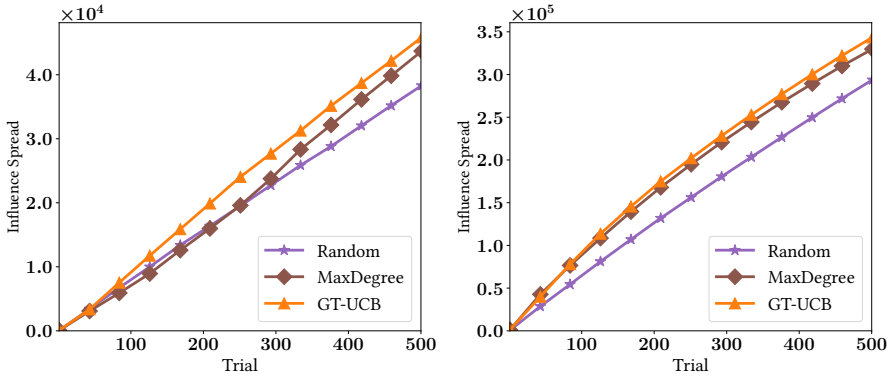


Figure 3.8: Twitter spread against rounds: (left) $L = 1$ (right) $L = 10$.

didates of GT-UCB and RANDOM. MAXDEGREE is given the entire reconstructed network (described in Table 3.1), that is, the network connecting active users to re-tweeters.

To test realistic spreads, at each step, once an candidate is selected by GT-UCB, a random cascade initiated by that candidate is chosen from the logs and we record its spread. This provides realistic, model-free spread samples to the compared algorithms. Since our data only contains successful activations (re-tweets) and not the failed ones, we could not test against EG, which needs both kinds of feedback.

In Fig. 3.8, we show the growth of the diffusion spread of GT-UCB against MAXDEGREE and RANDOM. Again, GT-UCB uses $K = 50$ if $L = 1$ and $K = 100$ if $L = 10$. We can see that GT-UCB outperforms the baselines, especially when a single node is selected at each round. We can observe that MAXDEGREE performs surprisingly well in both experiments. We emphasize that it relies on the knowledge of the entire network reconstructed from retweeting logs, whereas GT-UCB is only given a set of (few) fixed candidates.

3.6 EXPLORING FURTHER ONLINE IM MODELS

We now extend the model considered previously in two directions. First, we adapt our methodology with candidates to the semi-bandit IM setting of [23, 114, 121]. In the second

part of this section, we relax the influence stationarity assumption on the candidates in the OIMP framework.

3.6.1 *Semi-bandit online influence maximization via candidates*

In this chapter, we introduced a novel method to maximize the expected influence relying exclusively on a subset of the population, namely, the candidates. The rationale is that, arguably, in real life, marketing firms can only ask a few number of people to initiate spreads – e.g., influential people with whom they are contractually bound by a sponsorship. We now extend this idea to the semi-bandit IM setting of [23, 114, 121].

Problem 3.5 (Semi-bandit IM via candidates). *Given a set of candidates* $[K]$*, a* budget *of* N *trials, and a number* $1 \leq L \leq K$ *of candidates to be activated at each trial, the objective of the semi-bandit IM via candidates is to solve the following optimization problem:*

$$\underset{I_n \subseteq [K], |I_n|=L, \forall 1 \leqslant n \leqslant N}{\arg\max} \mathbb{E}\left[\sum_{n=1}^{N} |S(I_n)|\right].$$

Given the K candidates, the problem boils down to the following: *How should we select a candidate at each step?* As previously said in Section 3.2.4, a key difference w.r.t. the OIMP setting is that the optimal set of seeds – here, chosen among candidates – is *constant*, and thus, does not depend on previous actions. This is a great simplification w.r.t. OIMP and reduces the problem to estimating the value of each potential seed. We now describe our approach to estimate and build a confidence interval on the expected spread initiated by each candidate independently. Note that we make the non-intersecting support assumption, similarly to what we did in the OIMP missing mass bound derivation.

CONFIDENCE INTERVAL. Let n be the number of times a given candidate has be selected and $S_1, \ldots, S_n$ be the corresponding observed spreads. We want to estimate $\lambda := \sum_{u \in A} p(u)$ where $p(u)$ is the probability that the candidate activates node u. Estimating λ by $\hat{\lambda}_n := \frac{1}{n}\sum_{i=1}^{n} |S_i|$, we obtain the following confidence interval.

Proposition 3.1 *Let* $\epsilon > 0$*. For* $\lambda = \sum_{u \in A} p(u)$*, the following holds:*

$$\mathbb{P}\left(\hat{\lambda}_n - \lambda \geq \epsilon\right) \leq \exp\left(-n d^{Poi}(\lambda + \epsilon, \lambda)\right),$$

where $d^{Poi}(\lambda, \lambda_0) := \lambda_0 - \lambda + \lambda \log \frac{\lambda}{\lambda_0}$ *is the Kullback-Leibler divergence of a Poisson distribution.*

Proof *Let* $\epsilon > 0$ *and* $x = \lambda + \epsilon$*. For some* $t > 0$*, we have*

$$\mathbb{P}\left(\hat{\lambda}_n - \lambda \geq \epsilon\right) = \mathbb{P}\left(\frac{1}{n}\sum_{i=0}^{n} S_i - \lambda \geq \epsilon\right) \leq \exp\left(-n(tx - \log \mathbb{E}[e^{tS_1}])\right).$$

Assuming the independence between node activations, we can upper bound $\varphi(t) = \log \mathbb{E}[e^{tS_1}]$ *as follows:*

$$\varphi(t) = \sum_{u \in A} \log(1 - p(u) + p(u)e^t) \leq \sum_{u \in A} p(u)(e^t - 1) = \varphi^{Poi}(t).$$

Note that $\varphi^{Poi}(t)$ is the cumulant-generating function of a Poisson distribution of parameter λ. We obtain

$$\mathbb{P}\left(\hat{\lambda}_n - \lambda \geq \epsilon\right) \leq \exp\left(-n(tx - \varphi^{Poi}(t))\right).$$

We obtain the desired result by finding the supremum of $t \mapsto tx - \varphi^{Poi}(t)$. □

ALGORITHM. Using Proposition 3.1, we can build an index in the same spirit as the KL-UCB index from Garivier and Cappé [40]. This leads to Algorithm 8.

Algorithm 8: CANDIDATE-KLUCB ($L = 1$)

Data: Set of candidates $[K]$, time budget N
Initialization: play each candidate $k \in [K]$ once, observe the spread $S_{k,1}$, set $n_k = 1$;
for $t = K + 1, \ldots, N$ **do**
  Compute $b_k^{\mathrm{Poi}}(t)$ for every candidate k;
  Choose $k(t) = \arg\max_{k \in [K]} b_k^{\mathrm{Poi}}(t)$;
  Play candidate $k(t)$ and observe spread $S(t)$;
  Update statistics of candidate $k(t)$: $n_{k(t)}(t+1) = n_{k(t)}(t) + 1$ and $S_{k,n_k(t)} = S(t)$;
end

At each round, the learning agent computes the upper-confidence bound for each arm k as

$$b_k^{\mathrm{Poi}}(t) := \sup_{q \geq \hat{\lambda}(t)} \left\{ q \,\middle|\, n_k(t) d^{\mathrm{Poi}}\left(\hat{\lambda}(t), q\right) \leq \log(t) \right\}.$$

Algorithm 8 can be easily adapted to select $L > 1$ candidates at each round. Instead of choosing the candidate maximizing the KL-UCB index, we can select those having the L largest indices. Note that $k(t)$ then becomes a *set* of L candidates. A diffusion is initiated from the associated nodes and, at termination, all activations are observed.

NUMERICAL EXPERIMENTS. We conducted experiments on a publicly available dataset extracted from Facebook[4]. Note that this graph is relatively small – it contains 4,039 nodes for 88,234 edges – since we want to compare CANDIDATE-KLUCB to DILINUCB [113] which requires, for each selected seed node, the computation of $|V|$ matrix-vector products of dimension d, where d is the number of dimensions selected for the graph Laplacian embedding. In addition, DILINUCB needs to select each node from the graph once in the initialization phase, and thus, requires a much larger horizon to observe times when it can finally exploit the knowledge gathered during past exploration steps.

We tested DILINUCB with different settings of d, and found that $d = 50$ provided the best results, i.e., the best spread within the 10,000 steps horizon. Thus, we only report experiments with $d = 50$. Finally, we select 75 candidates for CANDIDATE-KLUCB using the DivRank criterion.

In Fig. 3.9, we show the results of the simulations where CANDIDATE-KLUCB is compared to DILINUCB on a horizon $T = 10,000$. On the left plot, we show the evolution of the regret against the number of trials. Unsuprisingly, DILINUCB needs a long initialization phase before it can finally start exploiting accumulated feedback. On the other hand,

4 http://snap.stanford.edu/data/egonets-Facebook.html

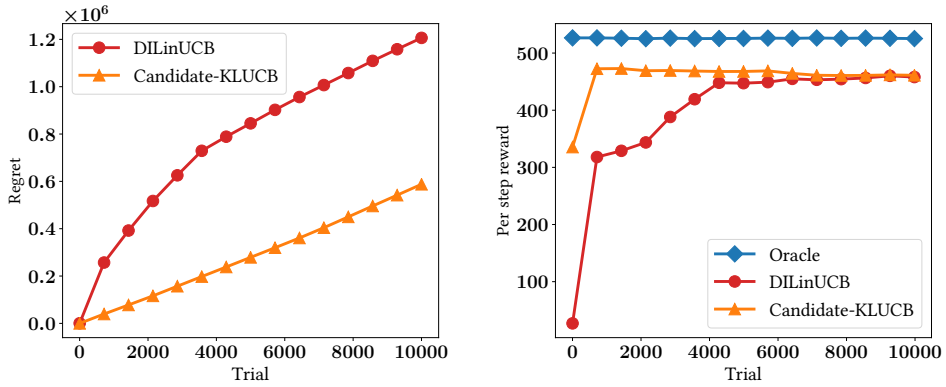


Figure 3.9: CANDIDATE-KLUCB vs DILINUCB on Facebook dataset, with horizon $T = 10,000$ and $L = 10$. (*left*) Regret against number of rounds (*right*) Per-step reward vs number of rounds.

CANDIDATE-KLUCB has approximatively the same performance all along the T steps. To verify this assumption, we display the evolution of the average per-step reward – in practice, we compute it on the last 50 steps – gathered by the algorithms in Fig. 3.9 (right). We can observe that CANDIDATE-KLUCB attains its asymptotic performance very early compared to DILINUCB.

SUMMARY. In this section, we provided a novel approach for the semi-bandit influence maximization. Our algorithm, CANDIDATE-KLUCB, has two advantages over its state-of-the-art competitor DILINUCB: (i) it has a very short initialization step, (ii) at each step, it selects the spread seed from a small subset of nodes which is much more realistic in many applications such as *influencer marketing* [15], in which a marketing firm only sponsors a few influential users.

3.6.2 *Online influence maximization via rotting candidates*

In Section 3.5.1, we studied the OIMP problem under the assumption that candidates have a *constant* tendency to activate their basic nodes. In certain situations, the candidates' ability to influence their basic nodes may diminish as they initiate new spreads. Intuitively, this means that candidates generate weariness if they persist in trying to convert people they are connected to, especially if the budget provided initially is given for a *single* campaign with a unique *semantics*. In the following, we assume that candidates can influence their basic nodes only decreasingly *with the number of times they initiated spreads*.

In terms of bandits, this work is in the same spirit as of Levine et al. [75] who study a setting – which they call *rotting bandits* – where each arm's value decays as a function of the number of times it has been selected. We also mention the work of Louëdec et al. [83] in which the authors propose to take into account the gradual obsolescence of items to be recommended while allowing new items to be added to the pool of candidates. Importantly, the item's value is modelled by a decreasing function of the number of steps since it was added to the pool of items, whereas in our work – and in that of [75]–, the value is a function of the number of times *the item has been selected*.

Our variant to the OIMP via candidates framework can be written as follows:

Problem 3.6 (OIMP via *rotting* candidates). *Given a set of candidates* $[K]$*, a* budget *of* N *trials, and a number* $1 \leq L \leq K$ *of candidates to be activated at each trial, the objective of the online influence maximization with persistence (OIMP) via* rotting *candidates is to solve the following optimization problem:*

$$\underset{I_n \subseteq [K], |I_n| = L, \forall 1 \leqslant n \leqslant N}{\arg\max} \mathbb{E}\left| \bigcup_{1 \leqslant n \leqslant N} S(I_n) \right|,$$

and such that, the probability that, at its s*-th selection, the candidate* $k \in [K]$ *activates its basic node* u *is:*

$$p_s(u) = \gamma(s)p(u),$$

where $\gamma : \mathbb{N} \to [0,1]$ *is a* known *non-increasing function and* $p(u) \in [0,1]$.

The traditional OIMP can be seen as a special instance of the online influence maximization with persistence via rotting candidates, where the non-increasing function γ – referred to as the weariness function in the following – is the identity. We will pursue the same strategy by estimating the missing mass of a given candidate by an adaptation of the Good-Turing estimator introduced for the OIMP problem. Note that the problem is more complex in the rotting setting because hapaxes must now incorporate the round of their activation.

As we did previously, to simplify the analysis, we assume that the candidates have *non intersecting support*. Let n be the number of times a given candidate has been selected and $S_1, \ldots, S_n$ the corresponding initiated spreads. We redefine the missing mass in the rotting candidate setting as $R_n := \sum_{u \in A} \mathbb{1}\{u \text{ never activated}\}\gamma(n)p(u)$ where $p(u)$ is the probability that the candidate activates node u, independently of the number of spreads initiated by the candidate. Again, the missing mass is equal to the expected number of additional conversions at time n given the nodes previously activated. The Good-Turing estimator adapted to the rotting setting is defined as follows:

$$\hat{R}_n = \sum_{u \in A} \frac{U_n^\gamma(u)}{n},$$

where $U_n^\gamma(u) := \sum_{i<n} \mathbb{1}\{X_0 = \ldots = X_{i-1} = X_{i+1} = \ldots = X_{n-1} = 0, X_i = 1\}\frac{\gamma(n)}{\gamma(i)}$. In short, if i is the round at which a hapax has been activated, we reweight it by the factor $\gamma(n)/\gamma(i)$ since we are interested in its contribution at the n-th round. We now justify formally this estimator by computing its bias.

ESTIMATOR BIAS. Lemma 3.3 shows that the estimator of the missing mass for the rotting candidates setting is hardly biased.

Lemma 3.3 *Denoting* $\lambda = \sum_{u \in A} p(u)$*, the bias of the missing mass estimator is*

$$\mathbb{E}[R_n] - \mathbb{E}[\hat{R}_n] \in \left[-\gamma(n)\frac{\lambda}{n}, 0\right].$$

Proof *Remember that* $U_n^\gamma(u) = \sum_{i<n} \mathbb{1}\{X_0 = \ldots = X_{i-1} = X_{i+1} = \ldots = X_{n-1} = 0, X_i = 1\}\frac{\gamma(n)}{\gamma(i)}$*. We have that*

$$\mathbb{E}[U_n^\gamma(u)] = \sum_{i<n} p_i(u) \prod_{j \neq i}(1 - p_j(u))\frac{\gamma(n)}{\gamma(i)} = p_n(u) \sum_{i<n} \prod_{j \neq i}(1 - p_j(u)).$$

We now can compute the bias of the estimator:

$$\begin{aligned}
\mathbb{E}[R_n] - \mathbb{E}[\hat{R}_n] &= \frac{1}{n}\sum_{u\in A} p_n(u)\left[\sum_{i<n}\prod_{j<n}(1-p_j(u)) - \sum_{i<n}\prod_{j\neq i}(1-p_j(u))\right] \\
&= \frac{1}{n}\sum_{u\in A} p_n(u)\sum_{i<n}\prod_{j\neq i}(1-p_j(u))[1-p_i(u)-1] \\
&= -\frac{1}{n}\sum_{u\in A} p_n(u)\sum_{i<n} p_i(u)\prod_{j\neq i}(1-p_j(u)) \\
&= -\frac{1}{n}\mathbb{E}\left[\sum_{u\in A} p_n(u)U_n(u)\right] \in \left[-\frac{\sum_{u\in A} p_n(u)}{n}, 0\right]
\end{aligned}$$

Note that the random variable $U_n(u)$ correspond to the hapax definition given in the OIMP problem, that is, $U_n(u) = \mathbb{1}\{u \text{ activated exactly once}\}$. □

Unsurprisingly, we obtain the same bias for the case where γ is the identity function. We omit the derivation of the confidence intervals in this thesis.

SUMMARY. We provided an extension to the OIMP problem that models users' weariness of repeatedly seeing the same piece of information We propose a new estimator which includes the diminishing influence of a candidate as the number of times it has been selected grows.

3.7 CONCLUSION

In this chapter, we proposed a diffusion-independent approach for online and adaptive IM, whose role is to maximize the number of activated nodes in an arbitrary environment, under the OIMP framework. We focus on scenarios in which influence campaigns consist of multiple consecutive trials conveying the same piece of information. Our method requires as only interfaces with the "real-world" the identification of potential seeds (the candidates) and the spread feedback (i.e., the set of activated nodes) at each trial.

Subsequent online iterations are very fast, making it possible to scale to very large graphs, where other approaches become infeasible. The efficiency of GT-UCB comes from the fact that it only relies on an estimate of a single quantity for each candidate seed – its potential or missing mass. This novel approach is shown to be very competitive on IM benchmark tasks.

Two extensions to this line of research were considered, one for adapting our OIMP approach via candidates to the semi-bandit influence maximization, one for incorporating the possible weariness generated by candidates if they persist in trying to convert people they are connected to.

APPENDIX 3.A ELEMENTS OF PROOF

3.A.1 *Useful lemmas*

Lemma 3.4 (Bennett's inequality – Theorem 2.9 and 2.10 [14]). *Let $X_1, \ldots, X_n$ be independent random variables with finite variance such that $X_i \leq b$ for some $b > 0$ for all*

$i \le n$. Let $S := \sum_{i=1}^n (X_i - \mathbb{E}[X_i])$ and $v := \sum_{i=1}^n \mathbb{E}[X_i^2]$. Writing $\varphi(u) = e^u - u - 1$, then for all $t > 0$,

$$\log \mathbb{E}\left[e^{tS}\right] \le \frac{v}{b^2}\varphi(bt) \le \frac{vt^2}{2(1-bt/3)}.$$

This implies that, $\mathbb{P}\left(S > \sqrt{2v\log 1/\delta} + \frac{b}{3}\log 1/\delta\right) \le \delta$.

Lemma 3.5 (Lemma 7 – [11]). *Let $n \ge 1$, $\lambda \ge 0$, $p \in [0,1]$ and $q = (1-p)^n$. Then,*

$$qe^{\lambda p(1-q)} + (1-q)e^{-\lambda pq} \le \exp(p\lambda^2/(4n)) \tag{3.4}$$
$$qe^{\lambda p(q-1)} + (1-q)e^{\lambda pq} \le \exp(p\lambda^2/(4n)) \tag{3.5}$$

3.A.2 *Analysis of the waiting time of GT-UCB*

Lemma 3.6 *For any $s \ge 3$,*

$$\mathbb{P}\left(\hat{R}_s \le \hat{R}_{s-1} - \frac{\lambda}{e(s-2)} - \sqrt{\frac{2\lambda}{s-1}\log(1/\delta)} - \frac{1}{3(s-1)}\log(1/\delta)\right) \le \delta.$$

Proof *Denote by $X_s(x) := \frac{U_{s-1}(x)}{s-1} - \frac{U_s(x)}{s} \le \frac{1}{s-1}$. We can rewrite $\hat{R}_{s-1} - \hat{R}_s = \sum_{x\in A} X_s(x)$ and can easily verify that*

$$v(x) := \mathbb{E}\left[X_s(x)^2\right] = p(x)(1-p(x))^{s-2}\left(\frac{1}{s-1} - \frac{1-p(x)}{s}\right) \le \frac{p(x)}{s-1}. \tag{3.6}$$

Let $t > 0$. By applying Lemma3.4, one obtains

$$\mathbb{P}\left(\hat{R}_{s-1} - \hat{R}_s \ge \mathbb{E}\left[\hat{R}_{s-1} - \hat{R}_s\right] + \sqrt{\frac{2\lambda}{s-1}\log(1/\delta)} + \frac{1}{3(s-1)}\log(1/\delta)\right) \le \delta.$$

We conclude remarking that $\mathbb{E}[X_s(x)] = p(x)^2(1-p(x))^{s-2} \le \frac{p(x)}{e(s-2)}$, that is, $\mathbb{E}[\hat{R}_{s-1} - \hat{R}_s] \le \frac{\lambda}{e(s-2)}$. □

Theorem 3.7 (Stopping time). *Denote $\lambda^{min} := \min_{k\in[K]} \lambda_k$ and $\lambda^{max} := \max_{k\in[K]} \lambda_k$. Assume that $\lambda^{min} \ge 13$. Then, for any $\alpha \in \left[\frac{13}{\lambda^{min}}, 1\right]$, if we define $\tau^* := T^*\left(\alpha - \frac{13}{\lambda^{min}}\right)$, with probability at least $1 - \frac{2K}{\lambda^{max}}$,*

$$T_{UCB}(\alpha) \le \tau^* + K\lambda^{max}\log(4\tau^* + 11K\lambda^{max}) + 2K.$$

Proof *Let us define the following confidence bounds:*

$$b^+_{k,s}(t) := (1+\sqrt{2})\sqrt{\frac{3\lambda_k\log(2t)}{s}} + \frac{\log(2t)}{s},$$
$$b^-_{k,s}(t) := (1+\sqrt{2})\sqrt{\frac{3\lambda_k\log(2t)}{s}} + \frac{\log(2t)}{s} + \frac{\lambda_k}{s}, \text{ and}$$
$$c^-_{k,t}(t) := \frac{\lambda}{e(s-2)} + \sqrt{\frac{6\lambda_k\log(t)}{s-1}} + \frac{\log(t)}{s-1}.$$

Let $S > 0$. Using these definitions, we introduce the following events:

$$\mathcal{F} := \left\{\forall k \in [K], \forall t > S, \forall s \leq t, \hat{R}_{k,s} - b^-_{k,s}(t) \leq R_{k,s} \leq \hat{R}_{k,s} + b^+_{k,s}(t)\right\},$$
$$\mathcal{G} := \left\{\forall k \in [K], \forall s \geq S, \hat{R}_{k,s} \geq \hat{R}_{k,s-1} - c^-_{k,s}(t)\right\},$$
$$\mathcal{E} := \mathcal{F} \cap \mathcal{G}.$$

Using Theorem 3.5, Lemma 3.6 and a union bound, one obtains $\mathbb{P}(\mathcal{E}) \geq 1 - \frac{2K}{S}$ (by setting $\delta \equiv \frac{1}{t^3}$). Indeed,

$$\mathbb{P}\left(\bar{\mathcal{E}}\right) \leq \mathbb{P}(\bar{\mathcal{F}}) + \mathbb{P}(\bar{\mathcal{G}}) \leq 2\sum_{k=1}^{K}\sum_{t>S}\sum_{s\leq t}\frac{1}{t^3} = 2K\sum_{t>S}\frac{1}{t^2} \leq \frac{2K}{S}.$$

In the following, we work on the event $\mathcal{E}$. Recall that we want to control $T_{UCB}(\alpha)$, the time at which every expert attains a missing mass smaller than α following Good-UCB strategy. We aim at comparing $T_{UCB}(\alpha)$ to $T^*(\alpha)$, the same quantity following the omniscient strategy. With that in mind, one can write:

$$T_{UCB}(\alpha) = \min\left\{t : \forall k \in [K], R_{k,N_k(t)} \leq \alpha\lambda_k\right\},$$
$$T^*(\alpha) = \sum_{k=1}^{K} T^*_k(\alpha), \text{ where } T^*_k(\alpha) = \min\left\{s : R_{k,s} \leq \alpha\lambda_k\right\}.$$

Following ideas from [17], we will control $T_{UCB}(\alpha)$ by comparing it to $U(\alpha)$ defined below, and which replaces the missing mass by an upper bound on the estimator of the missing mass (the Good-Turing estimator). Indeed, remind that we can control this on event $\mathcal{F}$.

$$U(\alpha) = \min\left\{t \geq 1 : \forall k \in [K], \hat{R}_{k,N_k(t)} + b^+_{k,N_k(t)}(t) \leq \alpha\lambda_k\right\}.$$

Let $S' \geq S$. On event $\mathcal{E}$, one has that $T_{UCB}(\alpha) \leq \max(S', U(\alpha))$. If $U(\alpha) \geq S'$, one has

$$\begin{aligned}
R_{k,N_k(U(\alpha))} &\geq \hat{R}_{k,N_k(U(\alpha))} - b^-_{k,N_k(U(\alpha))}(U(\alpha)) \quad \text{(we are on event } \mathcal{F} \text{ and } U(\alpha) > S' \geq S\text{)} \\
&\geq \hat{R}_{k,N_k(U(\alpha))-1} - b^-_{k,N_k(U(\alpha))}(U(\alpha)) - c^-_{k,N_k(U(\alpha))}(U(\alpha)) \\
&\qquad\qquad \text{(where are on event } \mathcal{G}\text{)} \\
&\geq \left(\alpha\lambda_k - b^+_{k,N_k(U(\alpha))-1}(U(\alpha))\right) - b^-_{k,N_k(U(\alpha))}(U(\alpha)) - c^-_{k,N_k(U(\alpha))}(U(\alpha))
\end{aligned}$$

The third inequality's justification is more evolved. Let t be the time such that $N_k(t) = N_k(U(\alpha)) - 1$ and $N_k(t+1) = N_k(U(\alpha))$. This implies that k is the chosen expert at time t, that is, the one maximizing the Good-UCB index. Moreover, since $t < U(\alpha)$, one knows that this index is greater than $\alpha\lambda_k$.

If $N_k(U(\alpha)) \geq S' + 2$, some basic maths calculations lead to

$$R_{k,N_k(U(\alpha))} \geq \alpha\lambda_k - 11\sqrt{\frac{\lambda_k \log(2U(\alpha))}{S'}} - \frac{3\log(2U(\alpha))}{S'} - \frac{3\lambda_k}{2S'}$$

We denote by $\lambda^{max} := \max_k \lambda_k$. If we take $S' = \lambda^{max}\log(2U(\alpha))$, we can rewrite previous inequality as

$$R_{k,N_k(U(\alpha))} \geq \alpha\lambda_k - 11 - \frac{3}{\lambda^{max}} - \frac{3}{2}$$

Thus, by definition of $T_k^*(\alpha)$*, and if* $\lambda^{max} > 6$*, one gets*

$$N_{k,U(\alpha)} \leq T_k^*\left(\alpha - \frac{13}{\lambda_k}\right) + S' + 2.$$

Finally, if we denote by $\lambda^{min} = \min_k \lambda_k$*, we obtain that*

$$U(\alpha) \leq K(S'+2) + T^*\left(\alpha - \frac{13}{\lambda^{min}}\right).$$

We now apply Lemma 3.7. We obtain that

$$\begin{aligned} U(\alpha) &\leq 2K + \tau^* + K\lambda^{max} \log\left(8K + 4\tau^* + 10K\lambda^{max}\right) \\ &\leq \tau^* + K\lambda^{max} \log\left(4\tau^* + 11K\lambda^{max}\right) + 2K. \end{aligned}$$

We conclude with $T_{UCB}(\alpha) \leq \max(S', U(\alpha))$. □

Lemma 3.7 (Lemma 3 from [17]). *Let* $a > 0$*,* $b \geq 0.4$*, and* $x \geq e$*, such that* $x \leq a + b\log x$*. Then one has*

$$x \leq a + b\log(2a + 4b\log(4b)).$$

Moreover, we add that if $b \geq 3$*, then* $x \leq a + b\log(2a + 5b)$.

4

ADAPTIVE MULTIPLE-ITEM RECOMMENDATION

In this chapter, we introduce a dynamic approach for adaptively learning to place items in multi-position displays (e.g., web pages offering advertising spaces). The proposed approach is particularly tailored for *cold start* situations in which the recommender agent (e.g. the film recommender system, the publicity agency, etc.) has no information on some of the items it recommends to users. For example, when a company releases new products, it may want to run a marketing campaign so as to give them visibility, but has no prior knowledge on people's response regarding them. As another example, think about film recommender systems: every time a new film is added to the system, its evaluation by users is unknown and we face a cold start situation. In order to answer these challenges, we propose a multi-armed bandit approach that alternates exploration and exploitation steps so as to maximize users' satisfaction regarding items they are recommended.

The present work proposes to exploit available information regarding the display position bias under the so-called *position-based* click model. Importantly, a major concern in this context is that the system receives *ambiguous* feedback. Indeed, much of the content may have been simply ignored by the user (e.g., the user did not scroll to the bottom of the page, and thus did not see the ad displayed there). We first discuss how this model differs from the cascade model and other variants of click models considered in several works on multiple-play bandits. We then provide a novel regret lower bound for this model as well as computationally efficient algorithms that display good empirical and theoretical performance.

The work has been presented at the NIPS conference in 2016 [71]. A preliminary version of this work was presented at the Workshop on Online Advertising Systems at ICML 2016 [116].

Contents

4.1 INTRODUCTION

During their browsing experience, users are constantly provided – without having asked for it – with clickable content spread over web pages. While users interact on a website, they send clicks to the system for a very limited selection of the clickable content. Hence, they let every unclicked item with an *equivocal* answer: the system does not know whether the content was really deemed irrelevant or simply ignored. In contrast, in traditional multi-armed bandit (MAB) models, the learner makes actions and observes at each round the reward corresponding to the chosen action. In the so-called multiple play semi-bandit setting, when users are presented with L items, they are assumed to provide feedback for each of those items.

Several variants of click models have been considered in the bandit literature. The necessity for the user to provide feedback for each item has been called into question in the context of the so-called *cascade model* [27, 28, 68] and its extensions such as the *Dependent Click Model* (DCM) [108]. Both models are particularly suited for search contexts, where the user is assumed to be looking for something relative to a query and examine sequentially items from a list. Consequently, the learner expects explicit feedback: in the cascade model each valid observation sequence must be either all zeros or terminated by a one, such that no ambiguity is left on the evaluation of the presented items. Conversely, multiple clicks are allowed in the DCM thus leaving some ambiguity on the last zeros of a sequence.

All previous click models assume that a portion of the recommendation list is explicitly examined by the user and hence that the learning algorithm eventually has access to rewards corresponding to the *unbiased* user's evaluation of each item. In contrast, we propose to analyze multiple-play bandits in the position-based model (PBM) [25]. In short, in the PBM, each position in the list is also endowed with a binary *examination variable* [28, 99] which is equal to one only when the user paid attention to the corresponding item. But this variable, that is independent of the user's evaluation of the item, is not observable. It allows to model situations where the user is not explicitly looking for specific content, as in typical *recommendation* scenarios.

Compared to the different variants of the cascade model, the PBM is challenging due to the *censoring* induced by the examination variables: the learning algorithm observes actual clicks but non-clicks are always ambiguous. Thus, combining observations made at different positions becomes a non-trivial statistical task. Some preliminary ideas on how to address this issue appear in the supplementary material of [67]. In this work, we provide a complete statistical study of stochastic multiple-play bandits with semi-bandit feedback in the PBM.

We begin this chapter with a short overview of the click models proposed in search and recommendation contexts and discuss existing multi-armed bandit studies under these feedback. We introduce the model and notations in Section 4.3.1 and provide the lower bound on the regret in Section 4.3.2. In Section 4.3.6, we present two optimistic algorithms, PBM-UCB and PBM-PIE, as well as an optimal theoretical analysis of the regret of the latter. In the last section dedicated to experiments, those policies are compared to several benchmarks on both synthetic and realistic data.

4.2 CLICK MODELS FOR SEARCH AND RECOMMENDATION

In this section, we present click models introduced in the last two decades aiming at modelling users' interaction with search and recommendation systems. A survey specifically dedicated to the presentation of these different click models and to their parameter estimation has been recently published by Chuklin et al. [25]. Here, we present a short summary of the models related to the position-based model which is the focus of our work, and refer the interested reader to this survey for a broader introduction to click models.

Propositions of click models have followed the rise of the Web, and, in particular, the development of search engines in the late 90's to access massive amounts of documents. Initially, experiments were performed to analize the behavior of users in real situations. For example, Joachims et al. [59] analyze the users' decision process through eyetracking tools and conclude that users' clicking decisions are biased by the position, the trust in the scoring function and the overall quality of the result set. The knowledge of these factors has dictated the choice of results page's design and several user models were introduced aiming at matching the observed user behavior with collected click logs.

In the following, alongside each click model, we describe related works – if any – on the corresponding bandit instance. The multi-armed bandit framework is particularly tailored to face *cold start* situations that are inherent to products newly introduced to an existing pool of documents / items. Indeed, the lack of knowledge regarding users' appreciation on these items requires that the search (or recommendation) system explore so as to gather information on items' value. Simultaneously, it needs to exploit cumulated observations in order to maximize the users' satisfaction. This problem is a typical instance of multi-armed bandits.

CASCADE MODEL. In the cascade model [28], the user scans the list from top to bottom and clicks on the first relevant item. The user always examines the first item and continues until finding an interesting item. More precisely, at each position, the user examines the item displayed. If the item is relevant, he/she clicks and the session stops. Otherwise, the user continues and examines the following position. Note that the user is considered satifisfied as soon as an item in the list was clicked and the attractiveness probabilities associated to items are the only parameters of the cascade model.

A key aspect of the cascade model is that it can only handle sessions with a single click. Furthermore, the positions of the items are not taken into account in the reward process because the user is assumed to continue scrolling the list as long as the items examined so far are not relevant, but the user satisfaction is only measured by the presence or absence of a click. In the bandit setting, this also implies that the optimal strategy in a learning context consists in showing the most relevant items at the end of the list in order

to maximize the amount of observed feedback [68] – which is counter-intuitive in both search and recommendation tasks.

To overcome these limitations, Combes et al. [27] introduce weights – these additional design parameters are to be defined by the learner – that are attributed to positions in the list, with a click on position $l \in \{1, \dots, L\}$ providing a reward w_l, where the sequence $(w_l)_l$ is decreasing to enforce the ranking behavior. However, no rule is given for setting the weights $(w_l)_l$ that control the order of importance of the positions. The authors propose an algorithm based on KL-UCB [40] and prove a lower bound on the regret as well as an asymptotically optimal upper bound. We will rely on their work to derive an asymptotically optimal analysis of PBM-PIE.

Importantly, all studies tackling the cascade model can easily estimate the unknown parameters as there is no ambiguity on every click / non-click. Even though the proposed algorithms are quite similar, some introduce small differences in order to simplify the analysis.

DEPENDENT CLICK MODEL. Another way to address the limitations of the cascade model is to consider the Dependent Click model (DCM) initially introduced by Guo et al. [51]. Here, continuation probabilities v_l are introduced for each position l: conditionally on the event that the user effectively scanned the list up to position l and clicked on this item, he/she can choose to continue with probability v_l. When trying to maximize the expected number of clicks, this framework naturally induces the necessity to rank the items in the optimal order. Indeed, if the least relevant items are displayed at the beginning of the list, the user scrolls down until he/she finds the first relevant item in, say, position l, and continues with probability v_l. Since continuation probabilities are decreasing values of l, the user thus observes less relevant items – in expectation – than if they were displayed at the beginning of the list.

Kveton et al. [108] study the DCM in the bandit setting and propose a variation of the KL-UCB algorithm tailored to the DCM. The algorithm estimates item attractivity using every click or non-click until the *last clicked* item. Indeed, these feedback are non-ambiguous whereas items displayed after the last click may correspond to items that were simply unobserved. Kveton et al. give an analysis of their algorithm that matches the lower bound of the DCM bandit setting up to logarithmic factors.

In summary, just as in the cascade model, the learner can estimates the desired quantities restraining the used feedback to have no ambiguity on examined items.

DYNAMIC BAYESIAN NETWORK MODEL. Chapelle and Zhang [20] introduce the Dynamic Bayesian Network model (DBN) as another extension to the cascade model. Instead of continuation probabilities, the authors propose to introduce for each item k *satisfaction* probabilities σ_k that add up to *attraction* probabilities θ_k: conditionally on the event that the user scanned the list up to position l, he/she clicks on the item k in position l with probability θ_k. He/she is satisfied by the clicked item with probability σ_k, and, *if unsatisfied* by the item, he/she continues scrolling to the next item with probability γ. Note that in the DBN model, there is a unique continuation parameter that is independent of the position. A special instance of the model with $\gamma = 1$ – referred to as the *Simplified* DBN model –, allows to estimate the parameters quite easily [20] without losing much prediction quality. If we further assume that all satisfaction probabilities are 1, the DBN reduces

to the cascade model. The study of the DBN model in the bandit setting remains an open question.

OTHER CLICK MODELS. Many other click models have been proposed in the last decade. We mention here the User Browsing model (UBM) by Dupret and Piwowarski [35] which can be seen as a mix of the PBM and the cascade model. The DCM has also been extended with the Click Chain model (CCM) by Guo et al. [51] that allows the user to stop scrolling the list without having clicked any item nor reached the end.

4.3 THE POSITION-BASED MODEL

In this section, we introduce notations that will be useful to describe formally the position-based model in the bandit setting. Then, we provide an analysis of the PBM problem with a lower bound of the regret which is shown to be tight with an upper bound analysis of algorithm PBM-PIE.

4.3.1 *Model and notations*

We consider the binary stochastic bandit model with K Bernoulli-distributed arms. The model parameters are the arm expectations $\theta = (\theta_1, \theta_2, \dots, \theta_K)$, which lie in $\Theta = (0,1)^K$. We will denote by $\mathcal{B}(\theta)$ the Bernoulli distribution with parameter θ and by $d(p,q) := p\log(p/q) + (1-p)\log((1-p)/(1-q))$ the Kullback-Leibler divergence from $\mathcal{B}(p)$ to $\mathcal{B}(q)$. At each round t, the learner selects a list of L arms – referred to as an *action* – chosen among the K arms which are indexed by $k \in \{1, \dots, K\}$. The set of actions is denoted by $\mathcal{A}$ and thus contains $K!/(K-L)!$ ordered lists; the action selected at time t will be denoted $A(t) = (A_1(t), \dots, A_L(t))$.

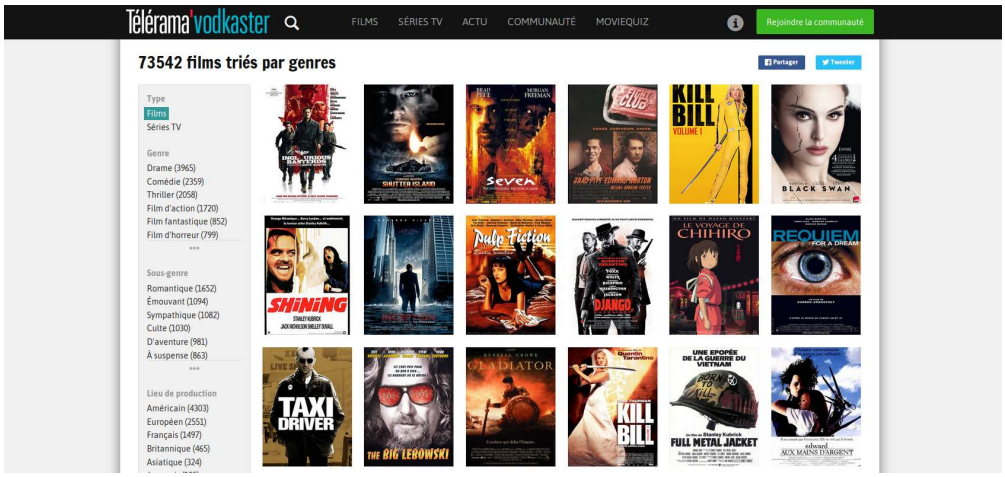


Figure 4.1: List of 18 recommended films displayed on 3 rows. Not surprisingly, independently of its content, the higher a film is displayed in a webpage, the higher its chances to be clicked. Interestingly, the films displayed in the middle columns receive more clicks on average than the films displayed on the left and right borders. Eyetracking experiments show that users tend to focus on the center of webpages. These observations support the need for a model like the PBM to describe recommendation contexts.

The PBM is characterized by examination parameters $(\kappa_l)_{1\le l\le L}$, where κ_l is the probability that the user effectively observes the item in position l [28]. At round t, the selection $A(t)$ is shown to the user and the learner observes the complete feedback – as in semi-bandit models – but the observation at position l, $Z_l(t)$, is *censored* being the product of two independent Bernoulli variables $Y_l(t)$ and $X_l(t)$, where $Y_l(t) \sim \mathcal{B}(\kappa_l)$ is non null when the user examined the item in position l – which is unknown to the learner – and $X_l(t) \sim \mathcal{B}(\theta_{A_l(t)})$ represents the actual user feedback to the item shown in position l. The learner receives a reward $r_{A(t)} = \sum_{l=1}^{L} Z_l(t)$, where $Z(t) = (X_1(t)Y_1(t), \ldots, X_L(t)Y_L(t))$ denotes the vector of censored observations at step t. In Fig. 4.1, we show a situation motivating the PBM in a recommendation scenario.

In the following, we will assume, without loss of generality, that $\theta_1 > \cdots > \theta_K$ and $\kappa_1 > \cdots > \kappa_L > 0$, in order to simplify the notations. The fact that the sequences $(\theta_l)_l$ and $(\kappa_l)_l$ are decreasing implies that the optimal list is $a^* = (1, \ldots, L)$. Denoting by $R(T) = \sum_{t=1}^{T} r_{a^*} - r_{A(t)}$ the regret incurred by the learner up to time T, one has

$$\mathbb{E}[R(T)] = \sum_{t=1}^{T}\sum_{l=1}^{L} \kappa_l(\theta_{a_l^*} - \mathbb{E}[\theta_{A_l(t)}]) = \sum_{a\in\mathcal{A}} (\mu^* - \mu_a)\,\mathbb{E}[N_a(T)] = \sum_{a\in\mathcal{A}} \Delta_a \mathbb{E}[N_a(T)], \quad (4.1)$$

where $\mu_a = \sum_{l=1}^{L} \kappa_l \theta_{a_l}$ is the expected reward of action a, $\mu^* = \mu_{a^*}$ is the best possible reward in average, $\Delta_a = \mu^* - \mu_a$ the expected gap to optimality, and, $N_a(T) = \sum_{t=1}^{T} \mathbb{1}\{A(t) = a\}$ is the number of times action a has been chosen up to time T.

In the following, we assume that the examination parameters $(\kappa_l)_{1\le l\le L}$ are known to the learner. These can be estimated from historical data [25], using, for instance, the EM algorithm [31] (as will be done in Section 4.4). In most scenarios, it is realistic to assume that the content (e.g., ads in on-line advertising) is changing much more frequently than the layout (web page design for instance) making it possible to have a good knowledge of the click-through biases associated with the display positions.

The main statistical challenge associated with the PBM is that one needs to obtain estimates and confidence bounds for the components θ_k of θ from the available $\mathcal{B}(\kappa_l\theta_k)$-distributed draws corresponding to occurrences of arm k at various positions $l = 1, \ldots, L$ in the list. To this aim, we define the following statistics: $S_{k,l}(t) = \sum_{s=1}^{t-1} Z_l(s)\mathbb{1}\{A_l(s) = k\}$, $S_k(t) = \sum_{l=1}^{L} S_{k,l}(t)$, $N_{k,l}(t) = \sum_{s=1}^{t-1} \mathbb{1}\{A_l(s) = k\}$, $N_k(t) = \sum_{l=1}^{L} N_{k,l}(t)$. We further require bias-corrected versions of the counts $\tilde{N}_{k,l}(t) = \sum_{s=1}^{t-1} \kappa_l \mathbb{1}\{A_l(s) = k\}$ and $\tilde{N}_k(t) = \sum_{l=1}^{L} \tilde{N}_{k,l}(t)$.

Observation *A time t, and* conditionally on the past actions $A(1)$ up to $A(t-1)$, *the Fisher information for θ_k is given by $I(\theta_k) = \sum_{l=1}^{L} N_{k,l}(t)\kappa_l/(\theta_k(1-\kappa_l\theta_k))$.*

Proof *Conditionnally to the actions $A(1)$ up to $A(t-1)$, the log-likelihood of the observations $Z(1), \ldots, Z(t-1)$ may be written as*

$$\sum_{s=}^{t-1}\sum_{k=1}^{K}\sum_{l=1}^{L} \mathbb{1}\{A_l(t) = k\}\,[Z_l(t)\log(\kappa_l\theta_k) + (1 - Z_l(t))\log(1 - \kappa_l\theta_k)]$$

$$= \sum_{k=1}^{K}\sum_{l=1}^{L} S_{k,l}(t)\log(\kappa_l\theta_k) + (N_{k,l}(t) - S_{k,l}(t))\log(1 - \kappa_l\theta_k).$$

Differenciating twice with respect to θ_k and taking the expectation of $(S_{k,l}(t))_l$, conditionnal to $A(1), \ldots, A(t-1)$, yields the expression of $I(\theta_k)$ given above. □

However, we cannot estimate θ_k using the maximum likelihood estimator since it has no closed form expression. Interestingly though, the simple pooled linear estimator

$$\hat{\theta}_k(t) = \frac{S_k(t)}{\tilde{N}_k(t)}, \tag{4.2}$$

considered in the supplementary material to [67], is unbiased and has a (conditional) variance of $v(\theta_k) = (\sum_{l=1}^{L} N_{k,l}(t)\kappa_l\theta_k(1-\kappa_l\theta_k))/(\sum_{l=1}^{L} N_{k,l}(t)\kappa_l)^2$, which is close to optimal given the Cramér-Rao lower bound. Indeed, $v(\theta_k)I(\theta_k)$ is recognized as a ratio of a weighted arithmetic mean to the corresponding weighted harmonic mean, which is known to be larger than one, but is upper bounded by $1/(1-\theta_k)$, irrespectively of the values of the κ_l's. Hence, if, for instance, we can assume that all θ_k's are smaller than one half, the loss with respect to the best unbiased estimator is no more than a factor of two for the variance. Note that despite its simplicity, $\hat{\theta}_k(t)$ cannot be written as a simple sum of conditionally independent increments divided by the number of terms and will thus require specific concentration results.

It can be checked that when θ_k gets very close to one, $\hat{\theta}_k(t)$ is no longer close to optimal. This observation also has a Bayesian counterpart that will be discussed in Section 4.4. Nevertheless, it is always preferable to the "position-debiased" estimator $(\sum_{l=1}^{L} S_{k,l}(t)/\kappa_l)/N_{k,l}(t)$ which gets very unreliable as soon as one of the κ_l's gets very small.

4.3.2 *Lower bound on the regret*

In this section, we consider the fundamental asymptotic limits of learning performance for online algorithms under the PBM. These cannot be deduced from earlier general results, such as those of [26, 50], due to the censoring in the feedback associated to each action. We detail a simple and general proof scheme – using the results of [63] – that applies to the PBM, as well as to more general models.

Lower bounds on the regret rely on changes of measure: the question is how much can we mistake the true parameters of the problem for others, when observing successive arms? With this in mind, we will subscript all expectations and probabilities by the parameter value and indicate explicitly that the quantities $\mu_a, a^*, \mu^*, \Delta_a$, introduced in Section 4.3.1, also depend on the parameter. For ease of notation, we will still assume that θ is such that $a^*(\theta) = (1, \dots, L)$.

4.3.3 *Existing results for multiple-play bandit problems*

Lower bounds on the regret will be proved for *uniformly efficient* algorithms, in the sense of [72]:

Definition 4.1 *An algorithm is said to be* uniformly efficient *if for any bandit model parameterized by θ and for all $\alpha \in (0, 1]$, its expected regret after T rounds is such that $\mathbb{E}_\theta R(T) = o(T^\alpha)$.*

For the multiple-play MAB, [4] obtained the following bound

$$\liminf_{T\to\infty} \frac{\mathbb{E}_\theta R(T)}{\log(T)} \geq \sum_{k=L+1}^{K} \frac{\theta_L - \theta_k}{d(\theta_k, \theta_L)}. \tag{4.3}$$

For the "learning to rank" problem where rewards follow the weighted Cascade Model with decreasing weights $(w_l)_{l=1,\dots,L}$, [27] derived the following bound

$$\liminf_{T\to\infty} \frac{\mathbb{E}_\theta R(T)}{\log T} \geq w_L \sum_{k=L+1}^{K} \frac{\theta_L - \theta_k}{d(\theta_k, \theta_L)}.$$

Perhaps surprisingly, this lower bound does not show any additional term corresponding to the complexity of ranking the L optimal arms. Indeed, the errors are still asymptotically dominated by the need to discriminate irrelevant arms $(\theta_k)_{k>L}$ from the worst of the relevant arms, that is, θ_L.

4.3.4 *Lower bound step by step*

STEP 1: COMPUTING THE EXPECTED LOG-LIKELIHOOD RATIO. Denoting by $\mathcal{F}_{s-1}$ the σ-algebra generated by the past actions and observations, we define the log-likelihood ratio for the two values θ and λ of the parameters by

$$\ell(t) := \sum_{s=1}^{t} \log \frac{p(Z(s); \theta \mid \mathcal{F}_{s-1})}{p(Z(s); \lambda \mid \mathcal{F}_{s-1})}. \tag{4.4}$$

Lemma 4.1 *For each position l and each item k, define the local amount of information by*

$$I_l(\theta_k, \lambda_k) := \mathbb{E}_\theta \left[\log \frac{p(Z_l(t); \theta)}{p(Z_l(t); \lambda)} \middle| A_l(t) = k \right],$$

and its cumulated sum over the L positions by $I_a(\theta, \lambda) := \sum_{l=1}^{L} \sum_{k=1}^{K} \mathbb{1}\{a_l = k\} I_l(\theta_k, \lambda_k)$. The expected log-likelihood ratio is given by

$$\mathbb{E}_\theta[\ell(t)] = \sum_{a \in \mathcal{A}} I_a(\theta, \lambda) \mathbb{E}_\theta[N_a(t)]. \tag{4.5}$$

The next proposition is adapted from Theorem 17 in Appendix B of [63] and provides a lower bound on the expected log-likelihood ratio.

Proposition 4.1 *Let $B(\theta) := \{\lambda \in \Theta \,|\, \forall l \leq L, \theta_l = \lambda_l \text{ and } \mu^*(\theta) < \mu^*(\lambda)\}$ be the set of changes of measure that improve over θ without modifying the optimal arms. Assuming that the expectation of the log-likelihood ratio may be written as in (4.5), for any uniformly efficient algorithm one has*

$$\forall \lambda \in B(\theta), \quad \liminf_{T\to\infty} \frac{\sum_{a \in \mathcal{A}} I_a(\theta, \lambda) \mathbb{E}_\theta[N_a(T)]}{\log(T)} \geq 1.$$

STEP 2: VARIATIONAL FORM OF THE LOWER BOUND. We are now ready to obtain the lower bound in a form similar to that originally given by [50].

Theorem 4.1 *The expected regret of any uniformly efficient algorithm satisfies*

$$\liminf_{T\to\infty} \frac{\mathbb{E}_\theta R(T)}{\log T} \geq f(\theta), \quad \text{where } f(\theta) = \inf_{c \geq 0} \sum_{a \in \mathcal{A}} \Delta_a(\theta) c_a, \text{ s.t. } \inf_{\lambda \in B(\theta)} \sum_{a \in \mathcal{A}} I_a(\theta, \lambda) c_a \geq 1.$$

Theorem 4.1 is a straightforward consequence of Proposition 4.1, combined with the expression of the expected regret given in (4.1). The vector $c \in \mathbb{R}_+^{|\mathcal{A}|}$, that satisfies the inequality $\sum_{a \in \mathcal{A}} I_a(\theta, \lambda) c_a \geq 1$, represents the feasible values of $\mathbb{E}_\theta[N_a(T)]/\log(T)$.

STEP 3: RELAXING THE CONSTRAINTS. The bounds mentioned in Section 4.3.3 may be recovered from Theorem 4.1 by considering only the changes of measure that affect a single suboptimal arm.

Corollary 4.1

$$f(\theta) \geq \inf_{c \geq 0} \sum_{a \in \mathcal{A}} \Delta_a(\theta) c_a \,, \quad \text{s.t.} \sum_{a \in \mathcal{A}} \sum_{l=1}^{L} \mathbb{1}\{a_l = k\} I_l(\theta_k, \theta_L) c_a \geq 1 \,, \quad \forall k \in \{L+1, \dots, K\}.$$

Corollary 4.1 is obtained by restricting the constraint set $B(\theta)$ of Theorem 4.1 to $\cup_{k=L+1}^{K} B_k(\theta)$, where $B_k(\theta) := \left\{\lambda \in \Theta | \forall j \neq k, \theta_j = \lambda_j \text{ and } \mu^*(\theta) < \mu^*(\lambda)\right\}$.

4.3.5 *Lower bound for the PBM*

Theorem 4.2 *For the PBM, the following lower bound holds for any uniformly efficient algorithm:*

$$\liminf_{T \to \infty} \frac{\mathbb{E}_\theta R(T)}{\log T} \geq \sum_{k=L+1}^{K} \min_{l \in \{1, \dots, L\}} \frac{\Delta_{v_{k,l}}(\theta)}{d(\kappa_l \theta_k, \kappa_l \theta_L)}, \tag{4.6}$$

where $v_{k,l} := (1, \dots, l-1, k, l, \dots, L-1)$.

Proof *First, note that for the PBM one has* $I_l(\theta_k, \lambda_k) = d(\kappa_l \theta_k, \kappa_l \lambda_k)$. *To get the expression given in Theorem 4.2 from Corollary 4.1, we proceed as in [27] showing that the optimal coefficients* $(c_a)_{a \in \mathcal{A}}$ *can be non-zero only for the* $K - L$ *actions that put the suboptimal arm* k *in the position* l *that reaches the minimum of* $\Delta_{v_{k,l}}(\theta)/d(\kappa_l \theta_k, \kappa_l \theta_L)$. *Nevertheless, this position does not always coincide with L, the end of the displayed list, contrary to the case of [27] (the detailed proof is given in Section 4.A.1).* □

The discrete minimization that appears in the r.h.s. of Theorem 4.2 corresponds to a fundamental trade-off in the PBM. When trying to discriminate a suboptimal arm k from the L optimal ones, it is desirable to put it higher in the list to obtain more information, as $d(\kappa_l \theta_k, \kappa_l \theta_L)$ is an increasing function of κ_l. On the other hand, the gap $\Delta_{v_{k,l}}(\theta)$ is also increasing as l gets closer to the top of the list. The fact that $d(\kappa_l \theta_k, \kappa_l \theta_L)$ is not linear in κ_l (it is a strictly convex function of κ_l) renders the trade-off non trivial. It is easily checked that when $(\theta_1 - \theta_L)$ is very small, i.e. when all optimal arms are equivalent, the optimal exploratory position is $l = 1$. In contrast, it is equal to L when the gap $(\theta_L - \theta_{L+1})$ becomes very small. Note that by using that for any suboptimal $a \in \mathcal{A}$, $\Delta_a(\theta) \geq \sum_{k=L+1}^{K} \sum_{l=1}^{L} \mathbb{1}\{a_l = k\} \kappa_l (\theta_L - \theta_k)$, one can lower bound the r.h.s. of Theorem 4.2 by $\kappa_L \sum_{k=L+1}^{K} (\theta_L - \theta_k)/d(\kappa_L \theta_k, \kappa_L \theta_L)$, which is not tight in general.

REMARK. In the uncensored version of the PBM – i.e., if the $Y_l(t)$ were observed –, the expression of $I_a(\theta, \lambda)$ is simpler: it is equal to $\sum_{l=1}^{L} \sum_{k=1}^{K} \mathbb{1}\{A_l(t) = k\} \kappa_l d(\theta_k, \lambda_k)$ and leads to a lower bound that coincides with (4.3). The uncensored PBM is actually statistically very close to the weighted Cascade model and can be addressed by algorithms that do not assume knowledge of the $(\kappa_l)_l$ but only of their ordering.

4.3.6 *Algorithms*

In this section we introduce two algorithms for the PBM. The first one uses the CUCB strategy of [21] and requires an simple upper confidence bound for θ_k based on the estimator $\hat{\theta}_k(t)$ defined in (4.2). The second algorithm is based on the *Parsimonious Item Exploration* – PIE(L) – scheme proposed in [27] and aims at reaching asymptotically optimal performance. For this second algorithm, termed PBM-PIE, it is also necessary to use a multi-position analog of the well-known KL-UCB index [40] that is inspired by a result of [85]. The analysis of PBM-PIE provided below confirms the relevance of the lower bound derived in Section 4.3.2.

PBM-UCB. The first algorithm simply consists in sorting optimistic indices in decreasing order and pulling the corresponding first L arms [21]. To derive the expression of the required "exploration bonus" we use an upper confidence for $\hat{\theta}_k(t)$ based on Hoeffding's inequality:

$$U_k^{UCB}(t,\delta) = \frac{S_k(t)}{\tilde{N}_k(t)} + \sqrt{\frac{N_k(t)}{\tilde{N}_k(t)}}\sqrt{\frac{\delta}{2\tilde{N}_k(t)}},$$

for which a coverage bound is given by the next proposition, proven in Section 4.A.2.

Proposition 4.2 *Let k be any arm in $\{1,\dots,K\}$, then for any $\delta > 0$,*

$$\mathbb{P}\left(U_k^{UCB}(t,\delta) \leq \theta_k\right) \leq e\delta\log(t)e^{-\delta}.$$

Following the ideas of [26], it is possible to obtain a logarithmic regret upper bound for this algorithm.

Theorem 4.3 *Let $C(\kappa) = \min_{1\leq l\leq L}[(\sum_{j=1}^{L}\kappa_j)^2/l + (\sum_{j=1}^{l}\kappa_j)^2]/\kappa_L^2$ and $\Delta = \min_{a\in\sigma(a^*)\setminus a^*}\Delta_a$, where $\sigma(a^*)$ denotes the permutations of the optimal action. Using PBM-UCB with $\delta = (1+\epsilon)\log(t)$ for some $\epsilon > 0$, there exists a constant $C_0(\epsilon)$ independent from the model parameters such that the regret of PBM-UCB is bounded from above by*

$$\mathbb{E}[R(T)] \leq C_0(\epsilon) + 16(1+\epsilon)C(\kappa)\log T\left(\frac{L}{\Delta} + \sum_{k\notin a^*}\frac{1}{\kappa_L(\theta_L - \theta_k)}\right).$$

The proof of the regret upper bound of PBM-UCB is omitted in this thesis and can be found in the original paper [71]. The presence of the term L/Δ in the above expression is attributable to limitations of the mathematical analysis. On the other hand, the absence of the KL-divergence terms appearing in the lower bound (4.6) is due to the use of an upper confidence bound based on Hoeffding's inequality.

PBM-PIE. We adapt the PIE(l) algorithm introduced by [27] for the Cascade Model to the PBM in Algorithm 13 below. At each round, the learner potentially explores at position L with probability 1/2 using the following upper-confidence bound for each arm k

$$U_k(t,\delta) = \sup_{q\in[\theta_k^{\min}(t),1]}\left\{q \,\middle|\, \sum_{l=1}^{L} N_{k,l}(t)\, d\left(\frac{S_{k,l}(t)}{N_{k,l}(t)}, \kappa_l q\right) \leq \delta\right\}, \tag{4.7}$$

where $\theta_k^{\min}(t)$ is the minimum of the convex function $\phi : q \mapsto \sum_{l=1}^{L} N_{k,l}(t) d(S_{k,l}(t)/N_{k,l}(t), \kappa_l q)$. In other positions, $l = 1, \dots, L-1$, PBM-PIE selects the arms with the largest estimates $\hat{\theta}_k(t)$. The resulting algorithm is presented as Algorithm 13 below, denoting by $\mathcal{L}(t)$ the L-largest empirical estimates, referred to as the "leaders" at round t.

Algorithm 9: PBM-PIE

Data: K, L, observation probabilities κ, $\epsilon > 0$
Initialization: first K rounds, play each arm at every position;
for $t = K+1, \dots, T$ **do**
  Compute $\hat{\theta}_k(t)$ for all k;
  $\mathcal{L}(t) \leftarrow$ top-L ordered arms by decreasing $\hat{\theta}_k(t)$;
  $A_l(t) \leftarrow \mathcal{L}_l(t)$ for each position $l < L$;
  $\mathcal{B}(t) \leftarrow \{k | k \notin \mathcal{L}(t), U_k(t, (1+\epsilon)\log(T)) \geq \hat{\theta}_{\mathcal{L}_L(t)}(t)\}$;
  if $\mathcal{B}(t) = \emptyset$ **then**
    $A_L(t) \leftarrow \mathcal{L}_L(t)$;
  else
    With probability 1/2, select $A_L(t)$ uniformly at random from $\mathcal{B}(t)$, else $A_L(t) \leftarrow \mathcal{L}_L(t)$;
  end
  Play action $A(t)$ and observe feedback $Z(t)$; Update $N_{k,l}(t+1)$ and $S_{k,l}(t+1)$;
end

The $U_k(t, \delta)$ index defined in (4.7) aggregates observations from all positions – as in PBM-UCB – but allows to build tighter confidence regions as shown by the next proposition proven in Section 4.A.3.

Proposition 4.3 *For all $\delta \geq L+1$,*

$$\mathbb{P}\left(U_k(t,\delta) < \theta_k\right) \leq e^{L+1} \left(\frac{\lceil \delta \log(t) \rceil \, \delta}{L}\right)^L e^{-\delta}.$$

We may now state the main result of this section that provides an upper bound on the regret of PBM-PIE.

Theorem 4.4 *Using PBM-PIE with $\delta = (1+\epsilon)\log(t)$ and $\epsilon > 0$, for any $\eta < \min_{k<K}(\theta_k - \theta_{k+1})/2$, there exist problem-dependent constants $C_1(\eta)$, $C_2(\epsilon, \eta)$, $C_3(\epsilon)$ and $\beta(\epsilon, \eta)$ such that*

$$\mathbb{E}[R(T)] \leq (1+\epsilon)^2 \log(T) \sum_{k=L+1}^{K} \frac{\kappa_L(\theta_L - \theta_k)}{d(\kappa_L \theta_k, \kappa_L(\theta_L - \eta))} + C_1(\eta) + \frac{C_2(\epsilon,\eta)}{T^{\beta(\epsilon,\eta)}} + C_3(\epsilon).$$

The proof of this result is provided in Section 4.A.3. Comparing to the expression in (4.6), Theorem 4.4 shows that PBM-PIE reaches asymptotically optimal performance when the optimal exploring position is indeed located at index L. In other case, there is a gap that is caused by the fact that the exploring position is fixed beforehand and not adapted from the data.

Remark 4.1 *It is possible to use the KL-UCB optimistic indices presented above to build a similar policy as PBM-UCB. In practice, it has comparable performances to PBM-PIE but its analysis is more complex and remains an open question.*

We conclude this section by a quick description of two other algorithms that will be used in the experimental section to benchmark our results.

RANKED BANDITS (RBA-KLUCB). The state-of-the-art algorithm for the sequential "learning to rank" problem was proposed by [98]. It runs one bandit algorithm per position, each one being entitled to choose the best suited arm at its rank. The underlying bandit algorithm that runs in each position is left to the choice of the user, the better the policy the lower the regret can be. If the bandit algorithm at position l selects an arm already chosen at a higher position, it receives a reward of zero. Consequently, the bandit algorithm operating at position l tends to focus on the estimation of l-th best arm. In the next section, we use as benchmark the Ranked Bandits strategy using the KL-UCB algorithm [40] as the per-position bandit.

PBM-TS. The observations $Z_l(t)$ are censored Bernoulli which results in a posterior that does not belong to a standard family of distribution. [67] suggest a version of Thompson Sampling called "Bias Corrected Multiple Play TS" (or BC-MP-TS) that approximates the true posterior by a Beta distribution. We observed in experiments that for parameter values close to one, this algorithm does not explore enough. In Figure 4.2, we show this phenomenon for $\theta = (0.95, 0.85, 0.75, 0.65, 0.55)$. The true posterior for the parameter θ_k at time t may be written as a product of truncated scaled beta distributions

$$\pi_t(\theta_k) \propto \prod_l \theta_k^{\alpha_{k,l}(t)} (1 - \kappa_l \theta_k)^{\beta_{k,l}(t)},$$

where $\alpha_{k,l}(t) = S_{k,l}(t)$ and $\beta_{k,l}(t) = N_{k,l}(t) - S_{k,l}(t)$. To draw from this exact posterior, we use rejection sampling with proposal distribution $\text{Beta}(\alpha_{k,m}(t), \beta_{k,m}(t))/\kappa_m$, where $m = \arg\max_{1 \le l \le L}(\alpha_{k,l}(t) + \beta_{k,l}(t))$.

4.4 NUMERICAL EXPERIMENTS

We conducted experiments on two types of datasets:

1. an arbitrary simple problem chosen so as to verify our theoretical claims,
2. a real problem with parameters estimated on click logs from a search engine.

4.4.1 *Simulations*

In order to evaluate our strategies, a simple problem is considered in which $K = 5$, $L = 3$, $\kappa = (0.9, 0.6, 0.3)$ and $\theta = (0.45, 0.35, 0.25, 0.15, 0.05)$. The arm expectations are chosen such that the asymptotic behavior can be observed after reasonable time horizon. All results are averaged based on 10,000 independent runs of the algorithm. We present the results in Figure 4.3 where PBM-UCB, PBM-PIE and PBM-TS are compared to RBA-KLUCB. The performance of PBM-PIE and PBM-TS are comparable, the latter even being under the lower bound (it is a common observation, e.g. see [67], and is due to the asymptotic nature of the lower bound). The curves confirm our analysis for PBM-PIE and lets us conjecture that the true Thompson Sampling policy might be asymptotically optimal. As expected, PBM-PIE shows asymptotically optimal performance, matching the lower bound after a large enough horizon.

# ads (K)	# records	$min\ \theta$	$max\ \theta$
5	216, 565	0.016	0.077
5	68, 179	0.031	0.050
6	435, 951	0.025	0.067
6	110, 071	0.023	0.069
6	147, 214	0.004	0.148
8	122, 218	0.108	0.146
11	1, 228, 004	0.022	0.149
11	391, 951	0.022	0.084

Table 4.1: Statistics on the queries: each line corresponds to the sub-dataset associated with a query.

4.4.2 *Real data experiments: search advertising*

The dataset was provided for KDD Cup 2012 track 2[60] and involves session logs of soso.com, a search engine owned by Tencent. It consists of ads that were inserted among search results. Each of the $150M$ lines from the log contains the user ID, the query typed, an ad, a position (1, 2 or 3) at which it was displayed and a binary reward (click/no-click). First, for every query, we excluded ads that were not displayed at least $1,000$ times at every position. We also filtered queries that had less than 5 ads satisfying the previous constraints. As a result, we obtained 8 queries with at least 5 and up to 11 ads. For each query q, we computed the matrix of the average click-through rates (CTR): $M_q \in \mathbb{R}^{K\times L}$, where K is the number of ads for the query q and $L = 3$ the number of positions. It is noticeable that the SVD of each M_q matrix has a highly dominating first singular value, therefore validating the low-rank assumption underlying in the PBM. In order to estimate the parameters of the problem, we used the EM algorithm suggested by [25, 31]. Table 4.1

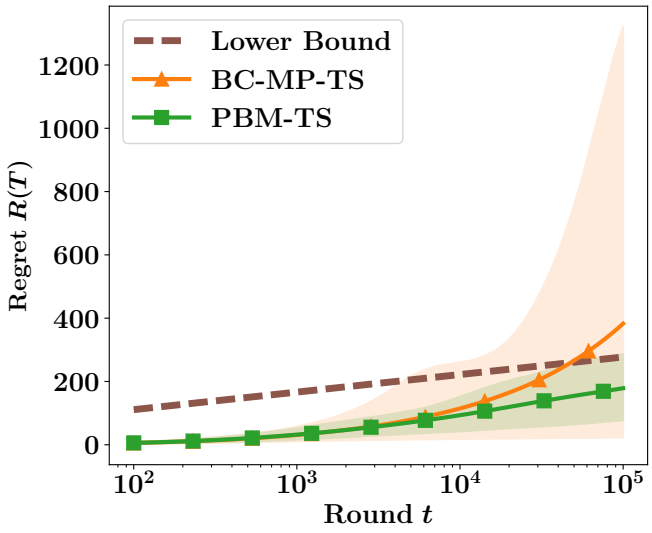


Figure 4.2: Average regret of PBM-TS and BC-MP-TS compared for high parameters. Shaded areas: first and last deciles.

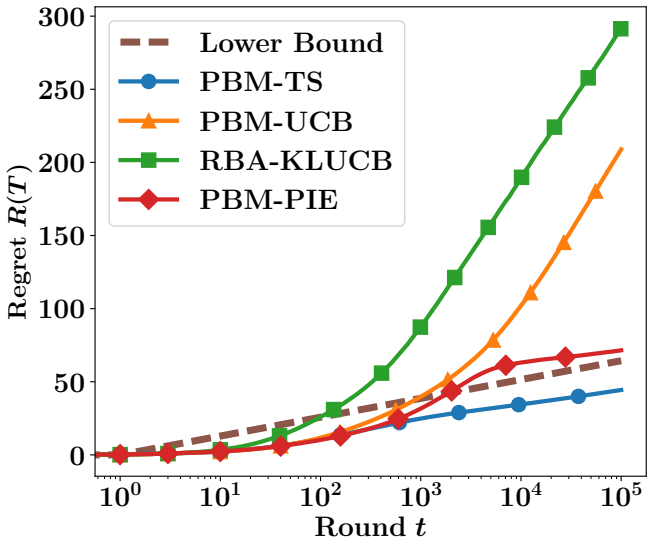


Figure 4.3: Average regret of various algorithms on synthetic data under the PBM.

reports some statistics about the bandit models reconstructed for each query: number of arms K, amount of data used to compute the parameters, minimum and maximum values of the θ's for each model.

We conducted a series of 2,000 simulations over this dataset. At the beginning of each run, a query was randomly selected together with corresponding probabilities of scanning positions and arm expectations. Even if rewards were still simulated, this scenario is more realistic since the values of the parameters were extracted from a real-world dataset. We show results for the different algorithms in Figure 4.4. It is remarkable that RBA-KLUCB performs slightly better than PBM-UCB. One can imagine that PBM-UCB does not benefit enough from position aggregations – only 3 positions are considered – to beat RBA-KLUCB. Both of them are outperformed by PBM-TS and PBM-PIE.

4.5 CONCLUSIONS AND EXTENSIONS

This work provides the first analysis of the PBM in an online context. The proof scheme used to obtain the lower bound on the regret is interesting on its own, as it can be generalized to various other settings. The tightness of the lower bound is validated by our analysis of PBM-PIE but it would be an interesting future contribution to provide such guarantees for more straightforward algorithms such as PBM-TS or a "PBM-KLUCB" using the confidence regions of PBM-PIE.

The main assumption in our work is the knowledge of the values of the $(\kappa_l)_{l\in[L]}$. In practice, the algorithms are robust to small variations of the κ's, but the proposal of an algorithm that is unaware of these parameters would be an interesting step forward. In this direction, Katariya et al. [61] study a bandit problem where, at each step, the learner selects a pair of row and columns arms, from a rank-1 matrix, and observes the product of their Bernoulli random variables. They make no assumption on neither row nor column parameters. However, the study of the PBM setting without the knowlege of position parameters remains an open question today.

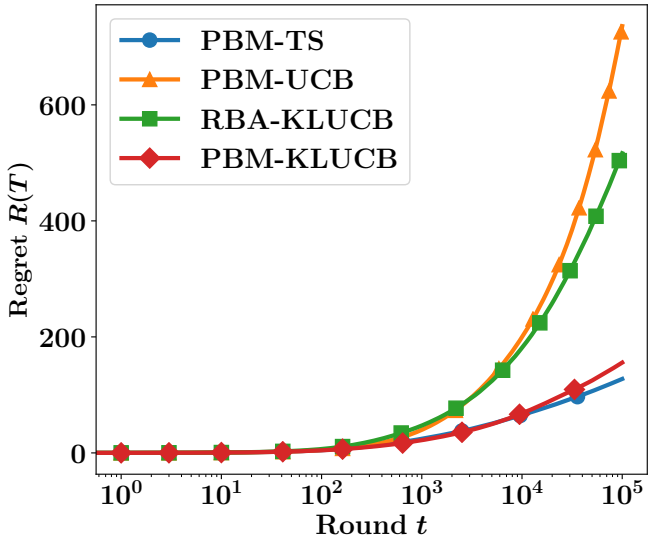


Figure 4.4: Performance of the proposed algorithms under the PBM on real data.

APPENDIX 4.A ELEMENTS OF PROOF

In this section, we gather most of the technical results from this chapter.

4.A.1 *Proof of Theorem 4.1*

4.A.1.1 *Proof of Lemma 4.1*

Proof *Under the PBM, the conditional expectation of the log-likelihood ratio defined in (4.4) writes*

$$\begin{aligned}
&\mathbb{E}_\theta[\ell(t)|A(1),\dots,A(t)] \\
&\quad= \mathbb{E}_\theta\left[\sum_{s=1}^{t}\sum_{a\in\mathcal{A}}\mathbb{1}\{A(s)=a\}\sum_{l=1}^{L}\log\frac{p_{a_l}(X_l(s)Y_l(s);\theta)}{p_{a_l}(X_l(s)Y_l(s);\lambda)}\,\middle|\,A(1),\dots,A(t)\right] \\
&\quad= \sum_{s=1}^{t}\sum_{a\in\mathcal{A}}\mathbb{1}\{A(s)=a\}\sum_{l=1}^{L}\mathbb{E}\left[\log\frac{p_{a_l}(X_l(s)Y_l(s);\theta)}{p_{a_l}(X_l(s)Y_l(s);\lambda)}\,\middle|\,A(s)=a\right] \\
&\quad= \sum_{a\in\mathcal{A}}N_a(t)\sum_{l=1}^{L}\sum_{k=1}^{K}\mathbb{1}\{a_l=k\}d(\kappa_l\theta_k,\kappa_l\lambda_k) \\
&\quad= \sum_{a\in\mathcal{A}}N_a(t)I_a(\theta,\lambda),
\end{aligned}$$

using the notation $I_a(\theta,\lambda)=\sum_{l=1}^{L}\sum_{k=1}^{K}\mathbb{1}\{a_l=k\}d(\kappa_l\theta_k,\kappa_l\lambda_k)$. □

4.A.1.2 *Details on the proof of Proposition 4.1*

Lemma 4.2 *Let* $\theta=(\theta_1,\dots,\theta_K)$ *and* $\lambda=(\lambda_1,\dots,\lambda_K)$ *be two bandit models such that the distributions of all arms in* θ *and* λ *are mutually absolutely continuous. Let* σ *be a stopping*

time with respect to $(\mathcal{F}_t)$ *such that* $(\sigma < +\infty)$ *a.s. under both models. Let* $\mathcal{E} \in \mathcal{F}_\sigma$ *be an event such that* $0 < \mathbb{P}_\theta(\mathcal{E}) < 1$. *Then one has*

$$\sum_{a \in \mathcal{A}} I_a(\theta, \lambda) \mathbb{E}_\theta[N_a(\sigma)] \geq d(\mathbb{P}_\theta(\mathcal{E}), \mathbb{P}_\lambda(\mathcal{E})),$$

where $I_a(\theta, \lambda)$ *is the conditional expectation of the log-likelihood ratio for the model of interest.*

The proof of this lemma directly follows from the above expressions of the log-likelihood ratio and from the proof of Lemma 1 in Appendix A.1 of [63].

We simply recall the following technical lemma for completeness.

Lemma 4.3 *Let* σ *be any stopping time with respect to* $(\mathcal{F}_t)$. *For every event* $A \in \mathcal{F}_\sigma$,

$$\mathbb{P}_\lambda(A) = \mathbb{E}_\theta[\mathbb{1}\{A\} \exp(-\ell(\sigma))].$$

A full proof of Lemma 4.3 can be found in the Appendix A.3 of [63] (proof of Lemma 15).

4.A.1.3 *Lower bound proof (Theorem 4.1)*

Proof *In order to prove the simplified lower bound of Theorem 4.1 we basically have two arguments:*

1. *a lower bound on* $f(\theta)$ *can be obtained by enlarging the feasible set, that is by relaxing some constraints;*
2. *Lemma 4.4 can be used to lower bound the objective function of the problem.*

The constant $f(\theta)$ *is defined by*

$$f(\theta) = \inf_{c \geq 0} \sum_{a \neq a^*(\theta)} \Delta_a(\theta) c_a \tag{4.8}$$

$$s.t \quad \inf_{\lambda \in B(\theta)} \sum_{a \in \mathcal{A}} I_a(\theta, \lambda) c_a \geq 1. \tag{4.9}$$

We begin by relaxing some constraints: we only allow the change of measure λ *to belong to the sets* $B_k(\theta) := \left\{\lambda \in \Theta | \forall j \neq k, \theta_j = \lambda_j \text{ and } \mu^*(\theta) < \mu^*(\lambda)\right\}$ *defined in Section 4.3.2:*

$$f(\theta) = \inf_{c \geq 0} \sum_{a \neq a^*(\theta)} \Delta_a(\theta) c_a \tag{4.10}$$

$$s.t \;\; \forall k \notin a^*(\theta), \; \forall \lambda \in B_k(\theta), \sum_{a \in \mathcal{A}} I_a(\theta, \lambda) c_a \geq 1. \tag{4.11}$$

The $K - L$ *constraints (4.11) only let one parameter move and must be true for any value satisfying the definition of the corresponding set* $B_k(\theta)$. *In practice, for each* k, *the parameter* λ_k *must be set to at least* θ_L. *Consequently, these constraints may then be rewritten*

$$f(\theta) = \inf_{c \geq 0} \sum_{a \neq a^*(\theta)} \Delta_a(\theta) c_a \tag{4.12}$$

$$s.t \; \forall k \notin a^*(\theta), \sum_{a \neq a^*(\theta)} c_a \sum_{l=1}^{L} \mathbb{1}\{a_l = k\} d(\kappa_l \theta_k, \kappa_l \theta_L) \geq 1. \tag{4.13}$$

□

Proposition 4.4 tells us that coefficients c_a are all zeros except for actions $a \in \mathcal{A}$ which can be written $a = v_{k,l_k}$ where $l_k = \arg\min_{l \leq L} \frac{\Delta_{v_{k,l}}(\theta)}{d(\kappa_l \theta_k, \kappa_l \theta_L)}$. Thus, we obtain the desired lower bound by rewriting (4.12) as

$$f(\theta) \geq \sum_{k=L+1}^{K} \min_{l \in \{1,\dots,L\}} \frac{\Delta_{v_{k,l}}(\theta)}{d(\kappa_l \theta_k, \kappa_l \theta_L)}.$$

Proposition 4.4 *Let $c = \{c_a : a \neq a^*\}$ be a solution of the linear problem (LP) in Theorem 4.1. Coefficients are all zeros except for actions a which can be written as $a = (1, \dots, l_k - 1, k, l_k, \dots, L-1) := v_{k,l_k}$ where $k > L$ and $l_k = \arg\min_{l \leq L} \frac{\Delta_{v_{k,l}}(\theta)}{d(\kappa_l \theta_k, \kappa_l \theta_L)}$.*

Proof *We denote by $\pi_k(a)$ the position of item $k \in \{1, \dots, K\}$ in action a (0 if $k \notin a$). Let l_k be the optimal position of item $k > L$ for exploration: $l_k = \arg\min_{l \leq L} \frac{\Delta_{v_{k,l}}(\theta)}{d(\kappa_l \theta_k, \kappa_l \theta_L)}$. Following [27], we show by contradiction that $c_a > 0$ implies that a can be written v_{k,l_k} for a well chosen $k > L$. Let $\alpha \neq a^*$ be a suboptimal action such that $\forall k > L, \alpha \neq v_{k,l_k}$ and $c_\alpha > 0$. We need to show a contradiction. Let us introduce a new set of coefficients c' defined as follows, for any $a \neq a^*$:*

$$c'_a = \begin{cases} 0 & \text{if } a = \alpha \\ c_a + \frac{d(\kappa_{\pi_k(\alpha)}\theta_k, \kappa_{\pi_k(\alpha)}\theta_L)}{d(\kappa_{l_k}\theta_k, \kappa_{l_k}\theta_L)} c_\alpha & \text{if } \exists k > L \text{ s.t. } a = v_{k,l_k} \text{ and } k \in \alpha \\ c_a & \text{otherwise.} \end{cases}$$

According to Lemma 4.4, these coefficients satisfy the constraints of the LP. We now show that these new coefficients yield a strictly lower value to the optimization problem:

$$\begin{aligned} c(\theta) - c'(\theta) &= c_\alpha \Delta_\alpha(\theta) - \sum_{k>L : k \in \alpha} \frac{d(\kappa_{\pi_k(\alpha)}\theta_k, \kappa_{\pi_k(\alpha)}\theta_L)}{d(\kappa_{l_k}\theta_k, \kappa_{l_k}\theta_L)} c_\alpha \Delta_{v_{k,l_k}}(\theta) \\ &> c_\alpha \left(\sum_{k>L : k \in \alpha} \Delta_{v_{k,\pi_k(\alpha)}}(\theta) - \sum_{k>L : k \in \alpha} \frac{d(\kappa_{\pi_k(\alpha)}\theta_k, \kappa_{\pi_k(\alpha)}\theta_L)}{d(\kappa_{l_k}\theta_k, \kappa_{l_k}\theta_L)} \Delta_{v_{k,l_k}}(\theta) \right). \end{aligned} \tag{4.14}$$

The strict inequality (4.14) is shown in Lemma 4.5. Let $k > L$ be one of the suboptimal arms in α. By definition of l_k, the corresponding term of the sum in equation (4.14) is positive. Thus, we have that $c(\theta) > c'(\theta)$ and, hence, by contradiction, we showed that $c_a > 0$ iff a can be written $a = v_{k,l_k}$ for some $k > L$. □

Lemma 4.4 *Let c be a vector of coefficients that satisfy constraints (4.13) of the optimization problem. Then, coefficients c' as defined in Proposition 4.4 also satisfy the constraints:*

$$\forall k \notin a^*(\theta), \sum_{a \neq a^*(\theta)} c'_a \sum_{l=1}^{L} \mathbb{1}\{a_l = k\} d(\kappa_l \theta_k, \kappa_l \theta_L) \geq 1.$$

Proof *We use the same α as introduced in Proposition 4.4. Let us fix $k \notin a^*(\theta)$. Let us define*

$$L(c) = \sum_{a \neq a^*(\theta)} c_a \sum_{l=1}^{L} \mathbb{1}\{a_l = k\} d(\kappa_l \theta_k, \kappa_l \theta_L).$$

We have

$$L(c') - L(c) = -c_\alpha \sum_{l=1}^{L} \mathbb{1}\{\alpha_l = k\} d(\kappa_l \theta_k, \kappa_l \theta_L) + \sum_{l:\alpha_l > L} \frac{d(\kappa_l \theta_k, \kappa_l \theta_L)}{d(\kappa_{l_k} \theta_k, \kappa_{l_k} \theta_L)} c_\alpha \times \mathbb{1}\{\alpha_l = k\} d(\kappa_{l_k} \theta_k, \kappa_{l_k} \theta_L).$$

If $k \notin \alpha$, clearly, $L(c') - L(c) = 0$. Else, $k \in \alpha$ and we note p its position in α: $p = \pi_k(\alpha)$. We rewrite:

$$L(c') - L(c) = c_\alpha d(\kappa_p \theta_k, \kappa_p \theta_L) \left(-1 + \frac{d(\kappa_{l_k} \theta_k, \kappa_{l_k} \theta_L)}{d(\kappa_{l_k} \theta_k, \kappa_{l_k} \theta_L)} \right) = 0.$$

Thus, the coefficients c' satisfy the constraints from Proposition 4.4. □

Lemma 4.5 *Let α be as in the proof of Proposition 4.4.*

$$\Delta_\alpha(\theta) > \sum_{k > L : k \in \alpha} \Delta_{v_{k, \pi_k(\alpha)}}(\theta).$$

Proof *Let $k_1, \ldots, k_p$ be the suboptimal arms in α by increasing position. Let $v(\alpha)$ be the action in $\mathcal{A}$ with lower regret such that it contains all the suboptimal arms of α in the same positions. Thus, $v(\alpha) = (1, \ldots, \pi_{k_1}(\alpha) - 1, k_1, \pi_{k_1}(\alpha), \ldots, \pi_{k_2}(\alpha) - 2, k_2, \pi_{k_2}(\alpha) - 1, \ldots, L - p)$. By definition, one has that $\Delta_\alpha(\theta) \geq \Delta_{v(\alpha)}(\theta)$. In the following, we show that $\Delta_{v(\alpha)}(\theta) \geq \sum_{k > L : k \in \alpha} \Delta_{v_{k, \pi_k(\alpha)}}(\theta)$ for $p = 2$ (that is to say α contains 2 suboptimal arms k_1 and k_2).*

For the sake of readability, we write π_i instead of $\pi_{k_i}(\alpha)$ in the following.

$$\begin{aligned}
\Delta_{v(\alpha)}(\theta) &= \sum_{l=1}^{L} \kappa_l (\theta_l - \theta_{(v_{k_1, \pi_1})_l}) + \sum_{l=1}^{L} \kappa_l (\theta_{(v_{k_1, \pi_1})_l} - \theta_{v(\alpha)_l}) \\
&= \Delta_{v_{k_1, \pi_1}}(\theta) + \left[\kappa_{\pi_2} \theta_{\pi_2 - 1} + \ldots + \kappa_L \theta_{L-1} \right] \\
&\quad - \left[\kappa_{\pi_2} \theta_{k_2} + \kappa_{\pi_2 + 1} \theta_{\pi_2 - 1} + \ldots + \kappa_L \theta_{L-2} \right] \\
&= \Delta_{v_{k_1, \pi_1}}(\theta) + \Delta_{v_{k_2, \pi_2}}(\theta) + \left[\kappa_{\pi_2} (\theta_{\pi_2 - 1} - \theta_{\pi_2}) + \ldots + \kappa_L (\theta_{L-1} - \theta_L) \right] \\
&\quad - \left[\kappa_{\pi_2 + 1} (\theta_{\pi_2 - 1} - \theta_{\pi_2}) + \ldots + \kappa_L (\theta_{L-2} - \theta_{L-1}) \right] \\
&= \Delta_{v_{k_1, \pi_1}}(\theta) + \Delta_{v_{k_2, \pi_2}}(\theta) + \mathcal{R}(\theta).
\end{aligned}$$

Thus, one has to show that $\mathcal{R}(\theta) = \kappa_{\pi_2} (\theta_{\pi_2 - 1} - \theta_{\pi_2}) + \kappa_{\pi_2 + 1} (2\theta_{\pi_2} - \theta_{\pi_2 - 1} - \theta_{\pi_2 + 1}) + \ldots + \kappa_L (2\theta_{L-1} - \theta_{L-2} - \theta_L) > 0$. In fact, using that $\kappa_l \geq \kappa_{l+1}$ for all $l < L$, we have

$$\begin{aligned}
\mathcal{R}(\theta) &\geq \kappa_{\pi_2 + 1} (\theta_{\pi_2 - 1} - \theta_{\pi_2} + 2\theta_{\pi_2} - \theta_{\pi_2 - 1} - \theta_{\pi_2 + 1}) \\
&\quad + \ldots + \kappa_L (2\theta_{L-1} - \theta_{L-2} - \theta_L) \\
&\geq \kappa_{\pi_2 + 2} (\theta_{\pi_2 + 1} - \theta_{\pi_2 + 2}) + \ldots + \kappa_L (2\theta_{L-1} - \theta_{L-2} - \theta_L) \\
&\geq \ldots \\
&\geq \kappa_L (\theta_{L-1} - \theta_L) \\
&> 0.
\end{aligned}$$

□

4.A.2 *Proof of Proposition 4.2*

In this section, we fix an arm $k \in \{1, \dots, K\}$ and obtain an upper confidence bound for the estimator $\hat{\theta}_k(t) := S_k(t)/\tilde{N}_k(t)$. Let τ_i be the instant of the i-th draw of arm k (the τ_i are stopping times w.r.t. $\mathcal{F}_t$). We introduce the centered sequence of successive observations from arm k

$$\bar{Z}_{k,i} = \sum_{l=1}^{L} \mathbb{1}\{A_l(\tau_i) = k\}(X_l(\tau_i)Y_l(\tau_i) - \theta_k \kappa_l). \tag{4.15}$$

Introducing the filtration $\mathcal{G}_i = \mathcal{F}_{\tau_{i+1}-1}$, one has $\mathbb{E}[\bar{Z}_{k,i}|\mathcal{G}_{i-1}] = 0$, and therefore, the sequence

$$M_{k,n} = \sum_{i=1}^{n} \bar{Z}_{k,i}$$

is a martingale with bounded increments, w.r.t. the filtration $(\mathcal{G}_n)_n$. By construction, one has

$$M_{k,N_k(t)} = S_k(t) - \tilde{N}_k(t)\theta_k = \tilde{N}_k(t)(\hat{\theta}_k(t) - \theta_k).$$

We use the so-called peeling technique together with the maximal version of Azuma-Hoeffding's inequality [13]. For any $\gamma > 0$ one has

$$\begin{aligned}
&\mathbb{P}\left(M_{k,N_k(t)} < -\sqrt{N_k(t)\delta/2}\right) \\
&\leq \sum_{i=1}^{\frac{\log(t)}{\log(1+\gamma)}} \mathbb{P}\left(M_{k,N_k(t)} < -\sqrt{N_k(t)\delta/2}, N_k(t) \in [(1+\gamma)^{i-1}, (1+\gamma)^i)\right) \\
&\leq \sum_{i=1}^{\frac{\log(t)}{\log(1+\gamma)}} \mathbb{P}\left(\exists i \in \{1, \dots, (1+\gamma)^i\} : \; M_{k,i} < -\sqrt{(1+\gamma)^{i-1}\delta/2}\right) \\
&\leq \sum_{i=1}^{\frac{\log(t)}{\log(1+\gamma)}} \exp\left(-\frac{\delta(1+\gamma)^{i-1}}{(1+\gamma)^i}\right) = \frac{\log(t)}{\log(1+\gamma)} \exp\left(-\frac{\delta}{(1+\gamma)}\right).
\end{aligned}$$

Choosing $\gamma = 1/(\delta - 1)$, gives

$$\mathbb{P}\left(\hat{\theta}_k(t) - \theta_k < -\frac{\sqrt{N_k(t)\delta/2}}{\tilde{N}_k(t)}\right) \leq \delta e \log(t) e^{-\delta}.$$

4.A.3 *Regret analysis for PBM-PIE (Theorem 4.4)*

The proof follows the decomposition of [27]. For all $t \geq 1$, we denote $f(t, \epsilon) = (1+\epsilon)\log t$.

4.A.4 *Controlling leaders and estimations*

Define $\eta_0 = \min_{k \in \{1,\dots,L-1\}}(\theta_k - \theta_{k+1})/2$ and let $\eta < \eta_0$. We define the following set of rounds

$$A = \{t \geq 1 : \mathcal{L}(t) \neq (1, \dots, L)\}.$$

Our goal is to upper bound the expected size of A. Let us introduce the following sets of rounds:

$$B = \{t \geq 1 : \exists k \in \mathcal{L}(t), |\hat{\theta}_k(t) - \theta_k| \geq \eta\},$$
$$C = \{t \geq 1 : \exists k \leq L, U_k(t) \leq \theta_k\},$$
$$D = \{t \geq 1 : t \in A \setminus (B \cup C), \exists k \leq L, k \notin \mathcal{L}(t), |\hat{\theta}_k(t) - \theta_k| \geq \eta\}.$$

We first show that $A \subset (B \cup C \cup D)$. Let $t \in A \setminus (B \cup C)$. Let $k, k' \in \mathcal{L}(t)$ such that $k < k'$. Since $t \notin B$, we have that $|\hat{\theta}_k(t) - \theta_k| \leq \eta$ and $|\hat{\theta}_{k'}(t) - \theta_{k'}| \leq \eta$. Since $\eta \leq (\theta_k - \theta_{k'})/2$, we conclude that $\hat{\theta}_k(t) \geq \hat{\theta}_{k'}(t)$. This proves that $(\mathcal{L}_1(t), \ldots, \mathcal{L}_L(t)$ is an increasing sequence. We have that $\mathcal{L}_L(t) > L$ otherwise $\mathcal{L}(t) = (1, \ldots, L)$ which is a contradiction because $t \in A$. Since $\mathcal{L}_L(t) > L$, there exists $k \leq L$ such that $k \notin \mathcal{L}(t)$. We show by contradiction that $|\hat{\theta}_k(t) - \theta_k| \geq \eta$. Assume that $|\hat{\theta}_k(t) - \theta_k| \leq \eta$. We also have that $\hat{\theta}_{\mathcal{L}_L(t)}(t) - \theta_{\mathcal{L}_L(t)} \leq \eta$ because $\mathcal{L}_L(t) \in \mathcal{L}(t)$ and $t \notin B$. Thus, $\hat{\theta}_k(t) > \hat{\theta}_{\mathcal{L}_L(t)}(t)$. We have a contradiction because this would imply that $k \in \mathcal{L}(t)$. Finally we have proven that if $t \in A \setminus (B \cup C)$, then $t \in D$ so $A \subset (B \cup C \cup D)$.

By a union bound, we obtain

$$\mathbb{E}[|A|] \leq [|B|] + [|C|] + [|D|].$$

In the following, we upper bound each set of rounds individually.

CONTROLLING $\mathbb{E}[|B|]$: We decompose $B = \bigcup_{k=1}^{K} (B_{k,1} \cup B_{k,2})$ where

$$B_{k,1} = \{t \geq 1 : k \in \mathcal{L}(t), \mathcal{L}_L(t) \neq k, |\hat{\theta}_k(t) - \theta_k| \geq \eta\}$$
$$B_{k,2} = \{t \geq 1 : k \in \mathcal{L}(t), \mathcal{L}_L(t) = k, |\hat{\theta}_k(t) - \theta_k| \geq \eta\}$$

Let $t \in B_{k,1}$: $k \in A(t)$ so $\mathbb{E}[k \in A(t) | t \in B_{k,1}] = 1$. Furthermore, for all t, $\mathbb{1}\{t \in B_{k,1}\}$ is $\mathcal{F}_{t-1}$ measurable. Then we can apply Lemma 4.9 (with $H = B_{k,1}$ and $c = 1$).

$$\mathbb{E}[|B_{k,1}|] \leq 2(2 + \kappa_L^{-2}\eta^{-2}).$$

Let $t \in B_{k,2}$: $k \in \mathcal{B}(t)$ but because of the randomization of the algorithm, $k \in A(t)$ with probability $1/2$, i.e. $\mathbb{E}[k \in A(t) | t \in B_{k,2}] \geq 1/2$. We get

$$\mathbb{E}[|B_{k,2}|] \leq 4(4 + \kappa_L^{-2}\eta^{-2})$$

By union bound over k, we get $\mathbb{E}[|B|] \leq 2K(10 + 3\kappa_L^{-2}\eta^{-2})$.

CONTROLLING $\mathbb{E}[|C|]$: We decompose $C = \bigcup_{k=1}^{L} C_k$ where $C_k = \{t \geq 1 : U_k(t) \leq \theta_k\}$

We first require to prove Proposition 4.3.

Proof *Theorem 2 of [85] implies that*

$$\mathbb{P}\left(\sum_{l=1}^{L} N_{k,l}(t) d\left(\frac{S_{k,l}(t)}{N_{k,l}(t)}, \kappa_l \theta_k\right) \geq \delta\right) \leq e^{-\delta} \left(\frac{\lceil \delta \log(t) \rceil \delta}{L}\right)^{L} e^{L+1}.$$

The function $\phi : x \mapsto \sum_{l=1}^{L} N_{k,l}(t) d\left(\frac{S_{k,l}(t)}{N_{k,l}(t)}, \kappa_l x\right)$ *is convex and non-decreasing on* $[\theta_k^{min}(t), 1]$*; the convexity is easily checked and* $\theta_k^{min}(t)$ *is defined as the minimum of this convex function. By definition, we have, either,* $U_k(t,\delta) = 1$ *and then* $U_k(t,\delta) > \theta_k$*, or,* $U_k(t,\delta) < 1$ *and* $\phi(U_k(t,\delta)) = \delta$*, consequently*

$$\mathbb{P}\left(U_k(t,\delta) < \theta_k\right) = \mathbb{P}\left(\phi(U_k(t,\delta)) \leq \phi(\theta_k)\right) = \mathbb{P}\left(\delta \leq \phi(\theta_k)\right). \qquad \square$$

Remember that $U_k(t) = U_k(t, (1+\epsilon)\log(t)) = U_k(t, f(t,\epsilon))$. Thus, applying Proposition 4.3, we obtain for arm k,

$$\begin{aligned}
\mathbb{E}[|C_k|] &\leq \sum_{t=1}^{\infty} \mathbb{P}(U_k(t) \leq \theta_k) \\
&\leq \lceil e^{L+1} \rceil + \frac{e^{L+1}}{L^L} \sum_{t=\lceil e^{L+1} \rceil + 1}^{\infty} \frac{(2+\epsilon)^{2L} (\log t)^{3L}}{t^{1+\epsilon}} \\
&\leq C_3(\epsilon),
\end{aligned}$$

for some constant $C_3(\epsilon)$.

CONTROLLING $\mathbb{E}[|D|]$: Decompose D as $D = \bigcup_{k=1}^{L} D_k$ where

$$D_k = \{t \geq 1 : t \in A \setminus (B \cup C), k \notin \mathcal{L}(t), |\hat{\theta}_k(t) - \theta_k| \geq \eta\}.$$

For a given $k \leq L$, D_k is the set of rounds at which k is not one of the leaders, and is not accurately estimated. Let $t \in D_k$. Since $k \notin \mathcal{L}(t)$, we must have $\mathcal{L}_L(t) > L$. In turn, since $t \notin B$, we have $|\hat{\theta}_{\mathcal{L}_L(t)}(t) - \theta_{\mathcal{L}_L(t)}| \leq \eta$, so that

$$\hat{\theta}_{\mathcal{L}_L(t)} \leq \theta_{\mathcal{L}_L(t)} + \eta \leq \theta_L + \eta \leq (\theta_L + \theta_{L+1})/2.$$

Furthermore, since $t \notin C$ and $1 \leq k \leq L$, we have $U_k(t) \geq \theta_k \geq \theta_L \geq (\theta_L + \theta_{L+1})/2 \geq \hat{\theta}_{\mathcal{L}_L(t)}$. This implies that $k \in \mathcal{B}(t)$ thus $\mathbb{E}[k \in A(t) | t \in D_k] \geq 1/(2K)$. We apply Lemma 4.9 with $H \equiv D_k$ and $c = 1/(2K)$ to get

$$\mathbb{E}[|D|] \leq \sum_{k=1}^{L} \mathbb{E}[|D_k|] \leq 4K(4K + \kappa_L^{-2}\eta^{-2}).$$

4.A.4.1 *Regret decomposition*

We decompose the regret by distinguishing rounds in $A \cup B$ and other rounds. More specifically, we introduce the following sets of rounds for arm $k > L$:

$$E_k = \{t \geq 1 : t \notin (B \cup C \cup D), \mathcal{L}(t) = a^*, A(t) = v_{k,L}\}.$$

The set of instants at which a suboptimal action is selected now can be expressed as follows

$$\{t \geq 1 : A(t) \neq a^*\} \subset (B \cup C \cup D) \cup (\cup_{k=L+1} E_k).$$

Using a union bound, we obtain the upper bound

$$\mathbb{E}[R(T)] \leq \left(\sum_{l=1}^{L} \kappa_l\right) \mathbb{E}[|B \cup C \cup D|] + \sum_{k=L+1}^{K} \Delta_{v_{k,L}}(\theta) \mathbb{E}[|E_k|].$$

From previous boundaries, putting it all together, there exist $C_1(\eta)$ and $C_3(\epsilon)$, such that

$$\left(\sum_{l=1}^{L} \kappa_l\right) (\mathbb{E}[|B|] + \mathbb{E}[|C|] + \mathbb{E}[|D|]) \leq C_1(\eta) + C_3(\epsilon).$$

At this step, it suffices to bound events E_k for all $k > L$.

4.A.4.2 *Bounding event E_k*

We proceed similarly to [40]. Let us fix an arm $k > L$. Let $t \in E_k$: arm k is pulled in position L, so by construction of the algorithm, we have that $k \in \mathcal{B}(t)$ and thus $U_k(t) \geq \hat{\theta}_{\mathcal{L}_L(t)}(t)$. We first show that this implies that $U_k(t) \geq \theta_L - \eta$. Since $t \in E_k$, we know that $\mathcal{L}_L(t) = L$, and since $t \notin B$, $|\hat{\theta}_L(t) - \theta_L| \leq \eta$. This leads to

$$U_k(t) \geq \hat{\theta}_{\mathcal{L}_L(t)}(t) = \hat{\theta}_L(t) \geq \theta_L - \eta.$$

Recall that $N_{k,L}(t)$ is the number of times arm k was played in position L. By denoting $d^+(x, y) = \mathbb{1}\{x < y\} d(x, y)$, we have that

$$\begin{aligned} &N_{k,L}(t) d^+(S_{k,L}(t)/N_{k,L}(t), \kappa_L(\theta_L - \eta)) \\ &\qquad \leq N_{k,L}(t) d^+(S_{k,L}(t)/N_{k,L}(t), \kappa_L U_k(t)) \\ &\qquad \leq \sum_{l=1}^{L} N_{k,l}(t) d^+(S_{k,l}(t)/N_{k,l}(t), \kappa_l U_k(t)) \leq f(t, \epsilon). \end{aligned}$$

This implies that $\mathbb{1}\{t \in E_k\} \leq \mathbb{1}\{N_{k,L}(t) d^+(S_{k,L}(t)/N_{k,L}(t), \kappa_L(\theta_L - \eta)) \leq f(t, \epsilon)\}$.

Lemma 4.6 *([40], Lemma 7) Denoting by $\hat{v}^L_{k,s}$ the empirical mean of the first s samples of $Z_{k,L}$, we have*

$$\begin{aligned} \sum_{t=1}^{T} \mathbb{1}\{A(t) = v_{k,L}, N_{k,L}(t) d^+(S_{k,L}(t)/N_{k,L}(t), \kappa_L(\theta_L - \eta)) \leq f(t, \epsilon)\} \\ \leq \sum_{s=1}^{T} \mathbb{1}\{s d^+(\hat{v}^L_{k,s}, \kappa_L(\theta_L - \eta)) \leq f(T, \epsilon)\}. \end{aligned}$$

We apply Lemma 4.6 which is a direct translation of Lemma 7 from [40] to our problem. This yields

$$|E_k| \leq \sum_{s=1}^{T} \mathbb{1}\{s d^+(\hat{v}^L_{k,s}, \kappa_L(\theta_L - \eta)) \leq f(T, \epsilon)\}.$$

Let $\gamma > 0$. We define $K_T = \frac{(1+\gamma) f(T,\epsilon)}{d^+(\kappa_L \theta_k, \kappa_L(\theta_L - \eta))}$. We now rewrite the last inequality splitting the sum in two parts.

$$\begin{aligned}
&\sum_{s=1}^{T} \mathbb{P}\left(sd^+(\hat{v}^L_{k,s}, \kappa_L(\theta_L - \eta)) \leq f(T,\epsilon)\right) \\
&\qquad \leq K_T + \sum_{s=K_T+1}^{\infty} \mathbb{P}(K_T d^+(\hat{v}^L_{k,s}, \kappa_L(\theta_L - \eta)) \leq f(T,\epsilon)) \\
&\qquad \leq K_T + \sum_{s=K_T+1}^{\infty} \mathbb{P}(d^+(\hat{v}^L_{k,s}, \kappa_L(\theta_L - \eta)) \leq d(\kappa_L\theta_k, \kappa_L(\theta_L - \eta))/(1+\gamma)) \\
&\qquad \leq K_T + \frac{C_2(\gamma,\eta)}{T^{\beta(\gamma,\eta)}},
\end{aligned}$$

where last inequality comes from Lemma 4.7. Fixing $\gamma < \epsilon$, we obtain the desired result, which concludes the proof.

Lemma 4.7 *For each $\gamma > 0$, there exists $C_2(\gamma,\eta) > 0$ and $\beta(\gamma,\eta) > 0$ such that*

$$\sum_{s=K_T+1}^{\infty} \mathbb{P}\left(d^+(\hat{v}^L_{k,s}, \kappa_L(\theta_L - \eta)) \leq \frac{d(\kappa_L\theta_k, \kappa_L(\theta_L - \eta)}{1+\gamma}\right) \leq \frac{C_2(\gamma,\eta)}{T^{\beta(\gamma,\eta)}}.$$

Proof *If $d^+(\hat{v}^L_{k,s}, \kappa_L(\theta_L - \eta)) \leq \frac{d(\kappa_L\theta_k, \kappa_L(\theta_L - \eta))}{1+\gamma}$, then there exists some $r(\gamma,\eta) \in (\theta_k, \theta_L - \eta)$ such that $\hat{v}^L_{k,s} > \kappa_L r(\gamma,\eta)$ and*

$$d(\kappa_L r(\gamma,\eta), \kappa_L(\theta_L - \eta)) = \frac{d(\kappa_L\theta_k, \kappa_L(\theta_L - \eta))}{1+\gamma}.$$

Hence,

$$\begin{aligned}
&\mathbb{P}\left(d^+(\hat{v}_{k,s}, \kappa_L\theta_L) < \frac{d(\kappa_L\theta_k, \kappa_L\theta_L)}{1+\gamma}\right) \\
&\quad \leq \mathbb{P}\left(d(\hat{v}_{k,s}, \kappa_L\theta_k) > d(\kappa_L r(\gamma,\eta), \kappa_L\theta_k), \hat{v}_{k,s} > \kappa_L\theta_k\right) \\
&\quad \leq \mathbb{P}(\hat{v}_{k,s} > \kappa_L r(\gamma,\eta)) \leq \exp(-sd(\kappa_L r(\gamma,\eta), \kappa_L\theta_k)).
\end{aligned}$$

We obtain,

$$\begin{aligned}
\sum_{t=K_T}^{\infty} \mathbb{P}\left(d^+(\hat{v}_{k,s}, \kappa_L\theta_L) < \frac{d(\kappa_L\theta_k, \kappa_L\theta_L)}{1+\gamma}\right) &\leq \frac{\exp(-K_T d(\kappa_L r(\gamma,\eta), \kappa_L\theta_k))}{1 - \exp(-d(\kappa_L r(\gamma,\eta), \kappa_L\theta_k))} \\
&\leq \frac{C_2(\gamma,\eta)}{T^{\beta(\gamma,\eta)}},
\end{aligned}$$

for well chosen $C_2(\gamma,\eta)$ and $\beta(\gamma,\eta)$. □

4.A.5 *Lemmas*

In this section, we recall two necessary concentration lemmas directly adapted from Lemma 4 and 5 in Appendix A of [27]. Although more involved from a probabilistic point of view, these results are simpler to establish than proposition 4.2 as their adaptation to the case of the PBM relies on a crude lower bound for $\tilde{N}_k(t)$, which is sufficient for proving Theorem 4.4..

Lemma 4.8 *For $k \in \{1, \dots, K\}$ consider the martingale $M_{k,n} = \sum_{i=1}^{n} \bar{Z}_{k,i}$, where $\bar{Z}_{k,i}$ is defined in (4.15). Consider ϕ a stopping time such that either $N_k(\phi) \geq s$ or $\phi = T + 1$. Then*

$$\mathbb{P}[|M_{k,N_k(\phi)}| \geq N_k(\phi)\eta, N_k(\phi) \geq s] \leq 2\exp(-2s\eta^2). \tag{4.16}$$

As a consequence,

$$\mathbb{P}[|\hat{\theta}_k(\phi) - \theta_k| \geq \eta, \ \phi \leq T] \leq 2\exp(-2s\kappa_L^2\eta^2). \tag{4.17}$$

Proof *The first result is a direct application of Lemma 4 of [27] as $(Z_l(t))_t$ with $Z_l(t) = X_l(t)Y_l(t)$ is an independent sequence of $[0,1]$-valued variables.*

For the second inequality, we use the fact that $\tilde{N}_k(t) \geq \kappa_L N_k(t)$. Hence,

$$\mathbb{P}[|\hat{\theta}_k(\phi) - \theta_k| \geq \eta, \ \phi \leq T] \leq \mathbb{P}\left[\frac{|M_{k,N_k(\phi)}|}{\kappa_L N_k(\phi)} \geq \eta, \ \phi \leq T\right].$$

which is upper bounded using (4.16). □

Lemma 4.9 *Fix $c > 0$ and $k \in \{1, \dots, K\}$. Consider a random set of rounds $H \subset \mathbb{N}$, such that, for all t, $\mathbb{1}\{t \in H\}$ is $\mathcal{F}_{t-1}$ measurable and such that for all $t \in H$, $\{k \in \mathcal{B}(t)\}$ is true. Further assume, for all t, one has $\mathbb{E}[\mathbb{1}\{k \in A(t)\} | t \in H] \geq c > 0$. We define τ_s a stopping time such that $\sum_{t=1}^{\tau_s} \mathbb{1}\{t \in H\} \geq s$. Consider the random set $\Lambda = \{\tau_s : s \geq 1\}$. Then, for all k,*

$$\sum_{t \geq 0} \mathbb{P}[t \in \Lambda, |\hat{\theta}_k(t) - \theta_k| \geq \eta] \leq 2c^{-1}(2c^{-1} + \kappa_L^{-2}\eta^{-2})$$

The proof of this lemma follows that of Lemma 5 in [27] using the same lower bound for $\tilde{N}_k(t)$ as above.

5

FINAL WORDS AND PERSPECTIVES

In this dissertation, we studied adaptive strategies that rely on the many types of feedback generated in user-centric applications to improve recommendation, and more generally the user experience in information access. We identified three important applications in which users constantly produce signals that can be incorporated in algorithms to deliver better services for answering information needs.

Concretely, in Chapter 2 we developed TOPKS-ASYT, an as-you-type algorithm to search on social media. The approach allows to improve user experimence on social platform by providing a list of *potentially* interesting results to a user who is typing a query. Our algorithm *adaptively* outputs user-centric recommendations under a network-aware query model by which information produced by users who are closer to the seeker can be given more weight. We introduced a novel trie data structure, Index, allowing ranked access over inverted lists to provide answers rapidly in strongly dynamic situations as the user is typing his / her query. We gave two extensions to our algorithm: (i) we gave an incremental version of TOPKS-ASYT that takes advantage of computations performed for previously typed letters so as to speed up subsequent computations, (ii) we proposed an anytime version of TOPKS-ASYT that allows to output the most likely answer within any chosen time limit.

In Chapter 3, we proposed a diffusion-independent approach for online and adaptive influence maximization. Our algorithm, called GT-UCB, maximizes the number people reached throughout a campaign (be it in politics, marketing, etc.). Our approach sequentially selects people chosen from a subset of the population and from whom spreads of diffusion are initiated. It requires as only interfaces with the "real-world" the identification of potentially influential people and the spread feedback at each trial. Unlike its competitors, GT-UCB is very fast in estimating the *remaining* value of each influencer, which is a major concern when dealing with short campaigns of tens to hundreds of spreads. We also described an extension that incorporates the diminuishing convertion impact as an influential user keeps promoting the same piece of information to his / her followers.

In Chapter 4, we studied the position-based model (PBM) – a click model particularly relevant in recommendation scenarios - in an online context. Through this adaptive approach, we focused on cold start situations where the recommender system has no knowledge about newly introduced items. For example, after a film is released, recommender systems need to gather feedback in order to evaluate properly how much users appreciate it. We provided a lower on the regret for the PBM bandit instance. The tightness of the lower bound was validated by the analysis of our proposed algorithm PBM-PIE.

We leave behind several interesting questions that deserve further research. In Chapter 2, we introduced a parameter (which we denoted α) that allows the as-you-type system

to specify how much social bias is included in the results displayed to query users. However, choosing the appropriate value for α is difficult because, on social platforms, some queries are social / subjective (e.g., searching for a restaurant for which friends wrote positive reviews), whereas some others are global / objective (e.g., searching for the official page of a famous user on the social platform). Incorporating user feedback in adaptive algorithms in order to learn the best value of α for each query topic is an important research direction for improving the user experience for accessing information. In Chapter 3, we proposed an adaptive algorithm for semi-bandit influence maximization which uses the Poisson Kullback-Leibler divergence. We showed that the algorithm performs well empirically by comparing our approach to the state-of-the-art algorithm. However, we have not provided a theoretical analysis yet, which would allow us to guarantee formally its superiority over competitor algorithms. In Chapter 4, even though we provided a theoretical analysis to PBM-PIE, we let open the question of the asymptotical optimality of PBM-TS, despite strong empirical evidences. Furthermore, our framework assumes the knowledge of the position bias probabilities. In practice, we observed that the studied algorithms are robust to small variations of these values, but the proposal of an algorithm that is unaware of these parameters would be an interesting step forward. In this direction, Katariya et al. [61] study a bandit problem where, at each step, the learner selects a pair of row and columns arms, from a rank-1 matrix, and observes the product of their Bernoulli random variables. They make no assumption on neither row nor column parameters. Nevertheless, the study of the PBM setting without the knowlege of position parameters remains an open question today.

A

APPENDIX

A.1 ONLINE MAXIMIZATION C++ PACKAGE

The work on online influence maximization with persistence presented in Chapter 3 led to the development of a software package in C++ available at `https://github.com/plagree/oim`. Note that the code is a fork of a project initially developed by Siyu Lei, Silviu Maniu and Luyi Mo from University of Hong Kong for their paper published at KDD 2015 [73]. The source code is provided as-in under the MIT License.

COMPILING. The *Makefile* is in the main folder and requires GCC 4.9.0 (or superior) as it uses C++14 features. The code needs Boost C++ library headers. It assumes the include files are present in `/usr/local/include`. If your Boost installation is someplace else, you have to modify the `INCLUDE_DIRS` directive in Makefile. The binary library does not need to be linked.

Compiling is as easy as:

```
# make clean; make
```

The output binary is `oim`.

METHODS AND USAGE. The program expects as input a tab delimited graph file of the following format:

```
node1 <TAB> node2 <TAB> prob
```

where `node1` and `node2` are the endpoints of a graph edge, and prob is the influence probability.

The following methods are supported:

1. *exponentiated gradient* [73], which runs as follows:

```
./oim -eg <graph> <alpha> <beta> <exploit> <trials> <L> [<model> <update>
<update_type> <cascades>]
```

2. *missing mass* [70], which runs as follows:

```
./oim -missing_mass <graph> <policy> <reduction> <trials> <L> <n_experts>
[<model> <cascades>]
```

3. *real graph*, which runs on the real diffusion graph – this corresponds to the ORACLE:

```
./oim -real <graph> <exploit> <trials> <L> [<model> <samples> <cascades>]
```

PARAMETERS. The parameters of the implemented methods are set as follows:

- `graph` is the name of the graph file,
- `alpha`, `beta` are the global prior on the edges of the graph,
- `exploit` can take any of the following values: 0 Random, 1 AdaptiveDegree, 2 Maxdegree, 3 CELF [47], 4 TIM [110], 5 SSA [94], 6 PMC [95],
- `samples` is the number of spreads to estimate the expected value of chosen seeds,
- `trials` is the number of trials N, `L` is the number of seeds in each trial,
- `update` is 1 if the graph is updated, 0 otherwise,
- `update_type` is the type of update: 0 local only, 1 least squares or 2 maximum likelihood,
- `reduction` can take the following values: 0 max cover, 1 highest degree, 2 DIVRANK,
- `policy` can take the following values: 0 Random, 1 GT-UCB,
- `model` can take the following values: 0 Linear Threshold, 1 Independent Cascade,
- `cascades` contains the path to the file containing real cascades (logs).

OUTPUT. The different methods write on the standard output with the following format:

1. *exponentiated gradient*:

```
stage <TAB> cum spread <TAB> expected spread <TAB> tselection <TAB> tupdate
<TAB> tround <TAB> ttotal <TAB> theta <TAB> memory <TAB> k <TAB> model
<TAB> seeds
```

2. *missing mass*:

```
stage <TAB> cumulative spread <TAB> treduction <TAB> tselection <TAB>
tupdate <TAB> tround <TAB> ttotal <TAB> memory <TAB> k <TAB> n_experts
<TAB> n_policy <TAB> n_reduction <TAB> model <TAB> seeds
```

3. *real graph*:

```
stage <TAB> cumulative spread <TAB> expected spread <TAB> tround <TAB>
ttotal <TAB> k <TAB> model <TAB> seeds
```

A.2 ANALYSIS OF TWEET LOGS

In Chapter 3, we conducted a series of experiments on data collected in August 2012 using Twitter streaming API. We extracted cascades in which the original author of a tweet is the seed who initated a spread composed of all users who retweeted the post. The resulting dataset contains 50, 537, 745 users among whom 32, 971, 976 have never been retweeted, that is, have potentially posted on their timeline but none of their posts were retweeted[1]. Not surprisingly, only 87, 940, 277 of the 726, 474, 937 total tweets have been retweeted. Intuitively, we believe that, since most users are non-influential, their posts are generally never reposted leading to all these isolated posts.

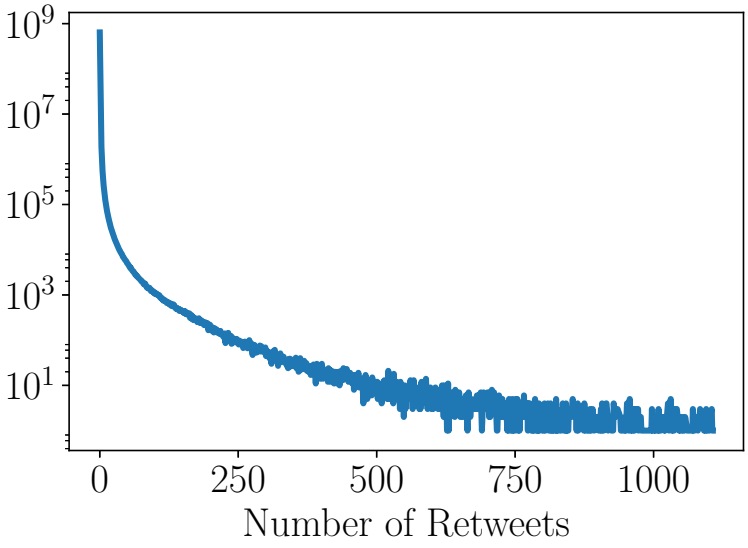


Figure A.1: Histogram of the number of retweets

In Fig. A.1, we show the histogram of the number of retweets by posts. We observe a power law distribution which is in line with many studies on social networks. In Table A.1, we provide the values from Fig. A.1 for tweets that were not retweeted much (less than 10 times). We observe that a very large majority of Twitter posts is never retweeted.

A.2.1 *Edge weights*

In Section 3.3.2.1, we conducted an analysis of the empirical activation probabilities to verify our claim that influencers (the candidates), despite being connected to many users, can barely activate most of them. Specifically, for every user v "influenced" by u, i.e., v retweeted at least one original tweet from u, we computed the estimated diffusion probabilities by

$$p_{u,v} = \frac{|u\text{'s tweets retweeted by } v|}{|\text{tweets by } u|}.$$

1 Note that we only have a small portion of Twitter traffic in August 2012. Thus, these non-influential users may have actually been retweeted without appearing in our dataset.

Table A.1: Values from Fig. A.1 for tweets with few retweets

Number retweets	Corresponding tweets
0	$638M$
1	$76M$
2	$6.2M$
3	$1.8M$
4	$915K$
5	$545K$
6	$363K$
7	$259K$
8	$194K$
9	$152K$
10	$124K$

We only computed these empirical diffusion probabilities when the source – the influencer u in the formula above – wrote at least 10 tweets in the entire dataset.

We obtained a set of 113, 375, 255 edges that involve a total of 22, 188, 987 distinct users[2]. The resulting graph is called the "empirical diffusion graph". In Fig. A.2a, we show the edge outdegree histogram of this reconstructed graph. We observe another power-law distribution in which most users have influenced almost no other users, that is, their tweets are hardly reposted by others. Interestingly, the right outlier – we found out the corresponding user is Justin Bieber – influenced more than 200, 000 people in our collection of tweets. Similarly, in Fig. A.2b, we display the edge indegree histogram of the empirical diffusion graph. We obtain a very regular power-law distribution with most values under 500 which is not very surprising as most users are very unlikely to repost content from more than a few tens or hundreds different other users.

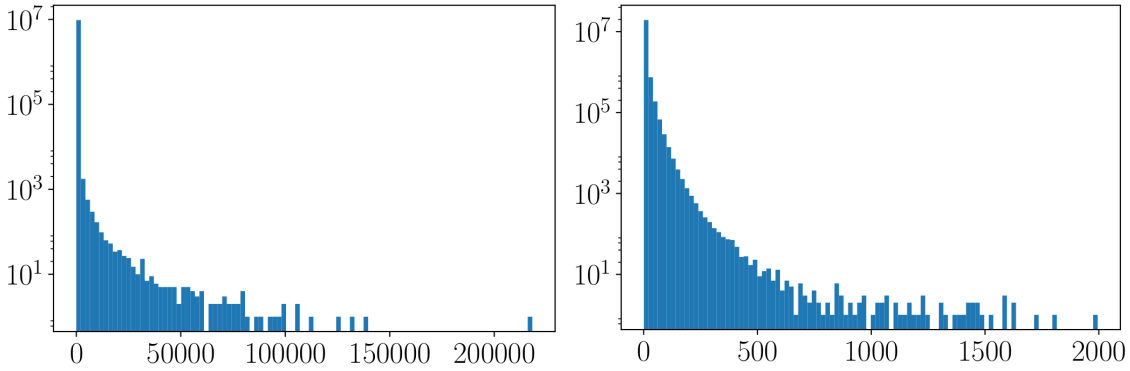
In Table A.2, we show the deciles of the empirical diffusion probabilities. Most probabilities are (very) small – the last decile has value 0.045 –, which is in line with our initial assumption that most nodes connected to an influencer have low activation probabilities.

10%	20%	30%	40%	50%	60%	70%	80%	90%
0.0012	0.0022	0.0035	0.0052	0.0075	0.0110	0.0164	0.0256	0.0455

Table A.2: Deciles of empirical diffusion probabilities

Our algorithm, GT-UCB, is given a set of candidates (e.g., influential users) and has to choose spread seeds among them at each step. Importantly, we assumed that the candi-

2 Note that around 30 edges had an empirical weight > 1 because some users retweeted several times the same posts thus leading to these unexpected numbers. Consequently, for each of these values, we thresholded at 1.



(a) Outdegree histogram (b) Indegree histogram

Figure A.2: Degree histogram of the empirical diffusion graph.

dates' support is non-intersecting to simplify the derivation of confidence bounds. Furthermore, this proves to be useful to avoid activating the same users via different candidates, which yields no additional reward in the persistence setting. After having extracted the set of candidates – e.g., using the DIVRANK criterion –, we computed their pairwise support intersection to verify that they activate mostly different users. Formally, we computed the Dice and Jaccard indices for every pair of candidates to measure the similarity of their support. We show the resulting matrices in Fig. A.3. We see that, for any two candidates chosen among the set of influencers, they activated almost no users in common in our dataset.

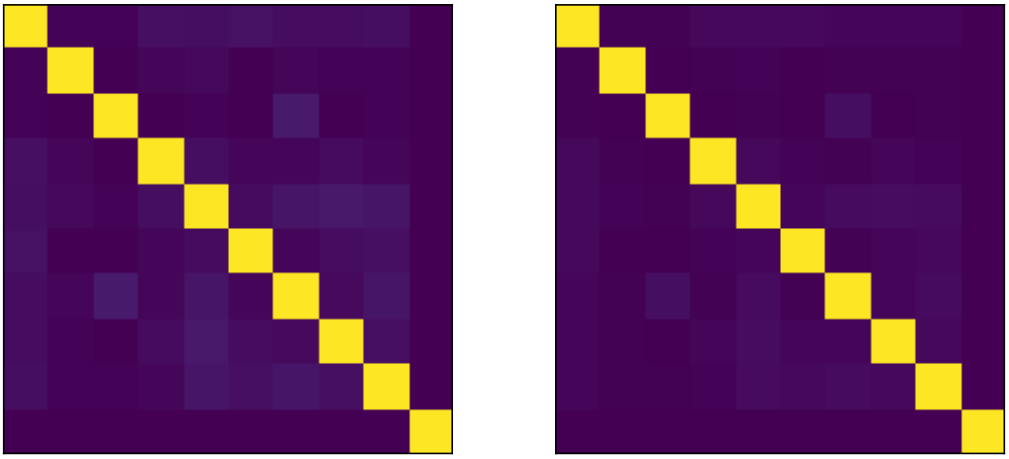


Figure A.3: Dice (left) and Jaccard (right) matrices for 10 candidates

A.2.2 *Graph reconstruction*

From the retweeting cascades, we applied the algorithm NPDC from [101] (code provided by the authors), which is a model-free approach to infer the underlying network, based solely on cascades. This method follows a simple yet effective idea, which is that, in cas-

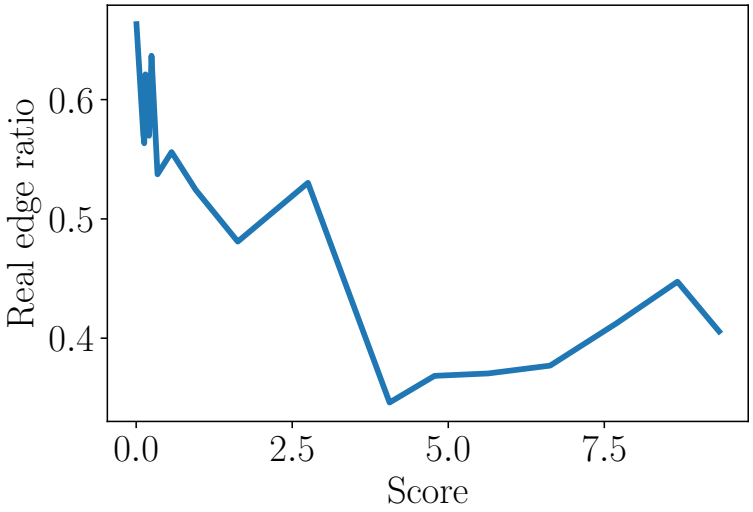


Figure A.4: Evolution of the real edge ratio against NPDC score.

cades, the "footprint" of pairs of users who are truly connected in the diffusion network should be *statistically distinguishable* from the one of pairs who are not connected.

For each pair of users, NPDC returns a positive score – the lower the score, the more likely the two users are truly connected in the underlying graph. To verify the quality of NPDC predictions, we computed the (true) edge ratio based on 500 edges and for different scores. More precisely, we sorted edges by increasing scores and computed the real edge ration for 20 different scores (e.g., the 500 edges with the lowest score, then 500 edges whose score are around the 5th percentile, etc.). The ratio formula is given by

$$\text{real edge ratio} = \frac{\#\text{ real edges}}{500},$$

where "real" edges – the ground truth – are given by Twitter API. We show the resulting curve in Fig. A.4.

A.2.3 *Why GT-UCB performs well*

We counted the number of times each candidate is played when GT-UCB is run on Twitter logs (experiments in Section 3.5.3). Note that we run 20 times the experiment with a horizon of 500 steps so as to improve results consistency. We show the values obtained of each of the 10 candidates in Table A.3.

Table A.3: Number of times each candidate has been selected after 500 steps.

Candidate	1	2	3	4	5	6	7	8	9	10
Number of plays	2,877	2,680	1,747	1,049	688	317	314	156	117	55

Interestingly, the algorithm can discard "bad" influencers (e.g. candidate 10) and naturally focuses on "good" candidates that are very influential. This is of strong interest in the

area of influencer marketing if a marketing firm sponsors influencers whose fame greatly changes from one to another.

BIBLIOGRAPHY

[1] Rajeev Agrawal. "Sample Mean Based Index Policies with O(log n) Regret for the Multi-Armed Bandit Problem." In: *Advances in Applied Probability* 27.4 (1995), pp. 1054–1078 (cit. on p. 8).

[2] Shipra Agrawal and Navin Goyal. "Analysis of Thompson Sampling for the Multi-armed Bandit Problem." In: *COLT*. Ed. by Shie Mannor, Nathan Srebro, and Robert C. Williamson. Vol. 23. JMLR Proceedings. 2012, pp. 39.1–39.26 (cit. on p. 8).

[3] Kumaripaba Ahukorala, Alan Medlar, Kalle Ilves, and Dorota Glowacka. "Balancing Exploration and Exploitation: Empirical Parameterization of Exploratory Search Systems." In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15. Melbourne, Australia, 2015, pp. 1703–1706 (cit. on p. 18).

[4] Venkatachalam Anantharam, Pravin Varaiya, and Jean Walrand. "Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays - Part I: IID rewards." In: *Automatic Control, IEEE Transactions on* 32.11 (1987), pp. 968–976 (cit. on p. 87).

[5] Akhil Arora, Sainyam Galhotra, and Sayan Ranu. "Debunking the Myths of Influence Maximization: An In-Depth Benchmarking Study." In: *SIGMOD*. SIGMOD '17. ACM, 2017 (cit. on pp. 46, 53, 54, 70).

[6] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. "Finite-time Analysis of the Multiarmed Bandit Problem." In: *Mach. Learn.* 47.2-3 (May 2002), pp. 235–256 (cit. on p. 8).

[7] Bahman Bahmani and Ashish Goel. "Partitioned Multi-indexing: Bringing Order to Social Search." In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12. Lyon, France: ACM, 2012, pp. 399–408. ISBN: 978-1-4503-1229-5 (cit. on p. 18).

[8] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. "Topic-aware social influence propagation models." In: *Knowl. Inf. Syst.* 37.3 (2013), pp. 555–584 (cit. on p. 55).

[9] Holger Bast, Christian W. Mortensen, and Ingmar Weber. "Output-sensitive Autocompletion Search." In: *Inf. Retr.* 11.4 (Aug. 2008), pp. 269–286. ISSN: 1386-4564 (cit. on p. 18).

[10] Holger Bast and Ingmar Weber. "Type Less, Find More: Fast Autocompletion Search with a Succinct Index." In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '06. Seattle, Washington, USA: ACM, 2006, pp. 364–371. ISBN: 1-59593-369-7 (cit. on p. 18).

[11] Daniel Berend and Aryeh Kontorovich. "On the Concentration of the Missing Mass." In: *Electronic Communications in Probability*. 2013, pp. 1–7 (cit. on pp. 65, 77).

[12] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. "Maximizing Social Influence in Nearly Optimal Time." In: *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '14. 2014, pp. 946–957 (cit. on p. 53).

[13] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. OUP Oxford, 2013 (cit. on p. 99).

[14] Stéphane Boucheron, Gábor Lugosi, Pascal Massart, and Michel Ledoux. *Concentration inequalities : a nonasymptotic theory of independence*. Oxford U. press, 2013 (cit. on pp. 65, 76).

[15] In: *Influencer Marketing*. Ed. by Duncan Brown and Nick Hayes. Oxford: Butterworth-Heinemann, 2008 (cit. on pp. 4, 74).

[16] Sébastien Bubeck and Nicolò Cesa-Bianchi. "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems." In: *Foundations and Trends in Machine Learning* 5.1 (2012), pp. 1–122 (cit. on p. 6).

[17] Sébastien Bubeck, Damien Ernst, and Aurélien Garivier. "Optimal discovery with probabilistic expert advice: finite time analysis and macroscopic optimality." In: *Journal of Machine Learning Research* 14.1 (2013), pp. 601–623 (cit. on pp. 5, 47, 60, 62, 65, 78, 79).

[18] Fei Cai, Shangsong Liang, and Maarten de Rijke. "Time-sensitive Personalized Query Auto-Completion." In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. CIKM '14. Shanghai, China: ACM, 2014, pp. 1599–1608. ISBN: 978-1-4503-2598-1 (cit. on p. 19).

[19] Yi Chang, Lei Tang, Yoshiyuki Inagaki, and Yan Liu. "What is Tumblr: A Statistical Overview and Comparison." In: *SIGKDD Explor. Newsl.* 16.1 (Sept. 2014), pp. 21–29. ISSN: 1931-0145 (cit. on p. 20).

[20] Olivier Chapelle and Ya Zhang. "A Dynamic Bayesian Network Click Model for Web Search Ranking." In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. 2009, pp. 1–10 (cit. on p. 84).

[21] Wei Chen, Yajun Wang, and Yang Yuan. "Combinatorial multi-armed bandit: General framework and applications." In: *Proc. of the 30th Int. Conf. on Machine Learning*. 2013 (cit. on p. 90).

[22] Wei Chen, Yifei Yuan, and Li Zhang. "Scalable Influence Maximization in Social Networks Under the Linear Threshold Model." In: *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM '10. 2010, pp. 88–97 (cit. on p. 51).

[23] Wei Chen, Yajun Wang, Yang Yuan, and Qinshi Wang. "Combinatorial Multi-armed Bandit and Its Extension to Probabilistically Triggered Arms." In: *Journal of Machine Learning Research (JMLR)* 17.1 (Jan. 2016), pp. 1746–1778 (cit. on pp. 46, 55, 57, 71, 72).

[24] Wei Chen, Tian Lin, Zihan Tan, Mingfei Zhao, and Xuren Zhou. "Robust Influence Maximization." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 2016, pp. 795–804 (cit. on p. 55).

[25] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. "Click Models for Web Search." In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 7.3 (2015), pp. 1–115 (cit. on pp. 82, 83, 86, 93).

[26] Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutière, et al. "Combinatorial Bandits Revisited." In: *Advances in Neural Information Processing Systems*. 2015 (cit. on pp. 87, 90).

[27] Richard Combes, Stefan Magureanu, Alexandre Proutière, and Cyrille Laroche. "Learning to rank: Regret lower bounds and efficient algorithms." In: *Proc. of the 2015 ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*. 2015 (cit. on pp. 6, 82, 84, 88–90, 97, 99, 103, 104).

[28] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. "An experimental comparison of click position-bias models." In: *Proc. of the Int. Conf. on Web Search and Data Mining*. ACM. 2008 (cit. on pp. 5, 82, 83, 86).

[29] Michael Curtiss et al. "Unicorn: A System for Searching the Social Graph." In: *Proc. VLDB Endow.* 6.11 (Aug. 2013), pp. 1150–1161. ISSN: 2150-8097 (cit. on p. 18).

[30] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters." In: *Commun. ACM* 51.1 (Jan. 2008), pp. 107–113 (cit. on p. 2).

[31] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm." In: *Journal of the royal statistical society. Series B* (1977), pp. 1–38 (cit. on pp. 86, 93).

[32] Jay L. Devore. *Probability and Statistics for Engineering and the Sciences*. 8th. ISBN-13: 978-0-538-73352-6. Brooks/Cole, 2011 (cit. on p. 8).

[33] Darcy DiNucci. "Fragmented future." In: *Print* 53.4 (1999), p. 32 (cit. on p. 1).

[34] Nan Du, Le Song, Hyenkyun Woo, and Hongyuan Zha. "Uncover Topic-Sensitive Information Diffusion Networks." In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS, Scottsdale, AZ, USA, April 29 - May 1*. 2013, pp. 229–237 (cit. on pp. 46, 47, 55).

[35] Georges E. Dupret and Benjamin Piwowarski. "A User Browsing Model to Predict Search Engine Click Data from Past Observations." In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. 2008, pp. 331–338 (cit. on p. 85).

[36] David A. Easley and Jon M. Kleinberg. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World*. Cambridge University Press, 2010 (cit. on p. 46).

[37] Pavlos Fafalios and Yannis Tzitzikas. "Type-ahead Exploratory Search Through Typo and Word Order Tolerant Autocompletion." In: *J. Web Eng.* 14.1-2 (Mar. 2015), pp. 80–116 (cit. on p. 19).

[38] Ronald Fagin, Amnon Lotem, and Moni Naor. "Optimal Aggregation Algorithms for Middleware." In: *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '01. Santa Barbara, California, USA: ACM, 2001, pp. 102–113. ISBN: 1-58113-361-8 (cit. on pp. 17, 18, 30).

[39] Jianhua Feng and Guoliang Li. "Efficient Fuzzy Type-Ahead Search in XML Data." In: *IEEE Transactions on Knowledge and Data Engineering* 24.5 (2012), pp. 882–895. ISSN: 1041-4347 (cit. on p. 19).

[40] Aurélien Garivier and Olivier Cappé. "The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond." In: *Proc. of the Conf. on Learning Theory*. 2011 (cit. on pp. 9, 73, 84, 90, 92, 102).

[41] Jacob Goldenberg, Barak Libai, and Eitan Muller. "Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth." In: *Marketing Letters* 12.3 (2001), pp. 211–223 (cit. on p. 48).

[42] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. "Uncovering the Temporal Dynamics of Diffusion Networks." In: *Proceedings of the 28th International Conference on Machine Learning, ICML, Bellevue, Washington, USA, June 28 - July 2*. 2011, pp. 561–568 (cit. on p. 46).

[43] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. "Inferring Networks of Diffusion and Influence." In: *ACM Trans. Knowl. Discov. Data* 5.4 (Feb. 2012), 21:1–21:37 (cit. on p. 46).

[44] Manuel Gomez-Rodriguez, Jure Leskovec, and Bernhard Schölkopf. "Structure and dynamics of information pathways in online media." In: *Sixth ACM International Conference on Web Search and Data Mining, WSDM, Rome, Italy, February 4-8*. 2013, pp. 23–32 (cit. on p. 46).

[45] I. J. Good. "The Population Frequencies of Species and the Estimation of Population Parameters." In: *Biometrika* 40.3-4 (1953), p. 237 (cit. on pp. 47, 60).

[46] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. "Learning influence probabilities in social networks." In: *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM, New York, NY, USA, February 4-6*. 2010, pp. 241–250 (cit. on p. 46).

[47] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. "CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks." In: *Proceedings of the 20th International Conference Companion on World Wide Web*. WWW '11. 2011, pp. 47–48 (cit. on pp. 53, 108).

[48] Przemyslaw A. Grabowicz, Niloy Ganguly, and Krishna P. Gummadi. "Distinguishing between Topical and Non-Topical Information Diffusion Mechanisms in Social Media." In: *Proceedings of the Tenth International Conference on Web and Social Media, Cologne, Germany, May 17-20*. 2016, pp. 151–160 (cit. on p. 47).

[49] M. Granovetter. "Threshold Models of Collective Behavior." In: *The American Journal of Sociology* 83.6 (1978), pp. 1420–1443 (cit. on p. 49).

[50] Todd L Graves and Tze Leung Lai. "Asymptotically efficient adaptive choice of control laws in controlled markov chains." In: *SIAM journal on control and optimization* 35.3 (1997), pp. 715–743 (cit. on pp. 87, 88).

[51] Fan Guo, Chao Liu, and Yi Min Wang. "Efficient Multiple-click Models in Web Search." In: *Proceedings of the Second ACM International Conference on Web Search and Data Mining*. WSDM '09. New York, NY, USA, 2009, pp. 124–131 (cit. on pp. 84, 85).

[52] Xinran He and David Kempe. "Robust Influence Maximization." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17*. 2016, pp. 885–894 (cit. on p. 55).

[53] Wing-Kai Hon, Rahul Shah, and Jeffrey Scott Vitter. "Space-Efficient Framework for Top-k String Retrieval Problems." In: *50th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '09. Atlanta, Georgia, USA, 2009, pp. 713–722 (cit. on p. 19).

[54] Bo-June (Paul) Hsu and Giuseppe Ottaviano. "Space-efficient Data Structures for Top-k Completion." In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, 2013, pp. 583–594. ISBN: 978-1-4503-2035-1 (cit. on pp. 15, 19, 30).

[55] Keke Huang, Sibo Wang, Glenn Bevilacqua, Xiaokui Xiao, and Laks V. S. Lakshmanan. "Revisiting the Stop-and-stare Algorithms for Influence Maximization." In: *Proc. VLDB Endow.* 10.9 (May 2017), pp. 913–924 (cit. on p. 54).

[56] Glen Jeh and Jennifer Widom. "SimRank: A Measure of Structural-context Similarity." In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '02. Edmonton, Alberta, Canada: ACM, 2002, pp. 538–543. ISBN: 1-58113-567-X. DOI: 10.1145/775047.775126 (cit. on p. 38).

[57] Shengyue Ji, Guoliang Li, Chen Li, and Jianhua Feng. "Efficient Interactive Fuzzy Keyword Search." In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: ACM, 2009, pp. 371–380. ISBN: 978-1-60558-487-4 (cit. on p. 18).

[58] Di Jiang, Kenneth Wai-Ting Leung, Jan Vosecky, and Wilfred Ng. "Personalized Query Suggestion With Diversity Awareness." In: *IEEE 30th International Conference on Data Engineering*. ICDE '14. Chicago, IL, USA, 2014, pp. 400–411 (cit. on p. 19).

[59] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. "Accurately Interpreting Clickthrough Data As Implicit Feedback." In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '05. 2005, pp. 154–161 (cit. on p. 83).

[60] *KDD Cup 2012 track 2*. http://www.kddcup2012.org/ (cit. on p. 93).

[61] Sumeet Katariya, Branislav Kveton, Csaba Szepesvári, Claire Vernade, and Zheng Wen. "Stochastic Rank-1 Bandits." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. 2017, pp. 392–401 (cit. on pp. 94, 106).

[62] Leo Katz. "A new status index derived from sociometric analysis." In: *Psychometrika* 18.1 (1953), pp. 39–43. ISSN: 0033-3123 (cit. on p. 21).

[63] Émilie Kaufmann, Olivier Cappé, and Aurélien Garivier. "On the Complexity of Best Arm Identification in Multi-Armed Bandit Models." In: *Journal of Machine Learning Research* (2015) (cit. on pp. 87, 88, 96).

[64] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. "Thompson Sampling: An Asymptotically Optimal Finite-Time Analysis." In: *Algorithmic Learning Theory: 23rd International Conference, ALT 2012, Lyon, France, October 29-31, 2012. Proceedings*. Ed. by Nader H. Bshouty, Gilles Stoltz, Nicolas Vayatis, and Thomas Zeugmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 199–213 (cit. on p. 8).

[65] David Kempe, Jon Kleinberg, and Éva Tardos. "Maximizing the Spread of Influence Through a Social Network." In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '03. ACM, 2003, pp. 137–146 (cit. on pp. 46, 48–53).

[66] Maksim Kitsak, Lazaros K. Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H. Eugene Stanley, and Hernán A. Makse. "Identification of influential spreaders in complex networks." In: *Nature Physics* (2010) (cit. on p. 55).

[67] Junpei Komiyama, Junya Honda, and Hiroshi Nakagawa. "Optimal Regret Analysis of Thompson Sampling in Stochastic Multi-armed Bandit Problem with Multiple Plays." In: *Proc. of the 32nd Int. Conf. on Machine Learning*. 2015 (cit. on pp. 82, 87, 92).

[68] Branislav Kveton, Csaba Szepesvári, Zheng Wen, and Azin Ashkan. "Cascading Bandits : Learning to Rank in the Cascade Model." In: *Proc. of the 32nd Int. Conf. on Machine Learning*. 2015 (cit. on pp. 82, 84).

[69] Paul Lagrée, Bogdan Cautis, and Hossein Vahabi. "A Network-Aware Approach for Searching As-You-Type in Social Media." In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. CIKM '15. 2015 (cit. on pp. 4, 13).

[70] Paul Lagrée, Bogdan Cautis, and Hossein Vahabi. "As-You-Type Social Aware Search." In: *ACM Transactions on Intelligent Systems and Technology* 8.5 (June 2017), 63:1–63:31 (cit. on pp. 4, 13, 107).

[71] Paul Lagrée, Claire Vernade, and Olivier Cappé. "Multiple-Play Bandits in the Position-Based Model." In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 1597–1605 (cit. on pp. 6, 81, 90).

[72] Tze Leung Lai and Herbert Robbins. "Asymptotically efficient adaptive allocation rules." In: *Advances in applied mathematics* 6.1 (1985), pp. 4–22 (cit. on pp. 7, 87).

[73] Siyu Lei, Silviu Maniu, Luyi Mo, Reynold Cheng, and Pierre Senellart. "Online Influence Maximization." In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. 2015 (cit. on pp. 47, 57–59, 61, 67, 68, 107).

[74] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. "Cost-effective Outbreak Detection in Networks." In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '07. 2007, pp. 420–429 (cit. on p. 53).

[75] Nir Levine, Koby Crammer, and Shie Mannor. "Rotting Bandits." In: *Working paper*. 2017 (cit. on p. 74).

[76] Guoliang Li, Jianhua Feng, and Chen Li. "Supporting Search-As-You-Type Using SQL in Databases." In: *IEEE Trans. on Knowl. and Data Eng.* 25.2 (2013), pp. 461–475 (cit. on p. 19).

[77] Guoliang Li, Shengyue Ji, Chen Li, Jiannan Wang, and Jianhua Feng. "Efficient fuzzy type-ahead search in TASTIER." In: *Proceedings of the 26th International Conference on Data Engineering*. ICDE '10. Long Beach, California, USA, 2010, pp. 1105–1108 (cit. on p. 18).

[78] Guoliang Li, Jiannan Wang, Chen Li, and Jianhua Feng. "Supporting Efficient Top-k Queries in Type-ahead Search." In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '12. Portland, Oregon, USA: ACM, 2012, pp. 355–364. ISBN: 978-1-4503-1472-5 (cit. on pp. 15, 18, 29, 40, 41).

[79] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. "A Contextual-bandit Approach to Personalized News Article Recommendation." In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. 2010, pp. 661–670 (cit. on p. 56).

[80] Xiang Li, J. David Smith, Thang N. Dinh, and My T. Thai. "Why approximate when you can get the exact? Optimal Targeted Viral Marketing at Scale." In: (2017) (cit. on p. 51).

[81] Yuchen Li, Zhifeng Bao, Guoliang Li, and Kian-Lee Tan. "Real time personalized search on social networks." In: *31st IEEE International Conference on Data Engineering, ICDE*. 2015 (cit. on p. 18).

[82] Thomas M. Liggett. *Interacting Particle Systems*. Springer Berlin Heidelberg, 1985 (cit. on p. 48).

[83] Jonathan Louëdec, Laurent Rossi, Max Chevalier, Aurélien Garivier, and Josiane Mothe. "Algorithme de bandit et obsolescence : un modèle pour la recommandation." Marseille, FR, 2016 (cit. on p. 74).

[84] Wei Lu, Xiaokui Xiao, Amit Goyal, Keke Huang, and Laks V.S. Lakshmanan. "Refutations on "Debunking the Myths of Influence Maximization: An In-Depth Benchmarking Study"." In: *Working paper*. 2017 (cit. on p. 54).

[85] Stefan Magureanu, Richard Combes, and Alexandre Proutière. "Lipschitz Bandits: Regret Lower Bounds and Optimal Algorithms." In: *Proc. of the Conf. on Learning Theory*. 2014 (cit. on pp. 90, 100).

[86] Fragkiskos D. Malliaros, Maria-Evgenia G. Rossi, and Michalis Vazirgiannis. "Locating influential nodes in complex networks." In: *Scientific Reports* (2016) (cit. on p. 55).

[87] Silviu Maniu and Bogdan Cautis. "Taagle: Efficient, Personalized Search in Collaborative Tagging Networks." In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. SIGMOD '12. Scottsdale, Arizona, USA: ACM, 2012, pp. 661–664. ISBN: 978-1-4503-1247-9 (cit. on p. 15).

[88] Silviu Maniu and Bogdan Cautis. "Network-aware Search in Social Tagging Applications: Instance Optimality Versus Efficiency." In: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. CIKM '13. San Francisco, California, USA: ACM, 2013, pp. 939–948. ISBN: 978-1-4503-2263-8 (cit. on pp. 3, 14, 15, 18, 19, 21, 26, 28, 32, 40, 42).

[89] David A. McAllester and Luis E. Ortiz. "Concentration Inequalities for the Missing Mass and for Histogram Rule Error." In: *Journal of Machine Learning Research* 4 (2003), pp. 895–911 (cit. on p. 64).

[90] David A. McAllester and Robert E. Schapire. "On the Convergence Rate of Good-Turing Estimators." In: *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory (COLT 2000), June 28 - July 1, 2000, Palo Alto, California*. 2000, pp. 1–6 (cit. on pp. 63, 64).

[91] Qiaozhu Mei, Jian Guo, and Dragomir Radev. "DivRank: The Interplay of Prestige and Diversity in Information Networks." In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. 2010 (cit. on p. 67).

[92] Rich Miller. *Estimate: Facebook Running 180,000 Servers*. http://www.datacenterknowledge.com/archives/2012/08/15/estimate-facebook-running-180000-servers/ (cit. on p. 2).

[93] G. L. Nemhauser, L. A. Wosley, and M. L. Fisher. "An analysis of approximations for maximizing submodular set functions." In: *Mathematical programming* 14 (1978), pp. 265–294 (cit. on p. 52).

[94] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. "Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks." In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. 2016 (cit. on pp. 53, 54, 108).

[95] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-Ichi Kawarabayashi. "Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations." In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. 2015 (cit. on pp. 53, 54, 108).

[96] Marco Pennacchiotti, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. "Making Your Interests Follow You on Twitter." In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. CIKM '12. Maui, Hawaii, USA: ACM, 2012, pp. 165–174. ISBN: 978-1-4503-1156-4 (cit. on p. 32).

[97] Michalis Potamias, Francesco Bonchi, Carlos Castillo, and Aristides Gionis. "Fast Shortest Path Distance Estimation in Large Networks." In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: ACM, 2009, pp. 867–876. ISBN: 978-1-60558-512-3 (cit. on p. 18).

[98] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. "Learning diverse rankings with multi-armed bandits." In: *Proc. of the 25th Int. Conf. on Machine learning*. ACM. 2008 (cit. on p. 92).

[99] Matthew Richardson, Ewa Dominowska, and Robert Ragno. "Predicting clicks: estimating the click-through rate for new ads." In: *Proc. of the 16th Int. Conf. on World Wide Web*. ACM. 2007 (cit. on p. 82).

[100] Daniel M. Romero, Brendan Meeder, and Jon Kleinberg. "Differences in the Mechanics of Information Diffusion Across Topics: Idioms, Political Hashtags, and Complex Contagion on Twitter." In: *Proceedings of the 20th International Conference on World Wide Web*. WWW '11. Hyderabad, India, 2011, pp. 695–704 (cit. on pp. 47, 55).

[101] Yu Rong, Qiankun Zhu, and Hong Cheng. "A Model-Free Approach to Infer the Diffusion Network from Event Cascade." In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. 2016 (cit. on pp. 55, 111).

[102] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. "Prediction of Information Diffusion Probabilities for Independent Cascade Model." In: *Knowledge-Based Intelligent Information and Engineering Systems, 12th International Conference, KES, Zagreb, Croatia, September 3-5, Proceedings, Part III*. 2008, pp. 67–75 (cit. on p. 46).

[103] Guillaume Salha, Nikolaos Tziortziotis, and Michalis Vazirgiannis. "Adaptive Submodular Influence Maximization with Myopic Feedback." In: *arXiv preprint 1704.06905* (2017) (cit. on p. 57).

[104] Thomas C. Schelling. *Micromotives and Macrobehavior*. W. W. Norton & Company, 1978 (cit. on p. 49).

[105] Ralf Schenkel, Tom Crecelius, Mouna Kacimi, Sebastian Michel, Thomas Neumann, Josiane X. Parreira, and Gerhard Weikum. "Efficient Top-k Querying over Social-tagging Networks." In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. Singapore, Singapore: ACM, 2008, pp. 523–530. ISBN: 978-1-60558-164-4 (cit. on pp. 14, 15, 18, 21, 28).

[106] Milad Shokouhi. "Learning to Personalize Query Auto-completion." In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '13. Dublin, Ireland: ACM, 2013, pp. 103–112. ISBN: 978-1-4503-2034-4 (cit. on p. 19).

[107] Milad Shokouhi and Kira Radinsky. "Time-sensitive Query Auto-completion." In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '12. Portland, Oregon, USA: ACM, 2012, pp. 601–610. ISBN: 978-1-4503-1472-5 (cit. on p. 19).

[108] Katariya Sumeet, Branislav Kveton, Csaba Szepesvári, and Zheng Wen. "DCM Bandits: Learning to Rank with Multiple Clicks." In: *Proc. of the 33rd Int. Conf. on Machine Learning*. 2016 (cit. on pp. 82, 84).

[109] Youze Tang, Yanchen Shi, and Xiaokui Xiao. "Influence Maximization in Near-Linear Time: A Martingale Approach." In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. 2015, pp. 1539–1554 (cit. on p. 54).

[110] Youze Tang, Xiaokui Xiao, and Yanchen Shi. "Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency." In: *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*. 2014, pp. 75–86 (cit. on pp. 53, 54, 108).

[111] William R Thompson. "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples." In: *Biometrika* (1933), pp. 285–294 (cit. on p. 7).

[112] Hossein Vahabi, Margareta Ackerman, David Loker, Ricardo Baeza-Yates, and Alejandro Lopez-Ortiz. "Orthogonal Query Recommendation." In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys '13. Hong Kong, China: ACM, 2013, pp. 33–40. ISBN: 978-1-4503-2409-0 (cit. on p. 19).

[113] S. Vaswani, B. Kveton, Z. Wen, M. Ghavamzadeh, L. Lakshmanan, and M. Schmidt. "Diffusion Independent Semi-Bandit Influence Maximization." In: *ICML*. 2017 (cit. on pp. 46, 55–57, 63, 73).

[114] Sharan Vaswani, V.S. Lakshmanan, and Mark Schmidt. "Influence Maximization with Bandits." In: *Workshop NIPS*. NIPS '15. 2015 (cit. on pp. 46, 55, 57–59, 71, 72).

[115] Ganesh Venkataraman, Abhimanyu Lad, Viet Ha-Thuc, and Dhruv Arya. "Instant Search: A Hands-on Tutorial." In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. 2016, pp. 1211–1214 (cit. on p. 18).

[116] Claire Vernade, Paul Lagrée, and Olivier Cappé. "Online Inference for Multiple-Item Display Under the Position-Based Model." In: *Workshop on Online Advertising Systems*. ICML. 2016 (cit. on pp. 6, 81).

[117] Chi Wang, Wei Chen, and Yajun Wang. "Scalable influence maximization for independent cascade model in large-scale social networks." In: *Data Mining and Knowledge Discovery* 25.3 (2012), pp. 545–576 (cit. on p. 51).

[118] Senzhang Wang, Xia Hu, Philip S. Yu, and Zhoujun Li. "MMRate: inferring multi-aspect diffusion networks with multi-pattern cascades." In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, New York, NY, USA - August 24 - 27*. 2014, pp. 1246–1255 (cit. on p. 55).

[119] D.J. Watts and P.S. Dodds. "Influentials, networks, and public opinion formation." In: *Journal of Consumer Research* 34.4 (2007), pp. 441–458 (cit. on p. 46).

[120] Duncan J. Watts. *Six Degrees: The Science of a Connected Age*. W. W. Norton, New York, 2003 (cit. on p. 46).

[121] Zheng Wen, Branislav Kveton, and Michal Valko. "Influence Maximization with Semi-Bandit Feedback." In: *Working paper*. 2016 (cit. on pp. 46, 55–59, 71, 72).

[122] Tom White. *Hadoop: The Definitive Guide*. 1st. O'Reilly Media, Inc., 2009. ISBN: 0596521979, 9780596521974 (cit. on p. 2).

[123] Sihem Amer Yahia, Michael Benedikt, Laks V. S. Lakshmanan, and Julia Stoyanovich. "Efficient Network Aware Search in Collaborative Tagging Sites." In: *Proc. VLDB Endow.* 1.1 (Aug. 2008), pp. 710–721. ISSN: 2150-8097 (cit. on pp. 15, 18).

[124] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster Computing with Working Sets." In: *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10 (cit. on p. 2).

[125] Aston Zhang, Amit Goyal, Weize Kong, Hongbo Deng, Anlei Dong, Yi Chang, Carl A. Gunter, and Jiawei Han. "adaQAC: Adaptive Query Auto-Completion via Implicit Negative Feedback." In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile, 2015, pp. 143–152 (cit. on p. 19).

[126] Ruicheng Zhong, Ju Fan, Guoliang Li, Kian-Lee Tan, and Lizhu Zhou. "Location-aware Instant Search." In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. CIKM '12. Maui, Hawaii, USA: ACM, 2012, pp. 385–394. ISBN: 978-1-4503-1156-4. DOI: 10.1145/2396761.2396812 (cit. on p. 19).

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX at

`https://bitbucket.org/amiede/classicthesis/`.

Université Paris-Saclay
ED STIC — 580
Université Paris Sud, Bâtiment 650 Ada Lovelace, 91405 Orsay Cedex, France.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Adaptive Methods for User-Centric Information Access Applications

Keywords: adaptive methods, online social networks, multi-armed bandits, recommendation, influencer marketing

Abstact: When users interact on modern Web systems, they let numerous footprints which we propose to exploit in order to develop better applications for information access. We study a family of techniques centered on users, which take advantage of the many types of feedback to *adapt* and improve services provided to users. We focus on applications like recommendation and influencer marketing in which users generate *discrete* feedback (e.g. clicks, "likes", reposts, etc.) that we incorporate in our algorithms in order to deliver strongly contextualized services.
The first part of this dissertation is dedicated to an approach for as-you-type search on social media. The problem consists in retrieving a set of k search results in a social-aware environment under the constraint that the query may be incomplete (e.g., if the last term is a prefix). Every time the user updates his / her query, the system updates the set of search results accordingly. We adopt a "network-aware" interpretation of information relevance, by which information produced by users who are closer to the user issuing a request is considered more relevant.
Then, we study a generic version of influence maximization, in which we want to maximize the influence of marketing or information campaigns by adaptively selecting "spread seeds" from a small subset of the population. Influencer marketing is a straightforward application of this, in which the focus of a campaign is placed on precise key individuals who are typically able to reach millions of consumers. This represents an unprecedented tool for online marketing that we propose to improve using an adaptive approach. Notably, our approach makes no assumptions on the underlying diffusion model and no diffusion network is needed.
Finally, we propose to address the well-known cold start problem faced by recommender systems with an adaptive approach. If no information is available regarding the user appreciation of an item, the recommender system needs to gather feedback (e.g., clicks) so as to estimate the value of the item. However, in order to minimize "bad" recommendations, a well-designed system should not collect feedback carelessly. We introduce a dynamic algorithm that aims to intelligently achieve the balance between "bad" and "good" recommendations.

Titre: Méthodes Adaptatives pour les Applications d'Accès à l'Information Centrées sur l'Utilisateur

Mots clefs: méthodes adaptives, réseaux sociaux, bandits multi-bras, recommandation, marketing d'influence

Résumé: Lorsque les internautes naviguent sur le Web, ils laissent de nombreuses traces que nous nous proposons d'exploiter pour améliorer les applications d'accès à l'information. Nous étudions des techniques centrées sur les utilisateurs qui tirent parti des nombreux types de rétroaction pour perfectionner les services offerts aux utilisateurs. Nous nous concentrons sur des applications telles que la recommandation et le marketing d'influence dans lesquelles les utilisateurs génèrent des signaux (clics, "j'aime", etc.) que nous intégrons dans nos algorithmes afin de fournir des services fortement contextualisés.
La première partie de cette thèse est consacrée à une approche interactive de la recherche d'information sur les médias sociaux. Le problème consiste à récupérer un ensemble de k résultats dans un réseau social sous la contrainte que la requête peut être incomplète (par exemple, si le dernier terme est un préfixe). Chaque fois que l'utilisateur met à jour sa requête, le système met à jour l'ensemble des résultats de recherche en conséquence. Nous adoptons une interprétation de la pertinence de l'information qui tient compte du réseau, selon laquelle l'information produite par les utilisateurs proches de l'utilisateur faisant la requête est jugée plus pertinente.
Ensuite, nous étudions une version générique de la maximisation de l'influence, dans laquelle nous voulons maximiser l'influence des campagnes d'information ou de marketing en sélectionnant de manière adaptative les utilisateurs initiant la propagation de l'information parmi un petit sous-ensemble de la population. Notre approche ne fait aucune hypothèse sur le modèle de diffusion sous-jacent ni même sur la structure du réseau de diffusion. Notre méthode a d'importantes applications dans le marketing d'influence qui vise à s'appuyer sur les influenceurs de réseaux sociaux pour promouvoir des produits ou des idées.
Enfin, nous abordons le problème bien connu du démarrage à froid auquel sont confrontés les systèmes de recommandation par une approche adaptative. Si aucune information n'est disponible concernant l'appréciation d'un article, le système de recommandation doit recueillir des signaux (clics, etc.) afin d'estimer la valeur de l'article. Cependant, afin de minimiser les mauvaises recommandations faites aux utilisateurs, le système ne doit pas recueillir ces signaux de façon négligente. Nous introduisons un algorithme dynamique qui vise à alterner intelligemment les recommandations visant à accumuler de l'information et celles s'appuyant sur les données déjà recueillies.