IMPROVING MARKETING INTELLIGENCE USING ONLINE USER-GENERATED CONTENTS

BY
JINLING JIANG

DISSERTATION SUBMITTED 2019

AALBORG UNIVERSITY
DENMARK

Improving Marketing Intelligence Using Online User-Generated Contents

Ph.D. Dissertation

Jinling Jiang

Dissertation submitted November 2019

Dissertation submitted:	December 2019
PhD supervisor:	Professor (MSO) Hua Lu Aalborg University
PhD committee:	Associate Professor Christian Thomsen (chairman) Aalborg University Professor Karine Bennis Zeitouni University of Versailles Saint-Quentin Associate Professor Raymond Chi-Wing Wong Hong Kong University of Science and Technology
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Computer Science

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-561-1

Published by:
Aalborg University Press
Langagervej 2
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2020

Abstract

In today's complex information environment, the original marketing relationship between consumers and brands has rapidly changed. The identity of consumers has changed from the original audience to the participators in the brand building process. How to find target audience and appropriately reach them so as to increase conversion rate is of great challenge.

Meanwhile, different kinds of marketing channels emerge and cover consumers' whole life-cycle. Nowadays there are mainly two channels for online advertising: digital advertising attract large consumer traffic on conventional web pages, whereas social media advertising, also known as social media marketing, uses social networks, blogs or other Internet communication platforms to conduct marketing and customer service development. The intensive interaction between brands and consumers increases marketing digitization and massive volumes of user behaviour data regarding online advertising has been generated and stored. This data foundation and the above challenges call for advanced techniques of utilizing online user generated contents to build intelligent marketing strategies.

In this thesis, we adopt machine learning and data management techniques to locate target audience and reach the audience via different channels. We solve the following four problems: 1) Precise Audience Expansion; 2) Accurate Prediction of Audience Demand; 3) Expert Recommendation for O2O Services; and 4) O2O Marketing by Utilizing Social Media Influencer.

First, traditional audience expansion model is usually based on user demographics which lacks the capability of capturing implicit user behaviour features. Yet, it is also normal that we are not able to collect ground-truth user profile information such as age and gender. Therefore in order to expand target audience, a more sophisticated model needs to be developed in the context of leveraging user behaviour in various advertising platforms. We aims to build an intelligent end-to-end audience expansion model under a real big data marketing environment.

Second, after targeting the right audience, advertisers are scrambling for a limited number of users to pay attention to their brands and pursuing more in-depth effects of advertising. The traditional CPM (Cost Per Mille) pay-

ment model is no longer sufficient as the advertising media will simulate lots of invalid traffic to charge. Thus the advertisers ask for CPC (Cost Per Click) payment model so that they only need to pay after users show interests in the ads. On the other hand, advertising media do not have click resources themselves and always think that the clicks on ads are not solely determined by the media audience, but also by the quality and creativity of the advertisement. To balance the benefits between advertisers and the advertising media, click-through rate (CTR) is a widely-used and crucial metric for evaluating ad performance. We develop a dedicated deep model for estimating CTR accurately while preserving user privacy.

Third, while e-commerce is growing at a fast rate, an advertising strategy would not just focus on its e-commerce functionality; it should provide online-to-offline (O2O) services as well. This is due to the growing importance of O2O marketing. The fundamental purpose of O2O marketing is to enhance the opportunity for online audience to become offline consumers. Meanwhile, the attention of consumers are moving increasingly to social media. Using large-scale data management techniques, we exploit social media data to recommend experts for O2O services regarding various geo-related information needs. In this way, users can look for online opinions efficiently and effectively in order to get satisfactory offline services. In this work, we design scalable solutions for hybrid index construction and efficient expert recommendation algorithms with advanced pruning techniques.

Fourth, as online pages are increasingly filled with various kinds of ads, advertisers are faced with the challenge of finding new ways to build relationships with their customers. There is a rising trend that deserves advertisers' attention, that is, "influencer marketing" which means using people's influence to break into your target market. Especially for O2O marketing, "influencer marketing" is an effective way to build a bridge between digital world and offline services. Imagine if audience are concerned about what someone wears on the social media, what to eat and drink, they will pay attention to it. We design hybrid user profiling techniques, a dedicated index structure and advanced pruning mechanisms for the search of local influencers around certain topics in Twitter.

We evaluate the proposed models and techniques by utilizing user-generated data from an advertising monitor platform and the social network platform Twitter. The advertising monitor data comes from Miaozhen Systems, the leading third-party advertising technology company in China. Tweets with geographical information are crawled for experimental use. The experiments conducted in each paper offer detailed understanding of the efficiency and effectiveness of the proposed methods.

Resumé

I dagens komplekse informationsmiljø det oprindelige markedsforhold mellem forbrugere og mærker har ændret sig hurtigt. Forbrugeridentitet er ændret fra det originale publikum til deltagerne i byggeprocessen for mærket, og endda forbrugere har større indflydelse på branddesign end selve mærket. At finde det rigtige publikum til at øge din konverteringsfrekvens er en stor udfordring.

I mellemtiden dukker de forskellige typer marketingkanaler op og dækker forbrugernes hele livscyklus. I dag er der hovedsageligt to kanaler til online-annoncering: Digital reklame tiltrækker stor forbrugertrafik på konventionelle websider, mens du annoncerer på sociale medier, også kendt som social media marketing, ved hjælp af sociale netværk, blogs eller andre internetkommunikationsplatforme at udføre markedsføring og kundeserviceudvikling. Intensiv interaktion mellem mærker og forbrugere søger markedsførings digitaliseringen, og der er genereret enorme mængder data om brugeradfærd om online-annoncering og gemt. Denne database og ovennævnte udfordringer kræver avancerede teknikker til at bruge online brugergenereret indhold til at opbygge intelligente marketingstrategier.

I denne afhandling bruker vi maskinlæring og datahåndtering teknikker til at lokalisere målgruppen og publikum via forskellige kanaler. Vi løser følgende fire problemer: 1) Eksakt publikum udvidelse; 2) Nøjagtig forudsigelse af publikums efterspørgsel; 3) Ekspertanbefaling til O2O-tjenester; 4) O2O Marketing ved hjælp af påvirkninger fra sociale medier.

For det første er den traditionelle publikums udvidelsesmodul normalt baseret på brugerdemografi, der mangler evnen til at fange implicitte funktioner til brugeradfærd. Ikke desto mindre er det også normalt, at vi ikke er i stand til at indsamle information om grundlæggende brugerprofiler som alder og køn. Derfor skal en mere sofistikeret model udvikles for at udvide publikum i sammenhæng med at udnytte brugeradfærd i forskellige reklameplatforme. Vi sigter mod at opbygge en intelligent ende-til-ende publikumsudvidelsesmodel under et ægte big data marketing-miljø.

For det andet, efter at have målrettet den rigtige målgruppe, skynder annoncører sig, efter at et begrænset antal brugere er opmærksomme på deres

mærker og forfølger mere dybe effekter af reklame. Den traditionelle betalingsmodel CPM (Cost Per Mille) er ikke længere tilstrækkelig, da reklamemediet simulerer mange ugyldige trafik til at opkræve. Annoncører anmoder således om en betalingsmodel for pris pr. Klik (CPC), så de kun skal betale, når brugerne viser interesse for annoncerne. På den anden side har reklamemedier ikke selv klik ressourcer, og mener altid, at klik på annoncer ikke kun bestemmes af mediepublikummet, men også af kvaliteten og kreativiteten i annoncen. For at afbalancere fordelene mellem annoncører og reklamemedier er CTR en meget brugt og kritisk beregning til evaluering af annonceydelsen. Vi udvikler en dedikeret dyb model til nøjagtigt at estimere CTR og samtidig bevare brugernes privatliv.

For det tredje, mens e-handel vokser hurtig, bør en reklamestrategi ikke kun fokusere på e-handelsfunktionalitet; Det skal også tilby online to-of-flow-tjenester (O2O). Dette skyldes den voksende betydning af O2O-markedsføring. Det grundlæggende formål med O2O-markedsføring er at forbedre online-målgruppernes evne til at blive flygtige forbrugere. I mellemtiden flytter forbrugerne mere og mere til sociale medier. Ved hjælp af storskala data styringsteknikker bruger vi sociale medier data til at anbefale eksperter til O2O-tjenester vedrørende forskellige georelaterede informationsbehov. På denne måde kan borgeren søge online meninger effektivt og effektivt for at opnå tilfredsstillende flyveservice. I dette arbejde designer vi en skalerbar løsning til konstruktion af hybrid indeks og effektive algoritmer til ekspertanbefaling med avancerede beskæringsteknikker.

For det fjerde, da websteder i stigende grad er fyldt med reklame, står annoncører over for udfordringen med at finde nye måder at opbygge relationer til deres kunder. Der er en voksende tendens, der fortjener opmærksomheden fra annoncører, det vil sige "influencer marketing", hvilket betyder at bruge folks indflydelse til at bryde ind på målmarkedet. Især for O2O-markedsføring er "influence marketing" en effektiv måde at bygge bro på den digitale verden med luftfartstjenester. Forestil dig, at hvis publikum er bekymret for, hvad nogen har på sociale medier, hvad de skal spise og drikke, vil de være opmærksomme på det. At vælge lokale påvirkere omkring specifikke emner på Twitter.

Vi designer hybrid brugerstablingsteknikker, en dedikeret indeksstruktur og avancerede beskærings mekanismer til blødere søgning på Twitter. Vi vurderede forudbestemte modeller og teknikker ved hjælp af brugergenererede data fra en reklamemonitorplatform og den sociale netværksplatform Twitter. Data fra reklamemonitoren kommer fra Miaozhen Systems, det førende tredjeparts reklameteknologiselskab i Kina. Tweets med geografisk information scannes til eksperimentel brug. Eksperimenterne udført i hver artikel giver en detaljeret forståelse af effektiviteten og effekten af de foreslåede metoder.

Acknowledgments

My utmost gratitude must go to my supervisor, Prof. Hua Lu, for his patience and professional guidance throughout my candidature period. I am quite impressed by his conscientiousness and could not have imagined I can finish my Ph.D. study without his extensive support. And I am very appreciative of the instructive suggestions and inspiring ideas he delivered. I feel very grateful for the time we have spent together on my Ph.D. study.

Next, I would like to express my sincere thanks to Prof. Xindong Wu, who is always there to cast the cares and support my research at Mininglamp Technology in recent two years. His endless encouragement supported me in going through the tough endeavors in my research and industrial activities.

An essential part of my research is delivered through active collaborations. My gratitude goes to Prof. Junjie Yao and Xiaoming Lin, with who I have actively worked and discussed. Without their technical support, this thesis would not have been made possible. I must also thank Prof. Bin Yang for his kindly help in my first publication. My office buddies in Daisy and Mininglamp, thank you for the lovely moments we have spent together. I would like to extend my thanks to all collaborators in my publication and colleagues. I am blessed with not only academic collaboration but also the friendship forged with them during these years.

Besides, I would like to express my gratitude to Dr. Jilin Hu for his guidance during my thesis writing. Special thanks to Helen Kristensen and Sututhi Perrananthasivam in Ph.D. School, who checked the ETCS I have got during my study. Special thanks to Prof. Jianyong Wang, who wrote a confirmation letter for my ICDM 2019 participation. Without your full help, I cannot pass the defense requirements.

Lastly, my heartfelt thanks would go to my beloved family and girlfriend for their loving considerations and great confidence in me all the time. Words fail to reveal my gratitude to my parents, who have raised me and guide me to be a person of integrity. You are the spiritual motivation that keeps me going.

Jinling Jiang
Aalborg University, December 12, 2019

Contents

Thesis Details

Thesis Title:

Improving Marketing Intelligence Using Online User-Generated Contents

Ph.D. Student: Jinling Jiang
Supervisors: Professor (MSO) Hua Lu, Aalborg University
The main body of the thesis consists of the following papers.

A. **Jinling Jiang**, Xiaoming Lin, Junjie Yao, Hua Lu: "Comprehensive Audience Expansion based on End-to-End Neural Prediction." Proceedings of the ACM SIGIR 2019 Workshop on eCommerce, 8 pages

B. **Jinling Jiang**, Xiaoming Lin, Junjie Yao, Ruogu Ding, Hua Lu, Xindong Wu: "Deep Monitor Network for Privacy-Preserving Click-Through Rate Prediction." (submitted to The ACM Web Conference, 2020)

C. **Jinling Jiang**, Hua Lu, Bin Yang, Bin Cui: "Finding top-k local users in geo-tagged social media data." IEEE International Conference on Data Engineering (ICDE) 2015: 267-278

D. **Jinling Jiang**, Hua Lu, Pengfei Li, Gang Pan, Xike Xie: "Finding Influential Local Users with Similar Interest from Geo-Tagged Social Media Data." IEEE International Conference on Mobile Data Management (MDM) 2017: 82-91

In addtion to the above papers, I have coauthored the following paper as part of my Ph.D. studies, which are not included in the thesis.

E. Chengbin Peng, Ka-Chun Wong, Alyn Rockwood, Xiangliang Zhang, **Jinling Jiang**, David E. Keyes: "Multiplicative Algorithms for Constrained Non-negative Matrix Factorization." IEEE International Conference on Data Mining (ICDM) 2012: 1068-1073

This thesis has been submitted for assessment in partial fulfillment of the Ph.D. degree. The thesis is based on the submitted or published scientific papers listed above. Parts of the content of the papers in the main body of the thesis are used directly or indirectly in the extended summary part of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The permission for using the published and accepted articles in the thesis have been obtained from the corresponding publishers with the condition that they are cited and copyright are placed prominently in the references. In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Aalborg University's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Part I

Thesis Summary

Chapter 1

Introduction

1.1 Background and Motivation

With the development of the Internet and the growth of users, the advertising industry, originated from the traditional offline advertising model, is gradually transforming into online model. No matter which marketing channel, there exist two primary types of online advertising. Brand advertising is a form of advertising used to establish connections and build strong, long-term relationships with consumers over time. Performance-based advertising is a form of advertising in which the advertisers pay only when there are measurable results. Either type of advertising aims to satisfy consumer needs and wants while creating profits for advertisers.

Compared with traditional advertising, the use of large data analysis technology for online advertising unveils business patterns from massive multi-sourced and heterogeneous user-generated content, providing intelligent support for business decisions on user insights, brand analysis, consumer profiling, personalized advertising and etc. These incorporated techniques are executed not only in booming display advertising industry, but also in various kinds of social media advertising. In display advertising, how to expand the audience pool and then push the right content to them become the two common requirements of advertisers. Regarding social media marketing, how to find an appropriate way of locating key opinion leaders (KOLs) is an important approach of reaching the target audience for various O2O (Online to Offline) scenarios. A KOL can be an expert or influencer who not only serve as a brand communicator, but also help brand advertisers to spread information to the audience and effectively affects consumer purchasing decisions.

In this thesis, we address the four problems that are introduced in the following parts:

1. Precise Audience Expansion: Provide intelligent audience expansion model.
2. Accurate Prediction of Audience Demand: Provide personalized recommendation model for display advertising while preserving user privacy.
3. Expert Recommendation for O2O Services: Look for expert opinions for geo-related services efficiently and effectively.
4. O2O Marketing by Utilizing Social Media Influencer: Select local influencers around certain topic in social media for maximizing marketing effects.

1.1.1 Precise Audience Expansion

Digital advertising has changed from passive acceptance into two-way interaction, and the effectiveness of advertising has become quantifiable and traceable. As a result, brand advertisers have become more and more demanding for audience targeting. By tracking and obtaining user behavior data, a third-party supplier of advertising monitor can analyze the data according to the advertiser needs, not only understanding the communication effects and sales conversion rate generated by the advertisement in time but also predicting the user conversion probability to some extent. Through analysis and modeling on massive data of user behavior, advertisers can accurately reach the target consumer. Therefore, how to better utilize the advertising monitor data in order to optimize ad serving and improve marketing conversion rate has become an important issue. There are two types of mainstream practices for the achievement of this task:

One way of selecting target audience is user profiling techniques through analysis on consumer data, including browsing, purchase behavior, user profile information, etc. In the end, the advertising platform will target new audience according to tags such as user demographics or interest preferences. However, it is difficult for advertisers to collect comprehensive user data for profiling and the tag-based audience expansion does not take into consideration the implicit and sophisticated user similarity, leading to lower advertising performance.

The second type is to select the target audience through the Lookalike algorithm. This approach relies more on big data and machine learning algorithms. It is more suitable for exploring new business logic, such as marketing for a new product, without a lot of business experience. We aims to build an intelligent end-to-end audience expansion model under a real big data marketing environment.

Example 1.1.1 (Audience Expansion)
Figure 1.1 reveals that advertisers submit a list of customers, which we call seed user set S, as positive samples and there are a universal user set U existing in advertising monitor platform. The user set $U - S$, which is called C , includes a large number of unlabeled samples. A lookalike machine is derived to predict a rank list of unlabeled users and the target audience set is taken out according to advertising requirements.

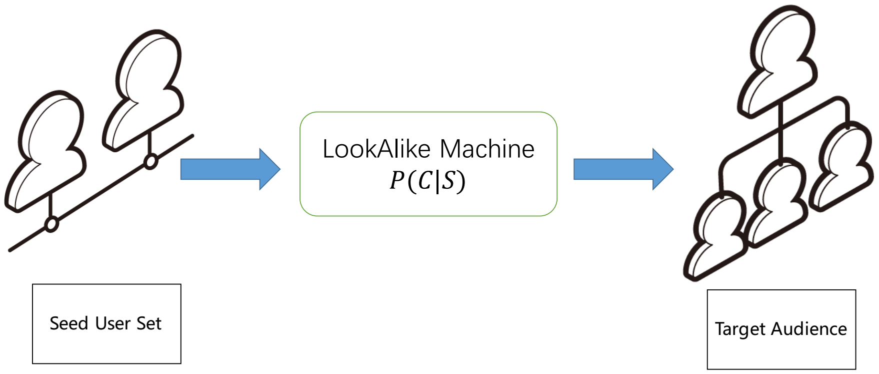


Fig. 1.1: Lookalike Model

1.1.2 Accurate Prediction of Audience Demand

In online advertising, click-through rate (CTR) is a crucial metric for evaluating ad performance and widely used for sponsored search, contextual advertising, display advertising and real-time bidding auctions. As in Example 1.1.2, the ad ranking strategy generally depends on CTR and bidding price which is the benefit the system will receive if an ad is clicked. According to performance-based advertising model, advertisers are only charged once their ads are clicked or actioned by users. Therefore, in order to maximize the revenue and to maintain a desirable user experience, it is critical for advertising platform to estimate the CTR of ads accurately.

Example 1.1.2 (Click Through Rate)
Figure 1.2 shows that in a computing advertising system, when a user traffic that can carry an advertisement request arrives, the system needs to return one sorted advertisement lists in a short time (usually required not to exceed 100 ms). Generally, the ranking score is based on the equation $bid * pctr * \alpha$ where parameter α controls the ranking mechanism: if $\alpha < 1$, the core factor is predicted click through rate (pctr) which is learned from user historical behaviors; otherwise the bidding price will dominate.

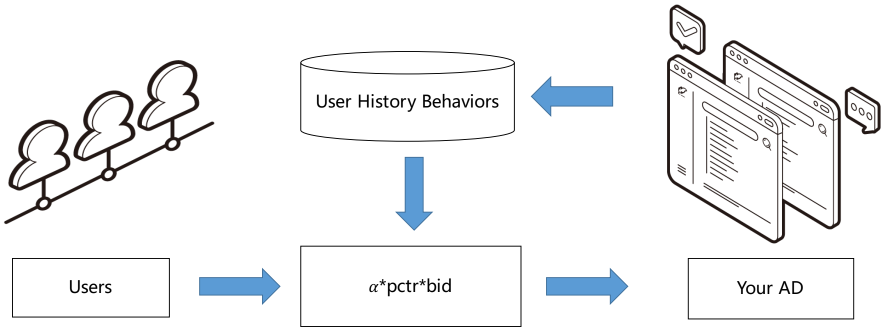


Fig. 1.2: CTR Illustration

There are a lot of work done in opimizing through learning from the user profile information and historical behaviors. However, under new data protection regulation like GDPR[1], user profile information is sensitive to access so that there is an urgent need of developing a CTR model with high accuracy under such circumstance.

1.1.3 Expert Recommendation for O2O Services

O2O marketing refers to the combination of the Internet platform with online payment capability and offline service business including catering, film, beauty, travel, fitness, etc. For example, consumers could see an ad online

[1]https://eugdpr.org/

and are driven to visit the restaurant, finally paying through the O2O applications. Example 1.1.3 describes a scenario where users are intelligently diverted into physical restaurants under online influence. The challenge is how to efficiently and effectively identify experts regarding certain geo-related service so as to look for their opinions.

Example 1.1.3 (Expert Recommendation for O2O Services)
For example, a couple travelling to Australia may ask "Are there any good restaurant near Sydney Opera House?" This query contains keyword "restaurant" and spatial location "Sydney Opera House". Directly retrieving the recommendations from the Internet couldn't exclude advertisements and consumers tend to look for advices from credible people. Figure 1.3 exhibits that consumer purchases at physical stores can be highly influenced by online interactions with experts.

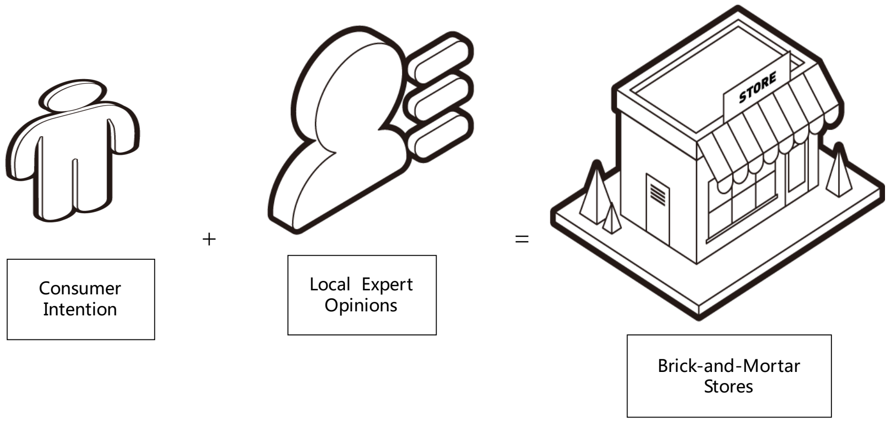


Fig. 1.3: Expert Recommendation for O2O Services

1.1.4 O2O Marketing by Utilizing Social Media Influencer

An effective means to build a bridge between digital world and offline physical stores is the so-called "influencer marketing" that ranks the fastest-growing online acquisition method in recent years[2]. Influencer marketing generally

[2]https://blog.crobox.com/article/influencer-psychology

means promoting services or products via social media influencers in Twitter, Facebook, Instagram, etc. Influencers are individuals who influence a consumer's opinion about a product or service and can have a strong influence on specific audience group on social media platforms. People follow influencers because their content matches the follower's interest, taste or opinion. Example 1.1.4 demonstrates the way of leveraging local influencers in an O2O marketing campaign. The question is how to select local influencers around certain topic as KOLs according to their published contents and social influence.

Example 1.1.4 (O2O Influencer Campaign)
Figure 1.4 depicts the effect brands line up with popular social media personas in order to expand the targeting audience. Suppose if you're running an event offline for a sport brand, you would consider bringing influencers to the event to capture the moments live and let the content stay online, generate traffic and eyeballs through Twitter live and Instagram stories. We need organise the event with a number of influencers (individuals between a thousand to a million followers) who are KOLs at sporting to support the campaign locally.

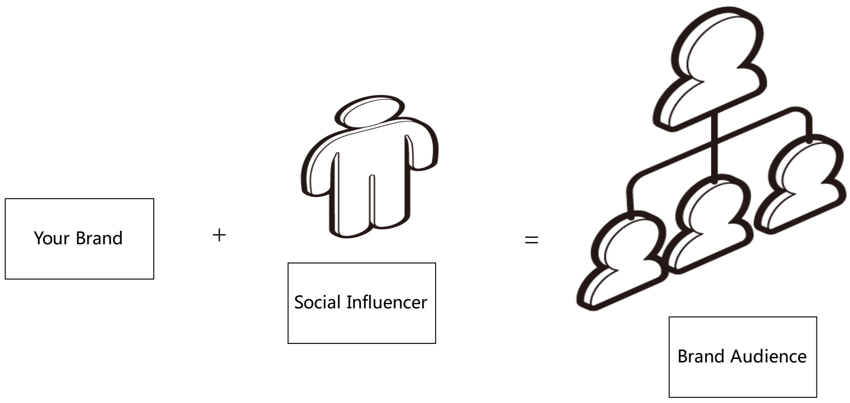


Fig. 1.4: Influencer Marketing

1.2 Thesis Structure

The thesis adopts various machine learning and data management techniques that can contribute to intelligent marketing applications. As already mentioned in the previous section, we aim to address four challenges in the thesis, whose overall structure is illustrated in Figure 1.5. Paper A and B focus on improving online marketing intelligence by utilizing user-generated actions in display advertising. In Paper A, under a real business setting, we propose a comprehensive solution incorporating an end-to-end audience expansion model so that the target audience can be located precisely. For those target audience, we enhance the capability of predicting click-through rate by introducing a novel Deep Monitor Network model while preserving user privacy in Paper B. Paper C and D put emphasis on improving O2O marketing intelligence by utilizing user-generated contents in social media platform. In Paper C, we retrieve the local experts in social media platform for users to look for opinions on geo-related services. Then in Paper D, we model and locate local social media influencers around certain topic as KOLs in order to maximize the effects of O2O influencer marketing.

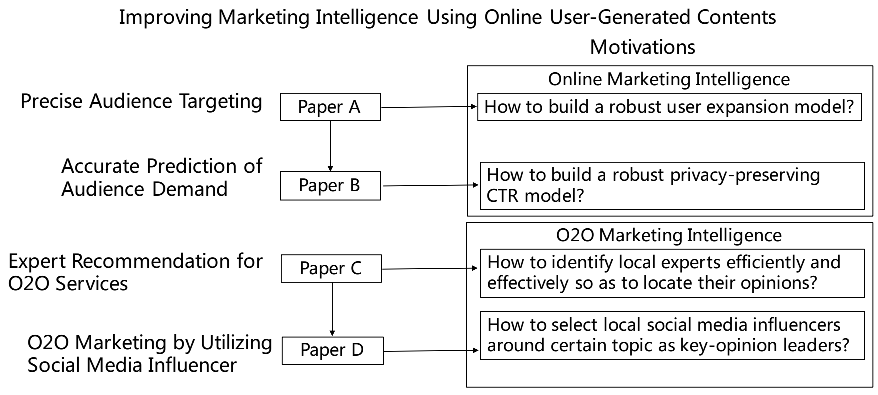

Improving Marketing Intelligence Using Online User-Generated Contents
Motivations
Online Marketing Intelligence
Precise Audience Targeting
Paper A
How to build a robust user expansion model?
Accurate Prediction of Audience Demand
Paper B
How to build a robust privacy-preserving CTR model?
O2O Marketing Intelligence
Expert Recommendation for O2O Services
Paper C
How to identify local experts efficiently and effectively so as to locate their opinions?
O2O Marketing by Utilizing Social Media Influencer
Paper D
How to select local social media influencers around certain topic as key-opinion leaders?

Fig. 1.5: Thesis Structure

Chapter 2

Audience Expansion For Display Advertising

This chapter gives an overall introduction of Paper A [13]. The chapter reuses content from the paper when that was considered most effective.

2.1 Problem Motivation and Statement

We first consider the scenario of audience expansion in display advertising. Suppose an advertiser has a seed consumer set who are their targeting audience, our goal it to find new clients who behave pretty similar to the current customers. We call this problem *look-alike modeling*. In the context of marketing, look-alike modeling can be in audience expansion task for the purpose of reaching new prospects with similar interests to the original target audience.

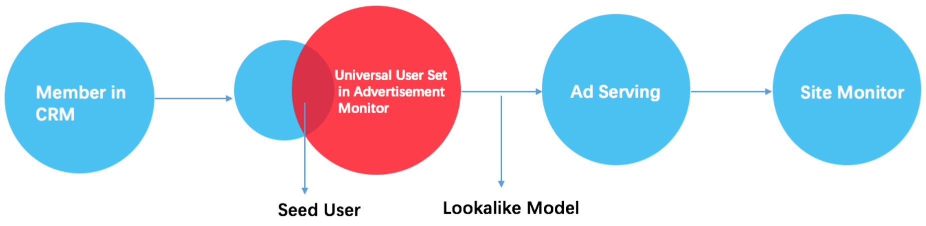


Fig. 2.1: Audience Expansion Dataflow [13]. ©2019 ACM

The data flow of audience expansion service is illustrated in Figure 2.1

reproduced from Paper A. The original users are provided by advertisers who upload the consumers recently exercising the purchase actions. Then the universal advertising monitor will match out some "seed" users. We build a lookalike model to predict the probability to be the target audience for all users of advertising monitor. Finally lookalike model performance will eventually evaluated by ad serving system and advertiers' site monitor.

The following example of audience expansion is reproduced from Paper A.

Example 2.1.1 (Problem Definition of Audience Expansion)
Advertisers submit a list of customers, which we call seed user set S, as positive samples and there are a universal user set U existing in advertising monitor platform. Then the problem is transformed into a Positive and Unlabeled learning problem: using a small number of labeled positive samples S and a large number of unlabeled samples $\|U - S\|$ to derive a prediction classifier.

Generally lookalike modeling which supports audience expansion system can be categorized in three lines: rule-based, similarity-based and model-based. **Rule-based approaches** focus on explicit positioning, where users with specific demographic tags (age, gender, geography) or interests are targeted directly for advertiser. The core technical support in the background is user profile mining, which means, the interest tags are inferred from the user behaviour [30] [33]. Furthermore, Mangalampalli et al. [25] builds a rule-based associative classifier for campaigns with less conversion; Shen et al. [32] and Liu et al. [23] present detailed in-depth analysis of multiple methods under different considerations(such as similarity, performance, whether or not campaign-agnostic) for online social network advertising. The main disadvantage of rule-based lookalike modeling is that it only captures the high-level features, therefore loses sophisticated details of user behaviour.

In order to verify that lookalike models based on the user behaviour work better than traditional demographics-based approach regarding the sales conversation rate, in Paper A, we integrate data flow during the whole advertising life cycle. There is not negative samples in audience expansion as a positive-unlabeled learning (PU learning) problem, we compare some methods used for sampling negative examples.In addition, we propose Scale-MLP, a network based on MLP, to get better generalization ability.

The contributions of this work can be summarized as follows.

- We have improved the commonly used ad serving mode from demographics-based crowd segmentation to a comprehensive audience expansion framework.

- We propose a lookalike model that has better generalization ability for audience expansion problem.
- We conduct extensive and effective experiments to extract negative samples from unlabeled data.
- We prove the effectiveness of the proposed lookalike models in an online environment.

2.2 Feature Extraction and Sample Selection

Here we introduce the features extraction and sampling techniques in the lookalike model.

2.2.1 User Behavior

We represent user behaviour with a high-dimensional sparse feature vector and each feature indicates the number of times an advertisement is clicked or just impressed by corresponding user. For example, we can see from Table 2.1, user with id "0043fbf4" is impressed by spid1 three times and only give one click. The original feature is then normalized according to the following equation:

$$norm\_freq = \begin{cases} \dfrac{1}{1+exp(-\frac{freq}{10})} & freq > 0 \\ 0 & freq = 0 \end{cases} \tag{2.1}$$

Table 2.1: User behaviour Representation [13]. ©2019 ACM

USER_ID	spid1_click	spid1_impression	spid2_click	spid2_impression	...	label
66a7988f	0	3	1	3	...	1
9b3fcc94	0	2	1	1	...	-1
0043fbf4	1	1	0	1	...	-1
9e664577	1	3	1	2	...	1
1df73293	0	1	0	1	...	-1

2.2.2 Sampling techniques

Besides randomly selecting negative samples and directly apply standard classifier to the PU learning problem, we compare the effectiveness of three other sampling techniques: spy, pre-train and bootstrap sampling. The "Spy" [22] [20] and "Pre-Train" sampling strategies are so-called "two-step" approach [18] where the general idea is described as follows: the first step is to identify a subset of unlabeled samples that can be reliably labelled as negative, then positive and negative samples are used to train a standard classifier that will be applied to the remaining unlabeled samples. Usually the classifier is learned iteratively till it converges or some stopping criterion is met. A more sophisticated approach [27] is a variant of bagging. For each iteration, a subset of unlabeled samples are bootstrapped from the unlabeled sample set U. Then a classifier is trained on the positive sample set and bootstrapping samples and the rest samples are predicted by the classifier. Finally we average the classifying scores of all iterations and select a subset samples with least average scores.

We compare the effectiveness of above three sampling techniques. Experimental results in Table 2.2 show that it is both efficient and effective to utilize spy sampling approach in our setting.

Table 2.2: The Impact of Sampling Approach [13]. ©2019 ACM

approach	sampling parameter	train loss	test accuracy	test auc	test recall	threshold
Random	0.9	0.4250	0.775	0.847	0.766	0.633
Random	0.5	0.4150	0.753	0.843	0.676	0.612
Spy	0.95	0.4138	0.775	0.847	0.768	0.640
Spy	0.9	0.4141	0.776	0.847	0.763	0.633
Spy	**0.5**	**0.3660**	**0.775**	**0.845**	**0.768**	**0.607**
Pre-Train	0.95	0.3677	0.775	0.845	0.771	0.624
Pre-Train	0.9	0.3829	0.776	0.846	0.771	0.632
Pre-Train	0.5	0.4382	0.702	0.839	0.632	0.628
Boostrap	0.95	0.4127	0.775	0.847	0.768	0.638
Boostrap	0.9	0.4153	0.776	0.847	0.763	0.633
Boostrap	**0.5**	**0.3976**	**0.775**	**0.845**	**0.766**	**0.640**

2.3 Comprehensive Modeling

We continue to introduce the lookalike model techniques used in our audience expansion system. Figure 2.2 illustrates a MLP network added by a scale layer.

The reason adding scale layer: The prediction equation of Scale-MLP model is defined as:

$$y = mlp(A(W \circ X) + bias), A \in \mathbb{R}^{k \times n} \tag{2.2}$$

According to Equation 2.2, after adding a scale layer to MLP, deep networks need feedforward an intermediate result $w_j x_j$. The essential difference between MLP and Scale-MLP is the way of updating the parameter matrix A during backpropagation. For MLP the partial derivative regarding a_{ij} is x_j, as a result, data noise on feature x_j can directly update the parameters $a_{ij}, i = 1...k$. The situation for Scale-MLP is different as feature x_j can only

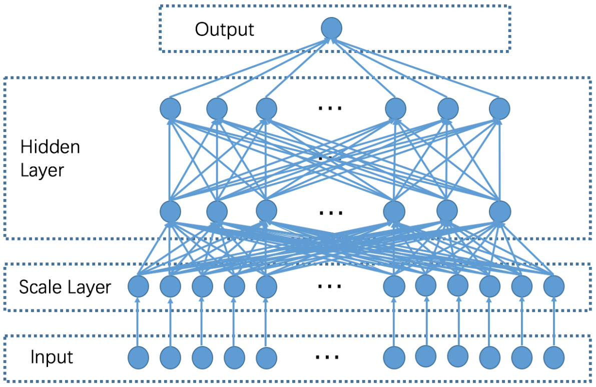


Fig. 2.2: Add Scale Layer to MLP Network [13]. ©2019 ACM

affect w_j. In display advertising, the randomness of x_j could be large so that Scale-MLP will be more robust.

2.4 Experiments

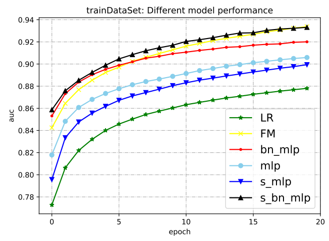
The dataset sizes are typically configured in real business environment as follows: $\|S\| = 0.1 - 0.2M(Million)$, $\|T\| = 10 - 20M$ and $\|U\| = 2000 - 3000M$. S is a small number of labeled positive samples submitted by advertisers, U is universal user set existing in advertising monitor platform, and T is the target audience set advertiser needs.

2.4.1 Model Performance

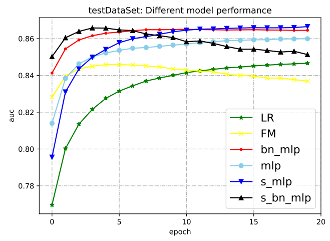
The model performance are shown in Figure 2.3. The results indicate that Scale-BN-MLP outperforms other five models at AUC value and convergence speed as well. This confirms the effects of adding batch normalization layer and scale layer to original MLP illustrated in the modelling section. In practice, Scale-BN-MLP model requires early stopping as it needs the least number of epochs to yield the optimal solution in testing phase. Figure 2.4 shows Scale-BN-MLP model has the least convergence epochs when learning rate is set 0.0001(1e-4) both in training and testing phases.

2.4.2 Online Effectiveness Evaluation

In order to evaluate the model effectiveness in a real business environment, we conduct online experiments for an advertiser using the data collected by

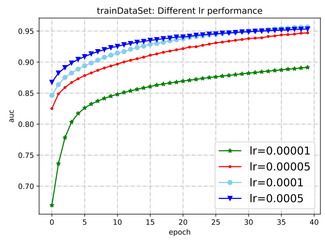


(a) Train Dataset

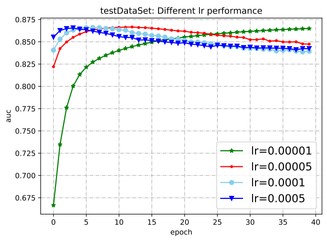


(b) Test Dataset

Fig. 2.3: Model Performance [13]. ©2019 ACM



(a) Train Dataset



(b) Test Dataset

Fig. 2.4: Comparison of Different Learning Rate [13]. ©2019 ACM

the third-party advertising monitor. Table 2.3 lists the testing result of a number of important business metrics such as ATV (Average Transaction Value), CPO (Cost Per Order) and CPA (Cost Per Action). Regarding the most important indicators CPO and CPA, our lookalike model outperforms nearly 6 times then the second-ranked method and far better than other demographic-based methods as well.

Table 2.3: Online A/B Testing Results [13]. ©2019 ACM

Metric	Random	F20-34	FEMALE	MALE	MODEL
Impression	18,367,151	6,493,314	3,910,355	1,454,655	1,221,095
Impression UV	8,578,859	3,152,614	2,052,897	912,468	594,456
Purchaser	597	123	117	29	217
Purchaser Rate	0.01%	0.00%	0.01%	0.00%	**0.04%**
Transaction	731	155	134	32	275
Sales	69,974	14,976	14,727	3,739	23,413
ATV	96	97	110	117	85
Media Cost	295,575	106,982	61,972	24,429	19,100
CPO	404.3	690.2	462.5	763.4	**69.5**
CPA	495.1	869.8	529.7	842.4	**88.0**
Incremental ROI	0.2	0.1	0.2	0.2	**1.2**

Chapter 3

Privacy-Preserving Click-Through Rate Prediction

This chapter gives an overall introduction of Paper B [12]. The chapter reuses content from the paper when that was considered most effective.

3.1 Problem Motivation and Statement

We first consider the problem setting. Given a specific user, an advertisement and corresponding ad campaign information, our goal is to estimate the probability of the user clicking on the advertisement. We call this the click-through rate prediction (CTR) problem in display advertising. CTR prediction continuously plays an important role in the advertising process given a budget and related information of a campaign, the campaign advertisers need decide how much they would like to bid for their advertisements in various media platforms in order to achieve the maximum clicks. The following example is reproduced from Paper B.

Example 3.1.1 (Real-world Click-Through Rate Example)
Table 3.1 shows a real example of advertising dataset consisting of typical multi-field categorical data for CTR prediction. If CLICK label equals to 1 it means the specific ad is clicked by corresponding user, otherwise it indicates an impression without click action. Each column attribute represents a field and each categorical feature (e.g., Nestle, iPhone X, IQIYI) belongs to one and only one field. For a new row without CLICK label, we need to estimate whether the CLICK label equals to zero or one.

Table 3.1: Multi-Field Records of Advertising Campaigns, used in CTR Prediction [12].

CLICK	USER_ID	Advertiser	Brand	Product	Industry	Platform	Device
1	66a7988f	L'Oreal	Maybelline	Maybelline	Beauty Care	Youku	MI MAX 2
1	9e664577	OPPO	OPPO	OPPO R7 PLUS	IT Product	Tencent	iPhone 6
0	9b3fcc94	Priceline	Agoda	Agoda	Tourist Accommo-dation	IQIYI	iPhone X
1	0043fbf4	P&G	SK-II	SK-II	Beauty Care	Tencent	HW 2A
1	1df73293	Nestle	Nan	Nestle Nan	Baby Prod-ucts	Tudou	OPPO V6
0	23ff3494	OPPO	OPPO	OPPO V6	IT Product	SINA	iPhone X

Now we are given a training dataset of n instances where each sample is (X_i, y_i). Under the setting of preserving user privacy, X_i excludes the user profile information (e.g., age, gender, location) and incorporates advertisement information such as advertiser, brand and product (all categorical features are transformed into one-hot vectors). $y_i \in \{0, 1\}$ is a label representing whether the user clicked on the item ($y_i = 1$ means clicked and $y_i = 0$ unclicked). We apply one-hot encoding to all categorical features and finally get a extremely sparse high-dimensional vector.

CTR estimation models can be categorized as FM-based [17] [16] [29] and deep learning based algorithms [34] [31] [3] [9]. The recent deep models [21] [2] [24] aim to design dedicated neural networks for high-order user feature extraction. DSTN [28] tries to learn an integrated representation of contextual and historical ads displayed to the same user. However, all these works do not take into consideration the user privacy issue. We design a novel model Deep Monitor Network (DMN) to tackle the challenges by incorporating creative user feature extraction and compressing historical training data.

Figure 3.1 depicts the general structure of our privacy-preserving prediction model. There are three main blocks: attention block activating the correlated events with respect to the predicted event; a dedicated feature extraction module for learning representation of user interests from historical events; a module integrating FM and MLP to learn higher order interactions. In the feature processing module, appropriately forecasting CTR for advertisers depends on fine designed processing of historical user behaviour recorded by advertising monitor at different media platforms (including clicked and unclicked records). The data of an advertising campaign used in our CTR prediction are usually multi-field categorical data where the feature space is very sparse and every feature belongs to one and only one field. Here we group the historical records by users in a natural way and learn the representation of user interests by a specific attention mechanism. In the training stage, though the scale of historical data seems challenging, we sample and well extract the sliding windows for the purpose of selecting training data.

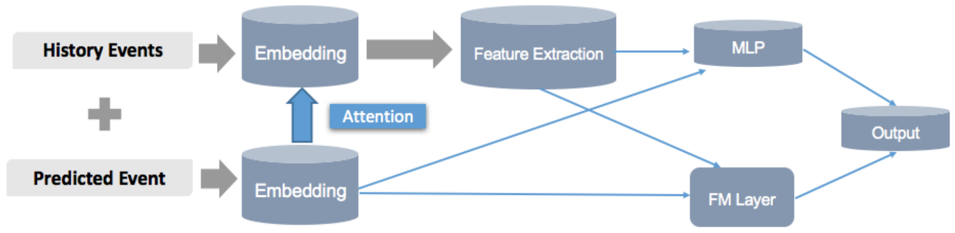


Fig. 3.1: Pipeline of DMN Prediction Model

The contributions of this work can be summarized as follows.

- We design a new Deep Monitor Network to capture high-order user features and reduce the model parameter space.
- DMN model preserves the user privacy by omitting user profile embedding, instead it learns the user interests representation by extracting features of user historical actions.
- Extensive experiments are conducted on both private and public datasets to demonstrate the advantages of DMN both on performance and model consistency under invalid traffic attack.

3.2 Deep Monitor Network

In this section we describe Deep Monitor Network (DMN) structure. Suppose a user without personal profile information, has n event records and each record contains m feature fields with an associated label y ($y \in \{0,1\}$). The traditional CTR model treats each record as an independent sample, and then explore interactions of the same user's records by embedding user identification at training phase. On the contrary, in our model, we sort the n event records in chronological order and treat them together as a sample. There are two benefits of this mechanism in general: it avoids the training challenge caused by the considerable training parameters of directly embedding user identification; it allows incremental model updating when new users emerge in the system, accordingly, there is no need for frequent model updates.

Shown in Figure 3.2 that generally depicts the model architecture with comparison of DeepFM [9]. The essential difference between the two models is the way of extracting user features. DeepFM utilizes the embedding of user identification while DMN relies on a delicate network structure in order to extract features of history events belonging to the same user. Deep Monitor

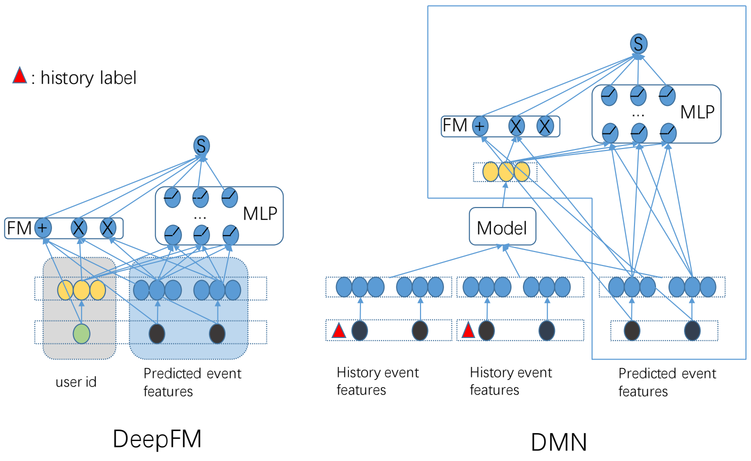


Fig. 3.2: Comparison of DeepFM and DMN Architecture [12].

Network mainly consists of three blocks: feature extraction of user historical cross-media behaviour, FM layer and MLP. There are two input pieces: history events belonging to one user, and the predicted event.

In fact, The very difference between DMN and other networks lies in the way of learning features representing user interest from history events. The most important part of DMN model is a feature extraction network extracting user interest representation with respect to an advertisement to be predicted. FM layer and MLP are utilized for extracting both low-order and high-order feature interactions based on the extracted user interest representation. We can also connect the feature extraction network of DMN with LR or other machine learning algorithms for CTR task. Regarding FM and MLP blocks, they are essentially constructed in the same way as DeepFM model.

3.3 Experiments

To study the effectiveness of DMN, we explore two dataset: Company* Dataset and Avazu Dataset[1]. Regarding Avazu dataset, we subsample the records of device_ids which has more than 10 instances. We compare 6 models in our experiments: LR, CCPM [24], DeepFM [9], MSM-RD [2], DSTN [28] and DMN.

[1]https://www.kaggle.com/c/avazu-ctr-prediction/data

3.3.1 Efficiency Comparison

In this part, we evaluate the efficiency of various deep CTR models. Regarding each epoch training time, Table 3.2 shows DMN requires less than other models except DeepFM. From Table 3.3 we can see that DMN is one of several models that require minimal epochs needed to converge.

Table 3.2: Model Efficiency Comparison [12].

Model	Company*	Avazu
LR	6000	2000
CCPM	8500	7000
DeepFM	24000	12000
MSM-RD	46800	19000
DSTN	30000	12000
DMN	28000	10000

Table 3.3: Number of Epochs to Converge [12].

Model	Company*	Avazu
LR	2	2
CCPM	4	9
DeepFM	11	4
MSM-RD	5	3
DSTN	5	4
DMN	5	5

3.3.2 Effectiveness Comparison

The prediction effects of different CTR models on the Company* and Avazu datasets are listed in Table 3.4.

Table 3.4: Model Effectiveness Comparison [12].

Model	Company*		Avazu	
	AUC	LogLoss	AUC	LogLoss
LR	0.6889	0.11045	0.7356	0.38805
CCPM	0.7248	0.10812	0.7433	0.38360
DeepFM	0.7219	0.10828	0.7461	0.38184
MSM-RD	0.7263	0.10799	0.7447	0.38327
DSTN	0.7223	0.10831	0.7475	0.38214
DMN	**0.7317**	**0.10746**	**0.7484**	**0.38169**

It can be seen from Table 3.4 that regarding AUC value: 1) on Company* dataset, DMN model has the best performance (0.74% improvement than MSM-RD that ranked second; 2) on Avazu dataset, DMN improve 1.23% compared to LR and 0.13% compared to the second best model DSTN.

Chapter 4

Expert Recommendation for O2O Services

This chapter gives an overall introduction of Paper C [15]. The chapter reuses content from the paper when that was considered most effective.

4.1 Problem Motivation and Statement

O2O marketing refers to the combination of the Internet platform with online payment capability and offline service business including catering, film, beauty, travel, fitness, etc. For example, consumers could see an ad online and are driven to visit the restaurant, finally paying through the O2O applications. Imagine people are looking for a Japanese food restaurant of good quality nearby, one can issue a query considering "Japanese restaurant" and current location in some application like TripAdvisor [1]. However, directly looking at recommendations from the application couldn't exclude advertisements and consumers tend to look for advices from credible people at social network platforms. Under expert influence, users are eventually guided to favourable physical restaurants. The key information need here is to efficiently identify the right person who are experts in the specific geo-related problem so that users can look for expert opinions in social media.

Traditional spatial-aware analyses conducted on social media [5, 19] focus on mining real-time and geo-spatial event information from geo-tagged tweets. One related industrial system Aardvark [10] retrieves the users for each raised question in order to satisfy one's information need. In this social search engine, individuals are connected through a location-based social network (LBSN) [36] as many questions are locality sensitive. Unlike previous

[1]https://www.tripadvisor.com

studies, in this work we consider the problem of searching local experts according to his/her microblog contents. Given a location q, a distance r, and a set of keywords W, our goal is to find the top-k users who have posted tweets relevant to the desired keywords in W at a place within the distance r from q. We call this the *Top-k Local User Search* (TkLUS) problem. The following example is reproduced from Paper C.

Example 4.1.1 (Example of Searching Local Experts)
Figure 4.1 shows a map where a TkLUS query is issued at the location indicated by the cross (43.6839128037, -79.37356590) with a single keyword "hotel" and a range of 10 km. The tweets containing "hotel" (and their users) are listed in Table 4.1, and their locations are also indicated in the map. According to different user scoring functions, user u_1 having more tweets or user u_5 having more replies/forwards (not shown in the table) will be returned.

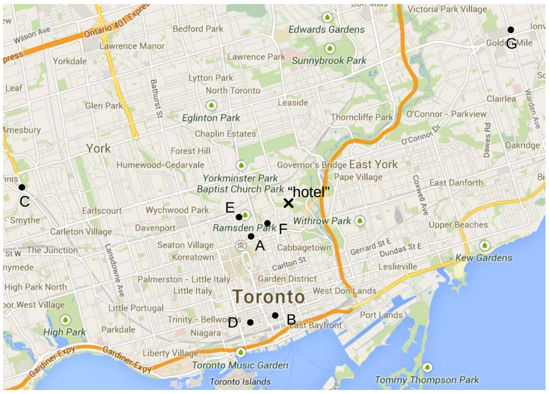


Fig. 4.1: TkLUS Example: Locations of Query and Tweets [15]. ©2015 IEEE

To find the top-k experts is a non-trivial problem under existing settings. The user set U and the post set P are very large on many social media platforms like Twitter. Meanwhile, it is not straightforward to measure the relevance of a user to a given query. Furthermore, an effective ranking mechanism need to be designed for a TkLUS query. In this paper, we systematically tackle the challenges of these data management issues.

The contributions of this work can be summarized as follows.

- We formulate *top-k local user search* (TkLUS) query to find local social media users who can provide relevant information to search needs.

pid	**uid**	**text**
A	u_1	I'm at Toronto Marriott Bloor Yorkville Hotel
B	u_2	Finally Toronto (at Clarion Hotel).
C	u_3	I'm at Four Seasons Hotel Toronto.
D	u_4	Veal, lemon ricotta gnocchi @ Four Seasons Hotel Toronto.
E	u_5	And that was the best massage I've ever had.(@ The Spa at Four Seasons Hotel Toronto)
F	u_6	Saturday night steez #fashion #style #ootd #toronto #saturday #party #outfit @ Four Seasons Hotel Toronto.
G	u_1	Marriott Bloor Yorkville Hotel is a perfect place to stay.

Table 4.1: Detailed Information of Example Tweets [15]. ©2015 IEEE

- We propose *tweet thread* to capture the query-dependent popularity of tweets, and devise two local user scoring functions that integrate social relationships, keyword relevance and spatial proximity in ranking local users for answering a TkLUS query.
- We design a hybrid index, which is aware of keywords as well as locations, for organizing high volume geo-tagged tweets.
- We devise an effective upper bound score and two efficient algorithms for processing TkLUS queries according to the proposed user scoring functions.
- We conduct extensive experiments on real Twitter data sets to evaluate our proposals.

4.2 Top-*k* Local User Search in Social Media

In this section, We firstly introduce the definition of social media post and then formalize TkLUS problem. The following definitions and examples in this section are reproduced from Paper C.

Definition 4.2.1 (Social Media Post)
(**Social Media Post**) A social media post is a 4-tuple $p = (uid, t, l, W)$, where *uid* identifies the user who publishes the post, t is the timestamp when the post is published, l is the location where the user publishes the post, and W is a set of words $(w_1, w_2, \ldots, w_n)$ that capture the textual content of the post.

In this paper, we focus on social media posts that have non-empty location fields, and make use of them to find relevant results for user queries. The size of $p.W$ is always small since social media platforms impose length constraints on posts. For example, a tweet contains at most 140 characters that can only convey a small number of words. On the other hand, not all textual words in an original social media post are included in the abstraction p. We assume the use of a vocabulary $\mathcal{W}$ that excludes popular stop words (e.g., this and that).

In the context of geo-tagged social media data consisting of post set P and user set U, we reproduce the following TkLUS problem formulation from Paper C.

Definition 4.2.2 (Top-*k* Local User Search in Social Media)
(**TkLUS problem**) In the context of geo-tagged social media data $\mathcal{D} = (P, U)$, given a query $q(l, r, W)$ where $q.l$ is a query location, $q.r$ is a distance value, and $q.W$ is a keyword set that captures user needs, a top-k local user search (TkLUS) finds a k-user set $\mathcal{E}_k \subseteq \mathcal{D}.U$ that satisfies the following conditions:

1. $\forall u \in \mathcal{E}_k, \exists p \in P_u$ such that $\|q.l, p.l\| \leq q.r$ [2] and $p.W \cap q.W \neq \emptyset$.
2. $\forall u \in \mathcal{E}_k$ and $\forall u' \in \mathcal{D}.U \setminus \mathcal{E}_k$, either u' does not satisfy condition 1 or $score(u', q) \leq score(u, q)$.

4.3 System Pipeline

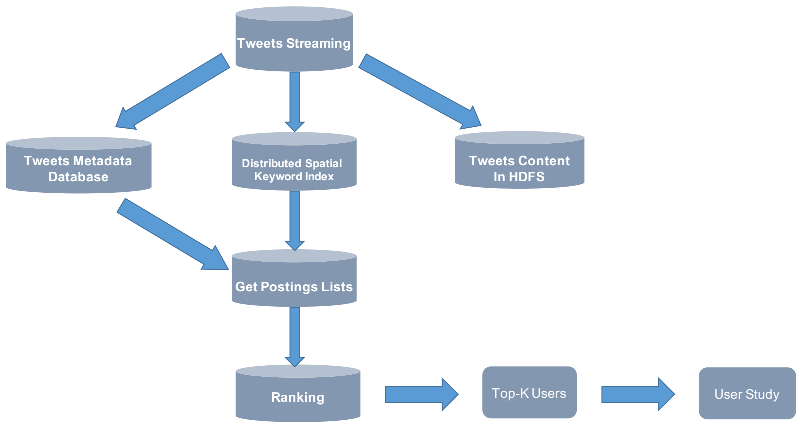


Fig. 4.2: System Pipeline

[2] $\|l_1, l_2\|$ denotes the Euclidean distance between locations l_1 and l_2. The techniques proposed in this paper can be adapted to other distance metrics.

The system pipeline is shown in Figure 4.2. We apply Twitter Rest API to crawl JSON-format tweets and then periodically collect the spatial tweets. A scalable hybrid index consisting of both spatial and text information has been built under Hadoop MapReduce framework and stored in Hadoop distributed file system (HDFS) with original spatial tweets for retrieval. The details of Index construction and query processing will be introduced in the following section. Finally, system efficiency and effectiveness are evaluated.

4.4 Hybrid Index Construction

The hybrid index is illustrated in Figure 4.3 reproduced from Paper C. It contains two components: forward index and inverted index. Each entry in the forward index is in the format of $\langle ge_i, kw_i \rangle$ where ge_i is a geohash code and kw_i refers to a keyword. The forward index is kept in the main memory and associates each of its entry to a postings list $\mathrm{P_i}$ in the inverted index that is stored in Hadoop HDFS. The postings list store tweet IDs linking to the tweets content in tweet database.

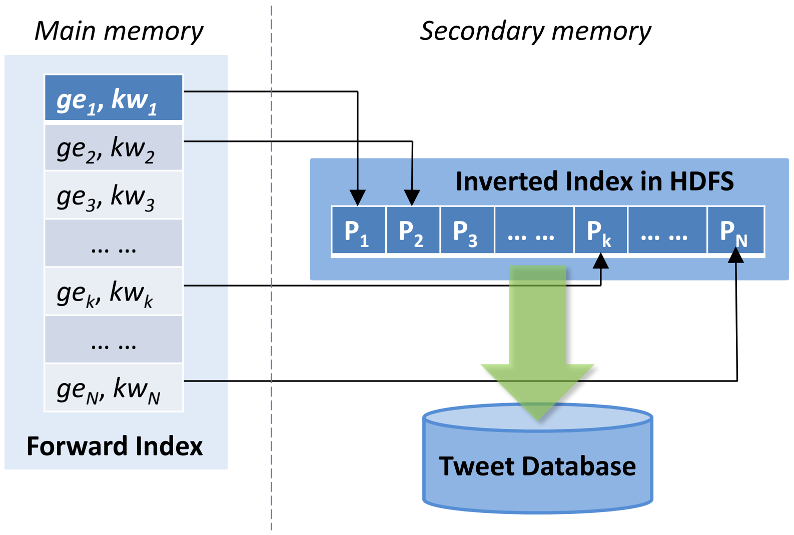


Fig. 4.3: Index Structure [15]. ©2015 IEEE

A MapReduce job is run over the inverted index files in order to construct the forward index that keeps track of the position of each postings list in HDFS. The composite key $\langle geohash, term \rangle$ is sorted under MapReduce programming framework so that close spatial points associated with the same keyword are probably stored in contiguous disk pages. Under such setting, the processing time of geo-related queries will be evidently reduced as the postings lists belonging to close areas are stored together.

4.5 Query Processing

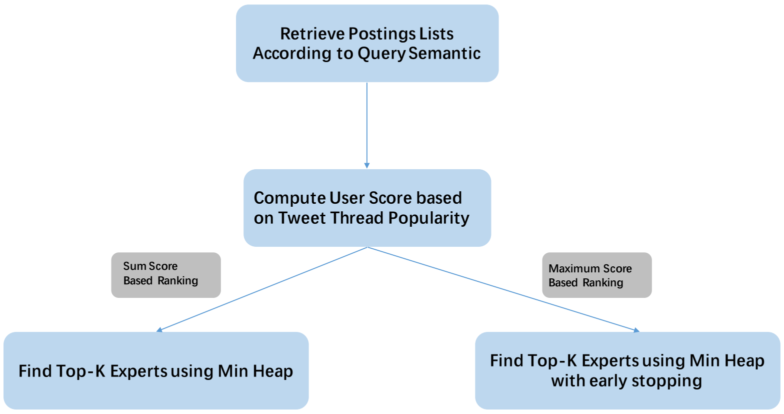

Retrieve Postings Lists According to Query Semantic
Compute User Score based on Tweet Thread Popularity
Sum Score Based Ranking
Maximum Score Based Ranking
Find Top-K Experts using Min Heap
Find Top-K Experts using Min Heap with early stopping

Fig. 4.4: TkLUS Query Processing

The general query processing strategy is depicted in Figure 4.4. The first step is to retrieve corresponding postings lists according to the query location and keywords. Then each candidate user is assigned a score according to different ranking mechanism: sum score and maximum score based ranking. Both user scoring approaches are based on the tweet thread popularity.

Figure 4.5 reveals an example of tweet thread and thread score is computed as in Definition 4.5.1. In Definition 4.5.1, $T.h$ means the height of the tweet thread and $|T_i|$ represents the number of forward/replied tweets in the i-th level. In particular, i starts from 1 at the top level in the tweet thread. In addition, ϵ is a smoothing parameter for a tweet thread consisting only one tweet. For example, in Figure 4.5, the thread score of tweet p_1 is $3 \times \frac{1}{2} + 4 \times \frac{1}{3} + 2 \times \frac{1}{4} = \frac{10}{3}$. For a user having several tweets, the sum score based ranking adds up all thread scores while the maximum score based ranking only considers the largest thread score.

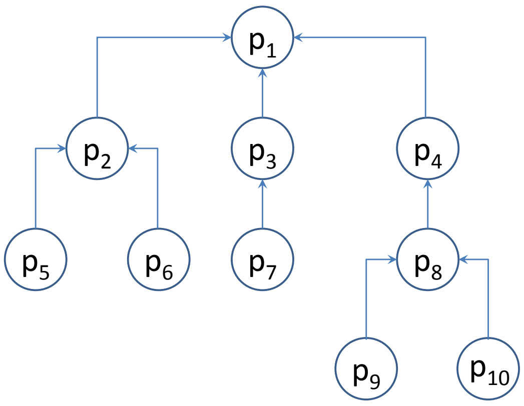


Fig. 4.5: Tweet Thread Example [15]. ©2015 IEEE

Definition 4.5.1 (Popularity of Tweet)

$$\phi(p) = \begin{cases} \epsilon, & \text{if } T.h = 1; \\ \sum_{i=2}^{n} |T_i| \times \frac{1}{i}, & \text{otherwise.} \end{cases}$$

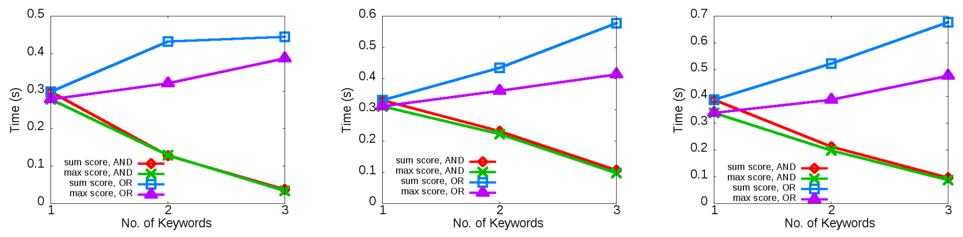
Finally we find top-k experts who have largest scores by maintaining a k-element priority-queue using min heap. For maximum score based ranking, upper bound popularity of one's tweet thread can be estimated for candidate pruning during query processing. Details of the algorithms will be illustrated in Paper C.

4.6 Experiments

In our experiments, queries with 1 to 3 keywords are chosen from AOL query logs and we randomly combine keywords and sampled locations to form a 90-query set.

4.6.1 Evaluation of Different Query Processing Strategies

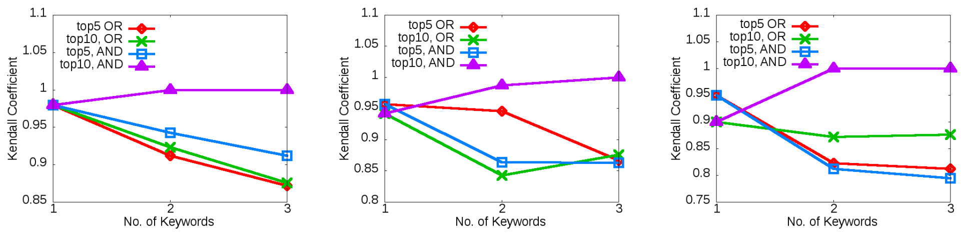
To evaluate the efficiency of designed mechanism, we study different combinations of user ranking strategy (maximum score based ranking or sum score based ranking) and query semantic (OR or AND semantic for multiple keywords). Figure 4.6 reveals that in general, the maximum score based user ranking outperforms sum score based alternative especially for large query range and OR semantic. The reason is that there are more room for pruning strategy of sum score based ranking mechanism to take effect.



(a) 10 km query range (b) 20 km query range (c) 50 km query range

Fig. 4.6: Query Efficiency with Multiple Query Keywords [15]. ©2015 IEEE

4.6.2 Consistency Measurement



(a) 10 km query range (b) 20 km query range (c) 50 km query range

Fig. 4.7: Kendall Tau for Multiple Query Keywords [15]. ©2015 IEEE

The result is shown in Figure 4.7 indicating the Kendall tau coefficient ranges from 0.8 to 0.95 for all considered setting. This suggests the consistency of two ranking strategies.

4.6.3 User Study

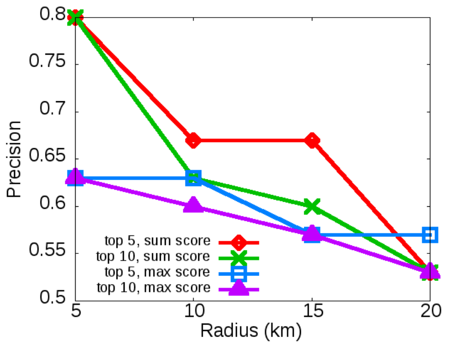


Fig. 4.8: User Study Results [15]. ©2015 IEEE

We adapt precision as the effectiveness metric in the user study. Figure 4.8 reveals the precision of top-5 and top-10 query results. Precision in our context is defined as the fraction of the returned users that are regarded as relevant expert by the user study. Overall, both user ranking methods (sum score based and maximum score based) are very effective for query ranges not larger than 10 km with precision from 60% to 80%.

Chapter 5

Location Based Influencer Targeting for O2O Marketing

This chapter gives an overall introduction of Paper D [14]. The chapter reuses content from the paper when that was considered most effective.

5.1 Problem Motivation and Statement

With the rapid change of media form, consumers' attitudes towards advertising have been different. They are more accepting ads with high-quality content than traditional ones. Whether on a social platform, a video site, or a news aggregation app, creative content from influencers is always the most important carrier of advertising. People follow influencers because their content matches the follower's interest, taste or opinion. In such way, "influencer marketing" is an effective means to build a bridge between online world and brick-and-mortar stores. Influencer marketing generally means promoting offline services or products via social media influencers. Influencers are also called key opinion leaders who can have a strong influence on certain target audience and shape user opinions on a specific product or service.

In order to select local influencers by utilizing their published contents at Twitter, there are three main technical challenges we need resolve: effectively profiling users with topics of interest and representative locations [26]; hybrid index construction considering user profile information, location and social influence; efficient and effective query processing algorithm design. Traditionally there are three user profiling techniques by mining tweets data: hashtag-based, entity-based and topic-based [26] [1]. Geo-social models [4, 6, 7] integrate social friendship and the locations through some creative way for new applications. Another related work [11] considers a GeoSN query that returns to a user the nearby friends sharing common interests. In this work, we consider the problem of searching local influencers as key opinion leaders around certain topics of interests. Given a topic list tl, a location l, a query range r and a threshold δ, query $q(tl, l, r, \delta)$ returns a set of k social media users who have the highest social influences within the distance r from l and with topic based query-user similarity of at least δ. We call this the *Top-k Influential Similar Local Query* (TkISL) problem. The following example is reproduced from Paper D. Table 5.1 shows a sample of user profiles in our experiments.

Example 5.1.1 (Example of Selecting Local Influential Users at Topic Level) Table 5.1 shows a sample of user profiles in our experiments. Suppose that k equals to 1, the query q's topic list is $\langle$*music*, *entertainment*, *cartoon*, *sports*$\rangle$, and the similarity threshold is 0.3. In this example, $sim(q, A) = sim(q, D) = sim(q, F) = 0.4$, and $sim(q, E) = \frac{1}{3}$. Only these four users qualify the similarity threshold 0.3. Among them, user F has the highest social influence and therefore F is returned in the query result.

User	(Lat., Lon.)	Profile	Influence
A	(24.2,-79.4)	business, sports, music	0.33
B	(46.6,118.4)	food, fiction, film	0.67
C	(34.6,137.2)	poetry, art, dance, internet	0.33
D	(37.1,114.5)	culture, sports, entertainment	0.26
E	(52.4, -2.5)	entertainment, music, dance, internet	0.43
F	(42.5,-72.7)	cartoon, music, internet	0.87
G	(33.6, -7.8)	fashion, opera, music	0.67
H	(-42.1,172.4)	culture, design	0.25

Table 5.1: Sample Data of User Profiles [14]. ©2017 IEEE

The contributions of this work can be summarized as follows.

- We formulate *Top-k Influential Similar Local Query* to find local Twitter users who have interests similar to the query and have highest social influence scores.
- We present techniques for effectively profiling and indexing users of geo-tagged social media with respect to the contents published in their social data.

- We design two query processing methods to process TkISL queries using the user management techniques.
- We conduct extensive experiments on real data sets to evaluate our proposals.

5.2 Top-k Influential Similar Local Query

We firstly give out the basic definitions and formulate the TkLUS problem. The following definitions on social media post and user are reproduced from Paper D.

Definition 5.2.1 (Social Media Post)
A social media post is a 4-tuple $p = (uid, t, l, W)$. It means that user identified by uid publishes the post in location l at time t, and the textual content of the post is captured as a set of words $W = (w_1, w_2, \ldots, w_n)$.

Definition 5.2.2 (Social Media User)
A social media user is captured as a list of tuples (uid, l, ψ), where uid is the user's identifier, and ψ is the user's profile that captures the user's social media posts published in the region represented by l. [1]

Definition 5.2.3 (User Profile)
A user's profile $\psi = (t_1, t_2, \ldots, t_n)$ is a list of topics.

The topics are generated by mining user interests based on user's published tweets. According to the definition, one user is about to have several representative locations and each location is associated with a user profile.

With the above definitions and the context of geo-tagged social media data consisting of post set P and user set U, we reproduce the following TkISL problem formulation from Paper C.

Definition 5.2.4 (Top-k Influential Similar Local Query)
In the context of geo-tagged social media data $\mathcal{D} = (P, U)$, given an integer k, a TkISL query $q(tl\langle t_1, t_2, ...t_n\rangle, l, r, \delta)$ finds a k-subset $\mathcal{E}_k \subseteq U$ such that:

1. $\forall u \in \mathcal{E}_k$, $\|q.l, u.l\| \leq q.r$ [2], and $sim(q.tl, u) \geq \delta$.
2. $\forall u \in \mathcal{E}_k$ and $\forall u' \in U \setminus \mathcal{E}_k$, $s(u') \leq s(u) \vee \|q.l, u'.l\| > q.r \vee sim(q.tl, u') < \delta$.

[1] To ease the presentation, we assume each user is represented by a single 3-tuple (uid, l, ψ) when giving the definitions. In our implementation of user indexing and searching, a user with m ($m \geq 1$) representative regions are represented by m such 3-tuples and thus treated as m different users.

[2] $\|l_1, l_2\|$ denotes the Euclidean distance between locations l_1 and l_2. Nevertheless, the proposed techniques can be extended to other distance metrics like Manhattan and network distances. When a user's representative region is involved, we calculate the distance using the region center.

5.3 Index Construction

This section covers the data management techniques utilized for user profiling and indexing.

5.3.1 User Profiling

Figure 5.1 depicts a hybrid approach of mining user interests of topics. Given a user, the first step is to execute entity detection and disambiguation on his/her posts (the URL embedded in tweets could be explored for enriching the contents). The second phase employs a multilingual topic detection to assign tags to the tweet texts by utilizing knowledge bases ncluding Wikipedia, Freebase [3] and DBPedia [4]. A user that mentions "Michael Jordan", "Los Angeles Lakers", "Spurs" can be tagged with "Basketball" and "Sports". There are two-level abstraction of the assigned tags: "topics" and "coarseTopics". For example, "Basketball" is at "topics" level while "Sports" is at "coarseTopics" level. As a result, this user profile is generated with topics at different levels.

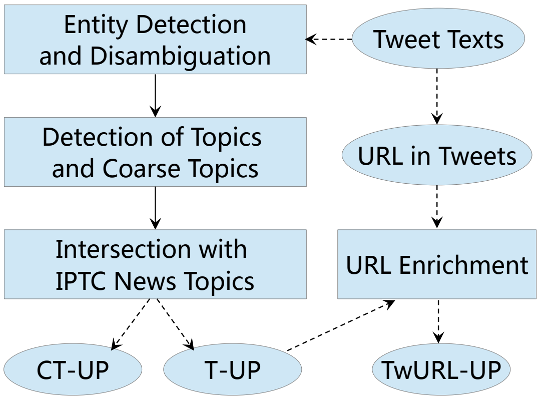

Entity Detection and Disambiguation
Tweet Texts
Detection of Topics and Coarse Topics
URL in Tweets
Intersection with IPTC News Topics
URL Enrichment
CT-UP
T-UP
TwURL-UP

Fig. 5.1: Topic-Based User Profiling [14]. ©2017 IEEE

According to Definition 5.2.2, each user will be assigned a list of representative regions. For a cluster of l locations $S_i = (s_{i1}, s_{i2}, \ldots, s_{il})$, we use the centroid of the cluster to capture the representative region r_i, i.e., $r_i.x = \sum_{j=1}^{l} s_{ij}.x/|S_i|$, and $r_i.y = \sum_{j=1}^{l} s_{ij}.y/|S_i|$. For a user having n tweets with geo information, we adapt the grid-based clustering algorithm proposed in [35] to generate the clustering result and calculate the representative regions. It is noteworthy that we represent each user as m 3-tuples (u, r_i, ψ_i)

[3]https://www.freebase.com/
[4]http://wiki.dbpedia.org/

($0 \leq i < m$) where user profile ψ_i is generated from the user's tweets posted within region r_i.

5.3.2 Indexing Users

We adapt the IR^2-tree [8] to index users which is essentially an R-tree associated with signature file in each node. Specifically, each signature file is a bitmap representing the user profile. The length of all such bitmaps is equal to the total number of topics being tagged for all users. Regarding the tree construction process, the main difference with the original IR^2-tree is the way of choosing a sub-node to insert a new object/entry in functions of ChooseSubtree(.) and Split(.). To be detailed, if ties exist, the adapted IR^2-tree will choose the sub-node that has a smaller bitmap similarity with the new object/entry. The bitmap similarity between bitmap signatures bt_i and bt_j is illustrated in the following equation where Function Count(.) represents the number of 1's.

$$sim(bt_i, bt_j) = \frac{Count(bt_i \wedge bt_j)}{Count(bt_i \vee bt_j)}. \tag{5.1}$$

5.4 Query Processing

The general procedure of baseline approach is divided into two phases. Firstly a candidate set is generated according to the query range and hyper-parameter δ representing similarity threshold. To be detailed, we define a query-node similarity as in Definition 5.4.1 and if it is less than similarity threshold, the corresponding node can be pruned during query processing. Finally we rank the candidate users by their social influence scores.

Definition 5.4.1 (Query-Node Similarity)
Given a TkISL query $q(tl\langle t_1, t_2, ...t_n\rangle, l, r, \delta)$ and a user set S_U, the query-node similarity is $sim(q, S_U) = \frac{|q.tl \cap \bigcup_{u_j \in S_U} u_j.\psi|}{|q.tl|}$.

To avoid the costly I/O in two-phase query processing, we propose a single-phase strategy which is depicted in Figure 5.2. We augment the tree node with a social influence score as in Definition 5.4.2. Therefore, the influence score of node entry e indicates the upper bound of all users under it. Finally we find top-k influencers who have largest scores by maintaining a k-element priority-queue using min heap. The pruning strategy is that If the influence score of visiting node is less than the maintaining k elements , the corresponding node can be pruned during query processing.

Definition 5.4.2 (Node Influence Score)
Given a node entry e,

1. If e indexes an object, $e.influence$ is the corresponding user's social influence.
2. Otherwise, $e.influence = \max\{e_i.influence \mid e_i \text{ is } e'\text{s sub-node entry}\}$.

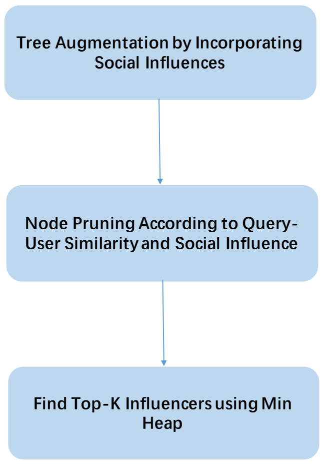

Tree Augmentation by Incorporating Social Influences
Node Pruning According to Query-User Similarity and Social Influence
Find Top-K Influencers using Min Heap

Fig. 5.2: TkLUS Query Processing [14]. ©2017 IEEE

5.5 Experiments

5.5.1 Effect of Profiling Techniques

In this part, we compare the effects of three different techniques for user profiling. From Figure 5.3, we can see that compared to CT-UP (coarse topic based) and T-UP (topic based), the profiling effectiveness of TwURL-UP (topic based with URL enrichment) is evidently enhanced. For example when the similarity threshold is 0.3, the number of user indexed by TwURL-UP is nearly twice as many as T-UP approach.

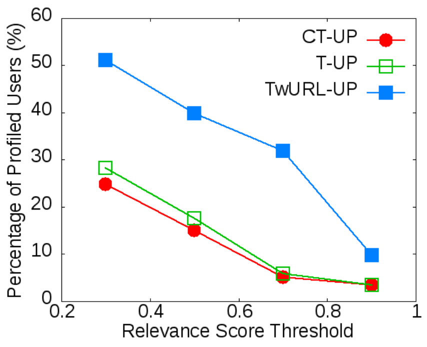


Fig. 5.3: User Profiling Effectiveness [14]. ©2017 IEEE

5.5.2 Evaluation of Different Query Processing Strategies

In this part, we compare the two query processing strategies under TwURL-UP profiling setting and vary the similarity threshold from 0.3 to 0.9. From Figure 5.4, due to the early pruning capability, the single-phase strategy outperforms the two-phase alternative regarding the average query time. Another interesting finding is that the less query time, the higher similarity threshold we set. It is sufficient to conclude that due to the higher similarity threshold, the more tree nodes are pruned at early stage.

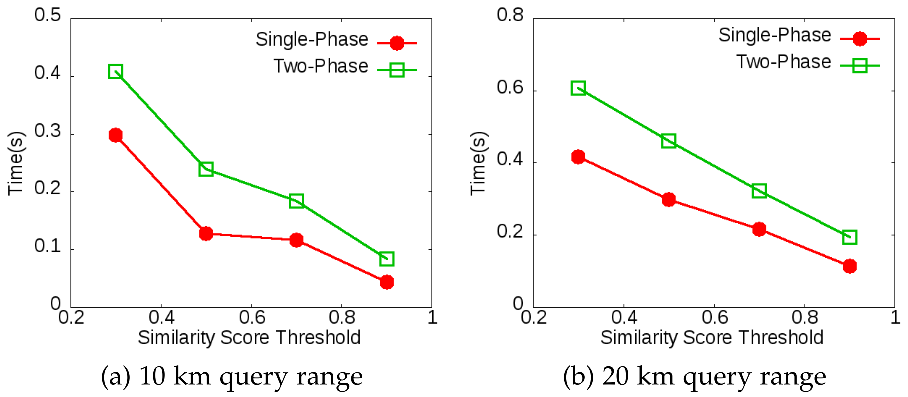
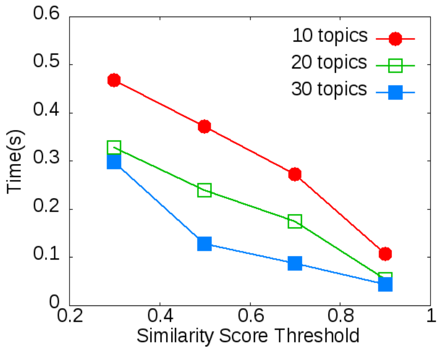


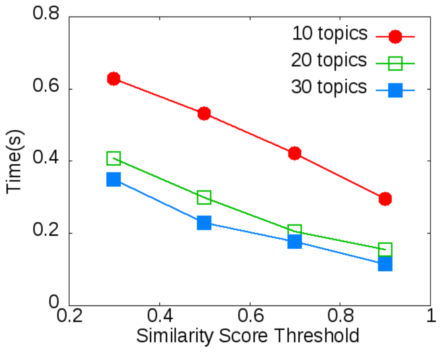
Fig. 5.4: Query Processing Efficiency [14]. ©2017 IEEE

5.5.3 Evaluation of Single-Phase Query Processing

Finally, we dive into the evaluation of single-phase query processing under TwURL-UP profiling setting and vary the similarity threshold from 0.3 to 0.9. From Figure 5.5, we see again the higher similarity threshold leads to early pruning for a shorter query time. In addition, the more number of topics, the less query processing time. The reason is that query-node similarity will be lower, leading to more aggressive node pruning.



(a) 10 km query range



(b) 20 km query range

Fig. 5.5: Single-Phase using TwURL-UP [14]. ©2017 IEEE

5.5.4 User Study

We adapt top-10 precision as the effectiveness metric in the user study and the results are shown in Table 5.2. Precision in our context is defined as the fraction of the returned users that are regarded as local influencers by the user study. Generally the corresponding query effectiveness of locality-sensitive topic "football" and "Chelsea Football Club" is 90% and 70%. In contrast, for a locality-insensitive topic (e.g., automobiles or entertainment), the top-*k* query effectiveness is less than 50%. This discrepancy indicates that it is effective for system to select local influencer around certain topic if there are enough local users and tweets discussing around the topic. It is evident that the TkISL query is able to find local influencers for locality-sensitive topics effectively.

Topic	Precision
football	90%
Chelsea Football Club	70%
music	60%
automobiles	40%
entertainment	30%

Table 5.2: User Study Result [14]. ©2017 IEEE

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis offers various data mining and data management techniques in order to provide marketing intelligence for online advertising. There are two problems each addressed in the context of display advertising and O2O commerce. We address precise audience targeting and personalized advertising recommendation by exploiting user behaviour data in display advertising. For O2O commerce, we address local traffic diversion and influencer targeting by exploiting user-generated data in social media platform. Each problem is formulated and addressed in one paper, and four papers are included in the thesis. A summary of each paper is given below.

- Paper A [13] proposes a new end-to-end solution for audience expansion problem. Specifically, we present a neural prediction framework and leverage it with the intuitive audience feature extraction stages. We investigate various sampling techniques in extremely imbalanced dataset. Extensive study is conducted on a large advertising dataset and the results demonstrate the advantage of the proposed approach. Finally we get evident performance improvement when testing the solution under a real business scenario.

- Paper B [12] designs a novel Deep Monitor Network (DMN) to tackle the challenges of predicting click-through rate under privacy-preserving setting. Instead of utilizing user profile embedding to extract user representation, we extend the recent deep network structures used in CTR prediction by a sophisticated feature extraction module and historical training data compression. Specifically, DMN separately leverage the clicked/positive and unclicked/negative samples and execute feature extraction in a creative way in order to learn the representation of user historical actions. The experimental results on private and public datasets reveal DMN outperforms state-of-art CTR models in all considered settings.

- Paper C [15] focuses on finding experts for local traffic diversion. We formalize a TkLUS query and propose two local user ranking approaches that integrates text relevance, social influence and location proximity. A hybrid index has been constructed under a scalable MapReduce framework, which is aware of keywords as well as locations, to organize high volume geo-tagged tweets. Corresponding to different ranking mechanisms, two query processing algorithms are devised for processing TkLUS queries and efficient pruning strategies are also presented. Finally we conduct an experimental study using real tweet dataset to evaluate the proposed techniques. The experimental results demonstrate the efficiency, effectiveness and scalability of our proposals.

- Paper D [14] put emphasis on finding local influencer around certain topic for O2O marketing. We design three hybrid user profiling techniques, an indexing tree, and an upper bound query-user similarity that enables efficient pruning in query processing. To process TkISL queries, we propose a baseline approach directly utilizing the indexing tree and the upper bound for pruning and a more efficient improved method, whereas a more efficient method speeds up the query processing by enhancing the tree structure and pruning strategy. Finally, the paper reports on extensive experiments with real twitter dataset, to offer insight its performance , showing that the proposed approach excels over different baselines and get excellent effectiveness.

6.2 Future Work

Several directions exist for future research. The rich information recorded in advertising platforms could be harnessed to investigate more sophisticated lookalike and CTR models. Meanwhile It is also challenging and meaningful to automatically incorporate the capability of invalid traffic identification into deep learning models. For those users having not enough behavioural data, there can be a possibility for us to exploit short-term sequence features in a time-series perspective.

Another direction is to extend existing works by introducing temporal considerations in mining and managing the user behaviour data. Both user-generated data on advertising and social media platforms have time-stamp attribute. For example, we can introduce recency weight in model learning and query processing for various applications. Furthermore, how to build scalable learning model and index construction in real-time system setting could also be an interesting problem.

References

[1] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Analyzing user modeling on twitter for personalized news recommendations," in *UMAP*, 2011, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=2021855.2021857

[2] P. P. K. Chan, X. Hu, L. Zhao, D. S. Yeung, D. Liu, and L. Xiao, "Convolutional neural networks based click-through rate prediction with multiple feature sequences," in *Proc. of IJCAI*, 2018, pp. 2007–2013.

[3] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. of the 1st Workshop on Deep Learning for Recommender Systems*, 2016, pp. 7–10.

[4] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD*, 2011, pp. 1082–1090.

[5] L. Derczynski, B. Yang, and C. S. Jensen, "Towards context-aware search and analysis on social media data," in *EDBT*, 2013, pp. 137–142.

[6] Y. Doytsher, B. Galon, and Y. Kanza, "Querying geo-social data by bridging spatial networks and social networks." in *GIS-LBSN*, 2010, pp. 39–46. [Online]. Available: http://dblp.uni-trier.de/db/conf/gis/lbsn2010.html#DoytsherGK10

[7] ——, "Querying socio-spatial networks on the world-wide web," in *WWW (Companion Volume)*, 2012, pp. 329–332.

[8] I. D. Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *ICDE*, 2008, pp. 656–665.

[9] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," in *Proc. of IJCAI*, 2017, pp. 1725–1731.

[10] D. Horowitz and S. D. Kamvar, "The anatomy of a large-scale social search engine," in *WWW*, 2010, pp. 431–440.

[11] Q. Huang and Y. Liu, "On geo-social network services," in *Geoinformatics*, 2009, pp. 1–6.

[12] J. Jiang, X. Lin, J. Yao, R. Ding, H. Lu, and X. Wu, "Deep monitor network for privacy-preserving click-through rate prediction (submitted)," in *The World Wide Web Conference, WWW 2020, Taipei, Apr 20-24, 2020*, 2020.

[13] J. Jiang, X. Lin, J. Yao, and H. Lu, "Comprehensive audience expansion based on end-to-end neural prediction," in *Proceedings of the SIGIR 2019 Workshop on eCommerce, co-located with the 42st International ACM SIGIR Conference on Research and Development in Information Retrieval, eCom@SIGIR 2019, Paris, France, July 25, 2019.*, 2019.

[14] J. Jiang, H. Lu, P. Li, G. Pan, and X. Xie, "Finding influential local users with similar interest from geo-tagged social media data," in *18th IEEE International Conference on Mobile Data Management, MDM 2017, Daejeon, South Korea, May 29 - June 1, 2017*, 2017, pp. 82–91.

[15] J. Jiang, H. Lu, B. Yang, and B. Cui, "Finding top-k local users in geo-tagged social media data," in *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, 2015, pp. 267–278.

[16] Y. Juan, D. Lefortier, and O. Chapelle, "Field-aware factorization machines in a real-world online advertising system," in *Proc. of WWW Companion*, 2017, pp. 680–688.

[17] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proc. of RecSys*, 2016, pp. 43–50.

[18] A. Kaboutari, S. Branch, J. Bagherzadeh, I. Urmia, and F. Kheradmand, "An evaluation of two-step techniques for positive-unlabeled learning in text classification," vol. 3, 2014, pp. 592–594.

[19] C.-H. Lee, H.-C. Yang, T.-F. Chien, and W.-S. Wen, "A novel approach for event detection by mining spatio-temporal information on microblogs," in *ASONAM*, 2011, pp. 254–259.

[20] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *Proc. of ICDM*, 2003, pp. 179–186.

[21] B. Liu, R. Tang, Y. Chen, J. Yu, H. Guo, and Y. Zhang, "Feature generation by convolutional neural network for click-through rate prediction," in *The World Wide Web Conference*, 2019, pp. 1119–1129.

[22] B. Liu, W. S. Lee, P. S. Yu, and X. Li, "Partially supervised classification of text documents," in *Proc. of ICML*, 2002, pp. 387–394.

[23] H. Liu, D. Pardoe, K. Liu, M. Thakur, F. Cao, and C. Li, "Audience expansion for online social network advertising," in *Proc. of KDD*, 2016, pp. 165–174.

[24] Q. Liu, F. Yu, S. Wu, and L. Wang, "A convolutional click prediction model," in *Proc. of CIKM*, 2015, pp. 1743–1746.

[25] A. Mangalampalli, A. Ratnaparkhi, A. O. Hatch, A. Bagherjeiran, R. Parekh, and V. Pudi, "A feature-pair-based associative classification approach to look-alike modeling for conversion-oriented user-targeting in tail campaigns," in *Proc. of WWW*, 2011, pp. 85–86.

[26] M. Michelson and S. A. Macskassy, "Discovering users' topics of interest on twitter: a first look," in *AND*, 2010, pp. 73–80.

[27] F. Mordelet and J. P. Vert, "A bagging svm to learn from positive and unlabeled examples," *Pattern Recogn. Lett.*, vol. 37, pp. 201–209, Feb. 2014.

[28] W. Ouyang, X. Zhang, L. Li, H. Zou, X. Xing, Z. Liu, and Y. Du, "Deep spatio-temporal neural networks for click-through rate prediction," in *Proc. of KDD*, 2019, pp. 2078–2086.

[29] J. Pan, J. Xu, A. L. Ruiz, W. Zhao, S. Pan, Y. Sun, and Q. Lu, "Field-weighted factorization machines for click-through rate prediction in display advertising," in *Proc. of WWW*, 2018.

[30] S. Pandey, M. Aly, A. Bagherjeiran, A. Hatch, P. Ciccolo, A. Ratnaparkhi, and M. Zinkevich, “Learning to target: What works for behavioral targeting,” in *Proc. of CIKM*, 2011, pp. 1805–1814.

[31] Y. Qu, B. Fang, W. Zhang, R. Tang, M. Niu, H. Guo, Y. Yu, and X. He, “Product-based neural networks for user response prediction over multi-field categorical data,” *ACM Trans. Inf. Syst.*, vol. 37, no. 1, pp. 5:1–5:35, Oct. 2018.

[32] J. Shen, S. C. Geyik, and A. Dasdan, “Effective audience extension in online advertising,” in *Proc. of KDD*, 2015, pp. 2099–2108.

[33] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen, “How much can behavioral targeting help online advertising?” in *Proc. of WWW*, 2009, pp. 261–270.

[34] W. Zhang, T. Du, and J. Wang, “Deep learning over multi-field categorical data -- A case study on user response prediction,” in *Proc. of ECIR*, 2016, pp. 45–57.

[35] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, “Collaborative location and activity recommendations with GPS history data,” in *WWW*, 2010, pp. 1029–1038.

[36] Y. Zheng, “Location-based social networks: Users,” in *Computing with Spatial Trajectories*, 2011, pp. 243–276.

Part II

Papers

Paper A

Comprehensive Audience Expansion based on End-to-End Neural Prediction

Jinling Jiang, Xiaoming Lin, Junjie Yao, Hua Lu

The paper has been published in the
Proceedings of the ACM SIGIR 2019 Workshop on eCommerce

Abstract

In current online advertising applications, lookalike methods are valuable and commonly used to identify new potential users, tackling the difficulties of audience expansion. However, the demographic information and a variety of user behavior logs are high dimensional,noisy, and increasingly complex, which are challenging to extract suitable user profiles. Usually, rule-based and similarity-based approaches are proposed to profile the users' interests and expand the audience. However, they are specific and limited in more complex scenarios.

In this paper, we propose a new end-to-end solution, unifying the feature extraction and profile prediction stages. Specifically, we present a neural prediction framework and leverage it with the intuitive audience feature extraction stages. We conduct extensive study on a real and large advertisement dataset. The results demonstrate the advantage of the proposed approach, not only in accuracy but also generality.

A.1 Introduction

The stunning growth of online advertisement enables the advertisers to sync up their products according to the fast changing needs of consumer. As the development of e-commerce platforms has introduced SMEs to enter consumers' sight, large enterprise advertisers face the crisis of slowing business growth and falling revenue. Therefore, brand advertisers have begun to pay more attention to the contribution of advertising to sales conversion, the actual revenue brought by advertising, requiring advertising agencies and third-party suppliers to provide more refined performance data of advertising effects.

Meanwhile, the emergence of big data technology has subverted the operation model of entire advertising industry and the traditional way of evaluating advertising effects. By tracking and obtaining user behaviour data, third-party supplier of advertising monitor can analyse the data according to the advertiser needs, not only understanding the communication effects and sales conversion rate generated by the advertisement in time, but also predicting the user conversion probability to some extent. Through analysis and modeling on massive data of user behaviour, advertisers can accurately reach the target consumer. Therefore, how to better utilize the advertising monitor data in order to optimize ad serving and improve marketing conversion rate has become an important issue.

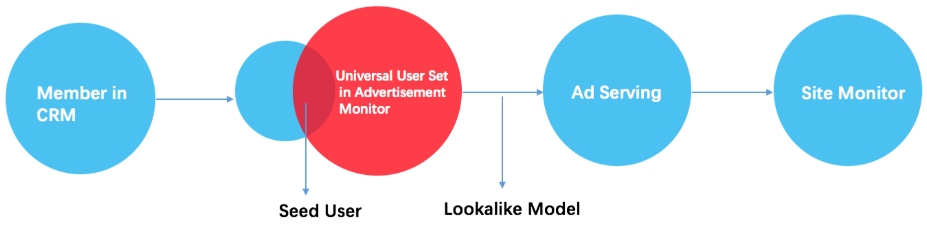


Fig. A.1: Audience Expansion Dataflow

One of the main challenges in ad serving is how to find the best converting prospects. A typical way is to do audience expansion, that is, to identify and reach new audiences with similar interests to the original target audience. Usually, the methodology used in audience expansion problem is called lookalike modeling. Given a seed user set S from a universal set U, lookalike models essentially find groups of audiences from $\|U - S\|$ who look and act like the audience in S.

But both traditional and current lookalike strategies for advertiser to look for target audience are mainly based on user demographics. There are two main problems with demographics-based audience segmentation: user demographics (age, gender and geographical location) itself is not precise as it is estimated via various statistical methods or machine learning models based on a small group of surveyed samples (10-100 thousands); the number of users that are specified by demographics is large, more sophisticated screening is required. Accordingly, the details of user behaviour data should be harnessed in machine learning models so as to target accurate audience segment. At the same time, there are two main problems that need to be solved based on user behaviour data modeling: user generated behaviour data through Internet is generally high-dimensional and sparse; advertisers usually can only provide positive samples, while negative samples need to be carefully picked up from a huge unlabeled sample set.

In addition, the ecologically closed Internet tycoons (represented by Facebook, Amazon, Tencent, Alibaba and etc.) provide for advertisers the capability to perform audience expansion within their own platforms. However, ad serving data of these platforms are not connected with the advertiser's CRM (Customer Relationship Management) system, so it is difficult to directly track the real conversion rate. In order to verify that lookalike models based on the user behaviour work better than traditional demographics-based approach regarding the sales conversation rate, we need to integrate data flow during the whole advertising life cycle.

The data flow of audience expansion service is illustrated in Figure A.1. The data runs between advertisers and our universal advertising monitor system across different media platforms. The original users come from the advertiser's CRM System selecting the consumers who recently exercise the purchase actions. Then the users who are tracked by the universal advertising monitor will be matched and treated as "seed" users. Based on the "seed" users and universal user set from advertising monitor, we build lookalike model to predict the probability to be target audience for all users. Afterwards, according to advertising budget, lookalike model will yield corresponding number of expanded users to be reached through ad serving system. Finally, the ad serving performance is evaluated by advertiser's site monitor system that record sales conversion in the near future.

In this paper, we build up a closed-loop data solution for brand advertisers and combines multiple techniques of selecting negative samples and extracting features, as well as machine learning lookalike models to reach targeted audience. which greatly enhances the conversion effect of ad serving.

The rest of the paper is organized as follows. Section A.2 gives out the formal problem statement and specifies the notations used in the paper. In Section A.3, we review the related work on various kinds of lookalike models and illustrate different design philosophy behind them. Section A.4.3 introduces our proposed lookalike models and Section A.4.4 reveals the sampling strategies. The evaluation of the algorithm is presented in Section A.5. Finally the conclusion and future work are discussed in Section A.6.

A.2 Problem Statement

We formalize the lookalike modeling as a prediction problem. Advertisers submit a list of customers, which we call seed user set S, as positive samples and there are a universal user set U existing in advertising monitor platform. Then the problem is transformed into a Positive and Unlabeled learning problem: using a small number of labeled positive samples S and a large number of unlabeled samples $U - S$ to derive a prediction classifier. Eventually unlabeled users are scored by the classifier and the target audience set T is taken out according to advertising requirements. The dataset sizes are typically configured in real business environment as follows: $\|S\| = 0.1 - 0.2M(Million)$, $\|T\| = 10 - 20M$ and $\|U\| = 2000 - 3000M$. Meanwhile, a user is represented by a feature vector which indicates the user's past behaviour collected by the advertising monitor system. The feature vector always occurs with high-dimension D and extreme sparsity. D is usually around 100-300 thousands and only 0.1 percent of the feature vector are non-zero elements.

A.3 Related Works

We briefly review the related literature of lookalike modeling. Generally in online user-targeted advertising areas, lookalike modeling which supports audience expansion system can be categorized in three lines: rule-based, similarity-based and model-based.

The rule-based approaches focus on explicit positioning, where users with specific demographic tags (age, gender, geography) or interests are targeted directly for advertiser. The core technical support in the background is user profile mining, which means, the interest tags are inferred from the user behaviour [1] [2]. Furthermore, Mangalampalli et al. [3] builds a rule-based associative classifier for campaigns with less conversion; Shen et al. [4] and Liu et al. [5] present detailed in-depth analysis of multiple methods under different considerations(such as similarity, performance, whether or not campaign-agnostic) for online social network advertising. The main disadvantage of rule-based lookalike modeling is that it only captures the high-level features, therefore loses sophisticated details of user behaviour.

The similarity-based approaches apply different similarity metrics to solve the problem of lookalike modeling. Naive similarity-based method computes pairwise similarities between and seed user and all the other users in the set while the locality-sensitive hashing (LSH) [6] technique is often applied to decrease the computation complexity of pairwise similarity. In addition, based on Ma et al. [7] [8] provide several similarity scoring methods to measure the potential value of the users to an specific advertiser. However, the similarity-based approach lacks the ability to catch the implicit interaction between features indicating user behaviour.

The model-based lookalike systems fall into two categories: unsupervised and supervised learning. For instance, k-means clustering [9] and frequent pattern mining [10] are the instances of unsupervised approach. Meanwhile, the supervised approach transforms the lookalike model into a positive-unlabeled learning (PU learning) problem [11] [12] [13] [14]. In PU learning, the positive samples are seed users while negative samples should be selected from the non-seed users. The main challenge of PU learning problem lies in three following aspects: negative samples not easy to obtain; negative samples are too diverse; negative samples are dynamically changing. In one word, different strategies on how to sample the negative users will definitely affect the model results. For example, besides random sampling, Ma et al. [7] select the past non-converter users as negative samples and Liu et al. [14] propose a "spy" method to aggregate negative users. Another challenge in model-based lookalike system is that it need have the capability to model in the very sparse feature space.

Another approach that is related to our lookalike model is collaborative filtering which is successfully applied in recommendation system . A key challenge in applying collaborative filtering lies also on the extreme sparsity of interaction between users and campaign and the way Kanagal et al. [15] address this challenge is to utilize a product taxonomy to reveal the relationships. Regarding the algorithms dealing with high-dimensional sparse data is an essential task in online advertising industry. Many models have been proposed to resolve this problem such as Logistic Regression (LR) [16] [17], lowPolynomial-2 (Poly2) [18], Factorization Machine-based models [19] [20] [21] and end-to-end deep learning related models [22] [23] [24].

A.4 The Proposed Approach

A.4.1 Feature Extraction and Analysis

Here we introduce the feature extraction and analysis for training lookalike model.

Each row of the original data collected by advertising monitor system represents an ad impression. The "CLICK" column is an indicator that shows whether or not the advertisement is clicked by the corresponding user. As

Table A.1: An example of data from advertising monitor system

CLICK	Timestamp	USER_ID	SPID
1	201809123278	66a7988f	107122831
-1	201809123346	9e664577	107108909
1	201809123456	9b3fcc94	107104618
-1	201809123787	0043fbf4	107102974
-1	201809132592	1df73293	107108909

shown in Table A.1, The main information of an ad impression includes timestamp, user_ id and an spid. The spid refers to the specific information of an advertisement where they are multi-field categorical data [25]which are commonly seen in CTR prediction and recommendation system.

The user behaviour is represented by a high-dimensional sparse feature vector where each feature corresponding to the times an advertisement is clicked or impressed. One typical feature extraction result is shown in Table A.2, User "$66a7988f$" is impressed by spid1 and spid2 both 3 times while he only clicks spid2 once. The user feature vector will be normalized afterwards. The normalization approach is as follows where $freq$ represents the original frequency and $norm\_freq$ is the frequency after normalization:

$$norm\_freq = \begin{cases} \dfrac{1}{1+exp(-\frac{freq}{10})} & freq > 0 \\ 0 & freq = 0 \end{cases} \tag{A.1}$$

To this end, every feature value is converted to a number between 0 and 1.

Table A.2: User behaviour Representation

USER_ID	spid1_click	spid1_impression	spid2_click	spid2_impression	...	label
66a7988f	0	3	1	3	...	1
9b3fcc94	0	2	1	1	...	-1
0043fbf4	1	1	0	1	...	-1
9e664577	1	3	1	2	...	1
1df73293	0	1	0	1	...	-1

It is noteworthy that the data label is the **purchase tag** (meaning the corresponding user has purchase action) from CRM system of a particular brand advertiser over a period of time, while the features represent the impression and click behaviour for ads of different brands. Unlike the high-dimensional sparse feature transformed by one-hot encoder in CTR prediction task, the original feature space is already sparse and high-dimensional.

The intuitive idea of utilizing spid as feature is that the ads are somehow correlated to the website which highly indicating user interests. That is to say, when an internet user is impressed by an specific ad, the ad itself could describe the user interests to some extend. Moreover, "CLICK" information directly connects user intention. The detailed comparison of different feature extraction methodologies will be incorporated in Section A.5.2.

A.4.2 Basic Expansion Models

Logistic Regression: Logistic Regression (*LR*) is probably the most widely used baseline classification model. Suppose there are n features $\{x_1, x_2, ..., x_n\}$ and x_i is either 0 or 1, consider an *LR* model without a regularization term:

$$y = bias + \beta^T X \tag{A.2}$$

where β is the coefficient vector. This naive linear model misses the crucial feature crosses, therefore, the Degree-2 Polynomial (Poly2) model is always provided to ease the problem.

$$y = bias + \beta^T X + XWX^T \tag{A.3}$$

where W is a symmetric parameter matrix with the elements on the diagonal are all equal to 0.

Factorization Machine In order to extract feature crosses while reducing the influence of high-dimensional sparse features, Rendle [19] proposes Factorization Machines to overcome the drawbacks of *LR*. Regarding *LR* model, the number of parameters in matrix W need to be learned is $\frac{n(n-1)}{2}$. When n is 100,000, the number of parameters is tens of billions. At the same time, when training the model using gradient descent optimization, the parameter w_{ij} can only be trained when x_i and x_j are both not zero, therefore there is a high demand on both the number of training samples and memory space at training phrase. As a result, for high-dimensional sparse features, the parameter matrix W is almost impossible to train.

To overcome this problem, we will decompose W into VV^T where each v_i in $V = (v_1, v_2, ..., v_n)^T$ can be seen as a latent k-dimensional factor of original feature. The Degree-2 *FM* model equation is defined as:

$$y = bias + \beta^T X + XVV^T X^T, V \in \mathbb{R}^{n \times k} \tag{A.4}$$

At this time, the number of parameters need to be estimated is $n \cdot k$ and easier to train even under sparsity setting as *FM* model break the independence of the interaction parameters by factorizing them.

A.4.3 Comprehensive Modeling

We continue to introduce the lookalike model techniques used in our audience expansion system.

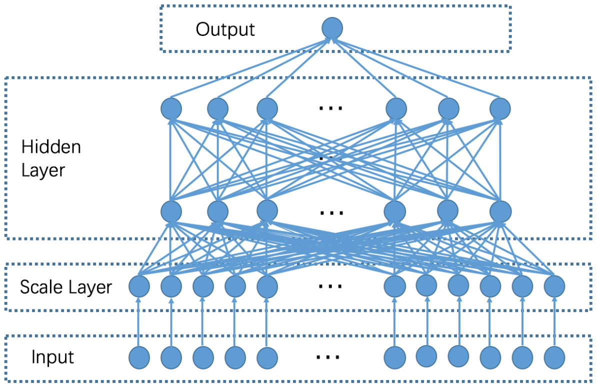


Fig. A.2: Add Scale Layer to MLP Network

Multilayer Perceptron (MLP) is a feedforward neural network consisting of several layers. By adding non-linear activation functions, MLP can fit high-order non-linear features. Figure A.2 illustrates a mlp network added by a scale layer.

The reason adding scale layer: The prediction equation of a standard MLP model is defined as:

$$y = mlp(AX + bias), A \in \mathbb{R}^{k \times n} \tag{A.5}$$

After adding a scale layer, the model we call Scale-MLP is updated as:

$$y = mlp(A(W \circ X) + bias), A \in \mathbb{R}^{k \times n} \tag{A.6}$$

The model expressibility of Equation A.5 and A.6 are the same so that there is no difference at model prediction stage. That is to say, the theoretical optimal solution of MLP and Scale-MLP are the same. However, deep models don't always converge to the same optimal solution in practice, therefore, the effectiveness of actual models obtained from Scale-MLP and MLP are often different on different datasets.

To be detailed, the essential difference lies in the way backpropagation update the network parameter during model training stage. Compared to a standard MLP, Equation A.6 reflects that the network need feedforward an intermediate result $w_j x_j$ after the scale layer added. When MLP updates the parameter matrix A during backpropagation , the partial derivative regarding a_{ij} is x_j; for Scale-MLP, the partial derivative regarding a_{ij} is $w_j x_j$ while regarding w_j is x_j. In another word, the value of feature x_j in MLP can directly affect the parameters $a_{ij}, i = 1...k$; for Scale-MLP, feature x_j can only update w_j.

Assuming that the influence of different features on the model is quite different, the fluctuation of feature values will make training process difficult to converge. Suppose that the feature x_j has little effect for the target, when the values of x_j drifts, it will cause training difficulty unless the absolute value of parameters $a_{ij}, i = 1...k$ are all small; on the other side, as long as the absolute value of the only affected parameter w_j in Scale-MLP model is small, the influence of the feature on the target can be made smaller. To conclude, adding the scale layer and updating the parameters of the scale layer during backpropagation can directly change the final influence of each feature on the model.

Generally saying, for MLP model, matrix A captures the first-order combinatoric features. In order to learn high-order features, the model need to fit the data by adjusting both the parameters of matrix A and the hidden layers of MLP. Due to the sparsity of feature space and importance of different features varies, the parameters of matrix A cannot be very effectively trained. Under such circumstances, the MLP model is easier to overfit. On the contrast, the Scale-MLP model only needs to train the parameters of the scale layer properly for the same purpose. Therefore, Scale-MLP model is much simpler to train in our setting.

Another angel to look at the functionality of the scale layer is that it adds randomness to the original user feature vector. In another word, if a user is not impressed by some ad, it doesn't mean that he/she is totally not interested in that ad. Therefore, the scale layer will help to learn a model which has better generalization capability.

A.4.4 Model Training

The Impact of Sampling Ratio

We evaluate the impact of sampling ratio based on different number of positive and unlabeled samples, seeing unlabeled as negative label. The standard classification algorithm we choose is Logistic Regression. The key metrics need to be taken care are *test recall* and *threshold*, meaning positive sample recall on testing data set and the corresponding probability boundary. The number of positive and negative samples in testing data set are 34657 and 72464. The evaluation result in Table A.3 shows when ratio of positive and unlabeled reaches 1:2 (the number of positive and negative samples are 69331 and 134584 respectively), the threshold doesn't change significantly when more unlabeled samples are added. Considering both training efficiency and effectiveness, it is practical to set the sampling ratio of positive:negative as 1:2.

Table A.3: The Impact of Sampling Ratio

positive	unlabeled	train loss	test accuracy	test auc	test recall	threshold
69331	69331	0.4693	0.764	0.835	0.740	0.493
97064	97064	0.4692	0.767	0.840	0.740	0.498
138663	138663	0.4700	0.770	0.843	0.740	0.493
69331	95743	0.4650	0.769	0.837	0.740	0.518
69331	**134584**	**0.4298**	**0.774**	**0.839**	**0.740**	**0.668**
69331	197328	0.3874	0.776	0.839	0.740	0.678
69331	245811	0.3576	0.776	0.839	0.740	0.682

Sampling Techniques

For general classification problem, to determine where the class boundary is, at least some of the negative samples to be close to the positive ones are chosen. Take "active learning" [26] as an example, algorithms will select out those samples that are most indistinguishable from the model for human expert to label. However, lookalike models deal with data without labelled negative samples, hence the goal of sampling is to pick out a reliable set of negative users.

Besides randomly selecting negative samples and directly apply standard classifier to the PU learning problem, we compare the effectiveness of three other sampling techniques: spy, pre-train and bootstrap sampling. The "Spy" [14] [11] and "Pre-Train" sampling strategies are so-called "two-step" approach [27] where the general idea is described as follows: the first step is to identify a subset of unlabeled samples that can be reliably labelled as negative, then positive and negative samples are used to train a standard classifier that will be applied to the remaining unlabeled samples. Usually the classifier is learned iteratively till it converges or some stopping criterion is met. Correspondingly, the "Spy" and "Pre-Train" sampling strategies are illustrated in Algorithm 1 and 2.

A more sophisticated approach [28] is a variant of bagging: first of all, a subset of unlabeled samples are bootstrapped from the unlabeled sample set U. The algorithm details are depicted in Algorithm 3. Here we set the number of iterations T and for each iteration, a standard classifier responsible for predicting U is trained on bootstrapped sample set U' and positive sample set P. The final predicted probability equals to the average score of T iterations.

Table A.4 shows the experimental result of different sampling approaches. The *sampling parameter* represents the percentage of unlabeled samples picked out as negative and *threshold* indicates the corresponding probability boundary. From the result table it can be seen that when spy and bootstrap approaches sample half size of the unlabeled data, it still guarantees almost the same level of recall on testing data while regarding pre-train sampling approach, the recall on test data is much lower. On the sampling efficiency, spy

Algorithm 1: Spy Sampling

```
   Input: Positive Sample Set $P$, Unlabeled Sample Set $U$
   Output: Negative Sample Set $N$ with size $k$
1  Randomly select a subset from $P$ as the spy set $P'$;
2  Train a classifier $M$ based on $P - P'$ and $U + P'$;
3  Select a subset $N$ of $k$ samples from $U$ with least prediction scores;
4  Return $N$;
```

Algorithm 2: Pre-Train Sampling

```
   Input: Positive Samples Set $P$, Unlabeled Sample Set $U$, Validation
          Set $V$
   Output: Negative Sample Set $N$ with size $k$
1  Randomly select a subset $N$ with size $k$ from $U$;
2  while true do
3      Randomly select a subset $N'$ from $N$;
4      Train a classifier $M$ based on $P$ and $N'$, and evaluate the model on
         $V$;
5      if the accuracy of $M$ doesn't improve on $V$ then
6          Return $N$;
7          break;
8      Predict $U$ using classifier $M$;
9      Select a subset $N$ of $k$ samples with least prediction scores;
```

Algorithm 3: Bootstrap Sampling

```
   Input: Positive Sample Set $P$, Unlabeled Sample Set $U$
   Output: Negative Sample Set $N$ with size $k$
1  for $t \leq T$ do
2      Bootstrap a subset $U'$ from $U$;
3      Train a classifier $M$ on $P$ and $U'$;
4      Predict $U - U'$ using classifier $M$;
5      Record the classifying scores;
6  Average the classifying scores of all iterations;
7  Select a subset $N$ of $k$ samples with least average scores;
8  Return $N$;
```

approach can only run one iteration compared to the other two which need converge after several rounds. Therefore, it is both efficient and effective to utilize spy sampling approach in our setting.

Table A.4: The Impact of Sampling Approach

approach	sampling parameter	train loss	test accuracy	test auc	test recall	threshold
Random	0.9	0.4250	0.775	0.847	0.766	0.633
Random	0.5	0.4150	0.753	0.843	0.676	0.612
Spy	0.95	0.4138	0.775	0.847	0.768	0.640
Spy	0.9	0.4141	0.776	0.847	0.763	0.633
Spy	**0.5**	**0.3660**	**0.775**	**0.845**	**0.768**	**0.607**
Pre-Train	0.95	0.3677	0.775	0.845	0.771	0.624
Pre-Train	0.9	0.3829	0.776	0.846	0.771	0.632
Pre-Train	0.5	0.4382	0.702	0.839	0.632	0.628
Boostrap	0.95	0.4127	0.775	0.847	0.768	0.638
Boostrap	0.9	0.4153	0.776	0.847	0.763	0.633
Boostrap	**0.5**	**0.3976**	**0.775**	**0.845**	**0.766**	**0.640**

A.5 Experiments

A.5.1 Setup

Regarding the model implementation, we use MXNet[1] on a stand-alone 1080TI GPU to compare different model effects and figure out model parameters. When predicting the universal user pool consisting of nearly 2.5 billion users, we used distributed MXNet on a 80-cores hadoop cluster to re-train the model and it took nearly 4 hours to finish the prediction of all users.

A.5.2 The Impact of Feature Engineering

Table A.5 shows the impact of different feature engineering approaches. In this table, *Time Slice* indicates the strategy of calculating the user behaviour by time slice (None: no time slice; day: slice by day; holiday: slice by holiday and weekday; month: slice by month). For example, if we extract features of user activities by month, one typical feature could be that one specific user is impressed by an ad of "Maybelline" 5 times in July. In general, only activities happening in last three months are to be extracted. *Click* means whether we distinguish between click action from impression. The experimental results based on *LR* model (training data volume: 428484; testing data volume: 107121; positive and negative ratio is 1:2) show that if the features are calculated by month and click action is separated from impression, the *AUC* value will reach 0.8465 in testing phrase which is the best among all settings. Therefore, this feature engineering strategy will be applied in various model methodologies afterwards.

[1]https://mxnet.apache.org/

Table A.5: The Impact of Feature Engineering

Feature Size	Time Slice	Click	Train AUC	Test AUC
144009	None	True	0.8721	0.8447
94932	None	False	0.8689	0.8443
249406	by holiday	True	0.8808	0.8445
196184	by month	True	0.8780	**0.8465**
133605	by month	False	0.8761	0.8461

A.5.3 Model Performance

In this section, the performance comparison of various models is introduced. The hyper-parameters configured in different models are listed at Table A.6. In this table, BN-MLP is a multi-layer perceptron with a batch normalization layer after each hidden layer; Scale-BN-MLP adds a scale layer before BN-MLP; *lr* and *wd* represent learning rate and L2 regularization parameter respectively.

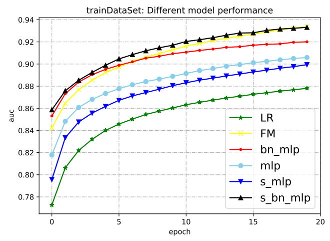
Table A.6: Hyper-parameter Setting

Model	Parameters
LR	lr=1e-4, wd=1e-6
FM	lr=1e-4, wd=3e-5, k=6
MLP	lr=1e-4, wd=3e-5
BN-MLP	lr=1e-4, wd=3e-5
Scale-MLP	lr=1e-4,wd=3e-5
Scale-BN-MLP	lr=1e-4,wd=3e-5

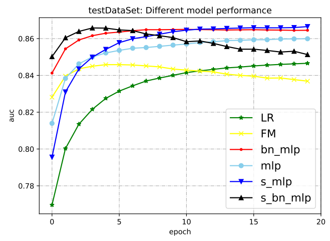
From the experiment results in Figure A.3, we can see that the effect of the multi-layer perceptron is better than that of LR and FM, and adding the batch normalization layer and the scale layer can both improve the generalization ability and convergence speed of the model. Therefore, Scale-BN-MLP outperforms other models regarding *AUC* value during training phrase. In addition, the convergence speed of Scale-BN-MLP (4 epochs) is the fastest one among all models, **requiring early stopping to yield the optimal model in practice**. The result confirms the derivation in section A.4.3. Figure A.4 shows different learning rates for Scale-BN-MLP model in training and testing data set, the convergence speed performs well when learning rate equals to 0.0001(1e-4).

A.5.4 Online Effectiveness Evaluation

Regarding effectiveness evaluation in a real closed-loop business setting, we corporate with a brand advertiser and a third-party advertising monitor sup-

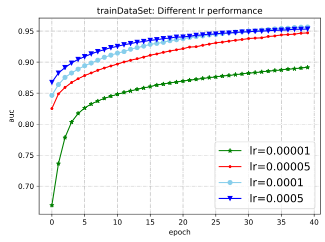


(a) Train Dataset

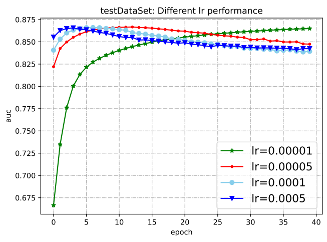


(b) Test Dataset

Fig. A.3: Model Performance



(a) Train Dataset



(b) Test Dataset

Fig. A.4: Comparison of Different Learning Rate

plier in order to conduct the online experiments. The final experiment results are shown at Table A.7. There are several important business metrics like Impression UV, Purchaser Rate, ATV (Average Transaction Value), CPO (Cost Per Order), CPA (Cost Per Action) and Incremental ROI listed in this table. All indicators of our model perform far better than traditional demographic-based approaches.

Table A.7: Online A/B Testing Results

Metric	Random	F20-34	FEMALE	MALE	MODEL
Impression	18,367,151	6,493,314	3,910,355	1,454,655	1,221,095
Impression UV	8,578,859	3,152,614	2,052,897	912,468	594,456
Purchaser	597	123	117	29	217
Purchaser Rate	0.01%	0.00%	0.01%	0.00%	**0.04%**
Transaction	731	155	134	32	275
Sales	69,974	14,976	14,727	3,739	23,413
ATV	96	97	110	117	85
Media Cost	295,575	106,982	61,972	24,429	19,100
CPO	404.3	690.2	462.5	763.4	**69.5**
CPA	495.1	869.8	529.7	842.4	**88.0**
Incremental ROI	0.2	0.1	0.2	0.2	**1.2**

A.6 Conclusions and Future Work

In this paper, we showed an data application architect to utilize advertisement monitor data in audience expansion system for brand advertisers, compared to traditional ad serving based on demographics, the lookalike model in our application focuses on analysing user behaviour. Regarding the way of picking up the negative samples from unlabeled data, we compared four sampling techniques and the impact of different sampling ratios in order to figure out the best setting. Meanwhile, to overcome the sparsity and high dimension of feature space, we proposed Scale-MLP, a modified MLP by adding a scale layer, although the training AUC is lower than other traditional learning strategies, however, it gains performance improvement when generalizing the model to testing data while the efficiency of Scale-MLP is comparable to other approaches. Lastly we prove that the lookalike model outperforms traditional ad serving mechanisms in real business environment.

Several directions exist for future research. The rich information contained in the advertisement could be harnessed to investigate more sophisticated lookalike models. For example, we could incorporate advertising information including advertiser, brand and product in order to explore more detailed feature interactions. There is also a possibility of probing user historical behaviour as a time series data to extract short-term or long-term sequence features. For different advertisers' campaign, adaptive user feature representation also need to be taken into consideration.

At the same time, CTR prediction task will be a challenging and interesting problem under the setting of growing diversity in targeting users and cross-media advertising platforms. CTR prediction results could be utilized for the purpose of omni-channel uniform budget allocation to effectively enhance ROI by matching brands/products with different media platforms.

References

[1] S. Pandey, M. Aly, A. Bagherjeiran, A. Hatch, P. Ciccolo, A. Ratnaparkhi, and M. Zinkevich, "Learning to target: What works for behavioral targeting," in *Proc. of CIKM*, 2011, pp. 1805–1814.

[2] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen, "How much can behavioral targeting help online advertising?" in *Proc. of WWW*, 2009, pp. 261–270.

[3] A. Mangalampalli, A. Ratnaparkhi, A. O. Hatch, A. Bagherjeiran, R. Parekh, and V. Pudi, "A feature-pair-based associative classification approach to look-alike modeling for conversion-oriented user-targeting in tail campaigns," in *Proc. of WWW*, 2011, pp. 85–86.

[4] J. Shen, S. C. Geyik, and A. Dasdan, "Effective audience extension in online advertising," in *Proc. of KDD*, 2015, pp. 2099–2108.

[5] H. Liu, D. Pardoe, K. Liu, M. Thakur, F. Cao, and C. Li, "Audience expansion for online social network advertising," in *Proc. of KDD*, 2016, pp. 165–174.

[6] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 128–131, March 2008.

[7] Q. Ma, E. Wagh, J. Wen, Z. Xia, R. Ormandi, and D. Chen, "Score look-alike audiences," in *Proc.of workshops on ICDM*, 2016, pp. 647–654.

[8] Q. Ma, M. Wen, Z. Xia, and D. Chen, "A sub-linear, massive-scale look-alike audience extension system a massive-scale look-alike audience extension," in *Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, 2016.

[9] A. Ramesh, A. Teredesai, A. Bindra, S. Pokuri, and K. Uppala, “Audience segment expansion using distributed in-database k-means clustering,” in *Proc. of ADKDD*, 2013, pp. 5:1–5:9.

[10] A. Bindra, S. Pokuri, K. Uppala, and A. Teredesai, “Distributed big advertiser data mining,” in *Proc. of Workshops on ICDM*, 2012, pp. 914–914.

[11] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, “Building text classifiers using positive and unlabeled examples,” in *Proc. of ICDM*, 2003, pp. 179–186.

[12] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, “Positive-unlabeled learning with non-negative risk estimator,” in *Proc. of NIPS*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1675–1685.

[13] M. N. Nguyen, X.-L. Li, and S.-K. Ng, “Positive unlabeled learning for time series classification,” in *Proc. of IJCAI*, 2011, pp. 1421–1426.

[14] B. Liu, W. S. Lee, P. S. Yu, and X. Li, “Partially supervised classification of text documents,” in *Proc. of ICML*, 2002, pp. 387–394.

[15] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, L. Garcia-Pueyo, and J. Yuan, “Focused matrix factorization for audience selection in display advertising,” in *Proc. of ICDE*, 2013, pp. 386–397.

[16] O. Chapelle, E. Manavoglu, and R. Rosales, “Simple and scalable response prediction for display advertising,” *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 4, pp. 61:1–61:34, Dec. 2014.

[17] R. Kumar, S. M. Naik, V. D. Naik, S. Shiralli, S. V.G, and M. Husain, “Predicting clicks: Ctr estimation of advertisements using logistic regression classifier,” in *2015 IEEE International Advance Computing Conference (IACC)*, 2015, pp. 1134–1138.

[18] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, “Training and testing low-degree polynomial data mappings via linear svm,” *JMLR*, vol. 11, pp. 1471–1490, Aug. 2010.

[19] S. Rendle, “Factorization machines,” in *Proc. of ICDM*, 2010, pp. 995–1000.

[20] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, “Field-aware factorization machines for ctr prediction,” in *Proc. of RecSys*, 2016, pp. 43–50.

[21] Y. Juan, D. Lefortier, and O. Chapelle, “Field-aware factorization machines in a real-world online advertising system,” in *Proc. of WWW Companion*, 2017, pp. 680–688.

[22] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. of the 1st Workshop on Deep Learning for Recommender Systems*, 2016, pp. 7–10.

[23] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," in *Proc. of IJCAI*, 2017, pp. 1725–1731.

[24] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proc. of ADKDD*, 2017, pp. 12:1–12:7.

[25] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data - - A case study on user response prediction," in *Proc. of ECIR*, ser. Lecture Notes in Computer Science, N. Ferro, F. Crestani, M. Moens, J. Mothe, F. Silvestri, G. M. D. Nunzio, C. Hauff, and G. Silvello, Eds., vol. 9626, 2016, pp. 45–57.

[26] B. Settles, "Active learning literature survey," Tech. Rep., 2010.

[27] A. Kaboutari, S. Branch, J. Bagherzadeh, I. Urmia, and F. Kheradmand, "An evaluation of two-step techniques for positive-unlabeled learning in text classification," vol. 3, 2014, pp. 592–594.

[28] F. Mordelet and J. P. Vert, "A bagging svm to learn from positive and unlabeled examples," *Pattern Recogn. Lett.*, vol. 37, pp. 201–209, Feb. 2014.

Paper B

Deep Monitor Network for Privacy-Preserving Click-Through Rate Prediction

Jinling Jiang, Xiaoming Lin, Junjie Yao, Ruogu Ding, Hua Lu and Xindong Wu

The paper has been submitted for publication.

Abstract

Compared with traditional advertising, online advertising shows great advantages mainly due to its adoption of large scale data analysis technologies, e.g., click-through-rate (CTR) prediction. In CTR prediction, to make online advertising effective, tens or hundreds of millions of user preference and records are analysed and matched with high probable advertisements. Recent CTR prediction methods are usually based on factorization machines or various sorts of deep learning models. Though such existing methods tackle the data sparsity challenge in CTR scenarios, they fail to provide reliable and accurate prediction results with the growing diversity in target users, advertising devices and media platforms. On the other hand, user profile information is increasingly sensitive to acquire, which demands CTR models with good performance and user privacy awareness.

In this paper, we introduce a novel Deep Monitor Network (DMN) model for privacy-preserving CTR task. DMN is able to model user behaviour across various advertising devices and media platforms, thereby incorporating rich user features without accessing user private information. We equip the new DMN model with creatively designed attention based feature matching layers and the later compact training layers structure. The unique model framework significantly reduce the model parameters by historical profile compression with privacy protection and accurate attention mechanism. Experiments on a public dataset and our own large CTR datasets demonstrate its advantages over several state-of-the-art approaches under privacy preservation scenario, not only in the prediction accuracy, but also the model stability under invalid traffic attack.
The layout has been revised.

B.1 Introduction

In the past few years, the market size of digital advertising has grown dramatically and it is expected that digital ad spending will exceed 200 billion U.S. dollars by 2021.[1] The problem of click-through rate prediction (CTR) can be formalized as the estimation of the percentage of clicks on the total number of impressions served for a given page and a specific user. As a challenging task, the bottom line of successful advertisement targeting is usually 0.2%.[2] CTR prediction continuously plays a fundamental role in the advertising process as given a budget and related information of an campaign, the campaign advertiser need decide how much they would like to bid for their advertisements in various media platforms in order to achieve the maximum clicks.

Therefore, accurately predicting the click-through rate of advertisements will surely improve the efficiency of the market, leading to a "three-wins" situation for advertiser, media platform and online user: advertisers want more people to click on their own ads and learn about their products; media platforms target to make more profit; users would like more suitable ads to be pushed so as to improve their own user experience. Either over-estimating or under-estimating CTR will lead to insufficient usage of budget, thereby achieving lower ROI in the end.

As a very important and difficult problem with many sparse features and tons of records, the recent solutions for CTR prediction can be categorized as **factorization machine** based [1] and **deep learning** based algorithms [2–5]. FM-based models can be seen as a family of shallow neural networks with one embedding layer imposed on sparse inputs, followed by specially designed neuron connections fitting the target. On the other hand, deep learning models always follow embedding and multilayer perceptron (MLP) paradigm. However, many challenges still exist in applying recent CTR prediction methods to large and complex scenarios. For example, the prediction model need capture different user action patterns across various advertising platforms. More importantly, under new data protection regulation like GDPR[3], user profile information is sensitive to access. Therefore, there is an urgent need of developing a CTR model with high accuracy under such circumstance.

The challenges of developing a robust privacy-preserving CTR model lies at following aspects:

- The model need capture different user action patterns across different advertising media.

[1]https://www.statista.com/topics/979/advertising-in-the-us/
[2]https://www.searchenginepeople.com/blog/925-adwords-ctr.html
[3]https://eugdpr.org/

- The training sample is quite imbalanced (the number of clicked sample is much less than unclicked sample).
- There is no user profile information provided except a virtual user identifier, therefore, feature extraction of user preference is difficult.

In this paper, we design a novel Deep Monitor Network (DMN) to tackle the challenges by incorporating creative feature extraction modules and compress the training stages within the neural networks. Instead of utilizing user profile embedding to extract user representation, we extend the recent deep network structures used in CTR prediction by new user feature extraction and historical training data compression. Figure B.1 generally depicts the model architecture with comparison of DeepFM [5]. The essential difference between the two models is the way of extracting user features. DeepFM utilizes the embedding of user identification while DMN relies on a dedicated network structure in order to extract features of history events belonging to the same user. The most recent and related work Deep Spatio-Temporal Neural Network (DSTN) [6] which learns the representation of contextual ads and historical displayed ads belonging to one user. Compared to DSTN, the general idea of DMN model design is to separately leverage the clicked/positive and unclicked/negative samples and execute feature extraction in a creative way in order to learn the representation of user historical actions.

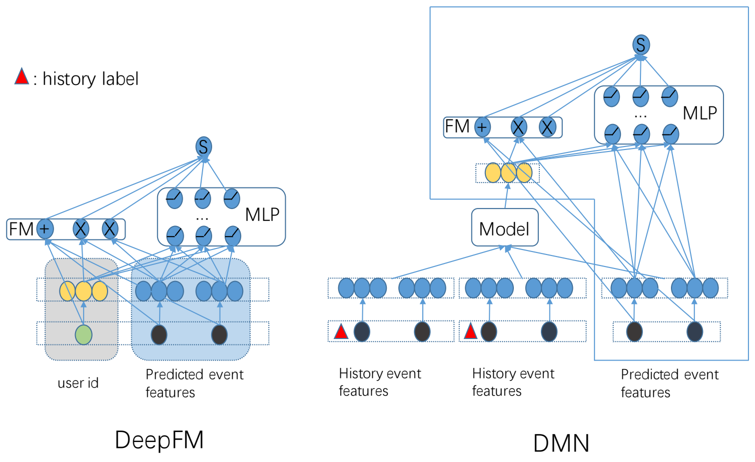


Fig. B.1: Comparison of DeepFM and DMN Architecture

Deep Monitor Network mainly consists of three blocks: feature extraction of user historical cross-media behaviour, FM layer and MLP. And there are two input pieces: history events belonging to one user, and the predicted event. In fact, the essential part of DMN model is a feature extraction network learning user interest representation with respect to an advertisement to be predicted. FM layer and MLP are utilized for extracting both low-order and high-order feature interactions based on the extracted user interest representation. We can replace the feature extraction network of DMN with LR or other machine learning algorithms for CTR task. Section B.5.3 will discuss the impact of connecting different learning modules. Regarding FM and MLP blocks, they are essentially constructed in the same way as DeepFM model did.

The contribution of this work can be summarized as follows.

- We design a new Deep Monitor Network to capture the sophisticated interactions of user cross-platform behaviour.
- Instead of user profile embedding, DMN model preserves the user privacy and learn the representation of user actions by feature extraction of user historical behaviour.
- We conduct extensive experiments on our own unique dataset with a public one to demonstrate the advantage of our proposed model.
- We conduct experiments to prove DMN is the most stable CTR model under invalid traffic attack.

The rest of the paper is organized as follows. Section B.2 formulates the problem definition and introduce the unique setting with data description in our environment. Section B.3 reviews the related work on various kinds of learning models for CTR prediction and illustrate different design principles. Section B.4 elaborates the model architecture and details. The evaluation of model efficiency and effectiveness is presented in Section B.5. Finally the conclusion are presented in Section B.6.

B.2 Problem Formulation and Data Description

In a typical CTR task, we are given a training dataset of n instances where each sample is (X_i, y_i). X_i is a vector of categorical or continuous features representing a pair of user and item (e.g., product, advertisement). Meanwhile, $y_i \in \{0, 1\}$ is the associated label indicating whether the user clicked on the item ($y_i = 1$ means the user clicked the item while $y_i = 0$ means the opposite). Usually X_i includes the features of user profile information (e.g., gender, location, age) and item attributes (advertiser, brand, product, etc.). Categorical features are encoded by one-hot vectors, and continuous features can also be represented by one-hot vectors imposed on discretization or remained as its original value. As each feature belongs to one and only one field, after one-hot encoding, each sample is converted into (x, y) where $x = [x_{field1}, x_{field2}, ..., x_{fieldj}, ..., x_{fieldn}]$. x is a extremely sparse d-dimensional vector with d usually being a very large number (in the order of hundreds of millions) and x_{fieldj} is the representation of the j-th field of X_i.

The essential difference in our environmental setting compared to traditional CTR task is that in order to preserve user privacy and make our application more general (some application don't have access to user demographics), we don't take into account user profile information and the only user-related information is virtual identification. That is to say, the related field information of user profile (e.g., gender, location, age) is omitted in our dataset. In common context, the CTR prediction task is to build a machine learning model $\hat{y} = CTR\_Model(X_i)$ to estimate the probability of a user clicking a specific advertisement.

Take a check of our recent CTR dataset. Table B.1 is an example of a real-world advertising data set for some big brand campaigns (L'Oreal, OPPO, P&G, etc) across different media platforms (Youku, IQIYI, Tencent, etc). It shows a typical multi-field categorical data for CTR prediction. Each row is an ad impression with a CLICK label indicating whether there is a click associated with the corresponding impression. Except CLICK Column, each of the rest column (Advertiser, Brand, Product, Industry, Platform and Device) is a field. Features are all categorical, e.g., OPPO, Maybelline, Tencent, and each of them belongs to one and only one field. For example, OPPO belongs to field Advertiser and Tencent belongs to field Platform. In this dataset, the number of user reaches the order of hundred of millions while the number of other fields (advertiser, brand, product, etc) is in the order of hundreds to thousands. Besides, users have different preferences on the targeting media platforms and unbalanced preference distributions. Finally, what needs to be emphasized is that user profile information (e.g., gender, location, age) is omitted due to user privacy preservation.

Table B.1: Multi-Field Records of Advertising Campaigns, used in CTR Prediction

CLICK	USER_ID	Advertiser	Brand	Product	Industry	Platform	Device
1	66a7988f	L'Oreal	Maybelline	Maybelline	Beauty Care	Youku	MI MAX 2
1	9e664577	OPPO	OPPO	OPPO R7 PLUS	IT Product	Tencent	iPhone 6
0	9b3fcc94	Priceline	Agoda	Agoda	Tourist Accommodation	IQIYI	iPhone X
1	0043fbf4	P&G	SK-II	SK-II	Beauty Care	Tencent	HW 2A
1	1df73293	Nestle	Nan	Nestle Nan	Baby Products	Tudou	OPPO V6
0	23ff3494	OPPO	OPPO	OPPO V6	IT Product	SINA	iPhone X

Table B.2 shows the basic statistics of feature sets. The dimensionality of user identity is in order of several billions while other feature fields is tens to thousands. Therefore, it is very inefficient of encoding user identifier directly as the model parameter space will be huge.

Table B.2: Statistics of feature sets collected in advertising monitor

Feature Field	Dimensionality	Type
USER_ID	~10^{10}	one-hot
Advertiser	~10^{2}	one-hot
Brand	~10^{2}	multi-hot
Product	~10^{3}	multi-hot
Industry	~10	one-hot
Platform	~10^{2}	one-hot
Device	~10^{3}	one-hot

B.3 Related Works

We briefly review recent work in related area.

Factorization Machine based: To capture the interaction between features in an efficient way, several Factorization Machine (FM)-based models are developed. Compared to FM [1], FFM model [7] [8] firstly raises the "field" concept, thereby decoupling the feature weight update during model training. To overcome the overfitting problem caused by huge number of training parameters of FFM, Field-aware Factorization Machine [9] alleviates the problem by considering different interaction strength between various fields.

Deep Learning based: Inspired by the technical development from deep learning in computer vision and natural language processing, state-of-art CTR prediction models evolve into various kinds of deep neural network which have powerful model expressibility. FNN [2] starts to integrate the feed forward neural network with a pre-trained FM-initialization layer. In addition, PNN [3] imposes different types of product layer (inner product, outer product and their different combinations) between the embedding layer and the first hidden layer. To catch both low-order and high-order feature interactions and overcome pre-training constraint, Wide & Deep [4] is proposed to combine expertise feature engineering and the learning ability of deep neural networks. In the same while, Deep & Cross [10] model pays attention to automating cross feature engineering and introduces a novel cross network that is more efficient in learning certain bounded-degree feature interactions. Compared to Wide & Deep and Deep & Cross model, DeepFM [5] substitutes the low-order feature interaction with an implementation of factorization machine integrated in network structure, thereby achieving a better accuracy.

Recently, [11] leverages the strength of CNN to generate local patterns and recombine them to generate new features. The Multi-Sequence Model

(MSM) [12] is based on the study [13] applying convolution network in CTR task and firstly investigates whether and how the feature sequence affects the performance of the CNN-based CTR prediction method. Work [14] proposes a Field-Aware Probabilistic Embedding Neural Network (FPENN) model combining ideas of Field-aware Factorization [7] and DeepFM [5] to achieve a good generalization ability. The state-of-art neural model in CTR prediction area is DSTN [6] that considers both contextual ads and historical behaviour so as to extract richer information on user interests.

Regarding CTR prediction task applied in Recommendation System (Alibaba[4] , Amazon[5] and other e-commerce platforms), instead of display advertising, Deep Interest Network (DIN) [15] and Deep Interest Evolution Network (DIEN) [16] propose models to effectively capture user's diverse interests from rich historical behaviour by designing a mechanism to adaptively learn the representation of user interests with respect to a certain ad. [17] puts emphasis on handling long sequential user behavior data with length scaling up to thousands for Alibaba recommendation advertising system.

Other Related Work: Besides the above-mentioned models, several CTR related problems are also explored by deep learning: Disguise Adversarial Network [18] firstly employs an adversarial learning framework in CTR prediction, however, it doesn't compare with other state-of-art algorithms regarding the model performance; CACNN [19] employs CNN to perform CTR forecasting at advertiser level; Deep Intent Network [20] investigates the use of recurrent neural networks (RNNs) in search-based online advertising; JUMP [21] adopts a novel three-layered RNN structure to jointly predict for user click and dwell time.

Another related research area is the data mining on user privacy. Privacy-Preserving Data Mining (PPDM) methodologies are designed to extract relevant knowledge from large amounts of data while guaranteeing a certain level of user privacy, such that data mining can still be performed on the transformed data efficiently [22] [23] [24].

B.4 Our Approach

B.4.1 Model Framework

Suppose a user with no profile information, has n event records and each record contains m feature fields with an associated label y ($y \in \{0, 1\}$). The traditional CTR model treats each record as an independent sample, and then explore interactions of the same user's records by embedding user identification at training phase. On the contrary, in our model, we sort the n event

[4]https://www.alibaba.com/

[5]https://www.amazon.com/

records in chronological order and treat them together as a sample. There are two benefits of this mechanism in general: it avoids the training challenge caused by the considerable training parameters of directly embedding user identification; it allows incremental model updating when new users emerge in the system, accordingly, there is no need for frequent model updates.

Different from DeepFM and other classic CTR models directly embedding user identification, in order to characterize sophisticated interests of a specific user across various media platforms, we model the relationship between the features of predicted event (e.g., campaign information) and cross-media behaviors belonging to the same user. A hyper-parameter *max_len* is set to limit the input length of history event sequence. For users who have less than *max_len* advertising events monitored, we apply zero padding to the input.

Figure B.2 shows a comprehensive version of model details. The main part of DMN is a delicate feature extraction module for learning representation of user interests from historical events (including clicked and unclicked records) while a combination of FM and MLP network is for the purpose of learning higher order interactions. The feature extraction module mainly consists of the following four parts: feature embedding, feature selection, model connection and feature transformation.

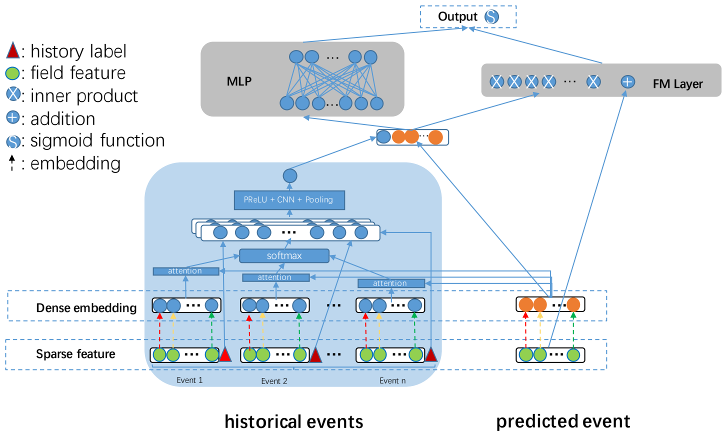


Fig. B.2: Framework of Deep Monitor Network

B.4.2 Feature Embedding

As the data shown in Table B.1 which is collected for the use of CTR prediction, coming into a multi-field categorical form, so that each instance is normally transformed into a high-dimensional binary vector (extremely sparse) via one-hot encoding. For example, (Advertiser=OPPO, Product=OPPO R7, Industry=IT Product , Platform=Tencent) can be represented as:

$$\underbrace{[0,...,1,...,0]}_{\text{Advertiser=OPPO}}\ \underbrace{[0,..,0,...,1]}_{\text{Product=OPPO R7}}\ \underbrace{[0,...,1,...,0]}_{\text{Industry=IT Product}}\ \underbrace{[1,...,0,...,0]}_{\text{Platform=Tencent}} \tag{B.1}$$

In order to transform high-dimensional sparse feature vectors into low-dimensional dense ones, an embedding layer is applied after one-hot encoding. Expressing the Embedding layer in the form of a matrix is essentially the process of solving a weight matrix of m (the dimension of the input high-dimensional sparse vector) x n (the dimension of the output dense vector). If the input vector is a one-hot eigenvector, the column vector in the weight matrix is the embedding vector of the one-hot feature of the corresponding dimension. Note that a feature field could be multivalent as mentioned in Table B.2, for example, there could be two brands or products in one advertisement (e.g., Product=OPPO R7, OPPO R7 PLUS), then the embedding value of corresponding field will be the sum of features' embeddings.

B.4.3 Feature Selection

The feature selection piece aims to adaptively learns the user feature representation with respect to a certain ad, we adopt attention mechanism in DMN model as well to calculate the correlation between different history events and current event. Users behave differently at different media platforms, for a specific predicted event, the model should particularly look at the related historical events (say events happening at the same media platform or containing the same brand/product). That is to say, the stronger correlation with the predicted event, the larger value of corresponding attention weight should be, and vice versa. The correlation between the feature vector e of predicted event and the feature vector h of a certain history event is calculated the same way as in [25]:

$$\begin{gathered} a(e_i, h_j) = v^T tanh(W_e e_i + W_h h_j) \\ i \in [1, 2...k^2], j \in [1, 2...max\_len] \end{gathered} \tag{B.2}$$

In Equation B.2, v, W_e and W_h are the parameters to be trained. In order to improve the model fitting ability, for feature vector e that has embedding size of k, there are k^2 attention weights to be calculated. The calculation procedure is shown in Figure B.3.

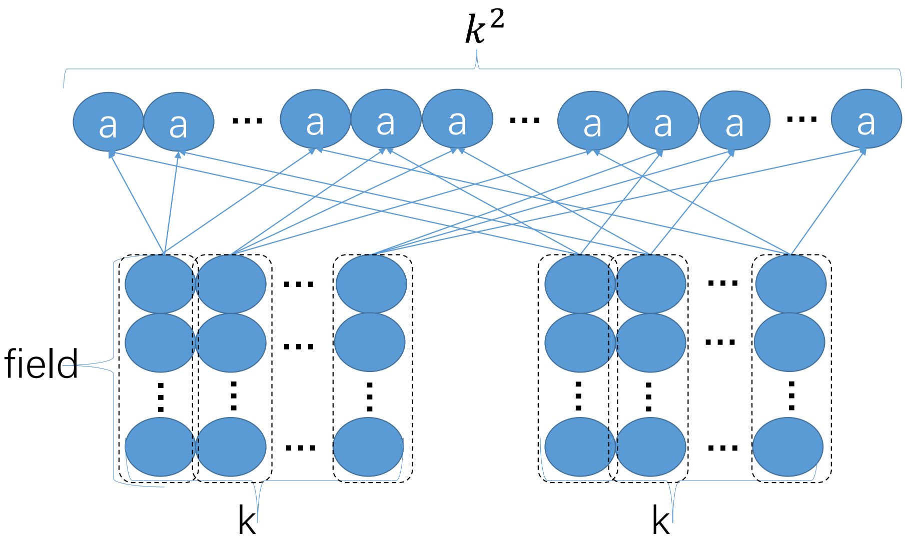


Fig. B.3: Attention Weights Calculation

In order to achieve the attention capability associated with the event label, We use the function in Equation B.3 to apply the attention result to the label. If we set the label of unclicked sample as 0, $f(label, a)$ will always be 0 regardless of the value of a. Then the BP algorithm cannot be effectively used to update the parameters. In order to solve this problem, we set the label of click events to 1 and the unclick ones to -1. To unify the sequence length, the label is filled with zero for the padding event. After this phase, the output is a matrix of $max\_len * k^2$ dimensions by multiplying the label and attention result.

$$\begin{gathered} f(label_j, a_{ij}) = label_j * a_i \\ i \in [1, 2...k^2], j \in [1, 2...max\_len] \end{gathered} \tag{B.3}$$

B.4.4 Model Connection

Since the samples in CTR task is quite imbalanced (the number of clicked samples is very small compared to the unclick ones; usually the ratio of clicked samples is 2%-4%), a robust learning model should activate more attention on the impact of clicked samples while lowering the impact of unclicked samples. That is to say, for the results obtained from Section B.4.3 , we need an activation function to connect the model into the next stage. The activation function need meet the following four conditions:

$$\begin{aligned} g(x) &\leq 0 \ (x \leq 0) \\ g(x) &> 0 \ (x > 0) \\ |g(-x)| &< |g(x)| \ (x > 0) \\ g(x_2) &> g(x_1) \ (-1 \leq x_1 < x_2 \leq 1) \end{aligned} \tag{B.4}$$

Hence, we adopt the PReLU function for the purpose of controlling the weights in our model. As in Equation B.5, the model connection part makes the coefficient of leakage into a parameter that is learned according to the event label.

$$PReLU(x_i) = \begin{cases} x_i & x_i > 0 \\ a_i x_i & x_i \leq 0 \end{cases} \tag{B.5}$$

B.4.5 Feature Transformation

After PReLU activation, the $max\_len * k^2$ dimensional matrix represents the user historical behavior features. To this end, the calculated feature vector is based on personal historical behavior/events. In order to extract composite and high-order feature interactions between features of historical events and predicted event, we need condense $max\_len * k^2$ dimensional matrix into the same feature dimension of the predicted event (i.e., k-dimension). The max_len historical events are ordered in timestamps, therefore, sequence feature may be extracted from max_len dimensions. In this way, the $max\_len * k^2$ dimensional matrix can be regarded as a three-dimensional tensor with a channel number of k^2, a length of max_len, and a width of 1.

In display advertising area, the sequence feature is not explicit in nature and random noise of the sequence is high, making it meaningless to employ recurrent network or convolution network structure with large kernel size to do the feature learning. As the larger convolution kernel size is, the more sequence feature it involves. For example, if we don't consider the sequence feature over the historical events at all, a $1 * 1$ convolution kernel should be applied. In our setting, we exactly employ a $1 * 1$ convolutional kernel in CNN layer and global mean pooling layer to process feature extraction on the $max\_len * k^2$ dimensional matrix. Finally a k-dimensional user features u_i is obtained. Generally u_i is computed as in equation B.6 and the computation pipeline is illustrated in Figure B.4.

$$u_i = GlobalAvgPool(cnn_i(X)), i \in [1, 2...k] \tag{B.6}$$

In Equation B.6, X represents the $max\_len * k^2$ dimensional matrix obtained after PReLU layer and cnn_i represents the i-th filter.

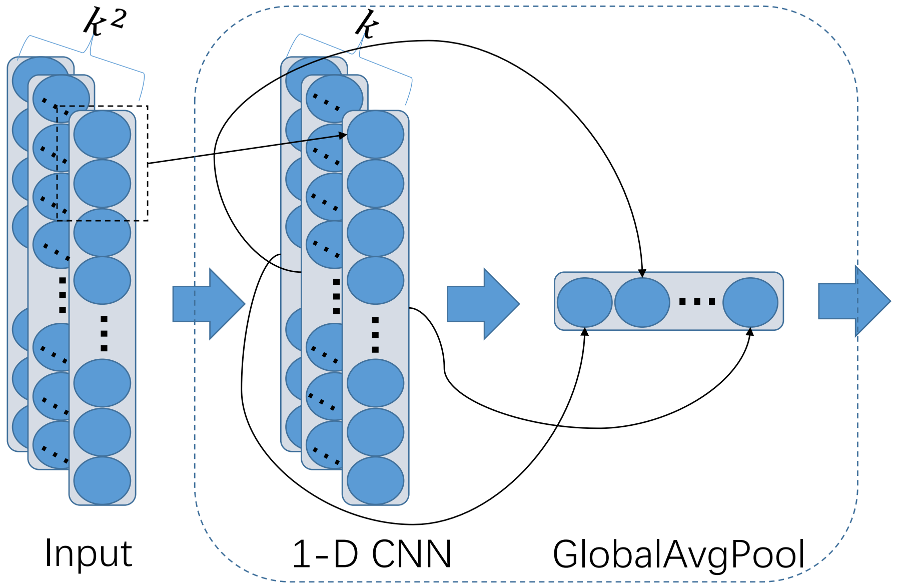


Fig. B.4: Convolution Calculation

B.5 Experiments

In this section, we compare our proposed Deep Monitor Network with the other state-of-the-art models empirically. The evaluation result indicates that DMN is more effective than any other state-of-the-art model and the efficiency of DMN is comparable to the best ones among all the deep models.

B.5.1 Experiment Setup

Datasets

We evaluate the effectiveness and efficiency of our proposed Deep Monitor Network on the following two datasets.

1) Company* Dataset: To verify the performance of DMN in real industrial CTR prediction task, we conduct experiment on Company* Dataset. We collect 0.2 billion samples (more than 1 million users) with 11 fields for 90 consecutive days of users' click records from cross-media display advertising monitor for big brand campaigns (e.g., P & G, O'Real). In this dataset, there are campaign features (e.g., advertiser, agency, and etc), brand features (e.g., brand, product, industry, and etc), and context features (e.g., campaign time, and etc). We obtain training and testing samples in two steps: given a user id and a timestamp, collect the specific user's *max_len* events before the timestamp as history events and event after the timestamp as predicted event; apply a 1 day sliding window to generate more samples. Finally we split the samples into two parts: 90% is for training, while the rest 10% is for testing.

2) Avazu Dataset: Avazu dataset[6] is provided in the competition of Kaggle in 2014. The dataset contains 40 million samples with 22 fields for 11 days. In this dataset, there is no user identifier field, instead two fields *device_id* and *device_ip* are strongly correlated to user identifier. There are more than 30 million samples with the same *device_id* which is considered meaningless. Therefore, we choose *device_ip* as a substitution of user identifer field. Meanwhile, we subsample the records of *device_id* which has more than 10 samples, result in more than 4 million users left. The training and testing samples are generated in the same way as what is done in Company* dataset.

Evaluation Metrics

We use two evaluation metrics in our experiments: AUC (Area Under ROC) and Logloss (cross entropy). The cross-entropy loss function is introduced in Equation B.7:

$$L = -\frac{1}{u}\sum_{i=1}^{N}[y_i log\hat{y}_i + (1-y_i)log(1-\hat{y}_i)] \tag{B.7}$$

where N is the training set of size N, with $y \in \{0,1\}$ as the true label of ith sample and $\hat{y}_i$ is the output of the network after the final softmax layer, denoting the predicted value on the ith sample.

Baselines

We compare 6 models in our experiments: LR, CCPM [13], DeepFM [5], MSM-RD [12], DSTN [6] and DMN. The reason we choose MSM-RD instead of other variants in [12] is that the effectiveness of MSM-RD is very close to the best variant MSM-SG while the efficiency is acceptable.

Parameter Settings

To evaluate the above models, all models follow the same hyper-parameter settings: (1) dropout: 0.5; (2) MLP network structure: 32-16; (3) optimizer: Adam; (4) learning rate: 1e-4; (5) λ: 3e-5; (6) embedding size/latent dimension: 6 (for FM and DMN); (7) batch size: 512; (8) activation function: relu; (9) max length (for DMN): 150. Two parameters (embedding size and max

[6]https://www.kaggle.com/c/avazu-ctr-prediction/data

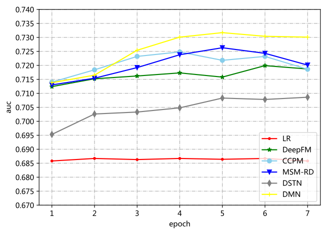
length) need to be paid attention as they affect DMN model performance, which will be discussed in Section B.5.4.

B.5.2 Performance Evaluation

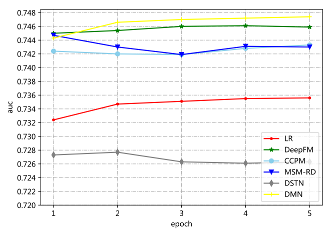
In this section, we evaluate the models listed in Section B.5.1 on the two datasets to compare their effectiveness and efficiency.

Efficiency Comparison

The efficiency of deep learning models is crucial in real industrial environment setting. We compare the efficiency of different models on the GPU running time (in second) each epoch and epochs to converge. Figure B.5 shows the process of training convergence regarding different models.



(a) Company* Dataset



(b) Avazu Dataset

Fig. B.5: Epochs for Convergence of Different Models

Table B.3: Model Efficiency Comparison

Model	Company*	Avazu
LR	6000	2000
CCPM	8500	7000
DeepFM	24000	12000
MSM-RD	46800	19000
DSTN	30000	12000
DMN	28000	10000

Table B.3 shows that DMN is comparable with DeepFM regarding the least one-epoch training time. Form Table B.4 we can draw the following observations: 1) Different deep models converge differently regarding the

Table B.4: Number of Epochs to Converge

Model	Company*	Avazu
LR	2	2
CCPM	4	9
DeepFM	11	4
MSM-RD	5	3
DSTN	5	4
DMN	5	5

epochs needed to converge, for example, CCPM runs 5 more epochs on Avazu dataset while DeepFM executes 7 more epochs on Company* dataset; 2) Both on two datasets, DMN is one of several models that require minimal epochs.

Effectiveness Comparison

The prediction effects of different CTR models on the Company* and Avazu datasets are listed in Table B.5. It can be found from Table B.5 that DMN model can effectively capture high-order interaction features and results in the best performance. On Company* dataset, DMN has a 4.0% improvement of the AUC value over the baseline model LR and 0.74% improvement over the second best model MSM-RD; on Avazu dataset, compared to LR and the second best model DSTN, the value is increased by 1.23% and 0.13% correspondingly.

Table B.5: Model Effectiveness Comparison

Model	Company*		Avazu	
	AUC	LogLoss	AUC	LogLoss
LR	0.6889	0.11045	0.7356	0.38805
CCPM	0.7248	0.10812	0.7433	0.38360
DeepFM	0.7219	0.10828	0.7461	0.38184
MSM-RD	0.7263	0.10799	0.7447	0.38327
DSTN	0.7223	0.10831	0.7475	0.38214
DMN	**0.7317**	**0.10746**	**0.7484**	**0.38169**

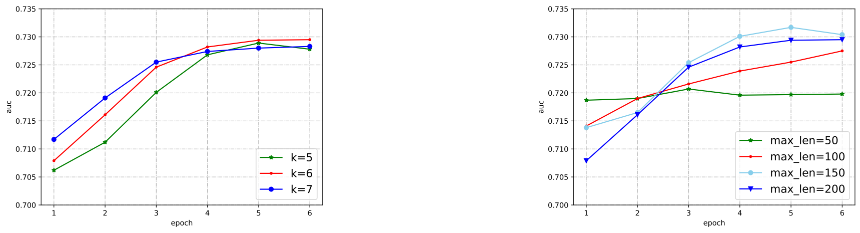
B.5.3 Impact of Different Learning Modules

As mentioned in Section B.4.1, the essential part of DMN prediction model is a specific network extracting a feature representation for user interest related to a certain predicted advertisement. Other machine learning models are able to utilize the extracted feature vector by DMN to make CTR prediction. Illustrated in Table B.6, when we connect LR with the feature extraction block of DMN, it improve the AUC from 0.6889 to 0.7078. Though the performance is still lower than DeepFM, it proves the feature extraction ability of DMN.

Table B.6: Impact of Different Learning Modules

Model	AUC	Logloss
LR	0.6889	0.11045
LR+DMN	0.7078	0.10923
DeepFM	0.7219	0.10828
DMN	0.7317	0.10746

B.5.4 Hyper-Parameter Study



(a) AUC Comparison of Different Embedding Size

(b) AUC Comparison of Different Max Length

Fig. B.6: Hyper-Parameter Tuning

We studied the performance of different hyper-parameters of DMN model on Company* dataset. Considering the main differences between DMN and other CTR learning models, we compare the effects of different parameter setting on k (embedding size) and max_len (maximum length of the history event). From Figure B.6, the analysis of the two hyper-parameters is illustrated as follows:

- embedding size k: different embedding size reflects the encode length of information mined from the fields. We tune the embedding size as 5, 6 and 7 when fixing max_len as 200. As shown in Figure B.6, there is a comparably small impact after three epochs when embedding size is set as different numbers. The model works the best when the embedding size is set 6.

- *max_len*: *max_len* represents the maximum length of historical events as model input. On one hand, in order to enable the model to be trained by mini-batch, it is necessary to align the input length. On the other hand, in order to increase the sample size by applying the sliding window, it is necessary to control the number of events of each sample by *max_len*. In the DMN model, *max_len* is a hyper-parameter that has a great impact on model effect, and it is strongly related to the experiment dataset. We compare the model performance of *max_len* to 50, 100, 150, 200 when fixing the embedding size as 6. As shown in Figure B.6, on Company* dataset, the model performs the best when *max_len* equals 150.

B.6 Conclusion

In this paper, we focus on CTR prediction task in the scenario of display advertising in advertisement monitor industry for big brand companies. We propose a DMN model for privacy-preserving CTR prediction. Due to the lack of personal profile information, DMN model aims to reduce the learning difficulties of the traditional deep CTR models by replacing user identity embedding with a sophisticated designed feature extraction network. Based on user historical behaviour, the feature extraction component obtains an adaptive encoding with regard to a certain ad, thereby improving the model expressibility. DMN relies on a Factorization Machine layer and MLP to extract both low-order and high-order feature interactions. Traditional prediction models focus on single media platform prediction, on the contrary, DMN is capable of capturing the diversity of user interests across media platforms. Meanwhile to an extent, we can easily scale DMN to traditional datasets for display advertising. The sufficiently conducted experiments indicate that DMN has better model effectiveness than current state-of-art solutions while model efficiency is comparable to the existing approaches. Furthermore, the feature extraction component of DMN can be easily re-combined with other machine learning models. This work explores a novel direction for CTR prediction that it is effective to automatically generate important and adaptive user interest representation in privacy preserving scenarios. We will explore more applications for DMN in privacy-preserving CTR scenarios for recommendation system.

References

[1] S. Rendle, "Factorization machines," in *Proc. of ICDM*, 2010, pp. 995–1000.

[2] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data - - A case study on user response prediction," in *Proc. of ECIR*, 2016, pp. 45–57.

[3] Y. Qu, B. Fang, W. Zhang, R. Tang, M. Niu, H. Guo, Y. Yu, and X. He, "Product-based neural networks for user response prediction over multi-field categorical data," *ACM Trans. Inf. Syst.*, vol. 37, no. 1, pp. 5:1–5:35, Oct. 2018.

[4] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. of the 1st Workshop on Deep Learning for Recommender Systems*, 2016, pp. 7–10.

[5] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," in *Proc. of IJCAI*, 2017, pp. 1725–1731.

[6] W. Ouyang, X. Zhang, L. Li, H. Zou, X. Xing, Z. Liu, and Y. Du, "Deep spatio-temporal neural networks for click-through rate prediction," in *Proc. of KDD*, 2019, pp. 2078–2086.

[7] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proc. of RecSys*, 2016, pp. 43–50.

[8] Y. Juan, D. Lefortier, and O. Chapelle, "Field-aware factorization machines in a real-world online advertising system," in *Proc. of WWW Companion*, 2017, pp. 680–688.

[9] J. Pan, J. Xu, A. L. Ruiz, W. Zhao, S. Pan, Y. Sun, and Q. Lu, "Field-weighted factorization machines for click-through rate prediction in display advertising," in *Proc. of WWW*, 2018.

[10] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proc. of ADKDD*, 2017, pp. 12:1–12:7.

[11] B. Liu, R. Tang, Y. Chen, J. Yu, H. Guo, and Y. Zhang, "Feature generation by convolutional neural network for click-through rate prediction," in *The World Wide Web Conference*, 2019, pp. 1119–1129.

[12] P. P. K. Chan, X. Hu, L. Zhao, D. S. Yeung, D. Liu, and L. Xiao, "Convolutional neural networks based click-through rate prediction with multiple feature sequences," in *Proc. of IJCAI*, 2018, pp. 2007–2013.

[13] Q. Liu, F. Yu, S. Wu, and L. Wang, "A convolutional click prediction model," in *Proc. of CIKM*, 2015, pp. 1743–1746.

[14] W. Liu, R. Tang, J. Li, J. Yu, H. Guo, X. He, and S. Zhang, "Field-aware probabilistic embedding neural network for ctr prediction," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 412–416.

[15] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proc. of KDD*, 2018, pp. 1059–1068.

[16] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proc. of AAAI*, 2019, pp. 5941–5948.

[17] Q. Pi, W. Bian, G. Zhou, X. Zhu, and K. Gai, "Practice on long sequential user behavior modeling for click-through rate prediction," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.*, 2019, pp. 2671–2679.

[18] Y. Deng, Y. Shen, and H. Jin, "Disguise adversarial networks for click-through rate prediction," in *Proc. of IJCAI*, 2017, pp. 1589–1595.

[19] H. Gao, D. Kong, M. Lu, X. Bai, and J. Yang, "Attention convolutional neural network for advertiser-level click-through rate forecasting," in *Proc. of WWW*, 2018, pp. 1855–1864.

[20] S. Zhai, K.-h. Chang, R. Zhang, and Z. M. Zhang, "Deepintent: Learning attentions for online advertising with recurrent neural networks," in *Proc. of KDD*, 2016, pp. 1295–1304.

[21] T. Zhou, H. Qian, Z. Shen, C. Zhang, C. Wang, S. Liu, and W. Ou, "Jump: a jointly predictor for user click and dwell time," in *Proc. of IJCAI*, 2018, pp. 3704–3710.

[22] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *KDD*, 2002.

[23] E. Bertino, D. Lin, and W. Jiang, *A Survey of Quantification of Privacy Preserving Data Mining Algorithms*. Boston, MA: Springer US, 2008, pp. 183–205.

[24] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*, 1st ed. Boston, MA, USA: Auerbach Publications, 2011.

[25] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014. [Online]. Available: http://arxiv.org/abs/1409.0473

Paper C

Finding Top-k Local Users in Geo-tagged Social Media Data

Jinling Jiang, Hua Lu, Bin Yang, Bin Cui

The paper has been published at the
31st IEEE International Conference on Data Engineering, 2015.

Abstract

Social network platforms and location-based services are increasingly popular in people's daily lives. The combination of them results in location-based social media where people are connected not only through the friendship in the social network but also by their geographical locations in reality. This duality makes it possible to query and make use of social media data in novel ways. In this work, we formulate a novel and useful problem called top-k local user search (TkLUS for short) from tweets with geo-tags. Given a location q, a distance r, and a set of keywords W, the TkLUS query finds the top-k users who have posted tweets relevant to the desired keywords in W at a place within the distance r from q. TkLUS queries are useful in many application scenarios such as friend recommendation, spatial decision, etc. We design a set of techniques to answer such queries efficiently. First, we propose two local user ranking methods that integrates text relevance and location proximity in a TkLUS query. Second, we construct a hybrid index under a scalable framework, which is aware of keywords as well as locations, to organize high volume geo-tagged tweets. Furthermore, we devise two algorithms for processing TkLUS queries. Finally, we conduct an experimental study using real tweet data sets to evaluate the proposed techniques. The experimental results demonstrate the efficiency, effectiveness and scalability of our proposals.

C.1 Introduction

Social networks like Twitter and Facebook are generating huge amounts of social media data. According to the Twitter blog, for example, Twitter users sent 4,064 tweets in a single second when 2011's Super Bowl came to its final moments[1], making the highest tweets per second (TPS) ever recorded for a sport event. In general, people use social network platforms for keeping in touch with friends and seeking various information that have social aspects.

Recently, the widespread of GPS-enabled mobile terminals and the rise of mobile location-based services (LBS) make social media data acquire geo-location information. Geo-tagged microblogs (e.g., tweets with geo-locations in metadata) play an important role in sharing observations and opinions, getting news, and understanding situations about events that are happening or that have happened in the real world. For example, local groups were quickly formed in Facebook in response to the earthquake in Haiti in 2010. As a result, so-called location-based social networks are becoming abundant sources of geo-related information.

Traditional information retrieval (IR) techniques behind search engines emphasize finding relevant information from long textual documents rich in keywords. They are not suitable for searching short-sized social media data that are characterized by few keywords. Twitter itself also offers a real-time search service[2], which returns highly-ranked tweets in response to user-input keywords. However, the spatial aspect is not handled in the search service.

With geo-tagged social media data, it is possible to offer novel services other than traditional search services. For example, a couple with kids moving to Seoul may ask "Are there any good babysitters in Seoul?" This query contains keyword "babysitter" and spatial location "Seoul". The social search engine Aardvark [1] indicates that such location-dependent and contextualized queries concerning travel, restaurants and local services in daily life are among the top popular categories.

However, social media data like tweets are inherently noisy as most pieces of them do not contain much useful information in limited number of bytes and are of little interest to a broad audience. Consequently, directly retrieving tweets may not suffice for such location-dependent queries. We argue that the information needed by such a location-dependent query instead lies in finding local social media users who are familiar with the relevant issues in a certain spatial region. Recommending local Twitter users for specific queries is very useful because people in need can directly communicate with those local users on Twitter platform. Also, Twitter maintains the social relationships among users, which we can make use of to score the users in

[1]http://blog.twitter.com/2011/02/superbowl.html
[2]https://twitter.com/search-home

finding and recommending local users.

In this work, we study the search of top-k local users in tweets that are associated with latitude/longitude information, namely geo-tagged tweets. Existing works on microblog indexing and search focus on a real-time architecture as Twitter produces a large volume of data each day. In contrast, geo-tagged tweets occupy less than 1 percent of the whole tweets data set[3]. Therefore, we find it reasonable to separately process the geo-tagged tweets in a batch mode. This allows us to collect the geo-tagged tweets data in a periodic manner and process them offline. In such a setting, we distinguish our research from the existing real-time indexing and search services in Twitter.

In addition, Section C.7 reviews the related work; Section C.8 concludes the paper with future work directions.

C.2 Problem Formulation

In this section, we formulate the problem of TkLUS queries.

C.2.1 Modeling Social Media Data

In the context of social media, we identify three different but interrelated concepts. In particular, social media consists of many *posts* that are posted by *users* who form a *social network* through their interactions in the posts.

Definition 1. ***(Social Media Post)*** *A social media post is a 4-tuple $p = (uid, t, l, W)$, where uid identifies the user who publishes the post, t is the timestamp when the post is published, l is the location where the user publishes the post, and W is a set of words $(w_1, w_2, \ldots, w_n)$ that capture the textual content of the post.*

The location field may be unavailable for many social media posts, depending on whether localization is supported (enabled) by a social media platform (a user). For example, Twitter users can choose to record their locations or not in tweets if their devices are GPS-enabled. In this paper, we focus on social media posts that have non-empty location fields, and make use of them to find relevant results for user queries.

The size of $p.W$ is always small since social media platforms impose length constraints on posts. For example, a tweet contains at most 140 characters that can only convey a small number of words. On the other hand, not all textual words in an original social media post are included in the abstraction p. We assume the use of a vocabulary $\mathcal{W}$ that excludes popular stop words (e.g., this and that).

Definition 2. ***(Social Network)*** *A social network referred in social media platform is a directed graph $G = (U, E_{reply}, l_{reply}, E_{forward}, l_{forward})$ where:*

[3]http://thenextweb.com/2010/01/15/twitter-geofail-023-tweets-geotagged/

1. *U is a set of vertices, each corresponding to a user.*
2. *E_{reply} is a set of directed edges that capture the "reply" relationships between users in U. Particularly, an edge $e = \langle u_1, u_2 \rangle \in E_{reply}$ means that user u_1* replies to *user u_2 in one or more posts.*
3. *l_{reply} : l_{reply} maps a reply edge to the set of involved posts. Given two users u_1 and u_2, $l_{reply}(u_1, u_2)$ returns all the posts in which u_1 replies to u_2.*
4. *$E_{forward}$ is a set of directed edges that capture the "forward" relationships between users in U. Particularly, an edge $e = \langle u_1, u_2 \rangle \in E_{forward}$ means that user u_1* forwards *user u_2's post(s).*
5. *$l_{forward}$: $l_{forward}$ maps a forward edge to the set of involved posts. Given two users u_1 and u_2, $l_{forward}(u_1, u_2)$ returns all u_2's posts forwarded by u_1.*

The location field may be unavailable for many social media posts, depending on whether localization is supported (enabled) by a social media platform (a user). For example, Twitter users can choose to record their locations or not in tweets if their devices are GPS-enabled. In this paper, we focus on social media posts that have non-empty location fields, and make use of them to find relevant results for user queries.

The size of $p.W$ is always small since social media platforms impose length constraints on posts. For example, a tweet contains at most 140 characters that can only convey a small number of words. On the other hand, not all textual words in an original social media post are included in the abstraction p. We assume the use of a vocabulary $\mathcal{W}$ that excludes popular stop words (e.g., this and that).

Definition 3. *(**Social Network**) A social network referred in social media platform is a directed graph $G = (U, E_{reply},$
$l_{reply}, E_{forward}, l_{forward})$ where:*

1. *U is a set of vertices, each corresponding to a user.*
2. *E_{reply} is a set of directed edges that capture the "reply" relationships between users in U. Particularly, an edge $e = \langle u_1, u_2 \rangle \in E_{reply}$ means that user u_1* replies to *user u_2 in one or more posts.*
3. *l_{reply} : l_{reply} maps a reply edge to the set of involved posts. Given two users u_1 and u_2, $l_{reply}(u_1, u_2)$ returns all the posts in which u_1 replies to u_2.*
4. *$E_{forward}$ is a set of directed edges that capture the "forward" relationships between users in U. Particularly, an edge $e = \langle u_1, u_2 \rangle \in E_{forward}$ means that user u_1* forwards *user u_2's post(s).*
5. *$l_{forward}$: $l_{forward}$ maps a forward edge to the set of involved posts. Given two users u_1 and u_2, $l_{forward}(u_1, u_2)$ returns all u_2's posts forwarded by u_1.*

The social network in a social media platform is complex as it involves both content-independent and content-dependent aspects. The definition above captures both aspects in different types of graph edges and auxiliary mappings. In particular, each reply edge in E_{reply} must involve at least one post where one user replies to another, whereas each edge in $E_{forward}$ must involve at least one post which one user forwards.

We use $P = \{(uid, t, l, W)\}$ to denote the set of all social media posts of interest, and $\mathcal{D} = (P, U, G)$ to denote the geo-tagged social media data. For convenience, we also use P_u to denote all the posts by a user $u \in U$.

C.2.2 Problem Definition

In the context of geo-tagged social media data, people may pose questions like *What are the events going on in New York City?* and *How can I find a babysitter in LA?* Among all social media users in U, there can be some local users, who have relevant knowledge indicated in their posts, to answer such questions. Without loss of generality, we formalize the questions as the following problem.

Problem Definition. **(Top-k Local User Search in Social Media)** In the context of geo-tagged social media data $\mathcal{D} = (P, U, G)$, given a query $q(l, r, W)$ where $q.l$ is a query location, $q.r$ is a distance value, and $q.W$ is a keyword set that captures user needs, a top-k local user search (TkLUS) finds a k-user set $\mathcal{E}_k \subseteq \mathcal{D}.U$ that satisfies the following conditions:

1. $\forall u \in \mathcal{E}_k$, $\exists p \in P_u$ such that $\|q.l, p.l\| \leq q.r$ [4] and $p.W \cap q.W \neq \emptyset$.
2. $\forall u \in \mathcal{E}_k$ and $\forall u' \in \mathcal{D}.U \setminus \mathcal{E}_k$, either u' does not satisfy condition 1 or $score(u', q) \leq score(u, q)$.

Here, $score(.)$ is a score function to measure the relevance of a user to a search request. Our score functions, to be detailed in Section C.3, consider both keyword and distance relevances.

We give a simplified example. Figure C.1 shows a map where a TkLUS query is issued at the location indicated by the cross (43.6839128037, -79.37356590) with a single keyword "hotel" and a range of 10 km. The tweets containing "hotel" (and their users) are listed in Table C.1, and their locations are also indicated in the map. According to different user scoring functions, user u_1 having more tweets or user u_5 having more replies/forwards (not shown in the table) will be returned.

To find the top-k users for a TkLUS query is a non-trivial problem. The user set U and the post set P are very large on many social media platforms like Twitter. It is definitely inefficient to check the sets iteratively. Also, it

[4] $\|l_1, l_2\|$ denotes the Euclidean distance between locations l_1 and l_2. The techniques proposed in this paper can be adapted to other distance metrics.

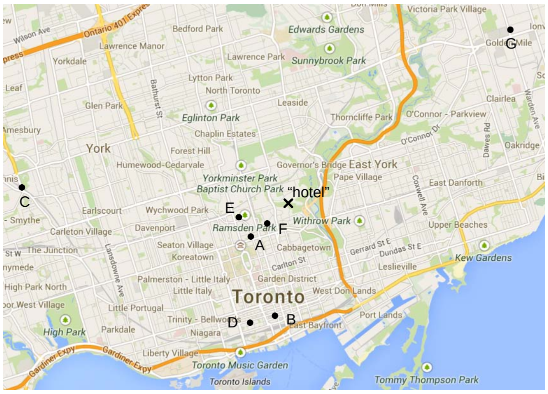


Fig. C.1: TkLUS Example: Locations of Query and Tweets

pid	uid	text
A	u_1	I’m at Toronto Marriott Bloor Yorkville Hotel
B	u_2	Finally Toronto (at Clarion Hotel).
C	u_3	I’m at Four Seasons Hotel Toronto.
D	u_4	Veal, lemon ricotta gnocchi @ Four Seasons Hotel Toronto.
E	u_5	And that was the best massage I’ve ever had.(@ The Spa at Four Seasons Hotel Toronto)
F	u_6	Saturday night steez #fashion #style #ootd #toronto #saturday #party #outfit @ Four Seasons Hotel Toronto.
G	u_1	Marriott Bloor Yorkville Hotel is a perfect place to stay.

Table C.1: Detailed Information of Example Tweets

is not straightforward to measure the relevance of a user to a given query. Furthermore, a TkLUS query can have many relevant tweets and local users, and therefore we need to rank them and return the most relevant users as the query result. In this paper, we systematically address such technical challenges.

C.3 Scoring Tweets and Users

In this section, we propose a way to measure tweet popularity, followed by the scoring functions for tweets and users.

C.3.1 Tweet Thread and Tweet Popularity

A local user that offers useful information is likely to have tweets popular in the social media. Therefore, we first consider a tweet p's popularity with respect to a query $q(l, r, W)$. Intuitively, p is not interesting if it has no keywords from $q.W$. We only consider tweets with query keyword(s) and give priority to those with more of them. On the other hand, a tweet tends to be popular if it gives rise to visible responses in the social media. For example, if a user has useful information, her/his tweets tend to be popular with many replies and/or forwards. In this sense, we also quantify tweet responses and give priority to those having more responses. To support these considerations, we introduce the concept *tweet thread*.

Definition 4. ***(Tweet Thread)*** *Given a query $q(l, r, W)$, a tweet thread T is a tree of tweets where*

1. *each node corresponds to a unique tweet;*
2. *the root $T.root$ is a tweet p that has keywords in $q.W$;*
3. *if tweet p_i is tweet p_j's parent node, p_j replies to or forwards p_i.*

Figure C.2 shows an example. Tweet p_1 has keywords requested in query $q(l, r, W)$. Tweets p_2, p_3, and p_4 reply to or forward p_1, each having further reply or forward tweets.

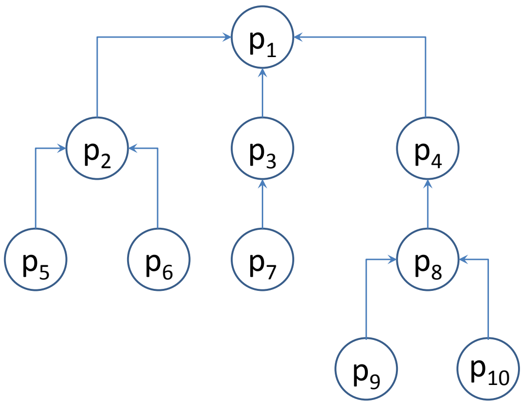


Fig. C.2: Tweet Thread Example

Intuitively, the more tweets in a tweet thread, the more popular the root tweet is. If one tweet shares no connection with other tweets, it will form a tweet thread by itself only. Apparently, such a tweet is not popular and should be given a low score in our search. Furthermore, a tweet thread with a larger number of tweets is more important than those with fewer as this means its root tweet leads a conversation involving a lot of concerns. As a result, the user who posts such a root tweet is regarded as a relevant local user for answering the given query. The algorithm of constructing tweet thread and computing the thread score will be described in Section C.4.1.

Based on the tweet thread concept, we define a tweet p's popularity as the score of tweet thread T whose root is p.

Definition 5. ***Popularity of Tweet***

$$\phi(p) = \begin{cases} \epsilon, & \textit{if } T.h = 1; \\ \sum_{i=2}^{n} |T_i| \times \frac{1}{i}, & \textit{otherwise.} \end{cases}$$

In Definition 5, $T.h$ means the height of the tweet thread and $|T_i|$ represents the number of tweets in the i-th level. In particular, i starts from 1 at the top level in the tweet thread. In addition, ϵ is a smoothing parameter set for tweet thread consisting only one tweet. For example, in Figure C.2, the score of tweet p_1 is $3 \times \frac{1}{2} + 4 \times \frac{1}{3} + 2 \times \frac{1}{4} = \frac{10}{3}$.

C.3.2 Individual Tweet's Score

To define a single tweet p's score with respect to a given query $q(l, r, W)$, we need to take into account the distance between p and the query, as well as the p's popularity with respect to q. For the former, we define the distance score $\delta(p, q)$ of a tweet as follows.

Definition 6. ***Distance Score of Tweet***

$$\delta(p, q) = \begin{cases} \frac{r - \|q.l, p.l\|}{r}, & \textit{if } \|q.l, p.l\| \le r; \\ 0, & \textit{otherwise.} \end{cases}$$

Note that $\delta(p, q)$ returns 0 if a tweet p is outside of the distance r from q. Within the distance r, a tweet gets a higher score if it is closer to q. Overall, the range of $\delta(p, q)$ is $[0, 1]$.

Next, we define the keyword relevance between a tweet p with a given query $q(l, r, W)$ as follows.

Definition 7. ***Keyword Relevance Score of Tweet***

$$\rho(p, q) = \frac{|q.W \cap p.W|}{N} \cdot \phi(p)$$

This definition counts the normalized occurrences of a query keyword in tweet p, multiplies it by p's popularity $\phi(p)$, and uses the product as p's keyword relevance. Here, N is a normalized parameter and we do not necessarily further normalize $\phi(p)$ since $\rho(p,q)$ is allowed to exceed 1. In our experimentation, N is empirically set around 40 such that keyword relevance score is comparable to the distance score. Also, the occurrences are actually counted according to a bag model of keywords. Precisely, $q.W$ is a set whereas $p.W$ is a bag/multiset. For example, if a query is "spicy restaurant" and tweet p contains one "spicy" and two "restaurant", the occurrences of a query keyword in tweet p is 3.

C.3.3 Ranking Users for Query

With the tweet score defined above, we are able to rank users who post tweets. We first consider the keyword aspect of a user with respect to a given query. We propose two ranking functions to compute the keyword relevance scores with different emphases.

Given a user u, we can look at all the tweets that u has posted and sum up their keyword relevance scores as user u's keyword relevance score. The intuition behind this is that all tweets of a user may matter if they contain keywords in a query. We call this scoring function *Sum Score*.

Definition 8. ***(Sum Score)*** *Given a tweet user u and a query $q(l,r,W)$, u's keyword relevance score is*

$$\rho_s(u,q) = \Sigma_{p\in P_u}\rho(p,q)$$

Alternatively, a user u's keyword relevance score can be calculated as the maximum keyword relevance score of all his/her tweets. The intuition behind this function is that if one tweet thread gets a very high score, the user of the root tweet in the thread will probably be the local user whom we are interested in. We call this scoring function *Maximum Score*.

Definition 9. ***(Maximum Score)*** *Given a tweet user u and a query $q(l,r,W)$, u's maximum keyword relevance score is*

$$\rho_m(u,q) = \max_{p\in P_u}\rho(p,q)$$

On the other hand, the distance score of user u is defined as the average distance of all u's posts and the query location.

Definition 10. ***(Distance Score of User)*** *Given a tweet user u and a query $q(l,r,W)$, u's distance score is*

$$\delta(u,q) = \frac{\Sigma_{p\in P_u}\delta(p,q)}{|P_u|}$$

By combining the two aspects, we define the user score.

Definition 11. ***(User Score)*** *Given a tweet user u and a query $q(l, r, W)$, u's score with respect to q is*

$$score(u, q) = \alpha \cdot \rho(u, q) + (1 - \alpha) \cdot \delta(u, q)$$

Here, the keyword part $\rho(u, q)$ and the distance part $\delta(u, q)$ are combined through a smoothing parameter $\alpha \in [0, 1]$.

Refer to the running example in Figure C.1 and Table C.1. Tweet threads are created for all the seven relevant tweets. If we use the sum score based ranking, user u_1 is ranked as the top local user because u_1 has two relevant tweets A and G (and thus two threads) and A is very close to the query location. The sum based ranking favors users with more relevant tweets. In contrast, the maximum based ranking returns u_5 as the top. In out data set, u_5's tweet E has considerably more replies and forwards than other tweets. As a result, E's tweet thread has a higher popularity than others, and u_5 yields the highest maximum score. Although E is not the closest to the query location, u_5 overall is still favored in the maximum score based user ranking due to the outstanding maximum score.

C.4 Data Organization and Indexing

In this section, we detail the organization and indexing of huge volumes of geo-tagged tweet data in our system.

C.4.1 Architecture and Data Organization

The system architecture is shown in Figure C.3. Twitter Rest API is commonly used to crawl sampled data in JSON format from Twitter. After extraction, transform and load (ETL) all the tweets metadata are stored in a centralized database. Since geo-tagged tweets occupy less than 1 percent of the whole tweets data set, we find it reasonable to separately process the geo-tagged tweets in a batch mode other than the existing real-time architectures which process general tweets. To be specific, we could periodically (e.g., one day) collect the spatial tweets and then build the index for these tweets. In such setting, a scalable index consisting of both spatial and text information is built under Hadoop MapReduce system and stored in Hadoop distributed file system (HDFS). The tweets content/text are stored in HDFS as well. The hybrid index is to be detailed in Section C.4.2. The query processing algorithms, to be detailed in Section C.5, are designed based on the database and the index.

All tweets in our system form a relation with the schema of (sid, uid, lat, lon, ruid, rsid) which is stored in a centralized metadata database as shown in Figure C.3. The meaning of each attribute is explained as follows.

Attribute "sid" represents the tweet ID which is essentially the tweet timestamp. Attribute "uid" indicates the user ID of this tweet. Attributes "lat" and "lon" contain the spatial information of this tweet. Attribute "ruid" expresses the user ID whose tweet is replied to or forwarded by this tweet and "rsid" corresponds to the tweet ID which is replied to or forwarded by this tweet. Furthermore, attribute "sid" is the primary key for which we build a B$^+$-tree. Another B$^+$-tree is built on attribute "rsid". These indexes are used to accelerate the query processing phase.

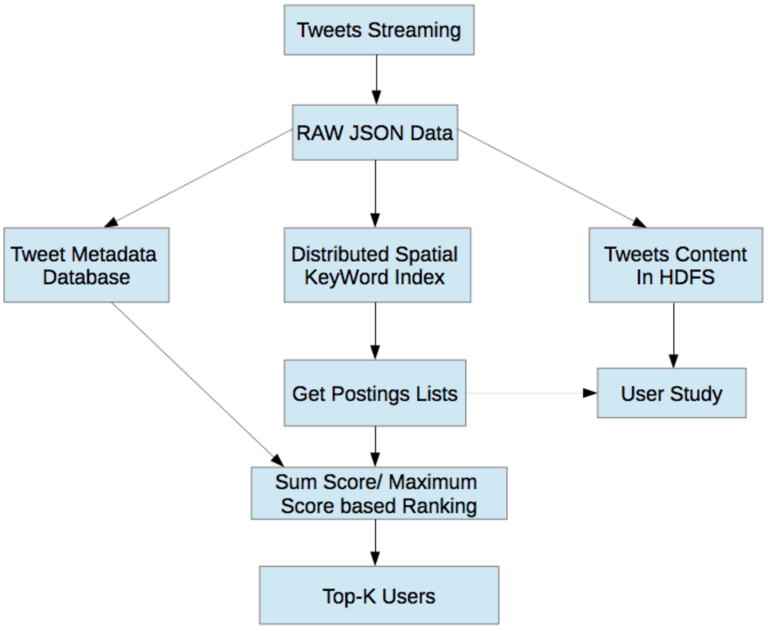


Fig. C.3: System Architecture

Based on this database schema and Definition 5, Algorithm 4 reveals the process of constructing tweet thread (detailed in Section C.3.1) and computing the thread score. Line 7 is an SQL statement where I/Os happen. Note that in a practical implementation, a thread depth d is always set to constrain the construction process since constructing a complete tweet thread can incur quite a number of I/Os.

C.4.2 Hybrid Index

Algorithm 4: Construct tweet thread and compute its score

```
Input: a tweetId $tid$, thread depth $d$
Output: score of tweet thread initiated by $tid$
1  Array ⟨Array⟩ $tweets$;
2  $Float\ score = \epsilon$;
3  $Int\ i = 1$;
4  $tweets[i]$.add ($tid$);
5  while $i \leq d$ do
6      for $j = 1; j \leq tweets[i].size; j++$ do
7          Array $temp$ =
8          select all where rsid = $tweets[i].get(j)$;
9          if $temp.size == 0$ then
10             break;
11         $tweets[i]$.add ($temp$);
12     $score$ += $tweets[i++].size \times \frac{1}{i}$;
13 return $score$
```

Index Design

Our hybrid index design adapts the Geohash encode generally derived from quadtree index. Quadtree [2] is an easily-maintained spatial index structure which divides the spatial space in a uniform way. In a quadtree, each node represents a bounding square and contains four children if it is not a leaf node. In each node split, each quadrant is obtained by dividing the parent node in half along both the horizontal and vertical axes.

If we disregard the tree structure itself and encode each children by adding two bits (00 for upper-left, 10 for upper-right, 11 for bottom-right, 01 for bottom-left) to the parent encode such that each leaf node in a *full-height* quadtree is a complete geohash. The height of the tree determines the encode precision, for example, if we would like to encode a latitude/longitude pair (-23.994140625, -46.23046875) and make the encode precision at 20 bits, the encode will be "11001111110011010100". Then we change this into Base32 encode scheme which uses ten digits 0-9 and twenty-two letters (a-z excluding a,i,l,o). Every five bits will be encoded into a character so that the final geohash is "6gxp".

From the above description, points in proximity mostly will have the same prefix so that a trie, or prefix tree could be used for indexing the geohash. To answer a circle query, a set of prefixes need to be constructed which completely covers the circle region while minimizing the area outside the query region. The Z-order [3] curve is popularly utilized to construct such set of prefixes. The reason why we adapt geohash encode system is that the data indexed by geohash will have all points for a given rectangular area in contiguous slices. In a distributed environment, data indexed by geohash will have all points for a given rectangular area in one computer. Such advantage could save I/O and communication cost in query evaluation.

The hybrid index is illustrated in Figure C.4. It contains two components: forward index and inverted index. Each entry in the forward index is in the format of $\langle ge_i, kw_i \rangle$ where ge_i is a geohash code and kw_i refers to a keyword. In our design, the forward index is not very large and it is therefore kept in the main memory. The forward index associates each of its entry to a postings list ($\mathrm{P_i}$) in the inverted index that is stored in Hadoop HDFS. Based on the inverted index, $\langle ge_i, kw_i \rangle$ pairs are associated with tweet IDs in the tweet database. In other words, the inverted index refers each pair $\langle ge_i, kw_i \rangle$ to those tweets in the database that have kw_i in its content.

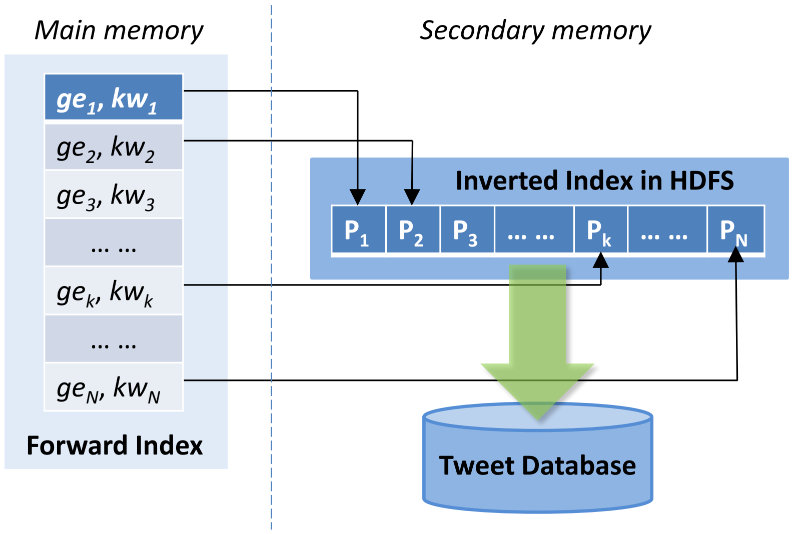


Fig. C.4: Index Structure

In our implementation, the key of the inverted index is a pair of geohash code and a keyword. In this paper, we use "term" and "keyword" exchangeably as they essentially have the same meaning in the literature. Each entry in a postings list is a pair $\langle TID, TF \rangle$. Specifically, TID is the tweet ID that is essentially the tweet timestamp and TF represents the term frequency which is useful to count the occurrences of query keyword in a tweet when processing a query. The forward index for fetching the postings in query processing is implemented according to a previous proposal [4].

As we handle large volumes of geo-tagged tweet data, we choose to exploit the Hadoop MapReduce [5] [6] programming paradigm that offers scalability and fault-tolerance. Geohash encoding scheme fits well when we build the hybrid index for both spatial and text aspects in MapReduce.

Index Construction

Algorithms 5 and 6 illustrate the Map and Reduce steps of index construction algorithm respectively under the Hadoop MapReduce framework. The input to the mappers is a social media post p defined in Definition 1. In the map function, the content of each post is tokenized and each term is stemmed. Stop words are filtered out during the tokenization process. An associative array H is used to keep track of term frequency for each term. Then the map function traverses all terms: for each term, compute the geohash on the post's spatial information and create a posting. The posting is a pair consisting of timestamp $p.timestamp$ and term frequency $H\langle w \rangle$. Note that timestamp $p.timestamp$ corresponds to the tweet ID in our context since each timestamp is unique. Finally, the mapper emits a key-value pair with a pair $\langle geohash, term \rangle$ as the key and the posting as the value. In the reduce phrase, a new list P is initialized. Then the reduce function appends all postings associated with each pair $\langle geohash, term \rangle$ to the list. Each posting is a pair $\langle n, f \rangle$ where n represents tweet ID (timestamp) and f means the frequency. Finally, the postings are sorted by the timestamp $p.timestamp$ before they are emitted. The subsequent intersection operations on the sorted postings can be very efficient in the query processing (refer to Algorithms 7 and 8 in Section C.5).

Algorithm 5: Pseudo-code of Map function

```
  Input: social media post $p$
1 AssociativeArray $\langle String, Integer \rangle$ $H$;
2 for all term $w \in p.W$ do
3     $H\langle w \rangle = H\langle w \rangle + 1$;
4 for all term $w \in H$ do
5     Emit ($\langle geohash(p.l), w \rangle$, $\langle p.timestamp, H\langle w \rangle \rangle$);
```

To construct the aforementioned forward index (see Figure C.4), another MapReduce job is run over the inverted index files in HDFS and a posting forward index is created to keep track of the position of each postings list in HDFS. Note that the Hadoop MapReduce framework can guarantee that the key of the inverted index is sorted. That means, the composite key $\langle geohash, term \rangle$ is sorted. As mentioned before, close points will probably have the same prefix of geohash so that close points associated with the same keyword are probably stored in contiguous disk pages. Such organization will substantially reduce the time accessing the postings lists which belong

Algorithm 6: Pseudo-code of the Reduce function

```
   Input: $\langle geohash\ g, term\ w \rangle$, postings $[\langle n_1, f_1 \rangle \ldots]$
1  List $P$;
2  for all posting $\langle n, f \rangle \in postings\ [\langle n_1, f_1 \rangle \ldots]$ do
3      $P$.Append ($\langle n, f \rangle$);
4  $P$.Sort( );
5  Emit($\langle geohash\ g, term\ w \rangle$, postings $P$)
```

to close areas with the same keyword. In addition, by using MapReduce we can easily scale our index construction to terabyte-scale and even petabyte-scale data set.

C.5 Query Processing Algorithms

We detail query processing algorithms in this section.

C.5.1 Algorithm for Sum Score Based User Ranking

Algorithm 7 shows the sketch of the sum score based user ranking algorithm for processing TkLUS queries. A list of geohash cells are returned by a circle query for given a location and radius (Line 1). Lines 2 to 8 get corresponding postings lists. Lines 9 to 14 show the difference between "AND" and "OR" semantics. The "AND" semantic requires the search results containing all the query keywords while the "OR" semantic relaxes the constraint by requiring a portion of the query keywords. Generally, "OR" semantic retains much more candidates than "AND" semantic. Note that in both "AND" and "OR" semantics, we maintain an associative array *tweetCount* in order to keep track of the occurrences of a query keyword in a tweet (Lines 12 and 16). Lines 17 to 26 show the process of traversing the postings list P and calculating the user score according to Definition 11.

C.5.2 Algorithm for Maximum Score Based User Ranking

In running the query processing algorithm proposed in Section C.5.1, we observe that the bottleneck of query processing lies in constructing the tweet thread for each relevant posting since every construction will cost several I/Os. To alleviate this problem, we define the upper bound score of a single tweet thread and make use of it to speed up query processing according to the maximum score (Definition 9).

In the following, we define the upper bound popularity of a tweet thread, where t_m means the maximum number of replied tweets a tweet can have in

Algorithm 7: Query Processing for Sum Score Based Ranking

```
Input: a set of Keywords $W\{w_1, w_2, \ldots, w_n\}$, a location $q$, radius $r$,
       number of returning results $k$
Output: $k$ UserIds $(u_1, u_2, \ldots, u_k)$
Array Geohashes = GeoHashCircleQuery($q$,$r$);
Array PostingsLists ;
AssociativeArray $\langle Long, Float\rangle$ topKUser;
for $i = 1; i \leq Geohashes.size; i++$ do
    for $j = 1; j \leq W.size; j++$ do
        get postings list PL for key $(GE_i, W_j)$;
        add PL into PostingsLists;
List P;
if AND semantic then
    intersect the postlings lists from PostingsLists;
    assign the intersect result to P;
else if OR semantic then
    union the postlings lists from PostingsLists;
    assign the union result to P;
for $k = 1; k \leq P.size; k++$ do
    if distance between $P_k$ and $q > r$ then
        continue;
    Compute thread score according to Algorithm 4;
    Compute relevance score RS according to Definition 8 and
      Definition 11;
    select userId where sid = $P_k.sid$;
    if $userId \notin topKUser$ then
        topKUser.add(userId, RS);
    else
        Update RS for userId;
topKUser.Sort( );
return topKUser;
```

our database.

Definition 12. ***Upper Bound Popularity of Tweet Thread***

$$\phi(p)_m = \sum_{i=2}^{n} |t_m| \times \frac{1}{i}$$

Algorithm 8 shows the sketch of the relevant query processing according to the maximum score. Specifically, a priority queue *topKUser* maintains the current top-k query results. Lines 1 to 14 are the same as the counterpart in Algorithm 7. The pruning process is showed at Lines 21 and 22. If the upper bound score of a tweet thread plus distance score is less than the top score maintained in the priority queue, there is no need to construct the thread for that tweet. In the running example (see Figure C.1 and Table C.1), user u_1 has two relevant tweets A and G. Algorithm 8 makes use of the upper bound popularity (Definition 12) and avoids constructing the threads for u_1 if u_1's overestimated user score based on the upper bound is less than u_5's user score that is already computed.

Definition 12 is the global upper bound for all keywords that can appear in all the tweets in a specific tweet thread, since t_m is the maximum number of replied tweets that a tweet can have in our database. As a matter of fact, the upper bound of any "specific keyword related" tweet threads should be much smaller than t_m. Therefore, it is desirable to maintain a specific bound for some hot keywords. Such a bound is expected to be lower than the global upper bound and thus tighter.

In our implementation, we identify from the data set top-10 frequent keywords as shown in Table C.3. For each top frequent keyword, a specific upper bound popularity is pre-computed by offline constructing tweet threads and selecting the largest thread score as the "specific keyword related" upper bound. For example, for hot keyword "restaurant", we may maintain a upper bound popularity for all tweet threads initiated by the tweet containing keyword "restaurant". When queries with keyword "restaurant" come in, we use the "restaurant related" upper bound popularity instead of the global upper bound popularity. For queries without any hot keyword, global upper bound popularity is still used. We evaluate the effect of such hot keyword upper bounds in Section C.6.2.

It is possible that one user has many tweet threads but none of them gets a top-k highest score. In such a case, the maximum score based query processing algorithm will not include the user as a top-k local user in the query result. Note that the user may be included in the query results by the sum score based query processing algorithm, due to the user's large number of tweet threads. We compare the two query processing algorithms by measuring the Kendall Tau coefficient of their query ranking results in Section C.6.2.

Algorithm 8: Query Processing for Maximum Score Based Ranking

```
Input: a set of keywords $W\{w_1, w_2, \ldots, w_n\}$, a location $q$, radius $r$,
       number of returning results $k$
Output: $k$ UserIds $(u_1, u_2, \ldots, u_k)$
Array $Geohashes$ = GeoHashCircleQuery($q$,$r$);
Array $PostingsLists$ ;
PriorityQueue $\langle PairOfLongFloat \rangle topKUser$;
for $i = 1; i \leq Geohashes.size; i++$ do
    for $j = 1; j \leq W.size; j++$ do
        get postings list $PL$ for key $(GE_i, W_j)$;
        add $PL$ into $PostingsLists$;
List $P$;
if $AND$ $semantic$ then
    intersect the postlings lists from $PostingsLists$;
    assign the intersect result to $P$;
else if $OR$ $semantic$ then
    union the postlings lists from $PostingsLists$;
    assign the union result to $P$;
for $k = 1; k \leq P.size; k++$ do
    if $distanceScore$ of tweet (Definition 6) $> r$ then
        continue;
    if $topKUser.size() == k \wedge UpperBound < topKUser.peek()$ then
        continue;
    Compute thread score according to Algorithm 4;
    Compute relevance score $RS$ according to Definition 9 and
     Definition 11;
    if $topKUser.size < k$ then
        if $userId \notin topKUser$ then
            $topKUser$.add($userId, RS$);
        else
            Update relevance score for $userId$;
    else if $topKUser.size == k$ then
        if $userId \notin topKUser \wedge topKUser.peek() < RS$ then
            $topKUser$.remove();
            $topKUser$.add($userId, RS$);
        else if $userId \in topKUser \wedge userId.score > RS$ then
            Update relevance score for $userId$;
return $topKUser$;
```

C.6 Experimental Study

We conduct an extensive experimental study to evaluate the proposed techniques for processing TkLUS queries. We sample a real Twitter data set with geographical coordinates collected by using Twitter REST API. The 20.3GB data set covers the period from September 2012 to February 2013, having nearly 514 million geo-tagged tweets in total. In this section, we report on the relevant experimental results.

C.6.1 Index Construction Time and Storage Cost

Our first experiments evaluate the index construction. We vary Geohash configuration and measure the index construction time and index size. We construct the proposed hybrid index (Section C.4.2) in a Hadoop MapReduce cluster of three PCs (see Table C.2 for details). The CPUs of three PCs are all Quad Core Intel(TM) i7 Intel(R) Core(TM) i7. One PC is configured as the master and the others are used as slaves.

Figure C.5 shows that the index construction efficiency is insensitive to the Geohash configuration. The index construction time is steady around 850 minutes. The state-of-the-art spatial keyword index I^3 [7], built in a single machine (Quad-Core AMD Opteron (tm) Processor 8356 and 64GB memory), processes 15 million geo-tagged tweets in around 1,500 minutes. In contrast, our MapReduce index construction algorithm processes one order of magnitude more tweets but incurs one order of magnitude less time than I^3 index. Note that our index is built and stored in a distributed way while the others like I^3 and IR-tree variants are centralized and unable to process large scale data; neither can they solve TkLUS queries.

Figure C.6 shows the results of the hybrid index sizes. The index size is very steady as the Geohash configuration varies, occupying about 3.5 gigabytes in HDFS. This index size is almost the same as I^3, although our indexes handle one order of magnitude more tweets.

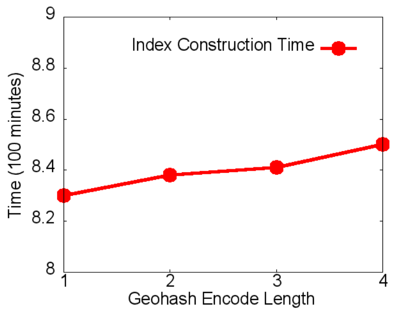


Fig. C.5: Index Construction Time

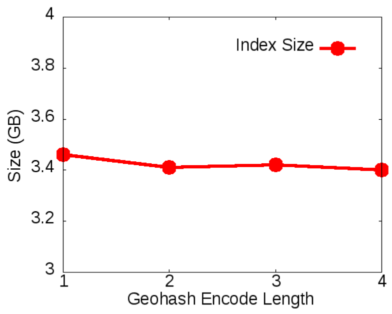


Fig. C.6: Index Size

C.6.2 Evaluation of Query Processing

In this part of experiments, we still store the index and tweets on HDFS, and keep tweet metadata in a central database. The cluster parameters are given in Table C.2.

Node Type	Memory	Hard Disk	CPU
Master	8GB	220GB	Quad Core Intel(TM) i7
Slave	8GB	500GB	8 Core Intel(TM) i7
Slave	8GB	500GB	8Core Intel(TM) i7

Table C.2: Cluster Summary

Query Settings

Based on the data set statistics, we select 30 meaningful keywords including the top-10 frequent ones shown in Table C.3. In the experiments, a 1-keyword query randomly gets one out of the 30. Queries with 2 and 3 keywords are constructed from AOL query logs that contain the single keyword from Table C.3. For example, queries "restaurant seafood" and "morroccan restaurants houston" occurring in AOL query logs are used in our experiment. Each query is randomly associated with a location that is sampled according to the spatial distribution in our data set. Finally, random combinations of keywords and locations form a 90-query set used in our experiments. Each query type in terms of keyword number (single keyword, two keywords and three keywords) has 30 corresponding queries in the query set.

According to Equation 11, we set α as 0.5 so that the two factors are considered as having the same impact. Both HDFS caches and database caches are set off in order to get fair evaluation results. Before query processing starts, the system first loads the postings forward index into memory since

Freq. Rank	Keyword		Freq. Rank	Keyword
1	restaurant		2	game
3	cafe		4	shop
5	hotel		6	club
7	coffee		8	film
9	pizza		10	mall

Table C.3: Top-10 Frequent Keywords

it is always small (less than 12MB in our experiments). Random access to inverted index in HDFS is disk-based.

When issuing the queries in the experiments, we vary the number of query keywords ($|q.W|$) from 1, 2 to 3, and the number k of local users to return from 5 to 10. On each configuration, we vary the query radius from 5 km to 20 km. The ϵ in Definition 4 is set 0.1 in our implementation.

Effect of Geohash Encoding Length

We first vary the Geohash configuration from 1-length encoding to 4-length encoding, and see the effect on the query processing. As an example, suppose the coordinate is (-23.994140625, -46.23046875), Table C.4 shows the geohash at different lengths.

length	geohash	length	geohash
1	6	3	6gx
2	6g	4	6gxp

Table C.4: Geohash Encoding Length Example

On each configuration, we vary the query radius from 5 km to 20 km. For each query radius, we issue 10 queries randomly chosen from the query set described in Section C.6.2. We report the average query processing time in Figure C.7.

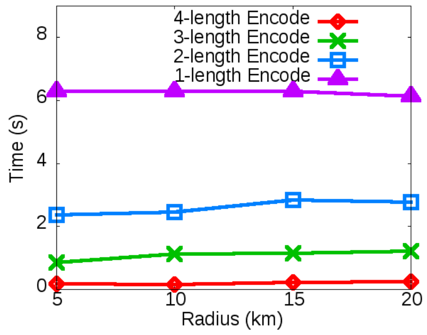


Fig. C.7: Effect of Geohash Encoding Lengths

A shorter length Geohash encoding means a coarser grid, i.e., fewer but larger grid cells. Although a longer encoding complicates the construction of a set of prefixes completely covering the query range and thus results in more candidate grid cells in search, it does not necessarily lead to more I/Os since the close points associated with the same keyword are probably stored in contiguous disk pages. With almost the same I/O cost, a shorter length encoding yields larger grid cells and needs to process more points in each cell. Thus, longer length encodings benefit TkLUS query processing.

Nevertheless, it does not mean that the longest Geohash encoding is favorable since the query efficiency also depends on the query range. If the query range is large and the encoding is long, I/O cost can increase substantially. In our experiments, the queries focus on searching for local users within the distance of 5 km to 20 km from the query location. These are practical settings for many location based services, for which the 4-length encoding works well. Moreover, the 4-length encoding is sufficient to handle the precision since the latitudes and longitudes have at most 8 bits of decimals in our data set. Based on these observations, we decide to use 4-length geohash encoding in the remaining experiments.

Results on Queries with Single Keyword

Next, we vary the query radius from 5 km to 100 km and use only the single keyword queries described in Section C.6.2.

We compare the query processing efficiency of the aforementioned two ranking methods (namely sum score based and maximum score based), and report the results in Figure C.8. Both methods incur longer query processing time when the query range increases. For a query range not greater than 20 km, the two methods perform closely and the maximum score based ranking method is only slightly better. For larger query ranges, the maximum score based ranking method clearly outperforms the sum score based ranking method. The superiority of maximum score based ranking method for large query ranges is attributed to its pruning power that works more visibly when there are more candidates involved in large query ranges. Small query ranges contain less tweets and thus the pruning does not make a big difference.

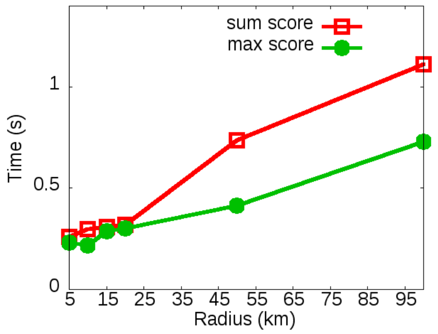


Fig. C.8: Single Keyword Efficiency

We investigate Kendall tau rank correlation coefficient between the query results using two different ranking functions. Suppose that the two top-k query results are ρ_b and ρ_d respectively. Let cp be the number of user pairs (u_i, u_j) whose rankings in ρ_b and ρ_d are concordant, i.e., if u_i is ranked before (after or in tie with) u_j in both ρ_b and ρ_d. Let dp be the number of user pairs (u_i, u_j) whose rankings are discordant in the two results. The Kendall tau coefficient is thus defined as

$$\tau(\rho_b, \rho_d) = \frac{cp - dp}{0.5k(k-1)}.$$

Since the query results using two different ranking functions are not exactly the same, a variant of Kendall tau rank correlation coefficient is proposed here to compare the two rankings. For example, suppose $k = 3$, ρ_b is $\langle A, B, C\rangle$ and ρ_d is $\langle B, D, E\rangle$. In order to measure the Kendall tau coefficient between ρ_b and ρ_d, we modify ρ_b and ρ_d to $\langle A, B, C, D, E\rangle$ and $\langle B, D, E, A, C\rangle$ respectively. The elements we add into a ranking have the same ordering value, e.g., elements D and E both are ranked the 4th in ρ_b. The same applies to ρ_d.

We thus measure the variant Kendall tau coefficient for single keyword top-5 and top-10 query results, and show the coefficient in Figure C.9. In all tested settings, the Kendall tau coefficient is higher than 0.863. This suggests that the two different ranking methods are highly consistent.

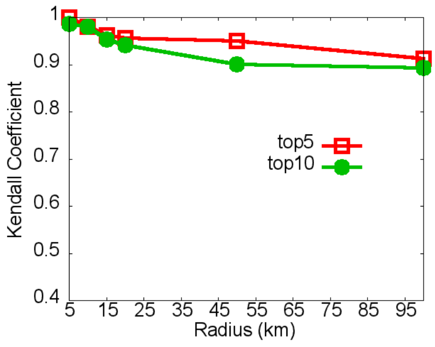


Fig. C.9: Kendall Tau for Single Query Keyword

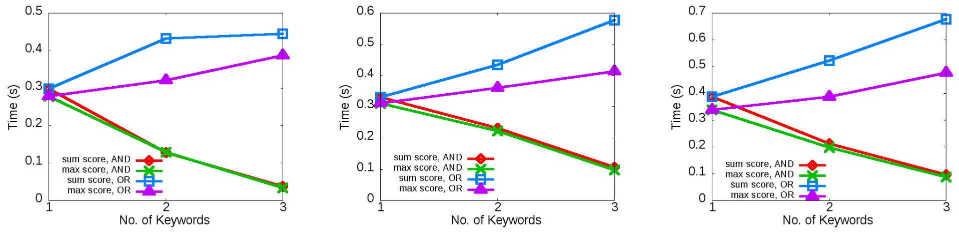
Results on Queries with Multiple Keywords

In this set of experiments, we vary the number of keywords in each query from 1 to 3 and see the effect on the query processing.

In the query processing, either OR or AND semantic can be used to handle the multiple keywords included in a query. The results on query processing efficiency are reported in Figure C.10. Overall, more keywords in the query incur longer query processing time in OR semantic while the opposite in

AND semantic. The reason is straight forward since AND semantic filters out more candidates.

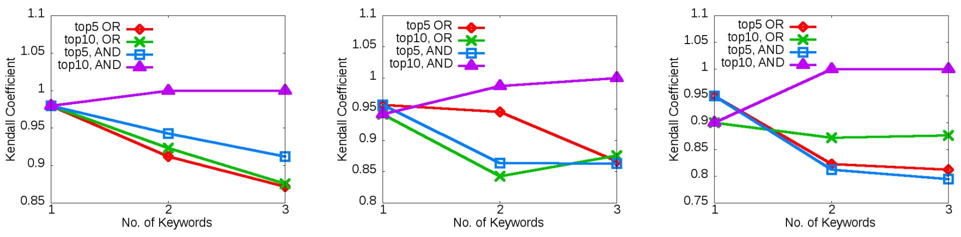
Generally, the maximum score based user ranking yields a better performance compared to the sum score based ranking, especially for 20 km and 50 km query ranges. The intersection operation used in processing the AND semantic prunes a lot of candidates, such that there is not much room for the pruning power for maximum score based ranking to take effect. In contrast, the OR semantic is processed by the union operation that results in more candidates and more room for pruning.



(a) 10 km query range (b) 20 km query range (c) 50 km query range

Fig. C.10: Query Efficiency with Multiple Query Keywords

In the same set of experiments, we also measure the Kendall tau coefficient in the same way as described in Section C.6.2. The experimental results are shown in Figure C.11. For the AND semantic, the Kendall tau coefficient is always higher than 0.95 at different query ranges. This indicates that two user ranking methods always return highly similar query results. For the OR semantic, the lowest Kendall tau coefficient is slightly below 0.8. The two ranking methods are still consistent.

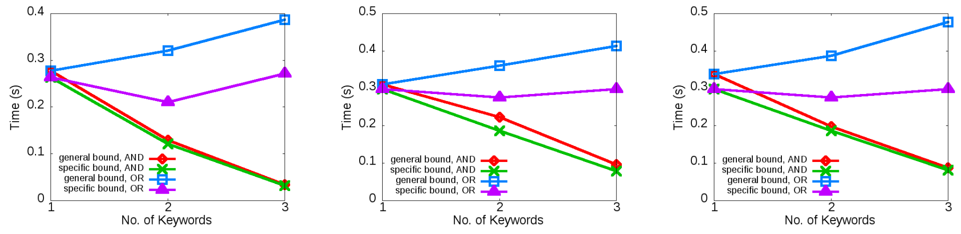


(a) 10 km query range (b) 20 km query range (c) 50 km query range

Fig. C.11: Kendall Tau for Multiple Query Keywords

Effect of Specific Popularity Bound for Maximum Score Based User Ranking

As described in Section C.5.2, the maximum score based query processing algorithm utilizes specific upper bound tweet popularity derived from hot keywords. Such specific bounds are pre-computed and expected to improve the performance for queries that contain those hot keywords. We maintain upper bound popularity for 10 hot keywords shown in Table C.3 and Figure C.12 reports the results about the query improvements in comparison to the use of the general upper bound (Definition 12 in Section C.5.2). For multiple keywords, "AND" semantic uses the smallest upper bound among the query keywords whereas "OR" semantic chooses the largest upper bound. For example, if "Mexican restaurant" is in a query and the upper bound popularity of "restaurant" is larger than "Mexican", "AND" semantic will use the upper bound popularity of "Mexican" whereas "OR" semantic chooses the upper bound popularity of "restaurant".



(a) 10 km query range (b) 20 km query range (c) 50 km query range

Fig. C.12: Effect of Specific Tweet Popularity Bound

It is clear that using such specific popularity bound of hot keywords fastens the query processing for both semantics. As the query range increases, the performance gain becomes more visible. The specific popularity bound is effective because those hot keywords help rule out irrelevant tweets when the query processing algorithm computes tweet threads.

User Study

We conduct a user study to evaluate the effectiveness of our scoring mechanisms (Section C.3) for top-*k* local users. A total of 30 queries with one to three keywords are issued at random. For each query, we output the top-10 query results with different query ranges (5, 10, 15, and 20 km). Each line of a top-10 query result is formulated as a pair (userId, tweet content), where userId identifies a user and the tweet content is the corresponding tweet keywords found by the query. Six participants familiar with Twitter are invited to give relevance feedbacks for the query results. A participant gives 1 to a query result line if she/he thinks it is relevant to the query, or 0 otherwise. We assign 20 top-10 query results to each participant so that each query result are evaluated four times. If a particular Twitter user's tweets are considered relevant twice or even more, that user will be regarded as a relevant user for the corresponding query.

We adapt precision as the effectiveness metric in the user study. Precision in our context is defined as the fraction of the returned local users that are regarded as relevant by the user study. We measure the precision for top-5 and top-10 query results, and the results are shown in Figure C.13.

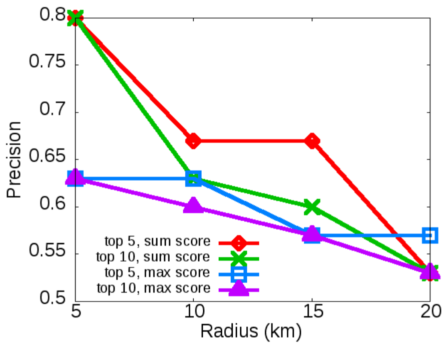


Fig. C.13: User Study Results

Overall, both user ranking methods (sum score based and maximum score based) are very effective for small and moderate query ranges—the precision is always between 60% and 80% for query ranges not larger than 10 km. The precision roughly decreases as the query range increases. These findings justify our distance score that assigns higher importance to tweets created at locations closer to the query location.

On the other hand, top-5 query results always achieve higher precision (up to 80%) than the top-10 counterparts. This certainly indicates that our user ranking methods are effective in prioritizing more relevant local users in the query results.

C.7 Related Work

C.7.1 SIR and Spatial Keyword Search

Spatial Information Retrieval (SIR) refers to providing access to geo-referenced sources by indexing, searching, retrieving and browsing [8]. "Web-a-Where" [9] attempts to build an IR system for tagging each web page with a geographic focus. "Steward" [10] provides a complete architecture of a spatio-textual search engine.

IR-trees combine R-tree with inverted index [11, 12]. In an IR-tree, each node represents an MBR that is associated with an inverted file. That inverted file refers to all the documents whose locations fall into the associated MBR. Although IR-tree family utilize both spatial and text attributes to prune the search space to facilitate query evaluation, it suffers from scalability since node splitting in such data structure will incur much I/O cost. Based on IR-tree, Rocha-Junior et al. [13] proposed the S2I index which improves the query evaluation performance. Recently, the I^3 [7] index is proposed to offer better scalability for large data sets and significantly outperforms in data update and query processing.

All these are centralized solutions and cannot scale seamlessly in a distributed way for extremely large data sets that a single computer cannot handle efficiently. In contrast, this work proposes an indexing mechanism to handle high-volume social media data in a decentralized manner. Furthermore, this work focuses on finding local users in social media, differing from mobile local search [14, 15] that considers no social context recommends businesses close to query locations.

C.7.2 Microblog and Social Media Data Management in General

TI [16] addresses real-time microblog indexing and search. The core part of TI is to classify tweets into distinguished and noisy tweets. The classification algorithm mainly depends on the statistics of past query log. The intuition behind the classification is that tweets getting high score on past queries are more probably to be retrieved in the future. After classification, the distinguished tweets are indexed immediately, namely in real time while the rest (noisy tweets) are processed in batch operation. In that way, TI substantially reduces the number of tweets it needs to process in real-time. However, TI suffers from scalability because of its centralized system design and the classification algorithm is not applicable when future queries are different from past query log.

Earlybird [17] is the proprietary distributed system developed by Twitter to overcome the practical system issues inside Twitter such as inverted index organization and concurrent control. The most recent academic work on real-time microblog indexing and search is Pollux [18] which is also a distributed system aiming to solve the concerns of fault tolerance and global storage effectiveness.

The provenance model [19] targets at the connections between microblog posts, clustering them into so-called bundles with respect to how the posts are originated and how they evolve in the microblog platform. Indexing and retrieval techniques are also proposed [19] to efficiently manage such microblog bundles. The provenance model of microblogs is query independent.

In contrast, the tweet thread proposed in this paper is dynamically generated according to the current user query. Moreover, the provenance model does not support location-dependent search in the microblog data.

Since geo-tagged tweets occupy less than 1 percent of the whole tweets data set, we propose a batch processing setting in Section C.4.1 other than the existing real-time architectures.

Our TkLUS problem differs from the microblog based jury selection problem [20] that involves no locations or keywords but employs a boolean voting to find a set of jurors that achieve high votes. Our TkLUS problem also differs from a friend search approach [21] in which only the query issuer's friends are explored and returned according to multiple scores.

A recent work [22] mines influential event organizers from online social networks. Compared to this paper, it solves a different problem with a different model of social network, and it considers no spatial locations.

C.7.3 Exploiting Locations in Social Media

Spatial-aware search and analysis of social media becomes an important research topic [23]. Lee et al. [24] develop algorithms to mine tweets stream for real-time and geo-spatial event information. Kim et al. [25] construct and visualize spatiotemporal trends of topics based on geo-tagged tweets.

A location-based social network (LBSN) consists of social structures made up of individuals, who are connected by the interdependency derived from their locations in the physical world, as well as their location-tagged media content such as photos, video, and text [26].

Aardvark [1] is a social search engine. When users ask a question in Aardvark, the system would route the question to other users who are most likely to answer this question. The key point in such social search engine is to find the right *person* to satisfy one's information need instead of retrieving the right *document*. In this social engine, location is a key factor utilized in the process of routing the question since many of the questions are locality sensitive.

Unlike related works, TkLUS studied in this paper is a new problem in the context of geo-tagged tweets. This paper distinguishes itself from the literature by the tweet thread concept, the scoring functions for tweets and users, the MapReduce-enabled hybrid index for tweets, and TkLUS query algorithms.

C.8 Conclusion and Future Work

This paper tackles top-k local user search (TkLUS) queries in geo-tagged social media. Given a location q, a distance r, and a set W of keywords that describe user needs, a TkLUS query finds the top-*k* local users who have posted tweets relevant to the keywords in *W* at a place within the distance *r* from *q*. A set of relevant techniques are designed to process such TkLUS queries. The concept of tweet thread is proposed to capture tweet popularity, based on which ranking functions that take into account both keyword relevance and spatial proximity are defined for tweets and users. A hybrid index is design for high volume geo-tagged tweets in a distributed mode. Pruning bounds and efficient algorithms are developed for processing TkLUS queries. The proposed techniques are evaluated through experiments on a large set of real tweets with geo-tags. The experimental results demonstrate that our proposals are scalable, efficient and effective.

Several directions exist for future research. The definition of TkLUS query can be extended by introducing temporal considerations in prioritizing tweets and local users. For example, we can define a query for a particular period of time and only search the tweets that are posted in that period. Also, we can still search all tweets but give priority to more recent tweets (and their users) in ranking.

There are also tweets that lack longitude/latitude in the metadata but mention place name(s) in the short content. It is worth studying how to exploit the implicit spatial information in such tweets to serve user needs like TkLUS queries.

We focus on geo-tagged tweets in this paper. It is also interesting to make the search for local users across the platform boundary, such that more informative query results can be obtained by involving different social networks.

Acknowledgments

The authors would like to thank Mr. Anders Skovsgaard for his generous help on collecting the large volumes of geo-tagged tweet data used in the experimental study.

References

[1] D. Horowitz and S. D. Kamvar, "The anatomy of a large-scale social search engine," in *WWW*, 2010, pp. 431–440.

[2] R. A. Finkel and J. L. Bentley, "Quad trees: A data structure for retrieval on composite keys," *Acta Inf.*, vol. 4, pp. 1–9, 1974.

[3] H. Samet, *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[4] J. J. Lin, T. Elsayed, L. Wang, and D. Metzler, "Of ivory and smurfs: Loxodontan mapreduce experiments for web search," in *TREC*, 2009.

[5] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI*, 2004, pp. 137–150.

[6] T. White, *Hadoop - The Definitive Guide: MapReduce for the Cloud*. O'Reilly, 2009.

[7] D. Zhang, K.-L. Tan, and A. K. H. Tung, "Scalable top-k spatial keyword search," in *EDBT*, 2013, pp. 359–370.

[8] R. R. Larson and P. Frontiera, "Geographic information retrieval (gir): searching where and what," in *SIGIR*, 2004, p. 600.

[9] E. Amitay, N. Har'El, R. Sivan, and A. Soffer, "Web-a-where: geotagging web content," in *SIGIR*, 2004, pp. 273–280.

[10] M. D. Lieberman, H. Samet, J. Sankaranarayanan, and J. Sperling, "Steward: architecture of a spatio-textual search engine," in *GIS*, 2007, p. 25.

[11] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects," *PVLDB*, vol. 2, no. 1, pp. 337–348, 2009.

[12] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang, "Ir-tree: An efficient index for geographic document search," *TKDE*, vol. 23, no. 4, pp. 585–599, Apr. 2011. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2010.149

[13] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg, "Efficient processing of top-k spatial keyword queries," in *SSTD*, 2011, pp. 205–222.

[14] D. Lymberopoulos, P. Zhao, A. C. König, K. Berberich, and J. Liu, "Location-aware click prediction in mobile local search," in *CIKM*, 2011, pp. 413–422.

[15] Y. Lv, D. Lymberopoulos, and Q. Wu, "An exploration of ranking heuristics in mobile local search," in *SIGIR*, 2012, pp. 295–304.

[16] C. Chen, F. Li, B. C. Ooi, and S. Wu, "Ti: an efficient indexing mechanism for real-time search on tweets," in *SIGMOD Conference*, 2011, pp. 649–660.

[17] M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin, "Earlybird: Real-time search at twitter," in *ICDE*, 2012, pp. 1360–1369.

[18] L. Lin, X. Yu, and N. Koudas, "Pollux: towards scalable distributed real-time search on microblogs," in *EDBT*, 2013, pp. 335–346.

[19] J. Yao, B. Cui, Z. Xue, and Q. Liu, "Provenance-based indexing support in micro-blog platforms," in *ICDE*, 2012, pp. 558–569.

[20] C. C. Cao, J. She, Y. Tong, and L. Chen, "Whom to ask? jury selection for decision making tasks on micro-blog services," *PVLDB*, vol. 5, no. 11, pp. 1495–1506, 2012.

[21] A. Nandi, S. Paparizos, J. C. Shafer, and R. Agrawal, "With a little help from my friends," in *ICDE*, 2013, pp. 1288–1291.

[22] K. Feng, G. Cong, S. S. Bhowmick, and S. Ma, "In search of influential event organizers in online social networks," in *SIGMOD Conference*, 2014, pp. 63–74.

[23] L. Derczynski, B. Yang, and C. S. Jensen, "Towards context-aware search and analysis on social media data," in *EDBT*, 2013, pp. 137–142.

[24] C.-H. Lee, H.-C. Yang, T.-F. Chien, and W.-S. Wen, "A novel approach for event detection by mining spatio-temporal information on microblogs," in *ASONAM*, 2011, pp. 254–259.

[25] K.-S. Kim, R. Lee, and K. Zettsu, "*m*trend: discovery of topic movements on geo-microblogging messages," in *GIS*, 2011, pp. 529–532.

[26] Y. Zheng, "Location-based social networks: Users," in *Computing with Spatial Trajectories*, 2011, pp. 243–276.

Paper D

Finding Influential Local Users with Similar Interest from Geo-tagged Social Media Data

Jinling Jiang, Hua Lu, Pengfei Li, Gang Pan, Xike Xie

The paper has been published in the
18th IEEE International Conference on Mobile Data Management, 2017

Abstract

Geo-tagged social media data provides abundant resources for people in need of local information. In this paper, we study how to find the top-k influential local users from geo-tagged social media data who have interests similar to a query. Such local users can be of particular importance for a variety of activities from events organizing to online advertising. We formulate the problem as Top-*k* Influential Similar Local Query *(TkISL) and provide a complete set of techniques for solving it. To effectively manage the social media users, we design three hybrid user profiling techniques, an indexing tree, and an upper bound query-user similarity that enables efficient pruning in query processing. To process TkISL queries, we propose a baseline method and a more efficient improved method. The former directly uses the indexing tree and the upper bound for pruning, whereas the latter speeds up the query processing by enhancing the tree and pruning. Finally, we conduct extensive experimental studies to evaluate our proposals on real geo-tagged tweet corpora. The experimental results demonstrate the efficiency and effectiveness of our proposals.*

D.1 Introduction

As the leading microblogging service, Twitter produces huge volumes of social media data. Within the 140 characters limit, Twitter users broadcast their daily life moments in short snippets of text called tweets. Compared to traditional Web pages, tweets are significantly shorter and contains much less links. Therefore, search behaviors on Twitter also differ from Web search. Many people are interested in searching for local information on Twitter as many use Twitter on the mobile. Unfortunately, the huge amounts of information in Twitter are not yet well organized for local search. People most of the time receive the individual tweets passively rather than being able to find relevant local information by themselves efficiently.

Due to the widespread of positioning functionality in mobile terminals, social media data is increasingly acquiring a geo-aspect. As a result, it is possible to support local search in social media. For example, a company that makes clothes for young people wants to brand their new products in San Francisco. Social media certainly is a promising means for online branding and promotion. If the company finds some local social media users that are interested in their products and have high social influence (e.g., with many replies and forwards of their tweets), such users can help attract potential buyers in San Francisco. Here, the interest in the products can be verified by a set of product related topics extracted from the users' tweets. In other words, those users with such topics should be considered.

Such a real-life example involves three aspects in geo-tagged social media: location, social influence, and interest that can be expressed as topics. Consequently, a novel search service is called for to integrate the three aspects in local search of social media users. Motivated by such practical and unsolved needs, we study a novel and useful *Top-k Influential Similar Local Query* in this paper.

Given a topic list tl, a location l, a query range r and a threshold δ, a TkISL query $q(tl, l, r, \delta)$ returns a set of k social media users who have the highest social influences within the distance r from l and with topic based query-user similarity of at least δ. In our problem formulation, a user's social influence is measured in terms of the explicit attention (e.g., forward and reply) which the user's contents receive. In addition, topics are employed because they result in concise yet informative user profiles and more flexible query-user similarity than that based on rigid keyword comparison. A TkISL query is useful for many activities like events organizing and online advertising.

To efficiently process TkISL queries, we propose a complete set of techniques. We present a comprehensive data management solution for users in geo-tagged social media, including three hybrid user profiling techniques, an indexing tree for users, and upper bound similarity that supports efficient pruning in query processing. Accordingly, we design a two-phase query processing method and a more efficient single-phase method with enhanced indexing and pruning. We experimentally evaluate the proposed techniques using a large set of real tweet data. The experimental results demonstrate the effectiveness and efficiency of our proposals.

The remainder of this paper is organized as follows. Section D.2 formulates the problem. Section D.3 details the data management for Twitter users. Section D.4 presents two query processing methods. Section D.5 reports on the experimental studies. Section D.6 reviews related work. Section D.7 concludes the paper and discusses future work directions.

D.2 Problem Formulation

In this section, we give the basic definitions and formulate the research problem.

Definition 13 (Social Media Post). *A social media post is a 4-tuple $p = (uid, t, l, W)$. It means that user identified by uid publishes the post in location l at time t, and the textual content of the post is captured as a set of words $W = (w_1, w_2, \ldots, w_n)$.*

Definition 14 (Social Media User). *A social media user is captured as a list of tuples (uid, l, ψ), where uid is the user's identifier, and ψ is the user's profile that captures the user's social media posts published in the region represented by l.* [1]

Definition 15 (User Profile). *A user's profile $\psi = (t_1, t_2, \ldots, t_n)$ is a list of topics.*

A user is modeled as in Definition 14 because each user may have a number of representative locations and corresponding profiles. User profiling and representative location determination are detailed in Section D.3.1 and Section D.3.2, respectively.

As a user profile captures topics in the user's social media posts, it indicates the user's interest. A social media platform can have a large pool of topics but each user may only be associated with part of them. For example, one user may have a 3-topic profile $(business, sports, music)$, which means all these three topics have occurred in the user's posts. Each user's profile(s) is generated based on all her/his published posts.

In general, a microblog social network is modeled as follows.

Definition 16 (Microblog Social Network). *A microblog social network is a directed graph $G = (U, E_{reply}, E_{forward})$ where:*

[1] To ease the presentation, we assume each user is represented by a single 3-tuple (uid, l, ψ) when giving the definitions. In our implementation of user indexing and searching, a user with m ($m \geq 1$) representative regions are represented by m such 3-tuples and thus treated as m different users.

1. *U is a set of vertices each of which corresponds to a user.*
2. *E_{reply} is a set of directed edges. An edge $e\langle u_1, u_2\rangle \in E_{reply}$ means that user u_1* replies to *user u_2 in one or more posts.*
3. *$E_{forward}$ is a set of directed edges. An edge $e\langle u_1, u_2\rangle \in E_{forward}$ means that user u_1* forwards *user u_2's post(s).*

In particular, E_{reply} and $E_{forward}$ capture content-dependent aspects. In particular, each reply edge in E_{reply} must involve at least one post where one user replies to another, whereas each edge in $E_{forward}$ must involve at least one post which one user forwards. In our problem setting, we do not differentiate edges types and simply use $G_S = (U, E)$ to represent a social network, where $G_S.E = G.E_{reply} \cup G.E_{forward}$. In the context of G_S, we define the social influence of a user.

Definition 17 (Social Influence). *A user u's social influence $s(u)$ is the average number of forwards/replies the user gets per tweet.*

In other words, we consider how many forwards or replies a user's tweet gets on average when we quantify a user's social influence. Intuitively, a user tends to be able to influence many people in a social network if the user has many tweets that have been forwarded or replied to. For example, such a user can disseminate online advertisements of a product to a large set of potential buyers through forwarding and/or replying, which helps achieve substantial effect on sales of the product.

With the definitions given above, we define our research problem as follows.

Top-k Influential Similar Local Query: Given a social network $G_S = (U, E)$ and an integer k, a TkISL query $q(tl\langle t_1, t_2, ...t_n\rangle, l, r, \delta)$ finds a k-subset $\mathcal{E}_k \subseteq U$ such that:

1. $\forall u \in \mathcal{E}_k$, $\|q.l, u.l\| \leq q.r$ [2], and $sim(q.tl, u) \geq \delta$.
2. $\forall u \in \mathcal{E}_k$ and $\forall u' \in U \setminus \mathcal{E}_k$, $s(u') \leq s(u) \vee \|q.l, u'.l\| > q.r \vee sim(q.tl, u') < \delta$.

In such a query, topic list tl contains n topics, l and r represent a location and a range respectively, and δ is a similarity threshold. The first condition requires to return users in the range r of the location l, and their profiles are similar enough to the query interest. The second condition further specifies that among the users satisfying the first condition, those with the highest social influences should be returned. In other words, a user is excluded

[2] $\|l_1, l_2\|$ denotes the Euclidean distance between locations l_1 and l_2. Nevertheless, the proposed techniques can be extended to other distance metrics like Manhattan and network distances. When a user's representative region is involved, we calculate the distance using the region center.

from the result because of low social influence, too far distance from l, or low similarity with the query topics. We use Jaccard similarity to compare a TkISL query's topics and those in a user's profile.

Definition 18 (Query-User Similarity). *Given a TkISL query $q(tl, l, r, \delta)$ and a user u_j, the query-user similarity $sim(q, u_j)$ is defined as $sim(q, u_j) = \frac{|q.tl \cap u_j.\psi|}{|q.tl \cup u_j.\psi|}$.*

We give an example to explain the constraints of influence and similarity in the problem definition. For simplicity we disregard locations here. Table D.1 shows a sample of user profiles in our experiments. Suppose that k

User	(Lat., Lon.)	Profile	Influence
A	(24.2,-79.4)	business, sports, music	0.33
B	(46.6,118.4)	food, fiction, film	0.67
C	(34.6,137.2)	poetry, art, dance, internet	0.33
D	(37.1,114.5)	culture, sports, entertainment	0.26
E	(52.4, -2.5)	entertainment, music, dance, internet	0.43
F	(42.5,-72.7)	cartoon, music, internet	0.87
G	(33.6, -7.8)	fashion, opera, music	0.67
H	(-42.1,172.4)	culture, design	0.25

Table D.1: Sample Data of User Profiles

equals to 1, the query q's topic list is $\langle$*music*, *entertainment*, *cartoon*, *sports*$\rangle$, and the similarity threshold is 0.3. In this example, $sim(q, A) = sim(q, D) = sim(q, F) = 0.4$, and $sim(q, E) = \frac{1}{3}$. Only these four users qualify the similarity threshold 0.3. Among them, user F has the highest social influence and therefore F is returned in the query result.

D.3 Data Management for Social Media Users

This section covers the data management techniques for social media users. Section D.3.1 addresses topic based user profiling. Section D.3.2 discusses how to find representative regions for such users. Section D.3.3 proposes a schema for indexing such users. Section D.3.4 derives an upper bound for the similarity between a query and a set of users.

D.3.1 User Profiling

Existing Approaches and Their Limitations

In the scope of this paper, we focus on content-based user profiling techniques. Existing techniques fall into three categories [1]. *Hashtag-based profiling* utilizes the hashtags (keywords starting with "#" indicating specific topics in tweets) in one's historical tweets to represent the corresponding user profile. *Topic-based profiling* adopts the OpenCalais taxonomy [3] to build a user profile as a subset of 18 IPTC News codes. Each code represents a top-level subject/topic such as Education, Entertainment_Culture, and Politics, etc. *Entity-based profiling* employs OpenCalais2 [4] to detect and identify 39 different types of entities such as persons, locations and organizations. Each entity is encoded as a uniform resource identifier (URI). Each user profile under this technique consists of several entities/URIs.

The dimensionality in hashtag-based and entity-based profiles can be very high and are usually restricted to 10,0000 [1]. In contrast, topic-based profiles only involve 18 topics, and thus less computational costs are needed in building and using such profiles. Also, microblogs abound with noises and semantic variations. A straightforward use of topic modeling [2] may fail to build profiles for those users whose historical tweets do not fall into a certain topic category, whereas hashtag-based and entity-based profiles may consist of information overlaps. For example, the hashtags #Egypt protest# in one user profile and #Jan 25# in another in fact refer to the same issue, a political protest in Egypt starting on January 25, 2011. As a result, the users with different hashtag or entity-based profiles may discuss the same contents and thus share common interests but their different profiles do not indicate that. If the topic-based profiling technique is used instead, the two users would ideally have the same profile with two topics: *Social Issues* and *War_Conflict* under the scheme of the 18 IPTC News topics.

Our Approach

In order to preserve computation efficiency while incorporating more detailed information, we introduce a new approach to build user profiles in the space of 1,152 IPTC News topics [5] instead of in the space of 18 top-level subjects only. The 4-phase process of user profile generation is illustrated in Figure D.1.

The generation process's first phase detects and disambiguates the entities from tweet texts of a given user. TextRazor [6] REST API is leveraged for that purpose in this phase. By utilizing a huge knowledge base of entity de-

[3]https://www.drupal.org/project/opencalais

[4]http://www.opencalais.com/

[5]http://cv.iptc.org/newscodes/mediatopic

[6]https://www.textrazor.com/

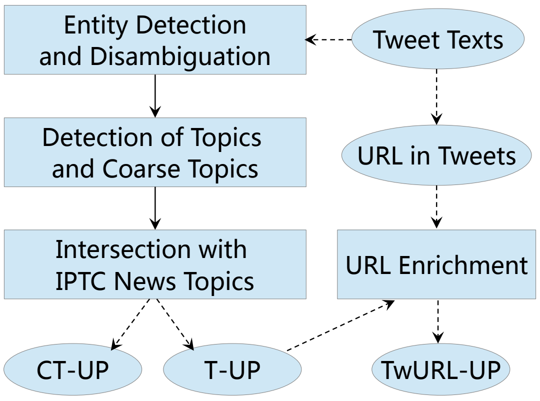

Entity Detection and Disambiguation
Tweet Texts
Detection of Topics and Coarse Topics
URL in Tweets
Intersection with IPTC News Topics
URL Enrichment
CT-UP
T-UP
TwURL-UP

Fig. D.1: User Profile Generation

tails extracted from various web sources, including Wikipedia, Freebase [7] and DBPedia [8], TextRazor builds a dictionary of millions of different entities like people, location and organization so as to allow rapid lookup in text analysis. TextRazor achieves disambiguation by exploring the context surrounding each entity with a comprehensive real-world knowledge base.

After the entities are obtained, the second phase employs a multilingual topic detection to tag the contents with topics at different levels. For example, a user that mentions "Chelsea", "Stamford Bridge", "Arsenal" can be tagged with "Soccer" and "Sports". TextRazor is able to recognize thousands of different tags at different levels of abstraction. To be specific, TextRazor uses two levels: "topics" and "coarseTopics". In our example, "Soccer" is at "topics" level while "Sports" is at "coarseTopics" level. As a result, this phase generates topics at different levels for the given user.

To avoid incorporating too many detailed topics, the third phase uses the aforementioned 1152 IPTC News topics to limit the topic dimensions in the user profiles. If a detected topic (at either "topics" or "coarseTopics" level) from the previous phase is not one of the 1152 IPTC News topics, it is abandoned in our generation process. This phase generates two kinds of profiles: *Coarse Topic based Users Profiles* (*CT-UP* for short) and *Topic based User Profiles* (*T-UP* for short).

The fourth phase in the user profile generation is an additional step to battle the likelihood of tweet texts containing noises and lacking explicit information. In order to enrich user profiles in such cases, the generation process identifies the URLs embedded in the tweets, detects topics indicated by the URLs, and combines such topics with those generated in the third phase

[7]https://www.freebase.com/
[8]http://wiki.dbpedia.org/

to formulate augmented user profiles.[9] We call the resultant user profile *Topic based User Profile with URL enrichment* or *TwURL-UP* for short.

In summary, our user profile generation outputs three kinds of profiles. It is worth mentioning that CT-UP and T-UP techniques check the relevance score of each topic encountered. A topic is abandoned if the score is lower than a given threshold. As T-UP's output is TwURL-UP's input, the score threshold affects all the three user profiling techniques. The effects of profiling techniques as well as the threshold values on user profiling and query processing are to be experimentally evaluated in Section D.5.

D.3.2 User's Representative Regions

A user may publish social media posts at many different places. Considering a user's all tweets, we may obtain the user's trajectory as a sequence of timestamped points $(s_1, s_2, \ldots, s_n)$, where $s_i = (x_i, y_i, t_i)$ $(0 \leq i < n)$ indicates that the user publishes a post at location (x_i, y_i) at timestamp t_i. If we keep all the raw locations in such a trajectory, we would have to create the same number (n) of user profiles. This is not space-efficient and it will incur high local search cost. Therefore, we cluster the n raw GPS locations into m $(m \ll n)$ *representative regions* for a user u, and use m 3-tuples (u, r_i, ψ_i) $(0 \leq i < m)$ to represent user u. In particular, user profile ψ_i is generated from u's tweets posted at those locations that are clustered to region r_i.

For a cluster of l locations $S_i = (s_{i1}, s_{i2}, \ldots, s_{il})$, we use the centroid of the cluster to capture the representative region r_i, i.e., $r_i.x = \sum_{j=1}^{l} s_{ij}.x / |S_i|$, and $r_i.y = \sum_{j=1}^{l} s_{ij}.y / |S_i|$. Previous spatial clustering methods fall into four categories [3]: partitioning methods (e.g., k-means), hierarchical methods (e.g., CURE [4]), density-based methods (e.g., DBSCAN [5] and OPTICS [6]), and grid-based method [7].

We need to construct representative regions through clustering for each user as different users are associated with different trajectories. If we use a clustering method from the first three categories, we need to set different parameters for different users because the parameters suitable for one user may be unsuitable for others. To avoid the troublesome parameter setting and tuning, we turn to grid-based clustering methods in the fourth category that are independent on the data characteristics. Furthermore, in order to control the clustering size at the same level for different user trajectories, we adapt the grid-based clustering algorithm [7] proposed for clustering GPS history data. In practice, the representative regions of a user should not be too large or too small in size. We use parameter d to control the size of grid cells. We omit the details of the clustering as it is not the focus of our research.

[9]TextRazor API is able to analyze URLs in addition to pure text.

D.3.3 Indexing Users

The IR^2-tree [8] is essentially an R-tree associated with signature file in each node. In particular, each node of an IR^2-tree contains both spatial and textual information. The spatial information is represented as a minimum bounding rectangle (MBR) and the textual information is stored as a bitmap signature file. All such bitmaps are of the same length that is equal to the total number of keywords (or topics in our paper) associated to objects that are indexed. In our setting, we adapt the IR^2-tree to index users. Recall that Definition 14 defines a user identified by *uid* as a list of 3-tuples in the form of (uid, l, ψ). Each such 3-tuple is regarded as an object to be indexed by the IR^2-tree.

Formally, a leaf node entry in IR^2-tree is in the form of $(ObjPtr,\ M, S)$ where $ObjPtr$ points to a 3-tuple (uid, l, ψ), M is the MBR formed by location l, and S is the bitmap signature of user profile ψ. In particular, a bit 1 in S means that the corresponding topic appears in user profile ψ. A non-leaf node entry is in the form of $(NodePtr, M, S)$ where $NodePtr$ points to a child node, M is the MBR of that child node, and S is the child node's signature file in bitmap format. In particular, S is the superimposition (OR-operation) of all signatures in the corresponding child node.

Figure D.2 shows the IR^2-tree for the data in Table D.1. The topic space is (*business, sports, music, food, fiction, film, poetry, art, dance, internet, culture, entertainment, cartoon, fashion, opera, design*). According to this order of 16 keywords, a bitmap representation of user A's profile (*business*, *sports*, *music*) is 11100000 00000000.

In tree construction, the adapted IR^2-tree basically employs the same functions for insertion and deletion as the original IR^2-tree. The main difference lies in how the adapted tree resolves ties in functions ChooseSubtree(.) and Split(.) used in object insertion. When choosing a sub-node to insert a new object/entry in these two functions, conventional R-tree considerations (e.g., area enlargement) are first taken. If ties exist, our construction procedure will choose the sub-node that yields a smaller similarity score with the new object/entry in terms of the bitmap signatures. The similarity of two bitmap signatures bt_i and bt_j is

$$sim(bt_i, bt_j) = \frac{Count(bt_i \wedge bt_j)}{Count(bt_i \vee bt_j)}. \tag{D.1}$$

Function Count(.) counts 1's in a bitmap. We omit the detailed tree construction algorithms as they are not the focus of our research.

D.3.4 Upper Bound Similarity

When processing the search for a given query $q(tl, l, r, \delta)$, it is desirable that we can prune tree branches using the similarity threshold δ. To that end, we

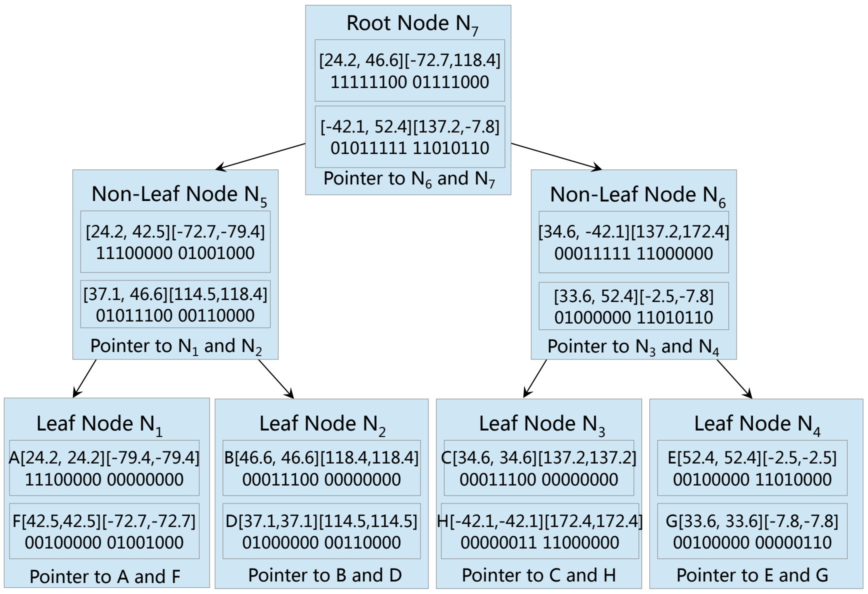


Fig. D.2: IR^2-tree Example

need to consider the upper bound similarity between a query q and a user u_j. Specifically, such an upper bound similarity is defined for a query q and a set S_U of users, where S_U contains u_j and represents all users covered in a tree node. To distinguish it from the query-user similarity (Definition 18), we call it *query-users similarity* instead.

Definition 19 (Query-Users Similarity). *Given a TkISL query $q(tl\langle t_1, t_2, ...t_n\rangle, l, r, \delta)$ and a user set S_U, the query-users similarity is $sim(q, S_U) = \frac{|q.tl \cap \bigcup_{u_j \in S_U} u_j.\psi|}{|q.tl|}$.*

The query-users similarity can serve as the upper bound of Jaccard similarity between a query q and any user u_j in set S_U. We have Lemma 1 for this purpose.

Lemma 1. *Given a TkISL query q and a set S_U of users, it holds that $\forall u_j \in S_U$, $sim(q, u_j) \leq sim(q, S_U)$.*

Proof. $\forall u_j \in S_U$, we have $u_j.\psi \subseteq \bigcup_{u_j \in S_U} u_j.\psi$, and therefore $q.tl \cap u_j.\psi \subseteq q.tl \cap \bigcup_{u_j \in S_U} u_j.\psi$, which means $|q.tl \cap u_j.\psi| \leq |q.tl \cap \bigcup_{u_j \in S_U} u_j.\psi|$.

On the other hand,

$$\begin{aligned}
sim(q,u_j) &= \frac{|q.tl \cap u_j.\psi|}{|q.tl \cup u_j.\psi|} \ (\because \text{Definition 18}) \\
&\leq \frac{|q.tl \cap u_j.\psi|}{|q.tl|} \ (\because |q.tl \cup u_j.\psi| \geq |q.tl|) \\
&\leq \frac{|q.tl \cap \bigcup_{u_j \in S_U} u_j.\psi|}{|q.tl|} \ (\because \text{Definition 19}) \\
&= sim(q, S_U). \qquad \blacksquare
\end{aligned}$$

When applying the query-users similarity to the adapted IR^2-tree, we need to consider $sim(q, node)$ between a query q and a tree node $node$. For all users covered in node $node$, their user profiles are aggregated through bit-wise OR operation into bitmap signature $node.bt$. In other words, if a bit in $node.bt$ is 1, node $node$ must cover at least one user with the corresponding topic in profile. If a bit in $node.bt$ is 0, node $node$ must cover no user with the corresponding topic in profile. Using $q.bt$ to denote the bitmap corresponding to the query's topic list, $sim(q, node) = sim(q.bt, node.bt)$ and it is calculated according to Equation D.1 in Section D.3.3.

The upper bound similarity enables the following pruning rule for query processing.

Pruning Rule 1. *Given a query $q(tl, l, r, \delta)$ and a tree node node, if $sim(q, node) < \delta$, the node node can be pruned as it contains no users with query-user similarity higher than δ.*

It is straightforward to prove the correctness of this pruning rule according to Lemma 1 and the presentation above. We give an example shown in Figure D.2. Suppose a query q has a topic list $tl = (business, sports, music, food)$ and a threshold $\delta = 0.5$. When considering the root node's two child nodes N_5 and N_6, the query-users similarity between q and N_5 is $sim(q, N_5) = \frac{4}{4} = 1 > \delta$ while that between q and N_6 is $sim(q, N_6) = \frac{1}{4} = 0.25 < \delta$. Therefore, node N_6 will be pruned in the search.

D.4 Query Processing

In this section, we present two query processing methods for TkISL search. The baseline method (in Section D.4.1) uses the adapted IR^2-tree and processes queries in two phases. The improved method (in Section D.4.2) further modifies the IR^2-tree and processes queries in a single phase.

D.4.1 Baseline Method: Two-Phase Query Processing

Figure D.3 shows that the general procedure of the two-phase query processing. The first phase is to generate a candidate user set based on the query range and similarity threshold. The second phase is to rank the candidate users by their social influence scores.

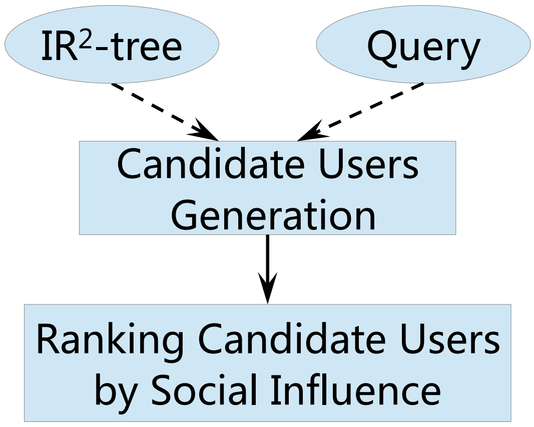

IR²-tree
Query
Candidate Users Generation
Ranking Candidate Users by Social Influence

Fig. D.3: Two-Phase Method

Algorithm 9 shows the sketch of the two-phase query processing. The input of the algorithm consists of an IR^2-tree tr, a query q, and an integer k. The first phase (lines 1–17) basically executes a range search via the tree in the depth-first fashion using a FIFO queue. During the search, unqualified nodes are pruned according to the distance constraint (lines 5, 9 and 14) and the similarity threshold (lines 9 and 14). Qualified users are added to the candidate list (line 17).

The second phase (lines 18–24) calculates the social influence for each candidate user, and returns the top-k with the highest influences. The social influence of a user is obtained through two SQL statements (lines 20–21) and a simple arithmetic operation (line 22) according to Definition 17.

The two SQL statements query against a tweet table that stores all tweets in our setting. The table is created off-line before any query comes in, according to the schema of $(tweetId,\ userId,\ repliedToUserId, repliedToStatusId)$. Specifically, *tweetId* identifies a tweet, *userId* refers to the user who published the tweet, *repliedToUserId* refer to the user whose tweet is replied to or forwarded by this tweet, and *repliedToStatusId* corresponds to the tweet identifier that is replied to or forwarded by this tweet.

Algorithm 9: Two-Phase Query Processing

```
   Input: $IR^2$-tree $tr$, Query $q$, Integer $k$
   Output: A list of $k$ users
   /* the first phase                                              */
1  Initialize a List $L$ and a FIFO Queue $Q$;
2  $Q$.Enqueue($tr$.RootNode);
3  while $Q.size()$ != 0 do
4      Entry $e = Q$.Dequeue();
5      if minDist($q.l, e$) $\leq q.r$ then
6          if $e$ is a non-leaf entry then
7              for each (NodePtr, MBR, S) in $e$ do
8                  Node $n$ = LoadNode($NodePtr$);
9                  if minDist($q.l, n$) $\leq q.r$ and $sim(q, n) \geq \delta$ then
10                     $Q$.Enqueue($n$);
11         else if $e$ is a leaf entry then
12             for each (ObjPtr, MBR, S) in $e$ do
13                 Object $Obj$ = LoadObject($ObjPtr$);
14                 if $\|q.l, Obj\| \leq q.r$ and $sim(q, Obj) \geq \delta$ then
15                     $Q$.Enqueue($Obj$);
16         else
17             $L$.add($e.userId$);
   /* the second phase                                             */
18 Initialize a List $US$ of pairs ($user\ u$, $social\_influcence\ si$) from $L$;
19 for each userId in $L$ do
20     Integer repliesNumber = SELECT COUNT(*) WHERE
         repliedToUserId=$userId$;
21     Integer tweetNumber = SELECT COUNT(DISTINCT tweetId)
         WHERE userId=$userId$ ;
22     $US$.add($userId$, repliesNumber/tweetNumber);
23 Sort $US$ on the descending order of right element;
24 Return the first $k$ elements from the sorted $US$;
```

D.4.2 Improved Method: Single-Phase Query Processing

To avoid the costly SQL queries in Algorithm 9, we propose a single-phase method.

IR^2-Tree Augmentation and Node Pruning

The critical issue for the single-phase query processing is to be able to know user social influences when visiting the IR^2-tree nodes. This is enabled by augmenting the tree as follows. Each node entry e is associated with an extra field *influence*. If e indexes an object, *e.influence* is the corresponding user's social influence (Definition 17). Otherwise, $e.influence = \max\{e_i.influence \mid e_i$ is $e's$ sub-node entry$\}$. Thus, a node entry e's *influence* is the upper bound influence for all users covered in e. This enables the following pruning rule.

Pruning Rule 2. *Suppose that $\mathcal{E}_k$ contains k users with the highest social influences and f_k is the smallest among them. Given a node entry e in the IR^2-tree, if $f_k >$ e.influence, then the node indexed by e can be safely pruned as it cannot contain any users with higher social influence than f_k.*

We refer to Table D.1 and Figure D.2 for example. The upper bound influence for node N_3 is max{0.33, 0.25} = 0.33, and that for node N_4 is max{0.43, 0.67} = 0.67. Therefore, the upper bound influence for node N_6 is max{0.33, 0.67} = 0.67. Suppose that user F has already been processed for a top-1 search. As F's influence is 0.87 that is higher than N_6's upper bound influence, node N_6 can be pruned in subsequent query processing as it cannot contain a user with higher influence.

Influence Lookup Table for Index Construction

When building the augmented IR^2-tree, we need to obtain users' social influence. To speed it up, we build a key-value lookup table that takes *UserId* as the key and the average number of retweets/replies (social influence) as the value for all users.

Algorithms 10 and 11 employ the Map-Reduce programming paradigm to scalably and efficiently build the lookup table for a large tweet corpus. The input to the map function is a social media post p defined in Definition 13 with additional meta-data *repliedToUserId* and *repliedToStatusId*. The former identifies the user whose tweet is replied to or forwarded by this tweet p, and the latter identifies the tweet that is replied to or forwarded by this tweet p. The output key of map function is *repliedToUserId* and the value is the *repliedToStatusId*.

In the reduce function, an associative array H is used to keep track of the occurrences for each *repliedToStatusId* (lines 4 and 5). From lines 7 to 8, the total number of posts (kept in variable c) having reply/forward information

Algorithm 10: Map function

```
Input: social media post $p$
1 Pair $\langle Long, String \rangle$ $mapOutput$ ;
2 Emit ($mapOutput\langle repliedToUserId \rangle$, $repliedToStatusId$);
```

Algorithm 11: Reduce function

```
Input: $\langle repliedToUserId \rangle$, $repliedToStatusIds[rSId_1, \ldots]$
1 AssociativeArray $\langle Long, Integer \rangle$ $H$;
2 Pair $\langle Integer, Integer \rangle$ $reduceOutputValue$ ;
3 int $c = 0$; int $sum = 0$;
4 for each $rSId \in repliedToStatusIds[rSId_1, \ldots]$ do
5     $H\langle rSId \rangle = H\langle rSId \rangle + 1$;
6 for each $key \in AssociativeArray\ H$ do
7     $c = c + 1$; $sum = sum + H\langle key \rangle$;
8 Emit($\langle repliedToUserId \rangle$, $reduceOutputValue(sum/c)$)
```

and the total number of replies/retweets (kept in variable *sum*) are computed. The output key of the reduce function is *repliedToUserId* (same as map function) and output value sum/c means the average number of retweets/replies to *repliedToUserId*.

Query Processing Algorithm

Algorithm 12 shows the single-phase query processing. The input parameters are the same as Algorithm 9, except that the tree *tr* is an augmented version as described above.

The algorithm employs a priority queue to keep all tree node entries encountered, giving priority to those entries with higher upper bound social influences. Before enqueuing any entry (lines 10 and 15), distance based pruning (lines 5, 9, and 14) and similarity based pruning (lines 9 and 14) are conducted. If an object is encountered from the priority queue, the corresponding user's identifier is added to the result list (lines 16 and 19). When there are k users in the result list, provided Pruning Rule 2 and the property of the priority queue Q, the while-loop breaks (lines 17–18) and the algorithm returns the result list with k users (line 20).

D.5 Experimental Studies

Algorithm 12: Single-Phase Query Processing

```
Input: Augmented $IR^2$-tree $tr$, Query $q$, Integer $k$
Output: A list $L$ of $k$ users
1  Initialize a PriorityQueue $Q$;
2  $Q$.Enqueue($tr$.RootNode, 0.0);    Int $count = 0$;
3  while $Q.size() \mathrel{!}= 0$ do
4      Entry $e = Q$.Dequeue();
5      if minDist$(q.l, e) \leq q.r$ then
6          if e is a non-leaf entry then
7              for each (NodePtr, MBR, S) in e do
8                  Node $n$ = LoadNode($NodePtr$);
9                  if minDist$(q.l, n) \leq q.r$ and $sim(q, n) \geq \delta$ then
10                     $Q$.Enqueue($n, n.influence$);
11         else if e is a leaf entry then
12             for each (ObjPtr, MBR, S) in e do
13                 Object $Obj$ = LoadObject($ObjPtr$);
14                 if $\|q.l, Obj\| \leq q.r$ and $sim(q, Obj) \geq \delta$ then
15                     $Q$.Enqueue($Obj, Obj.influence$);
16         else if e is an object then
17             if $++count = k$ then
18                 break ;
19             $L$.add($e.userId$);
20 Return $L$;
```

D.5.1 Settings and Data

The experiments are implemented in Java, and run with a 4GB JVM heap space on a PC enabled by Ubuntu 12.04 with Quad Core Intel(TM) i7 and 8GB main memory.

We sample a real Twitter data set with geographical coordinates collected by using Twitter REST API. In total, there are nearly 1 million users and 10.5 million geo-tagged tweets with the timestamp range from September 2012 to February 2013.

Figure D.4 shows the histogram statistics of the number of users and the number of tweets per user. The x-axis represents the number of tweets while the y-axis indicates the number of users who have published so many tweets. The statistics follow a power law distribution: many users publish very few tweets, while very few users publish many tweets. Concretely, nearly 174,745 users have 10 tweets while only 16,030 users have 40 tweets. In the plotting, we cut the long tail at 60 tweets where only 7,295 or fewer users publish at least so many tweets during that period of time.

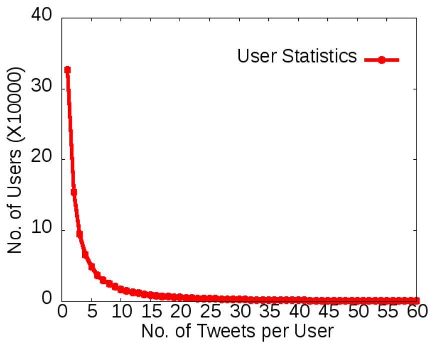


Fig. D.4: User Statistics

D.5.2 Effect of Profiling Techniques

Concerning profiling techniques CT-UP (coarse topic based), T-UP (topic based) and TwURL-UP (topic based with URL enrichment) detailed in Section D.3.1, we validate their effectiveness by measuring the percent of users for which a technique successfully builds profiles. We call such users *profiled users*. As users with less than 20 tweets tells very limited information but consume considerable processing time with TextRazor API, we exclude such users and obtain 77,000 remaining users. Recall that all the three user profiling techniques use a relevance score threshold to decide whether a topic is relevant enough for being included in a profile.

Figure D.5 shows the results of the effectiveness of user profiling techniques with respect to varying threshold values. In general, lower thresholds result in more topics for users and thus more users are profiled. For example, CT-UP only profiles about 3.5% of the users when the threshold is set to 0.9, each having more than 20 posts. In contrast, it profiles nearly 25% users when the threshold is 0.3. With the same threshold value, TwURL-UP significantly improves the profiling effectiveness by profiling much more users. For instance, when the threshold is set to 0.3, TwURL-UP profiles nearly 50% of the users, markedly more than T-UP (24.88%) and CT-UP (28.34%).

D.5.3 Effect of Grid Cell Size

We also study the effect of grid cell size $(d * d)$ used for determining the representative user regions. Intuitively, a small d tends to result in more more representative user regions and thus more user profiles; a larger d tends to

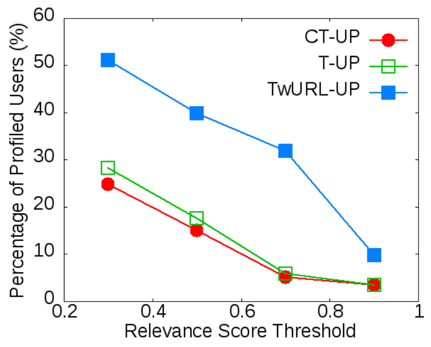


Fig. D.5: User Profiling Effectiveness

have the opposite effect. We vary *d* from 10 km to at most 30 km and the results are shown in Figure D.6. We count the number of users who have 1, 2 and at least 3 representative regions. Nearly 72,000 users have only one representative region when *d* is 30 km; and only 4,820 users have at least three when *d* is 10 km. When *d* is 10 km, still more than 62,000 users have only one representative region. Only 2,450 users have three or more representative regions when *d* equals 30 km. In our subsequent experiments, 20 km is used as the default *d* value because its effect is balanced according to the results shown in Figure D.6.

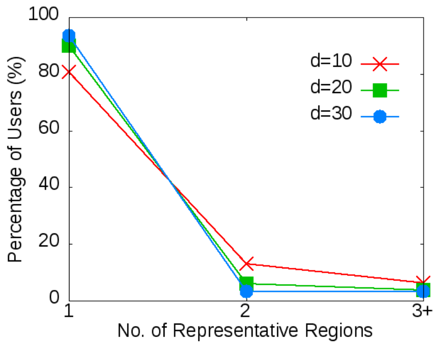


Fig. D.6: Statistics of User Representative Regions

D.5.4 Results on Query Processing Efficiency

This part of experiments evaluate the query processing efficiency under various query settings.

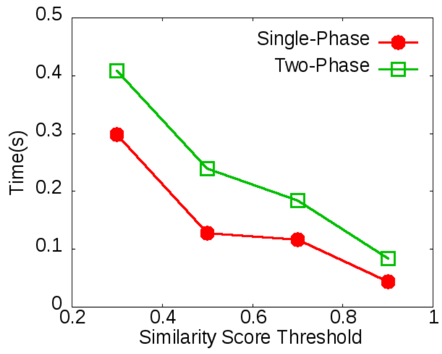
Query Settings

In our experiments, the topic list $tl\langle t_1, t_2, ...t_n\rangle$ in a TkISL query $q(tl, l, r, \delta)$ is a random sample from the 1,152 IPTC News topics. Each topic list is associated with a random location l sampled from the spatial domain of our data set. In total, we have 90 queries, each one-third having 10, 20 or 30 topics. The value of k in each query, i.e., the number of returned users, is a random value between 5 to 10 inclusively. The query range r is varied from 10km to 30km.

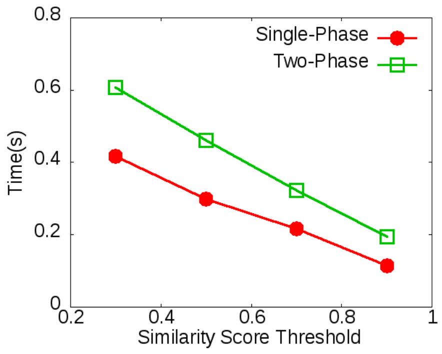
Two-Phase vs. Single-Phase Query Processing

In this part, we compare the two query processing methods under different settings. We randomly pick up 30 queries from the 90-query set. We vary the similarity threshold from 0.3 to 0.9. We focus on user profiles obtained by TwURL-UP as it generates the largest user profiles that certainly incur more query processing costs. Figure D.7 shows the average query processing time of the two methods.

Clearly, the single-phase query processing is more efficient that the two-phase alternative. The former avoids the I/Os required by the SQL statements used in the latter. It is also seen that larger query ranges incur longer query processing time for both methods. This is reasonable since a larger query range makes both methods tend to visit more tree nodes in query processing. Furthermore, a higher similarity threshold helps prune tree nodes more aggressively in both methods, which reduces the overall query processing time. The query processing on the range of 30km incurs higher time cost than 10km and 20km, but the similar decreasing trends are observed for increasing similarity score thresholds. Therefore, those results are omitted in the paper.



(a) 10 km query range



(b) 20 km query range

Fig. D.7: Query Processing Efficiency

Efficiency of Single-Phase Query Processing

In this part, we take a closer look at the efficiency of single-phase query processing under various settings.

First, we investigate the effect of different user profiling techniques and that of different similarity thresholds. We randomly pick up 30 queries from the 90-query set, and report the average query processing time results in Figure D.8. At a fixed query distance, a richer profiling technique like TwURL-

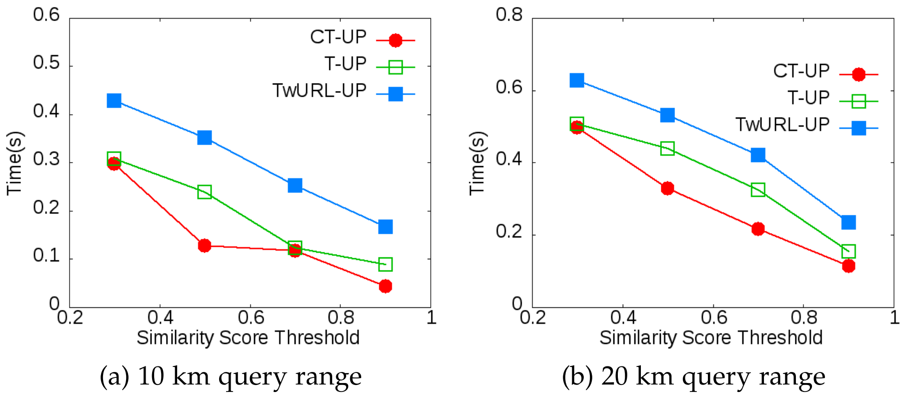
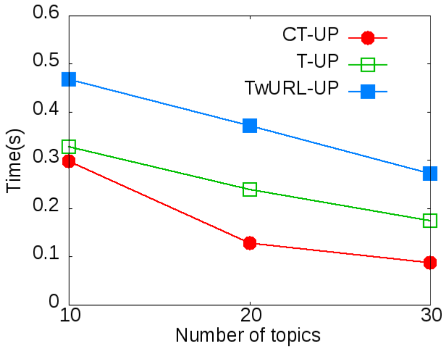


Fig. D.8: Single-Phase vs. Similarity Thresholds

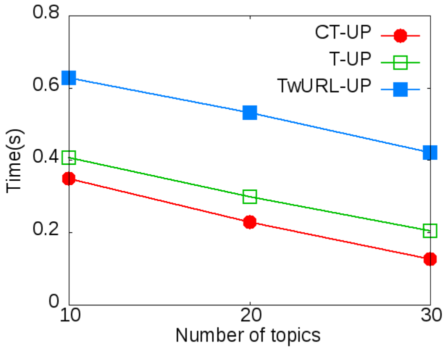
UP results in a larger number of non-zero bits in the bitmap associated with the index tree nodes. Therefore, the richer user profiles are, the less tree node pruning ratio is. The results show that, in each single setting, TwURL-UP results in the longest query processing time since it generates the user profiles with the largest numbers of topics. In contrast, CT-UP technique results in shortest query processing time—it prunes topics to a large extent in profiling users and thus generates poor profile information that needs much less processing time. The trends are the same for the results on the range of 30km and they are omitted due to page limit.

Second, we investigate the effect of varying query topic numbers for all user profiling techniques and query ranges. We use all the 30 queries for each topic number in {10, 20, 30}, and set the similarity threshold to 0.7. The results on the average query processing time are reported in Figure D.9. In general, more topics in a query make the similarity between the query and a tree node lower. Therefore, more tree nodes are likely to be pruned in the query processing. That is why the query processing time decreases as more topics are used in queries. We omit the results on the range of 30km since they exhibit higher costs but the same trends.

Finally, we focus on the user profiles generated by TwURL-UP, vary the similarity threshold from 0.3 to 0.9, and vary the query topics from 10 to 30.



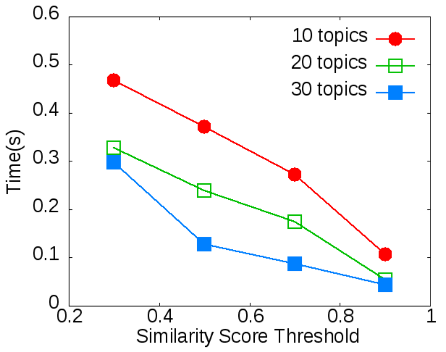
(a) 10 km query range



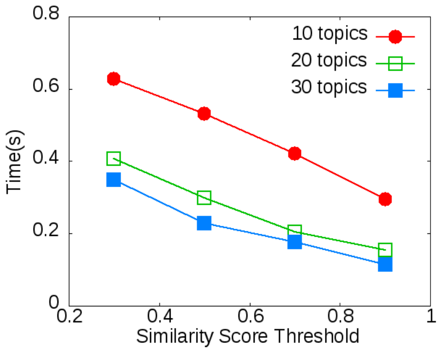
(b) 20 km query range

Fig. D.9: Single-Phase vs. Numbers of Topics

The query processing time results are reported in Figure D.10. Again, more topics in the query leads to shorter query processing time due to the reason pointed out above. Higher similarity thresholds also help prune more tree nodes and thus result in shorter query processing time. Again, the trends are the same for the results on the range of 30km and they are omitted due to page limit.



(a) 10 km query range



(b) 20 km query range

Fig. D.10: Single-Phase using TwURL-UP

D.5.5 User Study on Search Effectiveness

In addition to the experiments on search efficiency, we also conduct a user study to evaluate the effectiveness of our top-*k* local user search. We use all the English tweets with coordinates in London from February 19 to 20, 2013. In total, there are 60,897 tweets and 12,990 users. We select 5 topics (see Table D.2) at random, and set the query range to 20 km. For each query, we use a single topic and output the top-10 users in the result. Each line in the result is the user name that uniquely identifies the corresponding Twitter user.

We invite eight participants familiar with Twitter and the topics to give relevance feedbacks for the top-10 query results. A participant gives 1 to a returned user if s/he thinks the user is influential and relevant to the query, or 0 otherwise. Each participant evaluates the results of all the five top-10 queries. For a particular query, a returned user is regarded as a correct answer if it obtains a score not lower than 5. We define precision as the fraction of the returned users that are regarded as correct.

The user study results are shown in Table D.2. For a local topic like football or Chelsea Football Club, our top-*k* query is very effective in that the precision is 90% or 70%, respectively. Football is very popular in UK and people talk about it and the city-based clubs very actively in their tweets. In contrast, for a non-local topic (e.g., automobiles or entertainment) that is not associated to specific regions, the top-*k* query effectiveness is clearly lower. Such performance differences suggest that the TkISL query proposed in this paper is able to find local influencers for locality-sensitive topics effectively.

Topic	**Precision**
football	90%
Chelsea Football Club	70%
music	60%
automobiles	40%
entertainment	30%

Table D.2: User Study Result

D.6 Related Work

Social Media Data Management User profiling is an important issue for social media data management. Michelson and Macskassy [9] propose an approach for discovering users' topics of interest by detecting and processing entities in tweets. Abel et al. [1] design hashtag-based, entity-based and topic-based user profiles. We use hybrid user profiling techniques to facilitate local search of influential users with relevant interest, whereas previous research [10–13] ranks users without considering locations.

Geo-social models and queries have been studied for social media data. Cho et al. [14] study the relationship between friendship and mobility in location-based social networks. Doytsher et al. [15, 16] integrate a social network and a spatial network through the locations where activities take place. Armenatzoglou et al. [17] provide a framework containing different primitives to deal with three kinds of geo-social queries. Huang and Liu [18] suggest a GeoSN query definition that returns to a user the friends that are nearby and share common interests. Liu et al. [19] and Yang et al. [20] define two query types that essentially return a minimum circle of (partly) connected friends for a user in a geo-social network. In contrast, TkISL in this paper considers both social influence and topic similarity in search for local users in social media.

The top-*k* local user search (TkLUS) [21] finds the top-*k* local users who have posted tweets with keyword(s) from a set *W*. Our TkISL differs from it in substantial ways. First, the TkISL considers a user's influence in search while TkLUS does not. Second, users in the TkISL are represented as topic based profiles, whereas users in the TkLUS are not profiled. Third, the TkISL measures the topic based similarity between a user and the query topics, whereas the TkLUS does not consider such similarities. As a result, the techniques for the TkLUS are not applicable to the TkISL in this paper.

Geo-Textual Indexes Grid based and tree-based indexes have been proposed for geo-textual data. The former includes Spatial-first Index (ST) and text-first Index (TS) [22]. The latter includes the integrated inverted index (I^3) [23], the KR*-Tree (Keyword R*-tree) [24], the IR^2-tree [8], the IR-tree and its variants [25–27], and the Spatial Inverted Index (S2I) [28]. Chen et al. [29] conduct a comparative performance study on typical geo-textual hybrid indexes.

In this paper, grid-based clustering is used to get representative user regions. To index the geo-tagged tweets skewed in space, we adapt and modify the IR^2-tree. The user profiles in our setting are generated in a unified space of a fixed number of possible topics, for which the compact format of bitmap signatures in IR^2-tree is more suitable.

D.7 Conclusion And Future Work

This work tackles *Top-k Influential Similar Local Query* (TkISL). In the context of a large set of geo-tagged social media data, the TkISL finds the most influential local users with interest profiles most similar to a given query. We propose techniques for profiling and indexing the users in such data. We also design an upper bound query-user similarity that enables efficient pruning. Based on these user management techniques, we design a baseline method and an improved method to process TkISL queries. The proposals are evaluated using real geo-tagged tweets. The experimental results demonstrate the effectiveness and efficiency of the proposals.

Several directions exists for future research. First, it is interesting to use weights to prioritize different topics in user profiles. Second, it is relevant to study how to update user profiles efficiently when they have new tweets.

Third, it makes sense to take into account the time parameter in user profiling, e.g., giving priority to a user's recent posts in the profile. Fourth, it is interesting to generalize or adapt the proposed techniques to other social media data.

Acknowledgements

Pengfei Li and Gang Pan's work was partly supported by National Key Research and Development Plan of China (2016YFB1001200).

References

[1] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Analyzing user modeling on twitter for personalized news recommendations," in *UMAP*, 2011, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=2021855.2021857

[2] L. Hong and B. D. Davison, "Empirical study of topic modeling in twitter," in *SOMA*, 2010, pp. 80–88. [Online]. Available: http://doi.acm.org/10.1145/1964858.1964870

[3] J. Han, M. Kamber, and A. K. H. Tung, "Spatial clustering methods in data mining: A survey," in *Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS*, H. J. Miller and J. Han, Eds. Taylor and Francis, 2001. [Online]. Available: http://www-faculty.cs.uiuc.edu/~hanj/pdf/gkdbk01.pdf

[4] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," *SIGMOD Rec.*, vol. 27, no. 2, pp. 73–84, Jun. 1998. [Online]. Available: http://doi.acm.org/10.1145/276305.276312

[5] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996, pp. 226–231. [Online]. Available: http://www.aaai.org/Library/KDD/1996/kdd96-037.php

[6] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *SIGMOD*, 1999, pp. 49–60.

[7] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with GPS history data," in *WWW*, 2010, pp. 1029–1038.

[8] I. D. Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *ICDE*, 2008, pp. 656–665.

[9] M. Michelson and S. A. Macskassy, "Discovering users' topics of interest on twitter: a first look," in *AND*, 2010, pp. 73–80.

[10] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi, "Measuring user influence in twitter: The million follower fallacy," in *ICWSM*, 2010. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1538

[11] K. Feng, G. Cong, S. S. Bhowmick, and S. Ma, "In search of influential event organizers in online social networks," in *SIGMOD*, 2014, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/2588555.2612173

[12] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: Finding topic-sensitive influential twitterers," in *WSDM*, 2010, pp. 261–270. [Online]. Available: http://doi.acm.org/10.1145/1718487.1718520

[13] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: Quantifying influence on twitter," in *WSDM*, 2011, pp. 65–74. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935845

[14] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD*, 2011, pp. 1082–1090.

[15] Y. Doytsher, B. Galon, and Y. Kanza, "Querying geo-social data by bridging spatial networks and social networks." in *GIS-LBSN*, 2010, pp. 39–46. [Online]. Available: http://dblp.uni-trier.de/db/conf/gis/lbsn2010.html#DoytsherGK10

[16] ——, "Querying socio-spatial networks on the world-wide web," in *WWW (Companion Volume)*, 2012, pp. 329–332.

[17] N. Armenatzoglou, S. Papadopoulos, and D. Papadias, "A general framework for geo-social query processing," *PVLDB*, vol. 6, no. 10, pp. 913–924, Aug. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2536206.2536218

[18] Q. Huang and Y. Liu, "On geo-social network services," in *Geoinformatics*, 2009, pp. 1–6.

[19] W. Liu, W. Sun, C. Chen, Y. Huang, Y. Jing, and K. Chen, "Circle of friend query in geo-social networks," in *DASFAA (2)*, 2012, pp. 126–137.

[20] D.-N. Yang, C.-Y. Shen, W.-C. Lee, and M.-S. Chen, "On socio-spatial group query for location-based social networks," in *KDD*, 2012, pp. 949–957.

[21] J. Jiang, H. Lu, B. Yang, and B. Cui, "Finding top-k local users in geo-tagged social media data," in *ICDE*, 2015, pp. 267–278. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2015.7113290

[22] S. Vaid, C. B. Jones, H. Joho, and M. Sanderson, "Spatio-textual indexing for geographical search on the web," in *SSTD*, 2005, pp. 218–235.

[23] D. Zhang, K.-L. Tan, and A. K. H. Tung, "Scalable top-k spatial keyword search," in *EDBT*, 2013, pp. 359–370.

[24] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems," in *SSDBM*, 2007, p. 16. [Online]. Available: http://dx.doi.org/10.1109/SSDBM.2007.22

[25] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects," *PVLDB*, vol. 2, no. 1, pp. 337–348, 2009.

[26] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. L. Lee, and X. Wang, "IR-tree: An efficient index for geographic document search," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 4, pp. 585–599, 2011.

[27] D. Wu, G. Cong, and C. S. Jensen, "A framework for efficient spatial web object retrieval," *VLDB J.*, vol. 21, no. 6, pp. 797–822, 2012.

[28] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg, "Efficient processing of top-k spatial keyword queries," in *SSTD*, 2011, pp. 205–222.

[29] L. Chen, G. Cong, C. S. Jensen, and D. Wu, "Spatial keyword query processing: An experimental evaluation," *PVLDB*, vol. 6, no. 3, pp. 217–228, 2013.

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-561-1

AALBORG UNIVERSITY PRESS