MODELING USERS FOR ONLINE ADVERTISING

BY QIANG MA

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

S. Muthukrishnan

and approved by

____

____

____

____

New Brunswick, New Jersey

October, 2016

ABSTRACT OF THE DISSERTATION

Modeling Users for Online Advertising

by Qiang Ma

Dissertation Director: S. Muthukrishnan

Online advertising is able to target users at a fine level of granularity. To do this effectively, models are required to represent users and their behavior. In this thesis, we studied several problems related to models of online users. In online advertising, advertisers and ad platforms use user profiles as the language to target users, composed of user information from demographics, location, and interests. We implemented a user-profile-driven ad crawling framework and empirically investigated the relationship between user profiles and the ads to which they are exposed. We observed user profiles to play a greater role in display ads than in video ads. Furthermore, the main mode of accessing online content has been shifting from website browsing to mobile application usage. Mobile apps have become the building blocks to model user behavior on mobile devices. We designed a neural network model (`app2vec`) to vectorize mobile apps by studying how users employ these apps. We analyzed the learned app vectors qualitatively and quantitatively and used them to extract user app usage profiles for app-install advertising. Finally, advertisers are faced with the challenge of finding the optimal user profile properties to target. We designed a look-alike audience extension system, where advertisers provide a list of past converters as ”seed users” and our system determines users similar to the seed. Rather than assuming linear separability of lookalike and non-lookalike users, as in prior work, we propose a new approach with nearest-neighbor filtering. Our system works efficiently for billions of users and improves the ad campaign conversion rate in practice at Yahoo!.

Acknowledgements

This thesis could never have been completed without the hard work and support of several people. I am forever grateful to my advisor Prof. S. Muthukrishnan. Through our numerous conversations and joint projects, I was fortunate enough to be influenced by his system of questioning, reasoning and presenting. In addition to providing academic instruction, Prof. Muthukrishnan also helped me to become a more positive and healthy person. His active support played an essential role in ensuring the survival of me and my family in our new home.

I would like to thank my committee members–Prof. Muthukrishnan, Prof. Nath, Prof. Imielinski and Dr. Kale–for taking the time to serve on my committee and for providing me with valuable feedback on my thesis work.

Many thanks to my former advisor, Prof. Danfeng Yao, for your guidance at the start of my graduate study and for introducing me to the research world. My sincere thanks to my advisors in my master program: Prof. Frank Hsu, Prof. Gary Weiss and Prof. Damian Lyons. Without your guidance and assistance with my research work, I would never have had an opportunity to get into a PhD program.

Along my journey, I have been more than fortunate to have many supportive mentors who have helped me see beyond the ivory tower. My first internship mentor at MLP, Michael Kurtis, benefited me with his professionalism and his sharp mind. More importantly, he gave me the courage to pursue a PhD course. My research internship mentor at Narus, Han Hee Song, trained me for the R&D work environment. I would also like to thank my research internship mentor at Flurry, Wil Simpson, whose elegant way of breaking down complex practical problems and whose organized working style have greatly influenced my approach to work.

My fondest memories of graduate life are of working with my friends at the MassDAL lab, including Darja Krushevskaja, Priya Govindan, Vasisht Gopalan, Edan Harel, Brian Thompson, and John Robert Yaros. Thank you for the skills you taught me and for the courage and support you gave me during hard times. I am grateful to have worked closely with Darja Krushevskaja on many projects. She has always been an honest friend and helped me to overcome several difficulties.

I would not have been able to make so much progress without close collaboration with my talented and diligent colleagues at Yahoo!. My sincere thanks go out to Datong Chen, Zhen Xia, Musen Wen, Peiji Chen, Liang Wang, Jialing Liu, Jiayi Wen, Eeshan Wagh, and Robert Ormandi. I am indebted to Stratis Ioannidis for helping me to start breathing industrial scale big data. It has been my great honor to work with my co-authors and collaborators: Graham Cormode, Brian Thompson, Paul Barford, Igor Canadi, Mark Sandler, Ye Tian, Liang Wang, Kui Xu, Danfeng Yao, Alexander Crowell, Swati Rallapalli, Mario Baldi, Lili Qiu, and Antonio Nucci.

I can never thank my family enough for their continued love, support, patience and forgiveness. My father and mother have given everything they had to support my growth, including bravery to explore the world. In addition, my in-laws have always been supportive and I am very grateful.

Without the love and understanding of my dear loving wife, Qing Zhang, I would not be where I am today. She has given me the strength and encouragement necessary to face the world. I will try to get home early from work, my love.

Dedication

To my parents, my wife and my son.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With developments in Internet technology providing easier access to abundant information and services, people are more and more engaged in online activities. After just a few decades, several large online service providers have attracted hundreds of millions of active users (*e.g.,* Facebook [3], Google [4]) and are generating activities on the scale of billions every day. Massive data present a challenge to service providers to understand their users with the goal of improving their services. Meanwhile, opportunities for new applications abound.

With advances such as Javascript, which supports pixel tracking and beacon tracking, users' online activities can be logged with great detail, down to the browsing of a particular section of a web page. Such user tracking (discussed in Chapter 2) is ubiquitous in two aspects: (1) User tracking exists on most content providers [88], and (2) Numerous trackers can exist on a single web page. Over time, user tracking companies can accumulate user activity data and try to understand user behaviors. For example, it is natural to assume that if a user engage in multiple activities related to the content of a certain topic, the user is likely to be interested in that subject. Hence, user tracking companies invest in logging user activities, analyzing the contents consumed by users, and finally building models to construct a user representations which reflect online interests. However, user tracking within a single content provider is quite limited. Since users consume various types of online content from different content providers every day, the user representation from a single content provider can be biased. In order to obtain a complete representation, the anonymous browser cookie syncing technique plays a critical role. When a user visits a particular website, the content provider shares users (hashed) cookie under its domain and the users current activity with its partner companies. Today cookie syncing is ubiquitous [77, 11], and it enables the next level of user tracking tracking the same user across the Internet. Over the past decade, advertising has emerged as the primary source of revenue for many websites. The ability to track and model each user has inspired the creation of numerous applications, one of the most important being targeted advertising.

In the early stages of online advertising, advertisers arranged with content providers to display their ads on individual web pages to a certain number of visitors over a certain period of time. Nowadays, advertisers have much more flexibility in targeting audiences. Advertisers and ad platforms adopt the language of user profiling, which includes demographic and geographical information as well as fine granular interests. When advertisers set up ad campaigns, for instance, they can specify to target users interested in particular topics. Therefore, on a particular web page, instead of seeing the same ads, visitors will see different ads related to their past online activity. Furthermore, users are likely to encounter ads based on their activity history across the web. However, a lack of systematic study of the advertising landscape prevents understanding of which ads are shown to which types of users. In Chapter 3 of our work on user profile-driven advertising, we report a first-of-its-kind study of the features, mechanisms, and dynamics of ad targeting in practice. Our study takes the perspective of users who are the targets of ads shown on websites. We developed a scalable crawling capability that enabled us to produce synthetic user profiles and use them to gather the details of display and video ads, including creatives and landing pages. In the display ad landscape study, we found that while targeting is widely used, there remain many instances in which delivered ads do not depend on the user profile. We identified over 3.7K distinct advertisers from a variety of business segments. Finally, we found that when targeting is used, the types of ads delivered correspond to the details of user profiles, including patterns of visits. On the other hand, in the video ad study conducted on YouTube, we found that video ads mainly target video content rather than user profiles and can have a significant impact. Finally, we focused on an interesting type of video content on YouTube referred to as polymorphic videos, which involve different copies of the same video. While users watching polymorphic videos are likely to have similar interests, our study indicates that after watching different copies of the same video, users tend to have different interests added to their profiles.

Over the years, the primary mode of accessing online content has shifted to mobile devices, and user online activities have transitioned from website browsing to mobile application usage.

Based on reports from Flurry Analytics[1], US mobile users download more than eight apps per month on average [1], and 90% of the time spent on mobile devices was spent using apps [6]. This being the case, mobile apps are the key to understanding user behavior on mobile devices, and it is of great interest to understand the context in which mobile apps are used. In Chapter 4, we propose a neutral network model, `app2vec`, to represent apps in a vector space without a priori knowledge of their semantics. It involves visualizing a users mobile app usage over time as a "document" and applying to it the recent `word2vec` model. But the apps in a training context are carefully weighted by the usage time interval. We show that `app2vec` captures semantic relationships between apps much better than conventional bag-of-words and matrix factorization methods. The learned app vectors can be used to extract user app usage profiles for multiple applications in online app-install advertising. In this work, we present ad selection, app recommendation, and conversion prediction models. In mobile advertising, for a given impression there can be hundreds of thousands of eligible app ads to display. We abstract a step in this advertising pipeline that preprocesses this large set of eligible ads into a smaller set of relevant, diverse ads, which we refer to as the ad consideration set. We design a randomized algorithm to find such a set by choosing the set of k apps as the approximate mode of a Determinantal Point Process (DPP) distribution. While DPPs have been useful in clustering and other applications, their use in our context is novel. In offline experiments using app-install ad serving logs from Yahoo!, our proposed method achieves 36% lift in precision and 47% lift in recall compared to collaborative filtering methods (for example, at k = 5). We also use the technology of `app2vec` and DPPs to give a new algorithm for personalized app recommendation, using app clustering based on app distances in their vector space. With our additional app vectorization, collaborative filtering can achieve 7% lift in precision and 9% in recall. Finally, app clustering based on app distances in their vector space can effectively improve the accuracy of the conversion rate prediction model as well. According to our offline experiments with ad server logs, user app usage profiles extracted from our app clustering method demonstrate 4.7% AUC lift over baseline user profiles.

One key application of user online profiling is ad campaign targeting. A typical user targeting method is based on the pre-defined user interest taxonomy created by advertising platforms.

[1] `http://flurrymobile.tumblr.com/`

Advertisers must first analyze and understand the behaviors of their existing customers. They must then utilize the user representation taxonomy from a particular advertising platform to approximate the desired group of customers to target. It is therefore useful to relax the necessity of such explicit user profile targeting and automatically extend the current set of customers to a larger reach of audience with high potential value for the advertiser. In Chapter 5, we present an efficient large-scale lookalike audience extension system, where advertisers provide a list of past converters as "seed users" and our system automatically determines users exhibiting behavior patterns similar to the seed and makes them available for advertisers to target. Rather than assuming linear separability of lookalike and non-lookalike users as in prior works, we propose a campaign-specific nearest neighbor- based method. Extensive experiments have been conducted to compare our look-alike model with three other existing look-alike systems. The results demonstrate that our developed nearest neighbor filtering based method outperforms other state-of-the-art methods by more than 50% in terms of conversion rate in app-install ad campaigns. Our system works efficiently for billions of users, has been deployed to production in Yahoo! and boasts a significantly improved conversion rate compared to other types of user profile targeting methods.

Chapter 2

Overview of User Online Ad Targeting

2.1 Online Advertising

Advertising online is a compelling proposition for brands and e-commerce vendors that seek engagement with a broad cross-section of potential customers. The sheer volume of online users and the increasing amount of time that people spend online has led to an estimated $27.5B in online ad spending in the US for the first half-year (HY) of 2015, which represents a 19% increase over the previous year [24]. The majority of this spending is on advertising that most commonly appears in search results as text ads. There is, however, a growing preference for display ads typically image and video ads that appear in response to users browsing, app usage, and other activities on the Internet that can convey more robust and visual messages to users. A recent report by Forrester estimates that $16.8B was spent in the US on display and video advertising in FY2015, and this figure is growing at 13% annually [68]. Furthermore, according to this report, mobile display advertising was estimated to reach $6.8B in the US in FY2015 and grow at 42% annually.

The ubiquity of advertising on publisher websites and apps makes it easy to overlook the diversity and complexity of the online ad delivery ecosystem. A well-known depiction of the online ad ecosystem is the *Display Lumascape* that shows Marketers (brands) and Publishers/Consumers connected by hundreds of companies that provide a variety of intermediary services [81]. The task of delivering billions of ads from thousands of Marketers to millions of Publishers websites and apps on a daily basis is quite daunting, hence the complexity of the system.

Of central importance in the ad delivery process is the selection of a particular ad to display to a given visitor of a Publisher site. This is referred to as the *targeting problem*. Targeting is commonly based on such criteria as site/page context, placement size, user behavior and geolocation. Ad serving infrastructures provide targeting information that enables Marketers to bid on specific ad requests from a large number of Publishers. The presumption is that improvements in ad targeting will benefit all of the constituents in the ad ecosystem.

2.2 User Targeting

The main parties in the online advertising market are described below:

- Publishers: the online content providers, including traditional ones like the New York Times, Yahoo Finance, etc., and new types of content providers such as social networks (e.g., Facebook) and mobile apps (e.g., game apps, fitness apps). The goal of publishers in the market is to monetize their user traffic (ad impression inventory) by displaying ads to visitors.

- Advertisers: the companies or individuals that seek to promote their products. These include traditional advertisers who want to sell products or brand their companies and the new group of advertisers in the mobile domain who want to get more people to install and engage with their apps. The goal of advertisers is to promote their brands or sell as many products as possible within their advertising budget.

- Ad-networks[1]: the companies that connect publishers and advertisers, allowing advertisers to deliver ads to publisher web pages. The goals of ad-networks are two-fold, to help publishers maximize their inventory monetization and to help advertisers deliver ads to online users which maximize their return on advertising investment.

- Online users: the online content consumers, such as web page viewers or mobile app users. In the market, they are the creators of advertising inventory and are identified by anonymous IDs such as browser cookies.

In the context of online advertising, the main parties conduct practices to make user targeting possible and drive advances in user targeting performance. More specifically:

[1] There are more fine divisions of ad-networks functions, in our discussion the general entity of ad-networks is sufficient.

- Publishers embed user tracking companies user tracking code in their web pages, such that all their users detailed content consumption activities are logged by user tracking companies.

- Ad-networks typically do user tracking by themselves but may also purchase user data from other user tracking companies to get a complete view of online user behavior. Ad-networks build models to create a profile of online users, including demographics, geolocation, interests, etc. These user representations provide advertisers with the flexibility to target their desired groups of audiences.

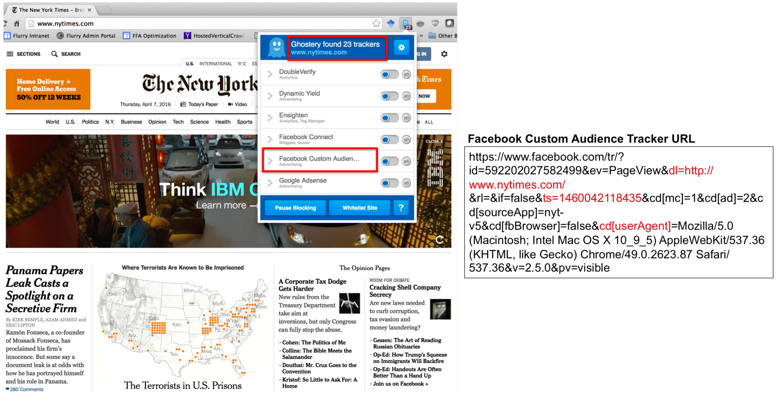
- Advertisers are the consumers of user online profile modeling results. Given limited budgets, advertisers cannot explore the whole user space to gain potential customers. Instead they use their business insights to design ad campaigns targeting particular groups of users, which are expected to bring a higher return on advertising investment.

- Users often show some consistency in their online content consumption patterns. These patterns are captured by user tracking companies to compose online user profiles, which are updated over time. User profiles not only benefit advertisers but also expose users to more relevant ads and improve the online experience.

User tracking plays an important role in collecting online user activities [88]. User tracking companies place JavaScript on their partners' web pages such that when a user opens the content page, the browser will also initiate a request to a page owned by the tracker. Information about this current page visit– usually including user browser cookie, page URL, time of day, user IP address, and browser type–will be sent to trackers. Depending on the type of content page, more detailed category information about the page may be passed as well.

Figure 2.1 shows an example of user tracking on the New York Times homepage. Using Ghostery[2] plug-in, we can see that there are 23 trackers (at the time of the screenshot) on the homepage of the New York Times. We can acquire some information about this page visit from the URL parameters: (a) content page URL: *dl=http://www.nytimes.com*; (b) time of the visit: *ts=1460042118435*; (c) user browser type from parameter *cd[userAgent]*; (d) There are two ads

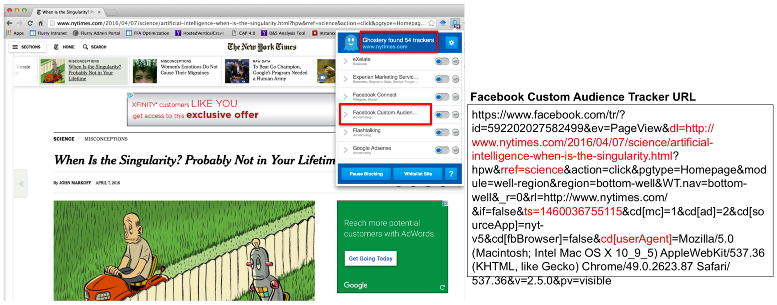
[2] https://www.ghostery.com/



Facebook Custom Audience Tracker URL

https://www.facebook.com/tr/?id=592202027582499&ev=PageView&dl=http://www.nytimes.com/&rl=&if=false&ts=1460042118435&cd[mc]=1&cd[ad]=2&cd[sourceApp]=nyt-v5&cd[fbBrowser]=false&cd[userAgent]=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/537.36&v=2.5.0&pv=visible

Figure 2.1: 23 trackers on New York Times homepage. Accessed on 04/07/2016.



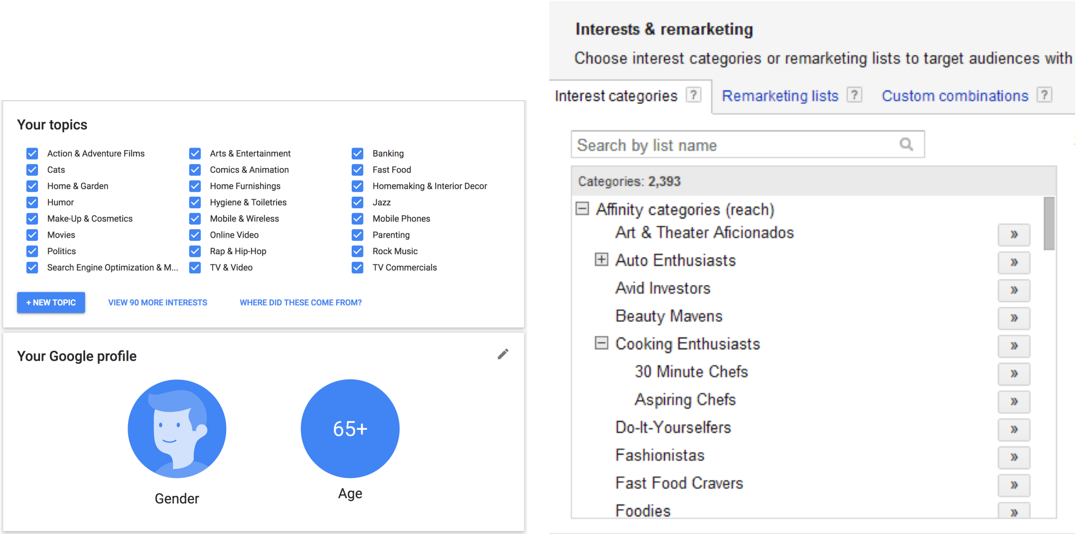
Facebook Custom Audience Tracker URL

https://www.facebook.com/tr/?id=592202027582499&ev=PageView&dl=http://www.nytimes.com/2016/04/07/science/artificial-intelligence-when-is-the-singularity.html?hpw&rref=science&action=click&pgtype=Homepage&module=well-region&region=bottom-well&WT.nav=bottom-well&_r=0&rl=http://www.nytimes.com/&if=false&ts=1460036755115&cd[mc]=1&cd[ad]=2&cd[sourceApp]=nyt-v5&cd[fbBrowser]=false&cd[userAgent]=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/537.36&v=2.5.0&pv=visible

Figure 2.2: 54 trackers on a science article page from the New York Times. Accessed on 04/07/2016.

on this page *cd[ad]=2*. If the user is reading a particular article, more details will be available in the URL parameters. For example, Figure 2.2 shows user tracking on a science article page from the New York Times. Browsing the New York Times homepage may indicate the user is interested in reading news articles in general, but this particular article page reveals a finer level of user interest. We can see that there are more than twice the number of trackers (54) compared to the home page. The tracker URL from *Facebook Custom Audience* reveals additional details, such as *rref=science* indicates the category of the article, while *rl=http://www.nytimes.com* discloses from where the user comes to this article.

Using the detailed logging of user online activities, user tracking companies build models to construct a profile for each user identified by anonymous browser cookies. The common elements of such profiles include user demographics (age, gender), user locations, and a list of interests revealed by historical content consumption. Figure 2.3(a) is an example of a user interests profile from Google ads settings[3]. This example was taken from a browser after browsing several random web pages. This is the profile which Google provides to users themselves. Figure 2.3(b) illustrates the user interest-based targeting interface provided to advertisers, allowing them to choose a combination of user interests to form a target audience.



Your topics

☑ Action & Adventure Films
☑ Cats
☑ Home & Garden
☑ Humor
☑ Make-Up & Cosmetics
☑ Movies
☑ Politics
☑ Search Engine Optimization & M...
☑ Arts & Entertainment
☑ Comics & Animation
☑ Home Furnishings
☑ Hygiene & Toiletries
☑ Mobile & Wireless
☑ Online Video
☑ Rap & Hip-Hop
☑ TV & Video
☑ Banking
☑ Fast Food
☑ Homemaking & Interior Decor
☑ Jazz
☑ Mobile Phones
☑ Parenting
☑ Rock Music
☑ TV Commercials

Your Google profile

Gender

65+
Age

Interests & remarketing

Choose interest categories or remarketing lists to target audiences with

Categories: 2,393

- Affinity categories (reach)
  - Art & Theater Aficionados
  - Auto Enthusiasts
  - Avid Investors
  - Beauty Mavens
  - Cooking Enthusiasts
    - 30 Minute Chefs
    - Aspiring Chefs
  - Do-It-Yourselfers
  - Fashionistas
  - Fast Food Cravers
  - Foodies

(a) Interests profile displayed to users themselves (b) User interests provided to advertisers for targeting

Figure 2.3: User profile example from Google.

Considering the big picture of the interactions among users, content and advertisers, there are billions of pieces of online content, each characterized by content category and other metadata. A common approach adopted by online advertisers, content providers and ad networks is to think of "profiles" of target users based on their interests. This underlying language of profiles and categories is updated as events progress, setting up a cycle with sophisticated dynamics as illustrated in Figure 2.4: (a) Users view the content of interest to them; (b) Contents viewed impact user profiles; (c) Profiles of users of a content page potentially impact categories of the content; (d) Advertisers upload ads which also have categories; and (e) Advertisers target content and user categories, which affects which ads users are shown, and so on.

[3] www.google.com/settings/ads

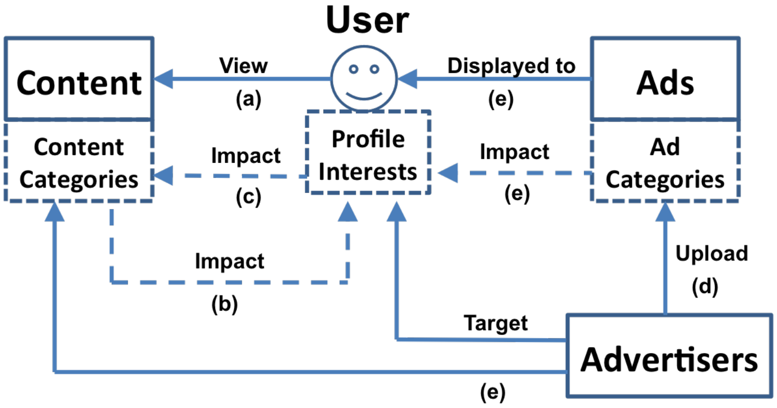


Figure 2.4: Interactions among user profile, content and ads.

2.3 Research Directions

In connection with the theme of user tracking in online advertising, there are a plethora of research directions for exploration.

Cross-device user tracking. Nowadays, users can access online content through multiple devices. Ad-networks have the incentive to track online user activities across devices, in order to get a complete picture of user online behavior and enable better ad targeting. However, privacy concerns dictate that cross-device tracking be accomplished without involving personally identifiable information. One way to address the problem is based on the fact that each device di has an IP address history $L_{d_i} = \{l_1, l_2, \ldots, l_n\}$. By comparing the similarity of different users IP history lists, one can calculate the probability that two devices belong to the same user, given the assumption that when two devices appear at the same locations many times, they are likely to belong to the same user. One challenge is how to efficiently compare among millions of devices. Hopefully, with time stamp information associated with each IP address, device matching accuracy will improve. However, there remains the problem of aligning user location and time.

Value of user profile. For a given user u, let the profile be $P_u = \{c_1, c_2, \ldots, c_n\}$, where c_i represents interest i.

- Several different models may be used to construct a user profile, and they may claim different sets of interests. To add a new interest c_k into user profiles, we need to know the incremental value of c_k. For example, when u clicks or converts for ad a_i, how do we measure the contribution of each interest c_i?

- If we consider the entire ad-network, different user profile elements have different costs associated with them. The cost can come from research and development, third-party data purchasing, etc. The problem becomes how to measure the monetary contribution of c_i to ad-network overall revenue to support business decisions.

- User interests are usually hierarchical. While the more detailed user interests are more valuable to some advertisers for ad targeting, the cost of getting such fine granularity data is also higher. Given this conflict, how does the level of profile interest granularity impact ad-networks revenue? [86] analyzed this problem in an informal marketplace setting without considering the dynamics of advertisers bidding strategy when precise user data are not available.

- Without being able to target particular groups of users, advertisers must compete (bid) for opportunities to reach the same set of users. Users are divided into groups of interest to different advertisers. Hence, we can expect the competition for a single user to be less. How does this affect the ad-networks revenue? Should ad-networks charge additional fees for user profile targeting, and if so, how much?

Ad targeting on user profile. In order to set up an ad campaign to promote a particular product g in terms of click-through-rate (conversion-rate), an advertiser supposes that optimal target users will have profile properties $P^* = \{p_1, p_2, \ldots, p_n\}$.

- How is a given advertiser, with or without ad campaign histories on the advertising platform, to find P^* for the advertiser in general or for the particular product g?

- Since different user profile properties are associated with different costs, suppose P^* is known, how to set up campaigns with profile properties $P^{'}$ under the constraint of $\arg\max_{P'} |U^{P^*} \cap U^{P^{'}}|$ and $C(P^{'}) \leq B$, where U^{P^*} is the set of users satisfying the targeting criteria P^* (similarly for $U^{P^{'}}$), $C(P^{'})$ is the cost of running campaigns by targeting $P^{'}$, and B is the budget of advertiser.

- In the field of economics, “price discrimination” is the practice of charging a different price for the same good or service, in order to capture the maximal consumer surplus.

In online advertising, an advertiser can take advantage of detailed user profiles to charge online users different prices (or offer different discounts) for the same service. Although there have been some studies of price discrimination on online shopping sites [44, 72], there is a need for further research into the status of such price discrimination in ads and its impact on the participating parties in the market. One of the challenges involved in such a study is the high cost of collecting targeted ads from the web. A large-scale profile-driven ad crawling system [22] would be useful in this regard.

Online user data storage and processing. Ad-networks may track billion of users (e.g., users identified by browser cookies) over time. In such cases, data storage space and data processing time both become challenging issues.

- Once in storage, online user activity data can be used in multiple applications, such as modeling user interests for content personalization or ad targeting, extracting user signals for ad click/conversion prediction models, and modeling user visit patterns to forecast future user traffic. The problem then becomes how to summarize user activity data to reduce data size while still retaining useful information for downstream user modeling.

- Multi-pass (or even single-pass) processing of such huge amounts of data is very expensive, and there is a need for streaming algorithms to summarize the characteristics of the data (e.g.,estimate median [64, 43]). In the case of such extensive data, even streaming through a single processing node is unaffordable. What is needed are better streaming algorithms under the map-reduce framework.

The above problems are of interest to both the industry and theoretical research.

Chapter 3

Observe User Online Advertising Profile and Ad Targeting

As discussed in the previous chapter, users online activities are tracked by companies (from content publishing companies to user data collection companies) to construct online user profiles. One significant usage of the user data is in online advertising targeting. Some natural questions to ask are: Do ads target user profiles in the field? What are the ads shown to different users? How do ads impact users profiles? In this chapter, we present a first-of-its-kind study of both display and video advertisements that are being delivered to online users.

Our objectives are to broadly characterize the online advertising landscape or *Adscape* and to elucidate targeting mechanisms from an empirical perspective. The study seeks to better understand the range of online ads, the degree of similarity between ads shown to different visitors of the same content (web page or video), the breadth of ads that are displayed on a given content, and the extent to which ads shown on various content pages are different from each other. In the long term, we hope to provide a foundation for improving ad targeting mechanisms and streamlining the ad-serving ecosystem.

The capability to gather display ads and video ads from across the web is central to our work. The vast number of websites that run display and videos ads, the diversity and dynamics of the targeting mechanisms, and the need to minimize the impact (load due to measurement) on any given website all present significant challenges. We addressed these challenges by developing novel methodologies, tools and infrastructure for ad-centric web-crawling. At the heart of our system is the notion of *profile-driven crawling*, that enables each crawler instance to interact with the ad ecosystem as though it were a unique user with particular characteristics. Our developed framework can efficiently generate a large amount of synthetic user profiles with desired profile properties. This capability is essential for collecting the full spectrum of ads that are delivered across websites.

3.1 Online Ad Landscape

3.1.1 The Anatomy of Online Advertising

The earliest online display ads, which appeared in the mid-1990's, were simply a bright, flashy billboard that was displayed to *any* visitor to the page over some period. The technology for serving ads has evolved tremendously since then, and now ads are typically *targeted* to users. We can look at the diversity of modern display ads from several different perspectives.

Advertiser View. Whether selling a product or promoting a particular discount, an advertiser wants to attract the attention of potential customers. They design advertising campaigns to achieve this goal. Ad campaigns include creatives, target audience demographics and interests, frequencies, placements, etc. Advertiser's budget is a key factor to balance all these considerations. Two example ad campaigns are as follows: (1) *A local car dealer in New Brunswick, NJ targets users living in New Brunswick from Friday to Sunday, and the ad should not be shown the user more than seven times in one day.* This campaign targets users based on location ("geo" in ads parlance), time of the day (one or more "dayparts"), and further limits number of times the ad may be shown within a time range ("frequency cap"); (2) *A local car dealer targets male users aged [35 - 50]. Besides, these male users are interested in "sports" and "travel".* This campaign targets users based on their interests ("profile" in ads parlance) and also demographics (gender, age).

Targeting strategies can be combined in sophisticated ways by advertisers, and the industry relies on the existence of parties who can track cookies and maintain user profiles.

Publisher View. A Publisher produces content or provides services that attract users, and with that, the opportunity to present ads to those users. On any given visit, ads can be served from a variety of sources including (i) *premium campaigns*, which are contracts with specific advertisers, (ii) *ad networks*, which represent multiple advertisers, and (iii) *ad exchanges*, which offers an auction-based environment for matching publishers with advertisers.

Typically, publishers combine the methods, even on a single page.

The Adscape View. Consider a user $u(t)$ accessing a webpage $w(t)$ at time t. Say the publisher of $w(t)$ shows a set of ads $a(t)$ to $u(t)$. There is some *allocation function* $f_w(t) : u(t) \rightarrow a(t)$. The set of all $f_w(t)$'s over all w's and all users $u(t)$ at any time t will be the

Adscape that is the focus of this paper. The functions $f_w(t)$'s may depend on: (i) user's demographics, interests, location, etc; (ii) site w, its contents and context; (iii) time t, and the past, including users' past actions, w's past contents, while f may vary over time; (iv) the set of ads $a()$'s, including mutual constraints that allow or disallow each other; (v) mechanisms, incentives and market conditions that govern the behavior of advertisers, networks, exchanges and publishers. The function $f_w(t)$ may ultimately be simple (all users see the same ads for a day) or sophisticated (each ad is personalized based on multiple criteria). Our research agenda is to broadly understand the user-targeting aspect of the Adscape of online display advertising. We pursue this goal by crawling w's, detecting and harvesting ads $a()$'s, and thereby observing and finding patterns in $f_w(t)$'s.

3.1.2 Ad Formats

Various types of publishers often serve a particular subset of ad types: text, display, stream and video ads. Figure 3.1 exemplifies the typical pairs of publisher and ad types. Display ads and video ads on YouTube are the focus of our study of advertising landscape.

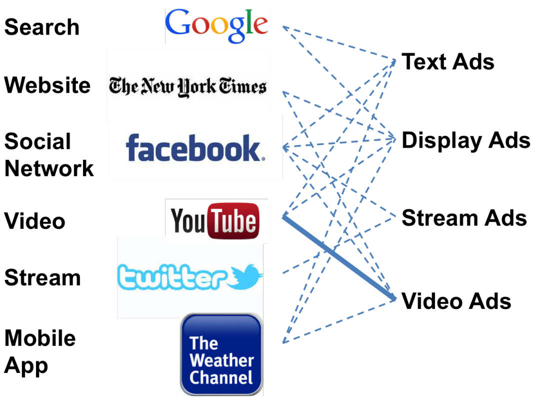
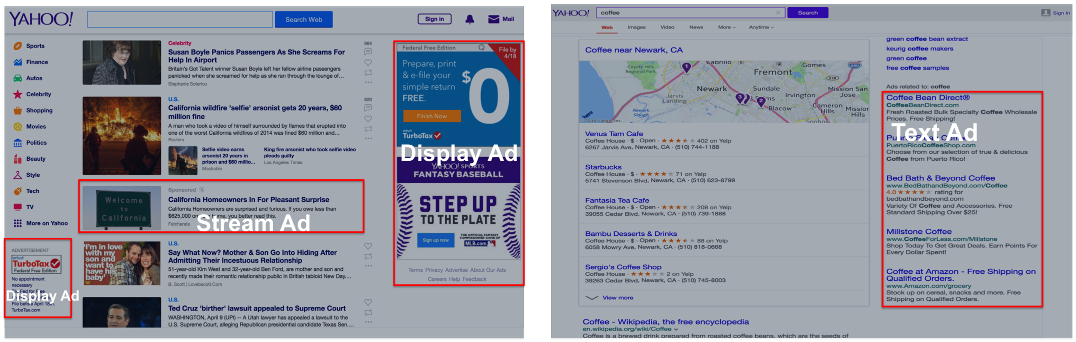


Figure 3.1: Publishers' common ad types.

- *Display ads* are usually in the format of image or flash objects, in the conventional content web pages (*e.g.,*news articles, blog posts) they are displayed at some fixed positions of the page. Figure 3.2(a) shows display ads examples on the home page of Yahoo!.

- *Stream ads* are the ads embedded in the content streams, such Facebook news feed and Yahoo! news from its home page. Users do not have to navigate to different pages to consume more contents, usually by just scrolling the page more contents will be fetched. Figure 3.2(a) shows an example of stream ad in Yahoo! news home page.

- *Text ads* are the simplest format where the ad is composed of succinct ad texts and a link to its landing page. Figure 3.2(b) shows examples of texts ads from Yahoo! search result.



(a) Example ads from Yahoo! home page

(b) Example ads from Yahoo! search result page

Figure 3.2: Ad format examples.

Video Ad Formats

Unlike textual and display ads, video ads are an emerging ad format on the Internet. They are expensive to manufacture and hard to personalize. Two of the leading publishers are YouTube and Hulu. There are several ad formats unique to video ads:

- *Pre-roll, mid-roll, post-roll* are video ad units shown before, during or after the main video of the page, respectively. These ads are similar in nature to TV advertising, except they have a greater capacity for interaction (*e.g.,*selection of an option).

- *Overlay* ads may periodically appear superimposed on the video. These are usually text, image, or Flash object.

- *Sponsored Videos* are present on some websites (*e.g.,*YouTube or Dailymotion). These are entries in the list of suggested videos (usually a sidebar), where some video ad uploader has paid for them to appear.

Web sites carrying the videos can also use standard ad formats, such as display or textual ads. When these ads are linked to video ads on the page, they are often called *companion* ads.

Note that a video itself can contain sponsored material (*e.g.,*a product placement). We do not consider this type of advertising in our study.

Taking the ads on YouTube as examples, Figure 3.3 shows the examples of the ad formats considered in our video ads study.

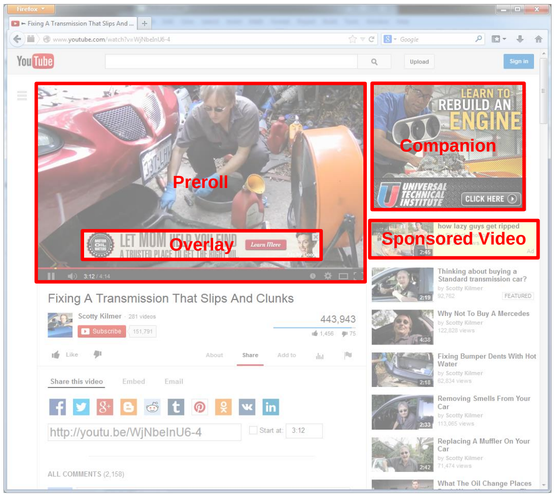

Preroll
Overlay
Companion
Sponsored Video
LEARN TO REBUILD AN ENGINE
UNIVERSAL TECHNICAL INSTITUTE
CLICK HERE
Fixing A Transmission That Slips And Clunks
Scotty Kilmer
443,943
http://youtu.be/WjNbelnU6-4
how lazy guys get ripped
Thinking about buying a Standard transmission car?
Why Not To Buy A Mercedes
Fixing Bumper Dents With Hot Water
Removing Smells From Your Car
Replacing A Muffler On Your Car
What The Oil Change Places

Figure 3.3: Example of different ad formats on YouTube.

3.2 Measurement Methodology

Our overall profile-based crawling system uses novel methods for profile generation, uncontaminated profile crawling, and ads collection and classification. In this section, we describe these methods.

3.2.1 Synthetic User Profiles

Profiles are generated for users based on their interactions with websites. Amongst the common browser cookies and non-cookies based (*e.g.,*Flash cookies, server-side profiling) web tracking technologies, ad networks, and ad exchanges typically associate user properties with users via browser cookies. This basic observation implies a mechanism for building profiles.

There are several ways to create a user profile: (1). The first approach is to mimic actions of a particular user (*e.g.,* [27, 63]). Characteristics of profiles built in this fashion can be arbitrary close to the profiles observed in reality. However, the approach is impractical for the exploration of the large space of profiles. (2). A second method is to establish many versatile profiles by going to web pages of ad networks and creating profiles manually[1]. However, using this approach, only the ad networks for which the profile was set, will recognize the user. Given the complexity of the ecosystem, this method is likely to be insufficient. (3). A third approach is to create *user interest-based* profiles by crawling a set of target sites. This method can potentially generate profiles for *any* user interest depending on the sites visited. It assumes that a user is assigned to profile categories based on websites visited and is presented on ad servers as a “bag” of interests.

We selected the crawler-based approach to explore targeted ad serving broadly. We have implemented Profile Builder as follows. For a given interest category, Profile Builder

1. fetches the top k websites associated with an interest category (*e.g.,*using Google Adwords Ad Planner, Alexa web page categories, etc);
2. opens Firefox with an empty profile (*e.g.,*using Selenium WebDriver[2]) and visits fetched URLs;
3. visits Google ads settings editor page and captures its content. This step is not required for profile generation. However it allows us to check the interest categories assigned to the profile;
4. zips the profile, including the cookies;
5. stores the profile for future use.

One can verify the *state of the profile*, or interest categories assigned to it, at any given moment, by opening Firefox with the profile and visiting the corresponding pages of ad networks. There are multiple sources for input interest categories. For instance, interest trees used by ad networks to allow advertisers to choose their target audiences.

This approach has several limitations. Profiles treated in this way do not take into account server-side profiling or Flash cookies. Another limitation is in the ability to discover *retargeted* ads, since for profile to get retargeted it has to visit the website of the advertiser first. However, retargeted advertising, browser and device-based targeting are out of the scope of this study.

[1] *e.g.,*www.google.com/settings/ads

[2] http://docs.seleniumhq.org/projects/webdriver/

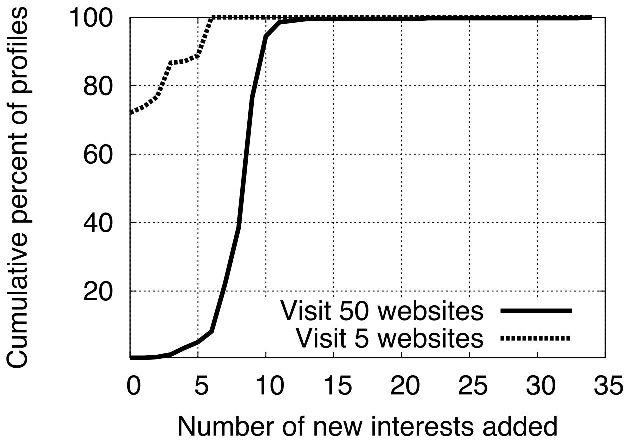


Figure 3.4: Dynamics of the number of profile interests as websites are visited.

Initial Profile Contamination. As discussed earlier, for each target interest category Profile Builder uses a list of websites. In general, websites have more than one category, for instance, Google Display Planner associates categories *News/Business News, Finance/Investing* with `bestonlinetrading.info`[3]. Intuitively, categories describing content are assigned to a user profile. Hence, the generated profile can have multiple categories, in addition to target category, associated with it. For instance, the profile created for interest category *Finance/Accounting & Auditing* contains interests *Business & Industrial*, *News*, and *Computers & Electronics* etc.

Uncontaminated Crawling. Profiles are dynamic, and change as more websites are visited (*e.g.,*new categories are added). For the purpose of our study, we refer to this phenomenon as *profile contamination*. Profile change is undesirable when we seek to understand how ads are targeted at a particular profile. To get a better understanding of this phenomena, we conducted the following experiment. We sampled 60 interest topics from Google's ad interests[4], and for each interest topic we create one synthetic user profile by crawling 50 random websites from Alexa[5] categorized websites. We checked the state of the profiles using Google's ads settings editor page after each visit. Figure 3.4 shows the distribution over the number of new categories

[3] These are the results when study was conducted, and they are subject to change over time due to changes of algorithm from Google.

[4] `www.google.com/settings/ads`

[5] `http://www.alexa.com/topsites/category`

obtained after visiting 50 websites. The figure indicates that 60% of profiles gained more than 9 new interests after 50 visits. That is significant, as an average number of interests at the beginning of the experiment was 8. However, this issue can be mitigated by limiting website visits to 5 or even fewer.

Harvesting and Identifying Ads Ads are often delivered to web pages by JavaScripts that are executed at the time of page load. For instance, it is not sufficient to just send an HTTP request and parse the response, since the ads will not be initialized. To address this, one could potentially call JavaScripts. However, this requires calling "proper" JavaScripts with "proper" parameters. Alternatively, one can let the browser do the work and see the page exactly the way a user would see it. We chose to stay agnostic to JavaScript execution and load the pages directly in a browser.

It is sometimes hard to find ads on a given web page, even for a human. We have developed a three-step decision process that identifies ads automatically. To be classified as an "ad" a target visual element has to pass following tests: (1) *AdBlock Test*. Adblock's easy list [6] is a database of regular expressions that can be used to detect ads. We test an element's URL, iFrame URL, `div` class, and a landing page URL against it to see if a match can be found; (2) *Dimension Test*. Display ads frequently have standard sizes (*e.g.,*a standard banner is 728×90), which enable them to fit into the design of many pages. We maintain a list of 25 different standard dimensions. The visual element has to match one of the entries in the list; (3) *Self Ads*. A visual element cannot link to a page within the same domain. Ad elements have to have an external link. If visual element passes all tests, it is classified as "ad".

We have implemented a distributed ad harvesting and parsing infrastructure based on the methods described in Figure 3.5. Ad harvesting is managed by the *Controller*. The Controller is configured with a *crawling plan* — a list of persona/site pairs to crawl with specified frequency — as an input, and executes it while balancing the load. The Controller manages the number of Firefox instances (also stated in the crawling plan) that are administered via our Firefly extensions. Most importantly, the Controller can open Firefox with profile p as per the crawling plan. The Controller sends commands to Fireflies to visit w's respecting profiles p that the

[6] http://easylist.adblockplus.org

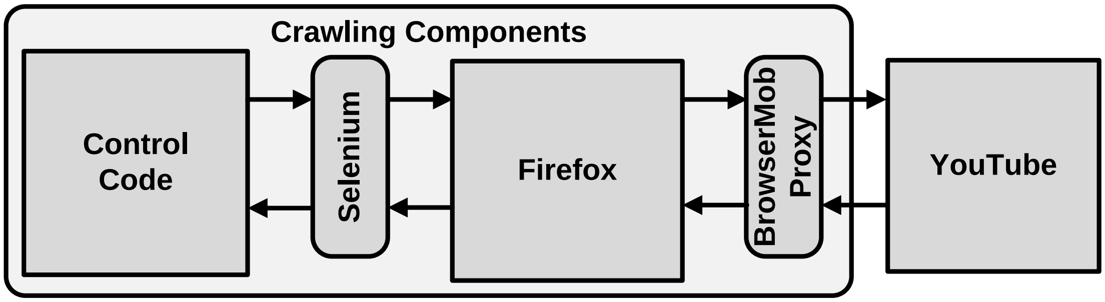


Figure 3.5: Ad Crawler Components.

Firefox's are using. The Fireflies follow the order and report back harvested visual elements, which are stored in database.

Parser instances function independently of the harvesters. They take unprocessed entries populated by harvesting and parse them. Also they (1) identify ads and (2) resolve ad landing pages. The Parser also downloads all visual elements and stores a local copy of them.

3.2.2 Controlled Webpage Crawling

To harvest display ads and study user profile targeting in display ads, we focus on the impact of user interest-based ad personalization or *user interest-based Adscape*. More formally, we restrict $f_w(t) : u(t) \rightarrow a(t)$ as follows: (1) we fix geolocation (by performing all data collection from a single location); (2) we do not consider time-of-day effects. Both dimensions are important and will be the subject of future work. This study focus is on $f_w(t) : p(u(t)) \rightarrow a(t)$ where $p(u(t))$ is the *profile* (or *persona* which we use interchangeably) of the user. We will make $p()$ more precise in the future, but it encompasses users' interests. We refer to (w, p) as a *pair* where w is a website and p is a persona. While the distribution of different types of personas on Internet can be arbitrary and can be a parameter of $f_w(t)$, in our work we restrict our attention to targeting algorithms given a (w, p) pair. In the future, one could expand the study by exploring actual frequencies of different user types and shape of the traffic (*e.g.,*using comScore).

Let $\mathcal{W} = (w_1, w_2, \dots)$ be set of all websites, and $\mathcal{P} = (p_1, p_2, \dots)$ be a set of all personas. In general, we will not be able to use all pairs formed from $\mathcal{W}$ and $\mathcal{P}$ for crawling because of the imposed load on our systems as well as ad ecosystem. Hence, we approach it in four steps:

- Given a single pair (w, p), we crawl it — crawl site w with browser depicting persona p – several times in a row. We study the distribution of ads over time, and propose a pattern of crawls (*crawling strategy*) we will ultimately deploy for the pair.

- We model a persona as a set of user profile interests that are associated with the user (or her browser, to be more precise). We build P from Google's advertising interests tree. For this study, we choose interest categories such that P is diverse and represents interests that are popular on the Internet.

- The number of distinct web pages on Internet is huge, and grows each day. However, in practice, only a few of these are frequently visited. For this study, we create a pool W from top popular websites using Alexa[7].

- Crawl all possible pairs formed from W and P for a short period. Analyze the data to identify a small *focus set*: a subset of pairs that we will crawl operationally. The choice is made on a budget regarding the number of crawls we can do with the crawling strategy above.

- We crawl the focus set of pairs as per the strategy, log the crawls and collect data about the ads observed.

3.2.3 Controlled Video Watching

In the case study of video ads targeting, our goal is to understand what video ads a user sees. A user u at time t, represented by profile $u_p(t)$ at site w sees a video ad a which is a function f_t:

$$f_t(u_p(t), w, t) \rightarrow a$$

where f may be sophisticated, *e.g.,*profile $u_p(t)$ may depend on the entire history of user's behavior; ads shown may vary over time t; contents of w changes over time; etc. Our formal goal is to study function f_t.

Crawling. One can consider crawling video serving websites with synthetic profiles (*e.g.,*[27,

[7] http://www.alexa.com/topsites

22]). This method is particularly attractive since one would have control over the service/video targets, access frequency, and user profile choices. However, there are difficulties in the creation of synthetic profiles.

- *Through browsing.* Synthetic user profiles can be established through targeted browsing similar to what is reported in [22], and use a popular user tracking service (*e.g.,*Google) as a proxy to find out the resulting user profile interests. Later one can use the created profiles for data harvesting and analysis. However, it was difficult to have a fine control of the outcome. For instance, if one wants to create a profile associated with "exotic pets", what content should one consume?

- *Through manual setup.* Some companies that profile users offer an interface for users to monitor or modify their profiles. For example, one can change demographic information, add or remove individual interest from Google at Ads Settings Manager[5], or from eXelate at interests settings page[8]. Potentially, one could visit some, or all, of these interfaces and set up suitable profiles. Later one can perform data collection, using established profiles. However, not all companies have such functionality. We are unaware of prior work that uses this approach to generate synthetic user profiles.

YouTube uses Google's first party profiles. Google allows a user to manage her profile through Ads Settings Manager. This makes profile creation and maintenance straightforward, provides fine control over the profile and its dynamics. Hence, one can choose to collect video ads from YouTube via *crawling with a manual setup of profiles*.

Our Approach to Data Collection YouTube's leading position among video sites makes it a strong candidate to study and understand its advertising, but there are other reasons. In particular, unlike most publishers, the user's profile is accessible and modifiable by the user. In our study, we choose *crawling with the manual setup of profiles*. This allows creation, modification and easy tracking of profiles. Further, ads on YouTube are placed by Google through AdWords, hence Google profiles are also used by advertisers for ad targeting.

Profile and Content Analysis It is not easy to compare *profiles* to *videos*. YouTube provides the following language to perform such operation:

[8] http://exelate.com/privacy/opt-in-opt-out/

- *Verticals* describe content. Vertical is the term used by AdWords to refer to content categories. Using verticals, advertisers can target users based on what they are watching.
- *Interests* describe the user's subject concerns (*e.g.,*the user likes *fishing* and *country music*). Using interests, the advertiser can target relevant users, regardless of what they are currently watching.

The hierarchies of verticals and interests are almost identical. However, some vertical categories are not used as interest categories, possibly because they are considered sensitive. For example, religion, ethnic and some health categories are never assigned to users. A user or video may be assigned a category at the hierarchy's leaves, which are the most specific categories, or any of the interior nodes, which increase in generality as they get closer to the root node. For example, a user may be assigned the highly specific "Shopping/Apparel/Clothing Accessories/Gems & Jewelry/Rings" interest, or the general "Shopping" interest.

In the interests portion of the profile manager (Figure 2.3(a)), each interest is marked as "YouTube", "Websites" or both. This marking is intended to indicate the source of user activity that caused the interest to be added to the profile. However, we have observed that in many cases, visiting a YouTube video page may cause Websites interests to be added, but not YouTube interests. If a user manually adds an interest to the profile manager, it will be marked as both YouTube and Websites.

These findings make YouTube a strong candidate for a case study of video ads delivery function f_t. It is one of the biggest platforms serving ads, it uses first party profiles and gives access to them. But most importantly, it provides the language of verticals and interests that eliminates discrepancies from automated category assignments to videos and ads. While the study could be extended to other platforms (*e.g.,*`on.aol.com`), we did not find any other candidates that would allow the level of transparency into the platform like YouTube. For instance, in case of `on.aol.com` one would: (1) create user profiles through browsing; (2) check profiles on third parties (w.r.t AOL) *e.g.,*Google; (3) harvest ads; (4) analyze ad content and map it to profiles acquired. These steps can potentially introduce a lot of noise and discrepancies into the dataset, *e.g.,*profile of Google may not match the profile of AOL.

3.3 Experiments and Analysis

3.3.1 Display Ad Targeting

Data Collection

Selecting W and P. For an initial pool of websites W, we made a selection of popular websites from Alexa[9]. We took the top 1500, removed non-English websites, websites with adult content and sites that contain no ads. The result was 314 websites.

We used Google's advertising interests tree as a basis for creating the initial pool of profiles P. We chose it because Google is the leader in display advertising, and has the largest user interest category tree that we could find. To diversify the pool, we picked all second level interests (251). Furthermore, to ensure that P included interests popular on the Internet, we proceeded as follows: (1) select the top 1,000 websites from Alexa that are part of Google Display Network; (2) for each website in the list, we used Google Ad Planner [10] to get the 10 most popular interests that were present in profiles of users who visit the website; (3) we formed the list of interests that were captured, and then added all third level interest categories found in the list to P. The result of this process was 340 interest categories, and a total of 340 profiles are created corresponding to each of them.

The resulting focus set was 1959 distinct pairs. These pairs contain 180 distinct websites and 340 distinct profiles *i.e.,* we selected *all* $p \in P$.

Dataset. Finally, we used our crawler to collect data on the focus set. Data was collected over a two day period from 10/1/2013 to 10/3/2013. This data collection produced 875,209 impressions and 175,495 distinct ads. We observed ads from 3,700 advertisers served using 106 distinct ad servers. With this data set, we believe that we can extract broad attributes of the Adscape.

Advertiser Rank Analysis. First, we seek to understand the dynamics of impression value over time. Specifically, it may be the case that the value of an impression goes down as w is visited repeatedly with a single p. To test this hypothesis we have considered the average rank of the advertisers (acquired from Alexa) and the sequential number of the visit in the strategy.

[9] http://www.alexa.com/topsites

[10] https://www.google.com/adplanner/

We used only ads acquired by long strategies for this analysis. 58 of total 3700 advertisers do not have rank information available. The average global rank for all available advertisers is 900K. For each visit we compute the average rank of advertisers present across all pairs. Interestingly, Figure 3.6 shows that for pairs with non-empty p, the rank of the advertisers drops significantly over the course of visits. If w is visited with an empty profile, the average rank of the advertiser stays virtually constant.

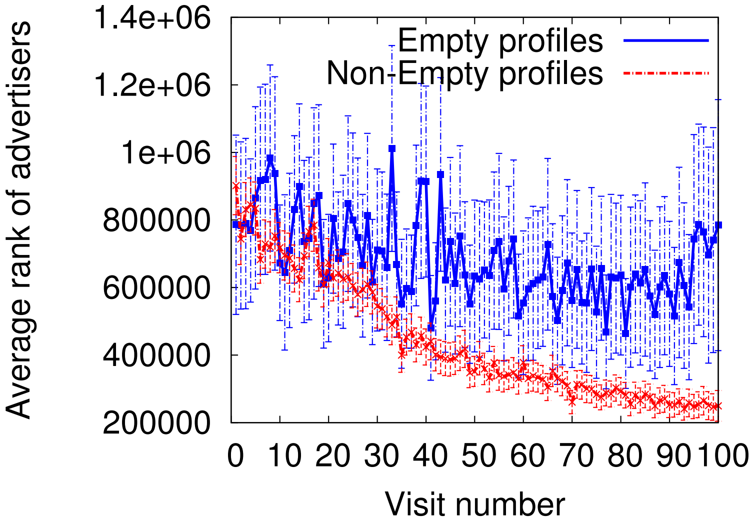


Figure 3.6: Average traffic rank of advertisers, with 95% confidence interval.

Counter-intuitively, we see that the average rank of advertisers is decreasing, which implies that the advertisers shown in the later stage of the 100 visits are more popular. In contrast, empty profiles do not see this trend of getting ads from more popular advertisers. Instead, those advertisers' ranks remain roughly constant over 100 visits.

As discussed earlier, profiles play an important role in our experimental setup. Here we analyze the relationship between profile interest categories and ads that are served.

Defining Targeted Ads. Intuitively, a targeted ad is shown to some profiles more frequently than to others. We use this intuition to define targeted ads. For each ad a shown on a fixed website w, we count how many times it was shown to each of the profiles. This gives us empirical distribution $g_a(p)$ of ad a over profiles p. If ad a is not targeted based on user's profiles, it is fair to assume that the observed distribution g_a should be close to uniform. To compare g_a and uniform distribution we use Pearson's χ^2 test. We say that ad a is *targeted*

Websites	Profile	Gender	Age
miamiherald.typepad.com	0.96	0.73	0.92
tech2.in.com	0.92	0.61	0.84
chicago.cbslocal.com	0.86	0.68	0.70
moneycontrol.com	0.86	0.71	0.80
goal.com	0.83	0.67	0.78
xda-developers.com	0.83	0.42	0.66
community.babycenter.com	0.83	0.36	0.70
celebritybabies.people.com	0.72	0.30	0.49
icanhas.cheezburger.com	0.71	0.26	0.34
women.webmd.com	0.71	0.38	0.49
lindaikeji.blogspot.com	0.69	0.48	0.52
ph.nba.com	0.61	0.34	0.36
shechive.files.wordpress.com	0.58	0.49	0.51

Table 3.1: Ratio of ads shown on websites whose distribution across the property does not follow a uniform distribution with statistical significance.

if the resulting p-value is less than 0.05. Note that for some ads our sample size is too small, which makes it impossible to reject the hypothesis of uniform distribution, even if g_a is not drawn from the uniform distribution. Therefore, we might get false negatives *i.e.,*ads that are targeted, but we falsely classify them as non-targeted.

Measuring Targeted Ads. We calculate the fraction of targeted ads for each of the websites that are used in sufficient number of pairs (in our case, 10). Results are shown in Table 3.1, column *Profile*. The table shows that 50% of the analyzed websites have at least 80% of their ad inventory targeted at profiles. This supports the observation that using a single profile to collect ads from many websites will not necessarily lead to a larger set of distinct ads.

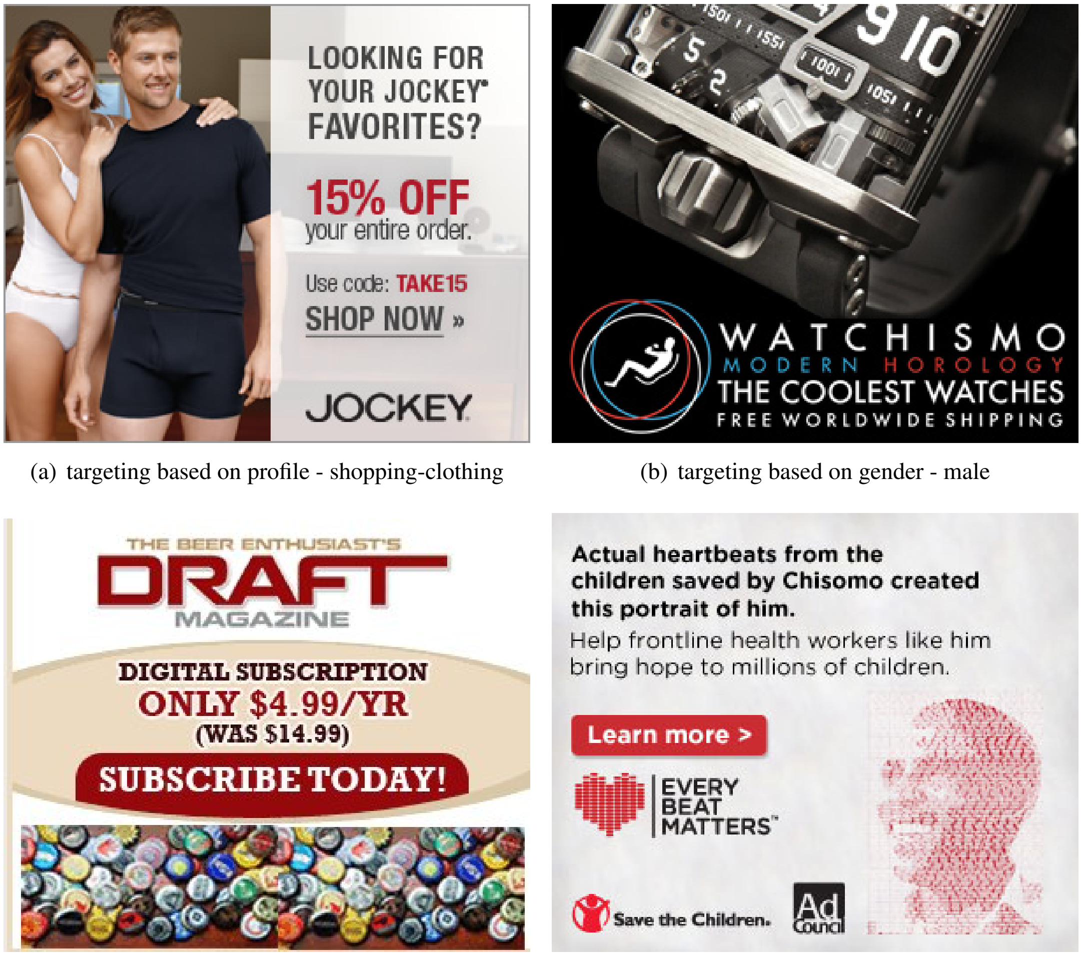
Demographic Targeting. While our profiles are based on interest categories, some online services have also attributed demographic information to some of our profiles. For example, Google's ads settings associates our "Pets" profile with a *Female* from the 25-34 age group. We fetch *gender* and *age groups* attribute values from profiles, and similar to interest targeting, we calculate for each website the fraction of ads targeted at age and gender. We excluded profiles that were not attributed, or attributed inconsistently, with demographic attributes. We considered only websites that were used in at least 10 pairs.

Since there are only 5 age groups and 2 gender groups, our sample sizes per group are larger than in distributions per interest analysis. Therefore, we classify ads into targeted/non-targeted groups with more confidence than in previous analysis. Table 3.1 columns *Gender* and *Age* show that both age and gender are highly targeted attributes. About half of the considered websites have more than 50% of their ads targeted by gender. There is a similar targeting percentage by age. In summary, websites that show a high level of targeting based on user interest profiles also show a high level of targeting based on gender and age group. Note, that here we quantify the portions of ads observed to bias towards some groups of age and gender. We do not attempt to disentangle demographic-based targeting from interest-based targeting. We leave that as a task for future work.

Examples of Targeted Ads. Figure 3.7 shows manually chosen ad creatives for which we observed highly targeted behavior. Figure 3.7(a) shows an ad for clothing brand Jockey. We observed the ad on websites `moneycontrol.com` and `shechive.files.wordpress.com`. On both of the websites, the ad was served exclusively to profile `shopping-clothing`. It is interesting to see that the ad is offering customers 15% off the entire order if they click on it.

Figure 3.7(b) shows an ad creative targeted based on gender. It is advertising Watchismo, online watch store. Website `news.com.au` served an ad total of 89 times to profiles with male gender attribution and 0 times to female profiles. Figure 3.7(c) shows an example of ad creative targeted based on age group. It is advertising Draft, a magazine for beer enthusiasts. The ad was shown 27 times on `moneycontrol.com`, exclusively to profiles attributed to age group 25-34 years. Figure 3.7(d) shows an example of ad creative that seems not to be targeted based on our profiles. It is advertising charity campaign called Every Beat Matters. The creative appeared on `preview.tinyurl.com` total of 90 times.

Advertiser Analysis As noted above, during our data collection we acquired ads from 3700 distinct advertisers. We associate a distinct advertiser with the distinct domain of the landing page of an ad. We observe that more than 80% of advertisers had no more than 100 ad impressions in our entire data corpus. The top advertiser is `https://www.lmbinsurance.com`, which had a total of 94,437 impressions or about 10% of all ad impressions. One of the websites we used in our crawling process shows consistently 2 ads linking to it on almost every visit by all profiles. One possible explanation is, that `https://www.lmbinsurance.com` had



(a) targeting based on profile - shopping-clothing

(b) targeting based on gender - male

(c) targeting based on age - 25-34 age group

(d) not targeted based on profile or demographic

Figure 3.7: Examples of ad creatives with different targeting strategies.

an agreement or contract with that website. This phenomenon of a huge number of impressions from a single advertiser was not captured in while selecting the focus set, and cases like this make the design of ad collection even more difficult.

Advertiser Categories. To find a category for each advertiser, we appeal to multiple sources, since we did not find an authoritative source that would assign a category to all of the advertisers in our data set. We used site categorization services from Alexa [11] and WebPulse [12] from Blue Coat (which offers a web content filtering product). These two sources have

[11] http://www.alexa.com/topsites/category

[12] http://sitereview.bluecoat.com/sitereview.jsp

Alexa	WebPulse	Mapped
Business/Automotive	Vehicles	Autos
Business/Hospitality/ Restaurant Chains	Restaurants	Restaurants
Business/Investing	Brokerage	Investing
Reference/Education/ Distance Learning/Online Courses	Education	Education
Business/Real Estate	Real Estate	Real Estate
Computers/Internet/ On the Web/Online Communities/ Social Networking	Social Networking	Social Networking
Recreation/Travel	Travel	Travel

Table 3.2: Sample category mappings for Alexa and WebPulse.

different sets of categories for labeling sites, but most of the categories differences can be resolved manually. Table 3.2 shows some samples of the category mappings made for Alexa and WebPulse.

We categorized 1480 of 3700 advertisers using Alexa and used WebPulse for the rest. Categorized impressions fall into more than 550 different detailed categories. Figure 3.8 shows ad distribution for the top 20 root level categories. The top advertiser category is *Financial Services*, which also contains the top advertiser. Some other broad categories are *Shopping*, *Computers*, Business, *Arts & Entertainment* and *Education*. We have also collected ads from 17 advertisers that have been categorized as *Spam*, which implicates ad crawling may help in online security research.

3.3.2 Video Ad Targeting

Data Collection

There are millions of videos on YouTube, and one can construct thousands of distinct user profiles. To keep data collection feasible, we have to limit the number of both videos and profiles in use.

For our pool of profiles P, in order to cover a diversity of interests while maintaining a small number of profiles, we create profiles corresponding to first-level categories of Google's interest tree (25 in total). We add an *empty* profile that represents a new user that has not yet been assigned any interest categories. In total, $|P| = 26$.

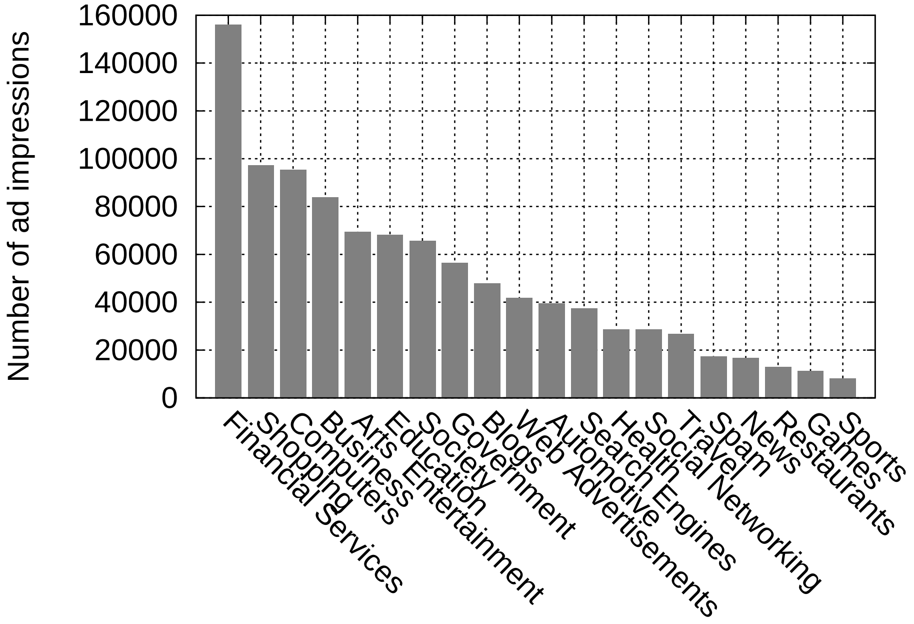


Figure 3.8: Distribution of impressions over categories.

For our pool of videos V, we focus on covering the space of verticals, rather than the space of videos. We use Google's Display Planner[13] to find top 50 channels for each of 25 first level categories. For each identified channel we take the 50 latest videos, resulting in 1250 videos total. We proceed by finding verticals for each of the videos in the list, and building a set cover on verticals. We form V consisting of 101 videos, s.t., (1) V contains maximal number of verticals; (2) most verticals are assigned to at least 3 videos in the cover, and only few are assigned to less than 3.

For data collection we form all possible *pairs* (p, v), s.t., $p \in P$ and $v \in V$. We visit each pair 20 times and stay on each video for 2 minutes, which is sufficient for pre-roll, companion and sponsored ads to load. We create a distinct instance of profile p for each visit. As a result we obtain 52,520 data points in one week from: $|P| \times |V| \times 20$. We refer to each data point as an *instance*.

In our data collection, we have observed 38,758 ad impressions, 23,600 of which are pre-rolls. At the earlier phase of data collection we see high rate of new ads, as the crawling process proceeds the rate of new ads slows down, as we start seeing duplicates. Interestingly, 61% of

[13] https://adwords.google.com/da/DisplayPlanner/home

(p, v)'s observed only one ad in 20 visits, and about half of the ads were collected from only 7 videos. As seen in Figure 3.9, *Reference* and *Science* profiles collected 2 times more ads than profiles *Real-estate* and *Travel*. Furthermore, *empty* profiles collected more ads than most of other profiles.

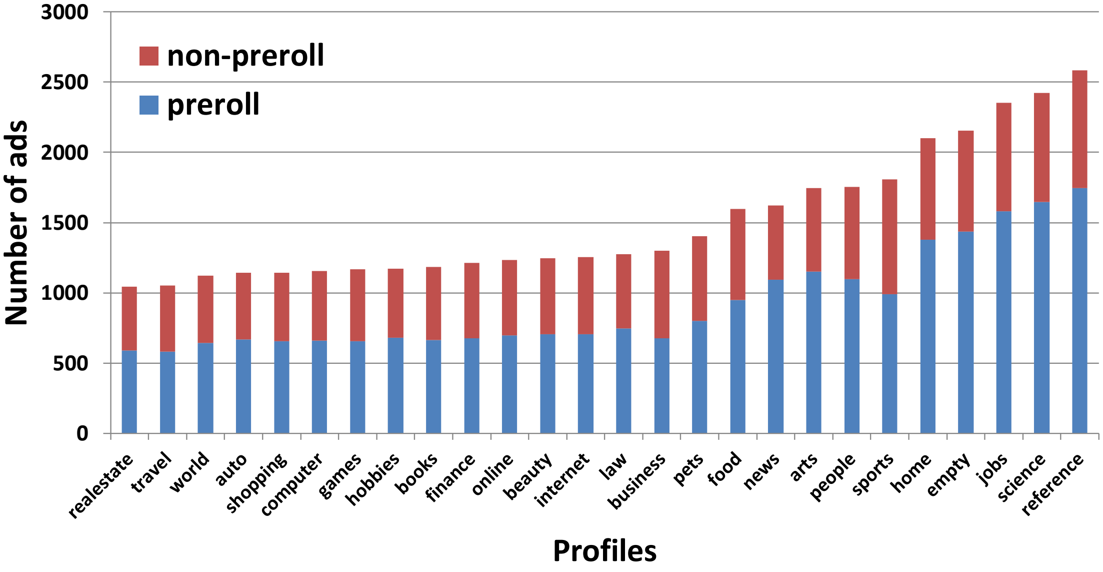


Figure 3.9: Collected ad amount distribution by profile.

The largest number of collected ads are in the category of *Arts & Entertainment* (see Figure 3.10), which can probably be explained by the fact that YouTube is an entertainment platform. However, we observe ads from all of the 1st level categories. For preroll video ads, only 15% of them are under 30 seconds, 46% of them are under 60 seconds, and 90% of them are under 3 minutes. We have also observed few extreme cases with a duration of more than 30 minutes, *e.g.,*documentaries.

Impact of Video Page Visit on User Profile. We have observed that a visit to a single YouTube video page potentially adds many interest categories to a profile. Figure 3.11 shows the distribution of number of added interests for *YouTube* and *Websites* interests individually, and together –*combined*. It is evident that the number of added *YouTube* interests tends to be either 0 or 15. Meanwhile, the number of *Websites* interests tends to be between 0 or 5, although the bimodal behavior is less pronounced than with YouTube interests. Consider distribution of *combined*, there are three peaks at $\approx$ 0, $\approx$ 5 and $\approx$ 20. Its shape is expected based on the distributions of *YouTube* and *Websites* frequencies (although it should be noted that an

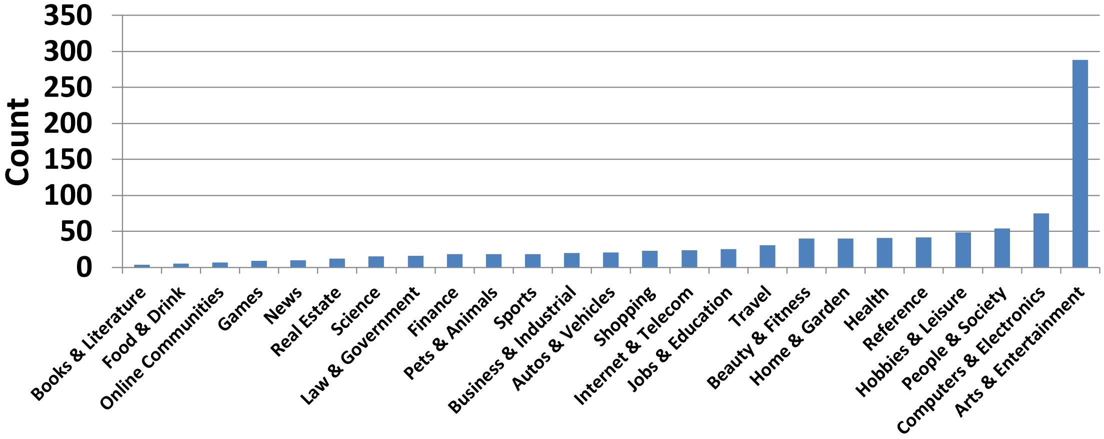


Figure 3.10: Video ads distribution by top level categories.

added interest could be marked both *YouTube* and *Websites*). Thus far, we have been unable to determine why the distributions for *YouTube* and *Websites* interests have such a bimodal behavior. Regardless of the distributions, those big numbers of interests added to make it clear that a single YouTube page visit makes significant changes to the profile.

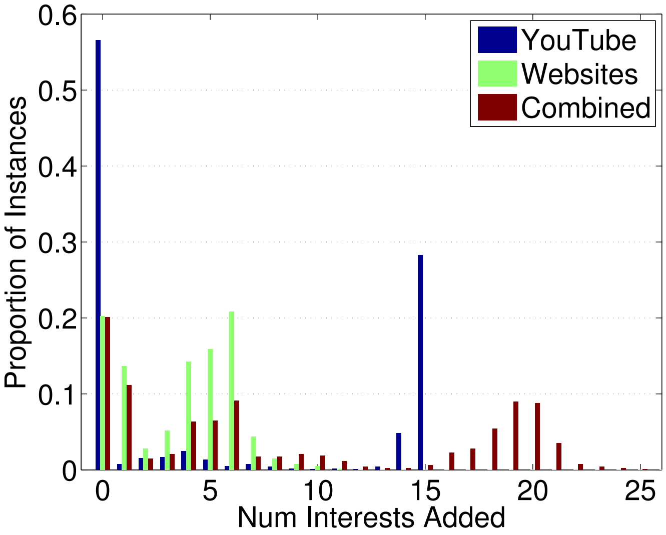


Figure 3.11: Interests added by marked activity.

Just as additions can occur, interests may also be removed when visiting a page. Figure 3.12 depicts the distribution of interests removed. Recall that when interests are added manually, as our Profile Builder does (see Section 3.2.3), interests are marked both *YouTube* and *Websites*. Thus, there is no reason to segregate the types of interests removed. As seen in the figure, in 37,435 of 52,520 (71.3%) instances, zero interests are removed. However, in some cases, a large number (up to 22 interests) can be eliminated. This again supports the observation that single page viewing can have a significant impact on the profile.

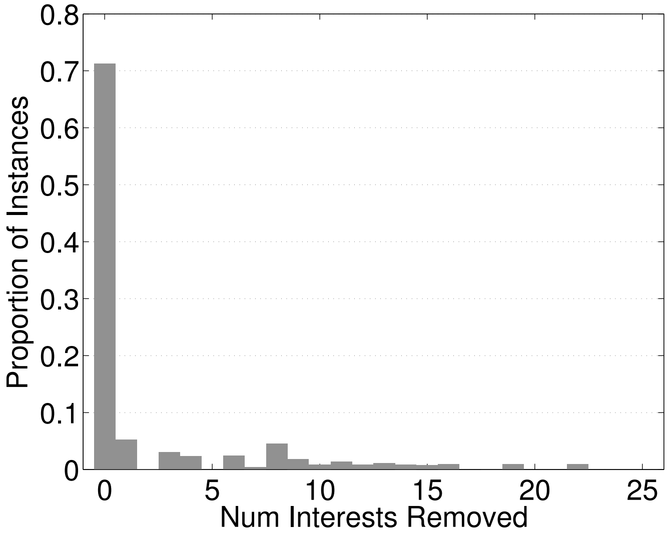


Figure 3.12: Interests removed.

Impact of Video Content on the Profile. We seek to understand whether profile changes can be attributed to the content of the video viewed. To do so, we compare verticals associated with the video viewed and the interest categories added to a profile after viewing. We perform the comparison using the first-level categories of the interest tree. For example, if the user has a third-level interest, such as "Games / Table Games / Billiards", we replace it by its root — "Games". The intuition for this simplification is as follows: a view of very specialized content might not add an exact specific interest, however, if caused by the video itself, it is reasonable to assume that the general "direction of the added category should align with the video.

For all 52,520 crawl instances, we find the overlap (*i.e.,*intersection) between profile added categories and video verticals. To gauge if the impact of the video is significantly different than random, we pair each crawl instance with a randomly selected video from our pool of 101 videos. We perform this pairing over all instances 20 times (20×52,520 = 1,050,400 pairs) and subsequently quantify the *expected* overlap by finding the intersection of added interests and verticals for each of these random pairings. Results are shown in Figure 3.13. A value of 0.4 along the y-axis indicates that 40% of instances (pairings in the case of "Expected") had the corresponding number of interests (as indicated on the x-axis) added to the profile that was also verticals for the video. From the figure, it is evident that the actual overlap is significantly higher than expected. These results suggest that the video content has a significant effect on the profile. At the same time, many added interests do not overlap with video verticals. In fact, the average number of added first-level interests not overlapping the verticals is 3.25, which indicates other factors beyond the video must be influencing the profile.

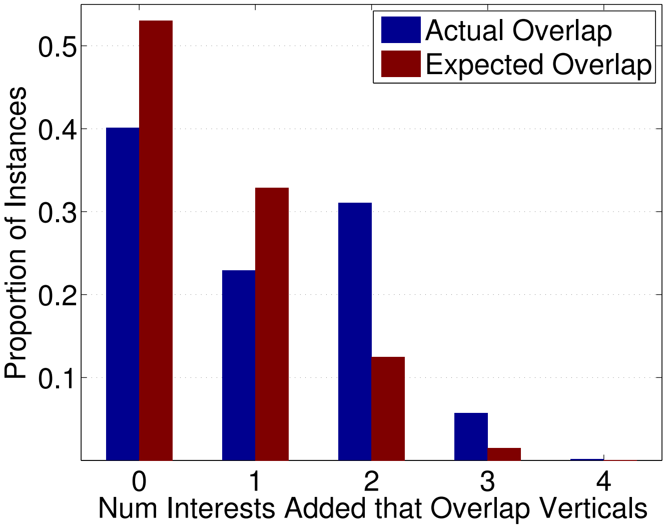


Figure 3.13: Added interests that are explained by video verticals.

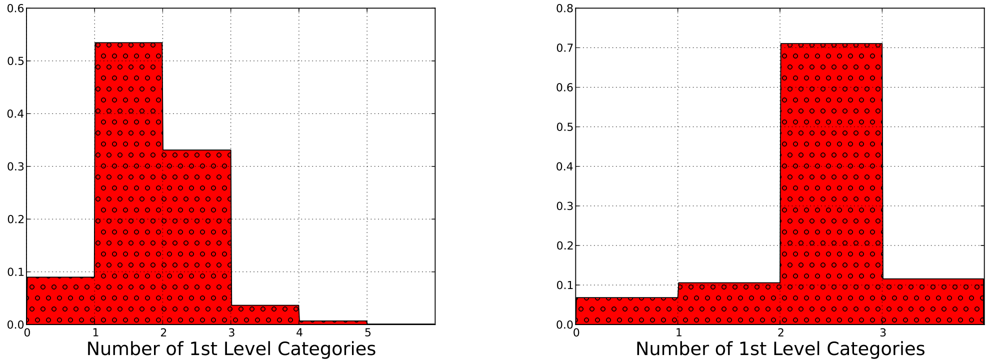
In the following analysis, we want to study the relationship between video ads and user profiles. We consider only instances in which we have observed preroll ads hosted on YouTube. The resulting set consists of 5365 instances of 771 distinct (p, v) pairs. In total, these instances saw 221 distinct YouTube hosted preroll ads. We perform analysis similar to Section 3.3.2: we consider overlaps between: (1) verticals of preroll ads and verticals of videos; and (2) verticals of preroll ads and profile interest categories (before video view).

Video Ads and Profile Interests. Interestingly, we found no instances where the interest overlapped the verticals. Thus, there is no indication that pre-rolls were targeting user profiles. While we did not consider geographic or demographic properties, we find that these results are strong evidence that video ads are targeted at the content rather than the profile. This can be potentially motivated in several distinct ways. For instance, video ads personalization is premature. Alternatively, it may make greater sense for the advertiser to target content because a user that comes to watch a particular video is “in the mood” about some particular topic (*i.e.,*the user’s transitive interest).

Impact of Ads on Profiles. In the previous section, we observe that user profiles can change dramatically after watching even a single video. We also observe that in a significant fraction of instances, we are unable to attribute changes of the profiles to the video's content. In this section we examine the impact of ads viewed on the user's profile.

Below we show two examples from our dataset that illustrate the potential connection between preroll ads and user profile changes.

To quantify the impact of preroll ads to the user profile, we consider instances that were exposed to a YouTube preroll ad and compare the first level verticals of video ads vs. profile interests added. We exclude interests that could have been attributed to the video. Figure 3.14(a) shows the distribution of number of categories in the overlap. Observe that around 90% of instances have at least one first-level interest added that is relevant to the ad, implying preroll ads have a significant effect on the profile.



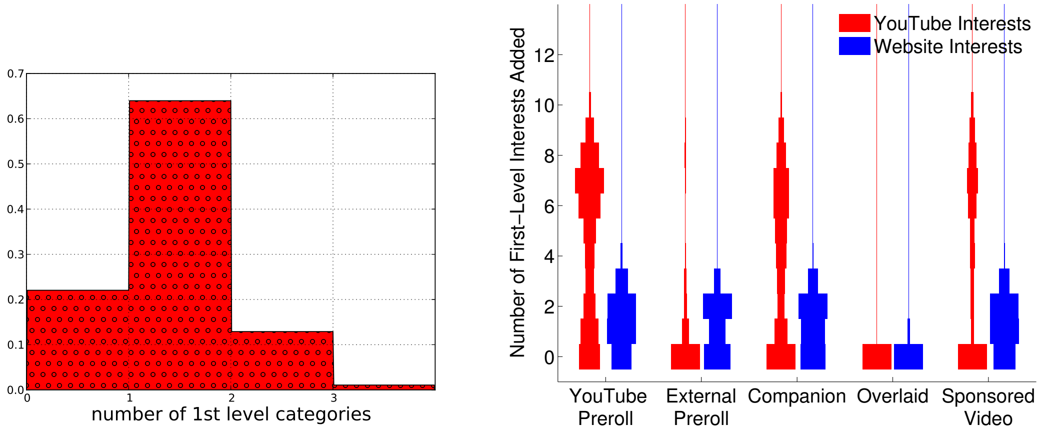
(a) Overlap sizes of preroll ads' categories and profiles' categories

(b) Overlap sizes of videos' categories and ads' categories

Figure 3.14: Impact of YouTube preroll ads to the profile.

Since we have seen that YouTube preroll ads have direct impact on the user's profile, do other ad types also affect the profile? Consider Figure 3.15(b), where violin plots are shown depicting the number of first-level interests added, excluding interests that match the first-level verticals of the videos' interests. For each ad type, a thicker horizontal bar means a higher frequency of instances had the corresponding number of interests added. The bars are normalized relative to the maximum frequency observed in the column. For example, most sponsored video ads add zero YouTube interests to the profile, thus the thickest red horizontal bar for

Sponsored Video ads occurs at zero. Each column in the picture is akin to a normalized histogram with values along the y-axis rather than the x-axis. As seen in the figure, the presence of a YouTube-hosted preroll, companion or sponsored video ad tends to lead to more changes to profile than externally-hosted preroll ads or overlaid ads. In particular, if an overlaid ad was shown, there were rarely any interest additions to the profile at all.



(a) Overlap sizes of sponsored videos' categories and profiles' categories

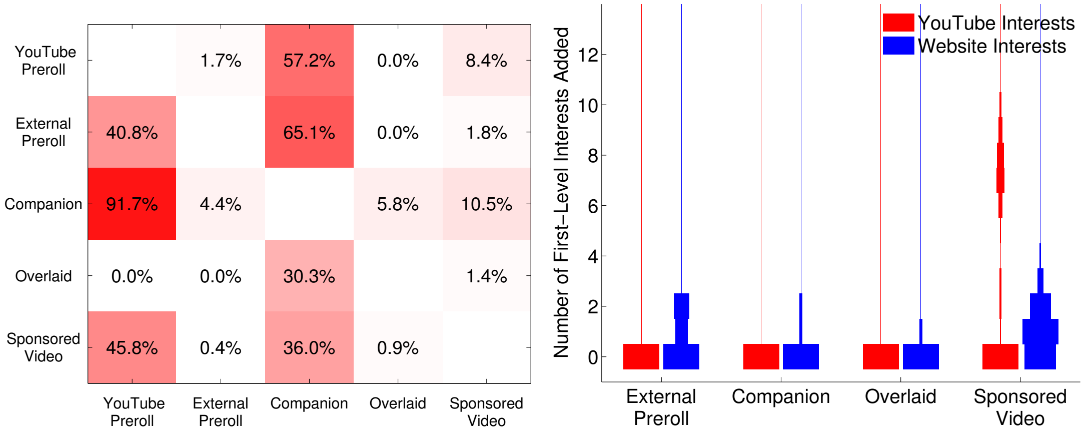
(b) Interests added by ad type

Figure 3.15: Impact of YouTube sponsored ads to the profile.

Figure 3.16(a) provides a heat map of the co-occurrences of ad types. The values shown in the cells are the proportion of instances with the row-labeled ad type that also contained the column-labeled ad type. For example, 8.4% of instances with a YouTube-hosted preroll ad were also shown a sponsored video ad. An important observation from this figure is that overlaid ads were never shown when preroll ads (both YouTube- and Externally-hosted) were shown. This provides support for the central thesis that "you are what ads you watch", since from Figure 3.15(b) preroll ads clearly affect the profile, while overlaid ads do not.

In Figure 3.16(a), it can also be seen that most companion ads are shown with a YouTube-hosted preroll ad. In fact, we have found that most companion ads are linked in the advertising message to the preroll ad (when YouTube-hosted). Thus, a large number of added interests for companion ads shown in Figure 3.15(b) can be attributed to the YouTube-hosted preroll ads. In fact, consider Figure 3.16(b), which has similar violin plots to Figure 3.15(b) except instances with YouTube-hosted preroll ads are excluded. We can see that the number of YouTube interests added for all ad types has almost reduced to zero, except in the case of sponsored videos.

Website interests are also significantly fewer for externally-hosted preroll and companion ads, although many instances are still above zero. These observations imply that presence of a YouTube-hosted preroll ad and, to a lesser extent, sponsored video ads are associated with the greatest impact on the user profile.



(a) Ad type co-occurrence heat map

(b) Interests added by ad type for instances without YouTube-hosted preroll ads

Figure 3.16: Impact of YouTube ads to the profile.

To further consider the impact of sponsored video ads, we perform a similar analysis to the YouTube-hosted preroll ads at the beginning of this section. We collect verticals for all sponsored videos and examine the number of first-level interests added to the profile that matches the first-level verticals of the sponsored videos but does not also match the content of the page. As shown in Figure 3.15(a), in 78% of the instances, at least, one additional interest was added that overlaps the sponsored video's verticals, but was not associated with the page's video (content). This further indicates that sponsored video ads significantly affect the profile.

In summary, the observed patterns suggest that YouTube is more aggressive in assigning interests to user profiles sourced by its own content, such as YouTube-hosted preroll ads, sponsored video ads, and video content. External content, such as externally-hosted preroll ads and overlay ads appear to play a less important role.

3.3.3 Polymorphic Videos and User Interests

There are YouTube videos that cover the same content but are uploaded multiple times by different users. We call these *polymorphic* videos. For example, if one searches for "*Charlie*

Bit Me” on YouTube, there are various versions of the video available.

Given the fact that polymorphic videos are covering the same event, we want to see whether (1) ads served are also similar and (2) similar ads affect user profiles in a similar manner.

Data collection. Even for a given video, polymorphic videos are not easy to discover automatically. For example, on the first page of search results for “*Charlie Bit Me*” there are several non-polymorphic videos. Automation of collecting polymorphic videos is out of the scope of this paper. Instead, we manually compose a polymorphic video set. We take 15 titles of the most popular (or viral) videos on YouTube. To achieve video category diversity, we choose viral videos from topics in Sports, Music, Comedy, and Education. For each video we find 10 polymorphic videos from search results on YouTube. The titles of the popular videos we selected are: 1. *Justin Bieber - Baby.* 2. *Telekinetic Coffee Shop Surprise.* 3. *Charlie bit my finger.* 4. *Diet Coke and Mentos Experiment.* 5. *A Letter From Fred.* 6. *Gangnam Style.* 7. *How Animals Eat Their Food.* 8. *THE NFL : A Bad Lip Reading.* 9. *Pitbull - Timber ft. Ke$ha.* 10. *Randy Pausch’s Last Lecture.* 11. *The Sneezing Baby Panda.* 12. *What Does The Fox Say.* 13. *Miley Cyrus - Wrecking Ball.* 14. *Barack Obama: Yes We Can.* 15. *Zidane vs. Materazzi.*

We use *empty* profiles to find out how polymorphic videos affect user profiles. For each of the 150 videos, we conducted 100 crawl instances. Out of these 15,000 total instances, 6,619 had ads, 862 of which are distinct.

Similarity of polymorphic videos. Polymorphic videos have virtually the same content, possibly with slight modifications by individual uploaders. It is natural to expect, that as the result they get similar verticals assigned. However, only 7 of 15 polymorphic video sets have at least one common vertical in all 10 of that set’s videos. Across the video sets, the average percentage of verticals that are common to all videos in the set is 18%. These results indicate that, either polymorphic videos do not have identical content from Google’s perspective, or verticals are not assigned based on the content only. Yet the question remains how similar the videos are compared to arbitrarily chosen videos.

We represent each video by the verticals associated with it, and compute its average Jaccard similarity with other videos in the same set (intra similarity), and average similarity with videos in other sets (inter similarity). The average intra- to inter-similarity ratio for all videos is $\approx$ 2.2. Therefore, videos in the same set are much more similar than videos across sets. So, while Google does not find their content identical, it does determine that they have substantial similarity.

Ads in Polymorphic Videos. In previous section, we have established that ads and videos often have verticals in common. In the previous section, we have also established that polymorphic videos have significant overlap in their verticals. However, will the user see ads of the same advertiser on different versions of the video?

Let advertiser be the uploader of the ad video. We found that for all sets there are no more than 2 advertisers (about 2% of all advertisers for a video set) that appear on all videos in a polymorphic set. One advertiser (Volkswagen) has ads on all 150 videos, which implies the advertiser's targeting is generic.

The overlap of advertisers among polymorphic videos is small. However, do polymorphic videos serve video ads with similar verticals? Looking at the first-level verticals associated with the preroll and sponsored video ads, we find that $Autos\&Vehicles$ and $Entertainment$ are the most common ad category across all 15 polymorphic video sets. Although most of the polymorphic video sets do not have a specific common vertical in their ads, interestingly all of "Randy Pausch - The Last Lecture" polymorphic videos have $Science\&Techonology$ ads.

We have also observed that polymorphic videos have different frequency of sponsored video or preroll ads. Our investigation did not find evidence that higher view counts or higher numbers of channel subscribers necessarily guarantee higher frequency of ads. The exact causes of different frequencies remain unknown.

Impact on Profiles. To check whether profiles that viewed polymorphic videos change in a similar way, we follow the paradigm of Section 3.3.2 and examine the similarity in first-level interests added to profiles. For 9 of the 15 video sets, only one interest was commonly added ($Arts\&Entertainment$). For the remaining 6 video sets, a set of 9 interests were universally added to all profiles by each of those videos (a surprisingly high level of homogeneity).

To quantify the similarity of the resulting profiles, we compute the following three Jaccard indexes considering only crawls that did not witness ads:

- *(Same-Video.)* We compute Jaccard index for each video, over 100 instances of crawls performed. We find that the profile is not deterministic (even with no ads displayed).

- *(Intra-group.)* To capture the similarity across the 10 videos of a polymorphic group, we compute the Jaccard index averaged of all pairs of crawls in the polymorphic group, excluding pairs of crawls of the same video.

- *(Inter-group.)* As a baseline examining videos without a polymorphic bond, we compute the Jaccard index over all pairs of crawls, s.t., videos in the pair are from different sets. We then average over all pairs for the given polymorphic set.

The results are presented in Figure 3.17. As expected, profiles' changes from different crawling instances of the same video have the highest similarity, followed by similarity of profiles within the same group, and finally across different groups. In a paired t-test, the difference between same video and intra-group profiles is statistically significant ($p = 0.0002$). Likewise, the difference between intra-group and inter-group profiles is also significant ($p = 0.002$). As results indicate, watching polymorphic videos do not result in the same level of profile similarity as do watching the same video. So, if two users with similar profiles are interested in the same video content but watch different versions of it, they are more likely to have a larger difference in profiles afterwards.

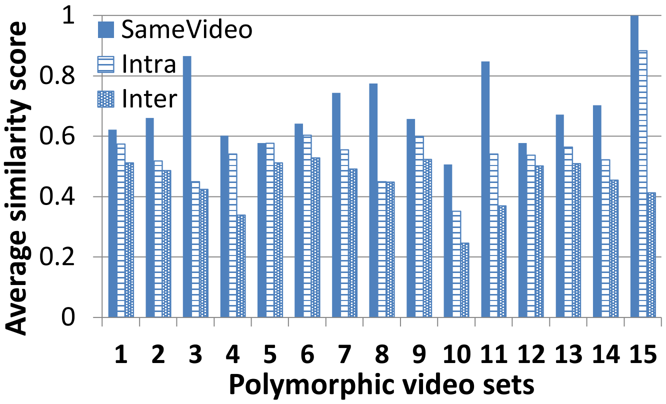


Figure 3.17: Similarities of profiles from crawls without ads, for same video, intra and inter polymorphic video sets.

The presence of the ad increases the difference between resulting profiles. The comparison of profile intra-group similarity is shown in Figure 3.18. For nearly all videos, witnessing ads resulted in less similarity in the profile. This result is significant under a paired t-test ($p = 0.007$).

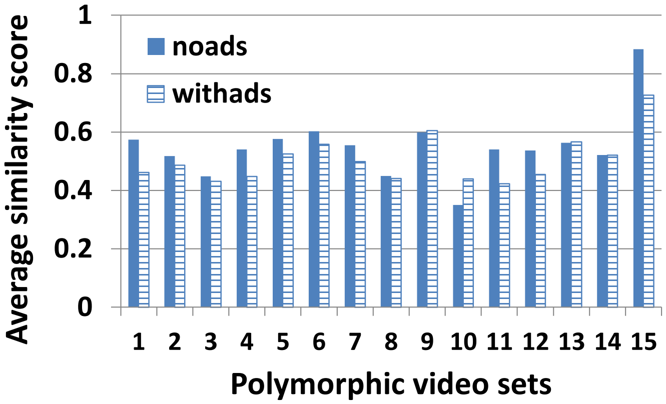


Figure 3.18: Profile similarity comparison for 15 sets of polymorphic videos.

3.4 Discussion

In our study, user interest profiles were simulated by automatic crawling web pages of specific topics. The huge advantage of such method is its scalability for collecting large data. While the main limitations are two folds: (1). it is hard to mimic real user's browsing behaviors, whose web page dwell time, page visit time of day, location are all possible factors that impact the user interest profiles produced by the companies; (2). the analysis of user interest profiles were based on the public information of web page, such as web page categories, and based on the information disclosed by the companies. This depends on the portions of the user profile information the companies willing to share with users being tracked.

The developped crawling infrastructure was used to gather over 175K distinct ads and associated data (landing pages, creatives, etc.) from 180 large English language web sites that run display ads[14]. We use 340 different user profiles in our crawling experiments. We make no claims that this sampling of web sites shows *all* available (English) display ads since local advertisers often focus on smaller, local sites. We employed different crawling strategies to assemble a data corpus that is extensive enough to expose key characteristics of the Adscape and provide insights on the most commonly-used targeting mechanisms.

When analyzing our data, we first consider the general targeting mechanisms that are used on different sites. Not surprisingly, we find that the majority of sites do indeed use targeting

[14] While we restrict our focus to English language web sites for this study, our methodology and tools can be more generally applied.

mechanisms on over 80% of their ad inventory. However, many sites show a substantial number of ads to all users regardless of profile. We next consider the Marketers who are engaged in online display advertising. Our analysis shows over 3.7K distinct Marketers from diverse business segments such as shopping, computer sales and financial services. Third, we drill down on the details of the ads themselves. Our analysis reveals that (i) interest-based targeting, which attempts to ensure that ad types shown generally align with the customer's interest profile characteristics, is widespread, and (ii) age and gender-based targeting is also widely used.

We choose to focus on YouTube, and use our user profile-aware crawler in a series of collection experiments conducted over a one week period. We configured the crawler with 26 distinct user profiles and target 251 videos with diverse content. The result of our efforts was a corpus of 2,290 unique video ads, and the profile data from the individual crawlers used. Some of our findings are as follows.

- Video watched affects user profile significantly: a visit to a single video can add several — in some cases up to 25 — new interest categories to a user profile. However, the connection between video content and specific changes in profiles is not always evident. There is significant overlap between video and ad contents.
- Our analyses show that video ads usually target the video content on the page, rather than targeting video viewer's profile interests.
- Interestingly, we also find that not only video content that is viewed but the *video ads* that are shown to users can also affect user profiles.
- YouTube has polymorphic content. That is, many videos are identical or differing little from each other. We find that different versions of the content can have a very different impact to profiles, and the difference is larger under the impact of video ads displayed.

Our empirical results imply that the mechanisms used by video ad platforms for user profile maintenance are complex and careful consideration of this behavior must be made in developing economic models of this ad ecosystem which is quite distinct from other online ads like display or sponsored search and also different from traditional video ads like in TV. Further, if we seek to understand the details of the ecosystem this is critical to understand how users' video viewing behavior online ends up impacting the ads they are shown.

3.5 Related work

Our ad crawling capability is most directly related to standard *web crawling*, which is widely used to gather content for search engines and a host of other applications. Early web crawlers emerged nearly two decades ago including WebCrawler [83], World Wide Web Worm [70] and RBSE [37]. Googlebot [15] and Bingbot [16] are two of the most prominent examples of modern web crawlers. The on-going challenges in content crawling include the ever-increasing number of websites, increasing use of dynamic content, and the tension between crawling frequency for information freshness and demand on Internet resources. Examples of studies that consider these problems include [20, 25, 29, 78].

Pandey and Olsten consider "user centric" web crawling in [80]. Their focus is on scheduling web crawls to specific pages in order to maintain the most up to date versions in search engine repositories. At the highest level, elements of our ad crawling system have similar objectives. More recently, Liu *et al.* consider the problem of using hints from user browser histories to organize URL lists for crawling [63]. We are aware of no prior work that builds and employs user profiles for ad crawling in the way that we do.

The task of crawling ads differs significantly from web crawling in a number of ways. These include but are not limited to the fact that *(i)* different ads can be shown to a user on each page reload, *(ii)* ads are delivered based on information beyond page context, *(iii)* display ads must be differentiated from other visual elements on a page. We are not aware of any work describing methods for crawling display ads. However, there are number of plugins that allow users to filter out display ads. AdBlock [17] is one of most widely used.

Our study on YouTube video ads and user profiles is most directly related to ad targeting, ad crawling and users' online advertising profiles. In [62] the authors performed large scale display ads data collection by developing a browser plug-in, their results showed that most of the ad categories are behaviorally targeted. From economics perspective, [39] studied companies advertising revenues as the function of user information, they showed that most of the

[15] http://support.google.com/webmasters/bin/answer.py-?hl=en&answer=182072

[16] http://www.bing.com/blogs/site_blogs/b/webmaster/-archive/2010/09/03/bingbot-is-coming-to-town.aspx

[17] http://adblockplus.org

advertising revenue was contributed by a small portion of users. In our study we found that video ads are mostly targeted by context in YouTube video ads market.

Shortly after YouTube got popular, studies were conducted to understand its system infrastructure [12], viewer characteristics and video popularity. [32, 30] crawled YouTube videos at large scale and found that YouTube videos are different from traditional streaming videos from length to life span. In [97], authors analyzed network traces collected by monitoring the traffic between YouTube and a university network, they found that global and local video popularities are uncorrelated, and alternative infrastructures were suggested based on the observations. As the popularity of mobile devices emerge, [87] studied YouTube video format and downloading mechanisms on mobile devices. [38] conducted a study on traffic generated from mobile devices and PCs, authors found that the YouTube access patterns are similar regardless of locations and user devices. Moreover their results revealed that more than the half of the video playback was abandoned at 20% of the whole duration. This observation aligns with our conclusion, that in order to observe all the changes of the profile it suffices to stay only a fraction of the video duration on the page.

There are multiple prior works on understanding YouTube video and user behaviors. Based on a large YouTube trace data set, [28] studied YouTube video popularity properties and viewers' focus evolution and provided insights into user behavior and video popularity distribution. [36] characterized YouTube usage from perspective of video and user characteristics in different geographical regions. We find, that frequency of ads does not depend on number of views of the video.

Content duplication phenomenon has been studied for a long time. [28] discovered the content duplication phenomenon on YouTube in their video popularity analysis, [93] analyzed YouTube search results and found a large amount of duplicates (27%) in popular search results. [82] analyzed video content to detect duplication and content overlap in YouTube, and have discovered significant rate of duplication (16%) in their test video collection. We have explored which categories get assigned to polymorphic videos, their similarity, as well as their impact to user profiles.

Several studies consider the problems associated with privacy and online advertising. Castelluccia *et al.* [27] demonstrated that one can reverse engineer users' profiles by looking at targeted ads displayed to her and making inferences about the target interests revealed in ads. In their work, they focused on root level categories of tree of profile interests that can be found on `www.google.com/ads/preferences/`. Roesner *et al.* [89] present a taxonomy of different trackers *i.e.,* in-site, cross site, cookie sharing, and social media trackers. In experiments, authors simulated users using AOL query search logs, and considered how prevalent tracking is in this dataset and propose an extension that helps to protect user privacy. Finally, the study by Guha *et al.* [42], which described challenges in measuring online advertising systems, informs our work. However, their primary focus is on privacy issues, while ours is on broader Adscape characterization.

3.6 Conclusion

User interest profile is the common language that companies use to represent users tracked by them and let advertisers find the right audience they want to target to show their ads.

In the efforts to find more user signals to infer their interests, companies try to understand each webpage visited by the user in great detail, so that each user's profile could learned and users are distinguishable by their detailed profiles. Inevitably, some noisy signals from user activity pages are also introduced in this process. It is still a non-trivial challenge in both research and industry, how to identify the underlying real user interests, even if the contents are similar by nature but with some differences in representations, and how to model the dynamics of users interests.

We surveyed the current state of art practices how online users are represented in the advertising industry. It is closed systems how companies store user activity information and produce interest profiles for them. In order study and observe those closed systems, we proposed a framework for creating synthetic user interest profiles by topic controlled web crawling, including regular web page browsing and video watching, to study the correlation between user online activities and the dynamics of user interest profiles in different companies.

Chapter 4

User Modeling on Mobile

Mobile device usage is a significant part of people's daily life. In 2014, comScore [9] reported that 60% of American digital media consumption time was from mobile devices. According to Flurry Analytics, in 2014, overall mobile application (app) usage grew by 76% [7], and there is little change of the app download rate per user per month since 2011 [2]; consistently, Americans average about 9 app downloads per month. The enormous number of app installs drives mobile app-install ad business. According to BI Intelligence [5] the US mobile app-install ad revenue will top 4.6 billion in 2015, and reach 6.8 billion by the end of 2019. All these motivate efforts to understand user interests and needs in mobile app usage and improve the way users discover useful apps.

Some time ago, website browsing was the primary mode of accessing online content. There was progress in understanding how users consume online content and how to use their historical patterns to predict future behavior through machine learning techniques[94, 31, 18, 71] Given behavior targeting is a norm in the online advertising industry, [22, 75] took third party's view to design and conducted careful experiments to measure the extent of behavioral targeting in online display ads.

Just as website visits reveal a user's online content consumption interests, user activities on mobile devices are represented by the mobile apps they install and use and the app sessions generated over time. According to a Nielsen report [8], in 2013, American users used on average about 27 apps per month. Therefore, it is critical to find out what (new) mobile applications are useful to a user. One can leverage users' mobile app interests to improve the mobile application discovery mechanisms in markets or target audiences in advertising campaigns.

In this chapter, we present a first-of-its-kind study of modeling mobile apps contextual similarity based on a large amount of users' app usage activities. We present an `app2vec` model [65] to vectorize apps using the app usage context, and we demonstrate the effectiveness of thus learned app vectors through applications.

Our contributions are summarized below:

- *(Vectorization App Usage)* Based on the usage history of apps across a huge sample user set from Yahoo!, we propose a model `app2vec` to represent apps in a vector space without *a priori* knowledge of their semantics. In contrast to the `word2vec` model on documents [10], we weight the apps in a training context by the usage time interval to the target app. We show that vector operations on the `app2vec` representations capture semantic relationships between apps, similar to how `word2vec` vectors capture linguistic relationships among words and phrases [74]. Further, we demonstrate a simple application of `app2vec` in user targeting.

- *(Quantitative Evaluation)* A common challenge for the word2vec model and newly developed models on top of `word2vec` is evaluating the quality of the word similarities using the learned word latent vectors. In our study, we conducted large-scale editorial evaluations on the quality of learned app similarities to compare several models including bag-of-words, matrix factorization, and `app2vec`. We also share the insights about the challenges and issues of each model in comparison, and we believe they can help the app ad research community to understand the results from `word2vec` framework better.

- *(Application 1)* In online advertising systems, there are usually hundreds of thousands of eligible ads for a given impression, and an ad serving model has to go through each eligible ad to calculate the probability of click or conversion to make ad selection. We propose using a method based on a *determinantal point process (`DPP`)* to pre-process the large set of eligible ads to generate a smaller set of important and diverse ads (what we call the *ad consideration set*) that can then be fed into an ad selection algorithm. A `DPP` is a distribution on subsets of apps assigning more weight to subsets containing popular as well as "diverse" items, where diversity is measured according to cosine distance between `app2vec` representations. Our algorithm outputs a set of k apps that is (approximately) the mode of a `DPP` distribution on subsets of size k. In off-line experiments, using app-install ad serving logs from Yahoo!, our proposed method can achieve a 36% lift in precision and 47% lift in recall compared to prior collaborative filtering methods (e.g. for $k = 5$).

- *(Application 2)* For personalized app recommendation, we give an algorithm using `DPP` 's and `app2vec` to cluster apps based on app distances in their vector space. We obtain a natural clustering of apps by sampling a set of apps from a `DPP` and using those apps as cluster centers. Through off-line experiments on Yahoo! data, we show that this combined technology gets 7% lift in precision and 9% in recall over conventional methods.

- *(Application 3)* In app-install advertising, conversion prediction model is used to estimate the probability of app install given an online impression from a user, which is then used to make the final selection of ad. In this scenario, through extensive experiments using ad impressions from ad server logs, we demonstrate that using app vectors learned from `app2vec` can achieve about 4.7% lift in AUC over baseline conversion prediction model.

4.1 The App Similarity Problem

In the mobile world, on one hand users are installing and trying new apps over time, on the other hand, there are thousands of mobile app developers who want to get more users to install their apps and grow their user bases. Therefore, how should app developers target mobile users to install new apps? More specifically, for app developers, the question is how to find users interested in their apps? First, let us look at the example below:

Example 4.1.1 *Assume there are three users X, Y and Z, and each of them uses a set of apps. User X uses {"Angry Bird", "NY Times", "House Finder"}; user Y uses {"Six Flags", "Boston Post", "Zillow"}; user Z uses {"Tumblr", "Yahoo Mail", "Flickr"}. Suppose the developers of "Redfin" app want to promote it and ask the question "who are the users interested in real estate?" It is natural to think that users X and Y are probably better candidates than Z to target for advertising, because they use apps "House Finder" and "Zillow" that are in the real estate category, similar to "Redfin".*

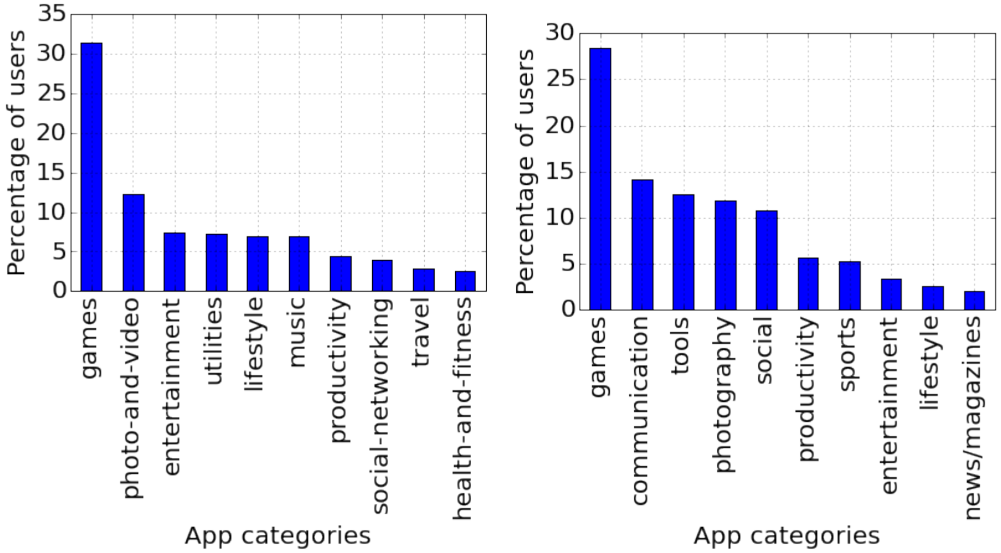
From this example above, the reason that users X and Y seem to be better candidates to target for advertising is that they use apps similar (e.g., of the same app store category) to the advertiser's app. However, how to determine which apps are similar to advertiser's apps? Formally, the fundamental problem we want to solve is: given a set of users U, and the historic app usage sessions S_u of a user $u \in U$, where each app session $s_i \in S_u$ is represented by $\langle u_i, a_j, t_s, t_e \rangle$, meaning user u_i used app a_j starting at time t_s and ending at time t_e. We would like to learn a similarity function $sim(a_i, a_j)$ for two apps a_i and a_j.

Two potential ways of solving this problem are:

- **Bag-of-words.** One approach to computing app similarity is through the bag-of-words method using app meta information. We can analyze the textual information in an app's title, description, market categories, user reviews, etc., or even enriched information from search engine results [95, 96]. There are several drawbacks to this approach: the app textual information are mostly provided by app developers, so the quality of the texts differ, and the data is noisy. For example, a game app description says how one can run a cafe, but based on the texts it is hard to distinguish it from a real cafe shop app. Another drawback of bag-of-words approach is that it is language specific, although people speak different languages and use apps developed in different languages, their app usage behavior may be similar. For example, people like to take photos of the highlights of a day, then use photo editing apps to beautify the photo and share it with friends in social apps. To identify such common user behavior patterns across languages, one has to build a different model for each language.

- **Vectorization.** Another approach is to learn latent vectors of apps [61], and then the distance between any pair of apps can be computed easily based on their latent vectors. In this approach, one can compose a user-app matrix based on user app usage history. In the matrix, a cell value is 1 if the corresponding user uses the specified app, and 0 if the user does not use the app. Or one can use normalized app usage (*e.g.,*based on the number of app sessions, the amount of time spent in apps, etc) as the cell values to represent the app usage intensity. Then low-rank approximation algorithms can be used to get the latent vectors of users and apps. This approach does not suffer from the high dimension problem as the bag-of-words approach. In later sections, we compare the learned app distances from this method to our proposed app vectorization method.

A common drawback of the approaches discussed above is they do not take app usage context into account. Similar apps that have different descriptions, or complementary apps which are used in similar contexts but have different functionalities, are not learned. In our study, we model apps based on their empirical app usage contexts (i.e., the sequence of apps used in a relatively short period) generated by millions of users. Based on usage context, we learn a latent vector representation for each mobile app; distances between vectorizations of the apps can then be used to measure similarity.



(a) iOS

(b) Android

Figure 4.1: Popularity of app categories.

Our data is obtained from app usage information available at Yahoo![1], which includes the app usage. From this data set, we can find out how many times an app was used, used with what other apps, and how users switch from one app to another[2]. Figure 4.1 shows the top 10 popular app categories in both platforms, based on 100 million users samples from February 2015 for both iOS and Android users. The most popular app category in both markets is “Games”, and the two markets have 7 app categories in common in the top 10 list, which implies that app

[1] https://developer.yahoo.com/.

[2] For anonymity, all user IDs were removed for data analysis.

usage behaviors have a lot in common regardless of their mobile device platforms. The studies in later sections focus on iOS users. In both platforms, although the most popular categories are installed by at least twice more users than the rest of the categories, there is no dominant app category. Most of the app categories are used by a few users, implying that beyond the most popular app categories, people have quite different tastes in apps.

4.2 Mobile App Modeling – `app2vec`

We now describe the background for and results of our vectorization in more detail.

4.2.1 `word2vec` Background

Our work is inspired by recent work on neural network models [74, 73], where the continuous bag-of-words (CBOW) and skip-gram (SG) models were proposed. In these works, the goal is to learn a latent vector for each word using a large corpus of documents, where the semantic distance between two words can be measured using their latent vectors.

Continuous bag-of-words. In this model, a log-linear classifier is used to predict the current word based on its preceding and succeeding words, where their representations are averaged as the input. With different application domains, how to form the input from context words should be adjusted according to domain knowledge. The objective of CBOW is to maximize the log-likelihood:

$$\mathcal{L} = \sum_{t=1}^{T} \log \mathcal{P}(w_t | w_{t-c} : w_{t+c})$$

where T is the size of word sequence, w_t is the t-th word in the word sequence, c is the context length (or half of the sliding window size), and $w_{t-c} : w_{t+c}$ is the sub-sequence $(w_{t-c}, \ldots, w_{t+c})$ without w_t. $\mathcal{P}(w_t|w_{t-c} : w_{t+c})$ is defined using softmax,

$$\mathcal{P}(w_t|w_{t-c} : w_{t+c}) = \frac{\exp\left(\bar{v}^T v'_{w_t}\right)}{\sum_{w=1}^{W} \exp \bar{v}^T v'_w}$$

where v'_{w_t} is the output vector representation of word w_t, and $\bar{v}$ is the average of word vectors in the context,

$$\bar{v} = \frac{1}{2c} \sum_{-c \le j \le c, j \ne 0} v_j$$

where v_w is the input vector representation of word w.

Skip-gram model. Instead of predicting current work using surrounding words, skip-gram does the opposite that uses current word to predict the surround words. Its objective function is defined as,

$$\mathcal{L} = \sum_{t=1}^{T} \log \mathcal{P}(w_{t-c} : w_{t+c} | w_t)$$

Assuming the independence among the contextual words, the probability distribution is simplified to,

$$\mathcal{P}(w_{t-c} : w_{t+c} | w_t) = \prod_{-c \leq j \leq c, j \neq 0} \mathcal{P}(w_{t+j} | w_t)$$

where the probability is defined by softmax:

$$\mathcal{P}(w_{t+j} | w_t) = \frac{\exp v_{w_t}^T v'_{w_{t+j}}}{\sum_{w=1}^{T} \exp v_{w_t}^T v'_w}$$

4.2.2 App Vectorization – `app2vec`

Taking an analogy to word and document modeling, we can treat each app as a word and each user as a document consisting of app sessions that take place in the order of time. However, there are unique properties to app session sequences that do not exist in text documents. Consider the below example:

Example: app session sequence of a user u is – (A, t_1, A, t_2, A, t_3, B, t_4, C, t_5, A). A, B, C, are different apps, t_i is the elapsed time between two app sessions. A could be a social app that u uses quite frequently.

The example above illustrates two challenges in processing app session "documents": (1) there are cases that an app is used multiple times in a row. In this case, the repeated app sessions of the same app are likely to be usage independent of other apps. Intuitively, we want to capture the similarity between apps within close usage context. Therefore, we pre-process the app session sequences to de-duplicate the repeated app sessions from the same app by only keeping the session followed by a different app. So the app session sequence in above example becomes $(A, t_3, B, t_4, C, t_5, A)$. (2) In the app usage context, if two app sessions are far apart from each other, they are likely to be less similar to each other. Therefore, we need a modified word2vec model to consider the weight between apps, where weight is measured by the time elapsed between two app sessions.

In the original CBOW and SG models proposed in word2vec, each word in the context is treated equally. It is a natural and simple assumption that words in the context are equally relevant to target word. However in the scenario of modeling mobile apps, app sessions have varying intervals in between, and intuitively the app sessions within a short interval to target app should contribute more in predicting the target app. Therefore, we modified CBOW algorithm to include the weight of context words. Let weight of word w_i to the target word w_t be:

$$r(w_i, w_t) = \alpha^l$$

where α is empirically chosen as 0.8, and l is the number of minutes between app w_i within the current context and target app w_t.

Plug in the recency weighting to CBOW model, $\bar{v}$ becomes,

$$\bar{v} = \frac{\sum_{-c \le j \le c, j \ne 0} r(w_j, w_t) v_j}{\sum_{-c \le j \le c, j \ne 0} r(w_j, w_t)}$$

In our study, the app vectors are trained by using this modified CBOW algorithm.

app2vec Training Results

Dataset. Our data is obtained from app usage information available to Yahoo![3]. If app developers use the SDK provided by Yahoo!, app installation and use are tracked and recorded. From this data set, we can find out how many times an app was used, used with what other apps in a short period, and how users switched from one app to another.

We sampled 300 million iOS users from February 2015 and used all app sessions generated by those users to build the app vectors. A user's app sessions are concatenated to form a user document [4]. Then we pre-process the user documents to de-duplicate the repeated app sessions and compute the elapsed time between app sessions. Then the whole data was read in one pass to train our app2vec model.

[3] https://developer.yahoo.com/

[4] User ids were removed after composing the app session sequences for keeping anonymity.

Query App	**Based on user-app matrix factorization**	**Direct application of `word2vec`**	**`app2vec` with preprocessing and weighting**
A social chatting app	A photo editor app	A screen theme app	A text and video chat app
	A music discovery and sharing app	A toolbox app	A music player app
	An emergency all app	A solitaire game app	A schoolmate chatting app (non-English app)

Table 4.1: Example of top similar app with data preprocessing.

The app2vec model generates a real vector for each app, such that the distance between two apps can be computed by taking their cosine distance. Apps that are commonly used in similar surrounding apps are more relevant than those used in different usage contexts.

Table 4.1 shows some examples to compare the quality of learned app similarity using matrix factorization method and our proposed app2vec model. For app2vec model, we also compare before and after preprocessing the user app session documents and considering weights of context apps[5]. We can see that the top similar apps make much better sense for the app2vec model after preprocessing the data and weighting than for the other models. In this comparison, we used the dataset that is used to train app2vec model to form a user-app matrix, and take low-rank approximation to get latent vectors of apps. One advantage of the app2vec model over bag-of-words approach is that it is language agnostic. For example, for the first query app, the app2vec model can identify a very similar chatting app in a different language.

To get a better understanding of the learned app similarity, Table 4.2 displays several more examples of relevant apps from the built app2vec model. And we compare with the similar apps obtained from the matrix factorization (MF) method, which composes a user-app matrix using the data set and takes low-rank approximation to get latent app vectors. In the first example, from the app2vec model, the dictionary app's most relevant apps are not only limited to other dictionary apps, but also the apps that are very likely to be used in the context. There are needs to query a (novel) word or phrase, such as in social communication and messaging apps. Whereas the similar apps obtained through the MF method are less meaningful, except the trivia app where one can imagine a dictionary is handy when to play trivia. In the second example, the querying app is used to search for a house for sale or rent. For the app2vec

[5] Due to the privacy policy, app names are not displayed here.

Queried App	Based on user-app matrix factorization	app2vec
A dictionary app	A fitness and exercise app	An app to watch, create and share short videos
	A food take-away order app	A horoscope app
	A trivia app	An app to share fun conversations, photos, videos with friends
	A casino game app	An app to share secrets and send messages anonymously
An app to search for house or apartments for sale or rent	A video streaming app	An app to search local real estate or find homes for sale
	An app to get free credit reports, credit scores, and daily credit monitoring	An app to get free credit reports, credit scores, and daily credit monitoring
	A coffee shop app	A finance app to manage bank or financial accounts
	An app to publish items for sale or to buy	An app for home improvement ideas and designs
An app to make online payment	A wallpaper app	An app to search for deals, coupons
	A navigation app	An app to search for cheap flights and compare prices
	A news app	An app that help mobile device to accept credit and debit cards payment
	A car racing app	An email client app
A role-playing game	All 4 apps are role-playing and RPG games	All 4 apps are role-play and RPG games

Table 4.2: Examples of relevant apps from app2vec model.

model, the most relevant apps include credit checking and home interior improvement design apps, which make sense in real life scenarios. But for the MF method, only two of the top 4 similar apps are related to house/apartment hunting. In the third example, the querying app is used to make online payments. The relevant apps from app2vec include shopping apps and mail client app (the latter one makes sense intuitively because it is used to check email receipts after making a payment). But the top 4 similar apps from the MF method do not make much sense. In the last example, all queried relevant apps for role-playing game apps are the same type of RPG game apps. For this query, both app2vec and the MF method work well.

Quantitative evaluation. In addition to the specific examples discussed above, we also measure the quality of app similarity in large scale through manual review. We sample 1000 apps from the data set used to train app2vec model and use different models to retrieve the top-1 similar app to those 1000 apps. The retrieved app pairs are categorized into three types through editorial efforts, (1). Strongly relevant: the two apps have (almost) the same functionality, e.g. two apps are the same type of game. (2). Relevant: the two apps have complementary functions and can be used together to achieve one task, or the use of one app needs actions from another app. (3). Not relevant: there is no obvious connection between two apps. The manual review process involves reading the apps' app store page and make an

App A	**App B**	**Evaluation Category**
Car racing game app developed by company X	Car racing game app developed by company Y	Strongly relevant
A restaurants review app	Food ordering and delivery service app	Relevant (complementary functions)
A news app	A flash light app	Not relevant

Table 4.3: App similarity quantitative evaluation criteria.

	`BoW` **(Bag-of -words)**	**`BoWCategory`** **(Bag-of-words within same app category)**	**`MFBinary`** **(Binary matrix factorization)**	**`MFIntensity`** **(Usage intensity matrix factorization)**	**`word2vecOnApp`** **(Direct application of `word2vec`)**	**`app2vec`**
Strongly relevant	51%	58%	21%	18%	45%	63%
Relevant	1%	2%	13%	17%	14%	19%
Not relevant	48%	40%	66%	65%	41%	18%

Table 4.4: App similarity quantitative evaluations using different models. Each model is used to query top similar app to 1000 sampled apps, and editorial evaluation accesses the quality of similar apps.

assessment based on the apps' description and screenshot images. Table 4.3 shows examples for those three categories.

Table 4.4 summarizes the evaluation results for several different models:

- `BoW`: bag-of-words approach, computes two apps' cosine similarity based on the keywords extracted from app's meta information, which includes app name, category, description. Standard text pre-processing steps, such as stop word removal and stemming, were taken before extracting keywords. Each keyword is then tf.idf weighted.
- `BoWCategory`: intuitively, two apps from the same category are more similar than apps from different categories. This method is also based on bag-of-words approach to compute two apps similarity, but we retrieve the top similar app from the query app's category.
- `MFBinary`[6]: matrix factorization approach, the matrix used to do low rank approximation is user-to-app matrix M, where $m_{i,j} = \{0, 1\}$ to indicate whether user u_i uses app a_j. The rank is set to be the same as `app2vec` latent vector size.
- `MFIntensity`: the matrix used to do low-rank approximation is user-to-app matrix M, where $m_{i,j}$ is the percent of app (app j) sessions used by user (u_i) among all apps used. The rank is set to be the same as `app2vec` latent vector size.

[6] For matrix factorization approach, one month app usage of 300 million users are used, since a user may not use all installed apps in one day.

- word2vecOnApp: direct application of word2vec: treat each user as a *document*, apps used as *words* and the sequence of app sessions as *sentence*. Directly apply word2vec on this set of user documents.
- app2vec: proposed in section 4.2.2, where a user's app sessions are pre-processed to combine adjacent sessions from the same app, to apply a larger weight to apps used within a short time frame.

We can see from Table 4.4 that across all models, numbers of app pairs belong to the same type are much larger than those have relevant functionalities. Especially for BoW and BoWCategory, almost all the semantically relevant pairs (about 60%) are apps of the same type. It is expected in that if two apps' keyword sets overlap, they are likely used to describe the same functions of the apps. However, due to the ambiguity of keywords, sometimes, the same set of keywords are used in different types of apps. For example, {"food", "drink", "order", "check-out"} can be used in an app to make food delivery orders and in a restaurant simulation game app. A simple way to improve on this aspect, one can limit to retrieve top similar apps from the same app store category. By applying this method, BoWCategory retrieves 13% more apps belong to the same type. BoW method relies on the quality of the app meta information (e.g., app description provided by developers), it does not take into account of user usage information. As to the few cases of relevant functionality app pairs, it is largely by chance. For example, in one such pair, one app is an RPG game, and the other app is a user guide app for this game.

MFBinary and MFIntensity generate similar results, semantically relevant app pairs are about 35%. The amount of same type app pairs is much less than BoW approach, but app pairs that have relevant functionalities get improved significantly. And by using user-app usage intensity information, MFIntensity captures 26% more relevant function app pairs than MFBinary. It is expected to have more app pairs that have relevant functionalities using MF model since by formulation it captures the app co-use patterns across the users. For example, one of such app pairs includes an app to help users to look for nearby outdoor fitness parks which have exercise equipment, and another app is a fitness exercise tracking app to record amount of exercises done. However, overall MF model does not generate better interpretable app pair relationships than simple BoW approaches. Our hypothesis is that an individual user installs a diverse set of apps on his/her device, spanning from utility app, wallpaper app, battery life monitoring app, to stock trading app. A well trained MF model probably retrieves pairs of apps that are co-used by many similar users. However, those pairs of apps are not necessarily semantically related.

Comparing direct application of `word2vec` on users app usage data (`word2vecOnApp`) with the simple `BoWCategory` method, the total numbers of semantically relevant app pairs are almost the same. But we can see 23% of the semantically relevant app pairs are identified as having a relevant function, comparing to 3% by `BoWCategory` model. It is because `word2vecOnApp` utilizes the app usage contexts from users, where the function relevant apps are likely to appear in the same contexts, and the signals can be picked up by `word2vec` model. However, the difference from word documents are (1). in users' app sessions the same app can have many consecutive sessions (e.g., usage sessions from messaging apps, social apps). These are noise to the `word2vec` model which does training on moving window concept, and the context windows are not well formed; (2) the semantic contexts can be formed better by utilizing the time intervals between app sessions. Therefore, `word2vecOnApp` can be improved further by approaching these two issues.

`app2vec` has 82% of the retrieved app pairs evaluated as semantically relevant. Comparing to `word2vecOnApp`, `app2vec` identifies 40% more pairs that are of the same type and 33% more pairs that have relevant functionalities.

App analogy examples. In the original `word2vec` paper [74], the authors illustrated interesting examples of word analogies can be found through word vector operations, such as ("man + king - woman = queen"). When the same operations are conducted on the learned app vectors from `app2vec`, interesting app analogies also can be found. Table 4.5 shows some examples of app pairs with nice semantic implications. For example (as shown in the first row), an email client app is related to notes taking and reminder app. When we apply the vector operation with a public transportation schedule app, we discovered that a social events app to complete the analogous pair. It makes sense that these two are used in complementary for going to and coming back from social events. Another example in the table (second row), for a pair of apps including a travel guide and trail maps app. The analogous pairs found include a daily nightlife events discovery app and an app for fitness exercise and training tracking. This

v_A - v_B	+ v_C	= v_D
Mail client app for hotmail & outlook - Sticky notes & reminders app	An app of real time bus, subway & metrotracker with off-line schedules.	An app for finding cool clubs, fun bars, and other events.
	Advanced photography app.	App that encrypts and protects documents, e.g., photos, videos etc.
An app for personal trip advisor, local travel guide & on road trip planner - An app of US trail maps	Social network app for television fans and movie buffs.	Community app for writers, authors and readers.
	An app that shows which bars, night clubs and pop-up parties are most popular every night of the week.	An app that helps people do exercises, track training progress.
An app that provides healthy restaurant nutrition guide - An fitness app that focus on walking training plan, and gives how-to-lose-weight tips	An app that lists grocery cash back, coupons, freebies, and rewards on healthy & popular brands.	An app that lets users earn rewards just for taking photos of their shopping receipts.
	An app for small business that enable phone to accept credit card payments.	An app that lets users send money for free. Transfer to any bank overnight.

Table 4.5: Examples of app analogies.

example makes sense in that people who are social butterflies probably care more about being in shape.

4.3 app2vec in Diverse Ad Consideration Set Selection

In online advertising systems, there are usually hundreds of thousands of eligible ads for a given impression, and ad serving model has to go through each eligible ad to calculate the probability of click or conversion to make ad selection. Response time to ad requests should be quite small. Therefore, efficiency is critical in eligible ad evaluation and ad selection. We propose using DPP method to pre-process the large set of eligible ads and generate the smaller sets of important and diverse ads and try to optimize the total likelihood of ad click. This smaller ad set is called **consideration set** (advertiser bid is not considered at this stage).

4.3.1 Determinantal Point Process (DPP) Background

Determinantal point processes (DPPs) [52] were initially used in physics to model repulsion. Recently this technique has received attention in machine learning; the focus of DPP-based models has been on diverse subset selection from a discrete and finite base set. A DPP-based method selects a subset of most diverse items from a given set. Some variants of DPPs can consider the quality of items and restrict the size of selected subset. In this section, we give some background to DPPs, and present our proposed applications in app clustering (see section 4.4), and personalized app-install ad selection (see section 4.3).

A point process P on a discrete set $\Gamma = \{1, 2, \ldots, N\}$ is a probability measure on 2^{Γ}, the set of all subsets of Γ. P is called a determinantal point process (DPP) if when Y is a random set drawn according to P, and its probability is represented as

$$P_L(Y) = \frac{\det(L_Y)}{\sum_{Y' \in \Gamma} \det(L_{Y'})}$$

Where L is a matrix indexed by elements of Γ, $L_{i,j}$ is a measure of similarity of elements i and j, and L_Y is a sub-matrix of L with only the elements in Y.

Intuitively, if elements included in the sub-matrix L_Y are dis-similar (diverse) the values of the diagonal are smaller, then $det(L_Y)$ is larger, and $P_L(Y)$ has larger probability.

Original DPP models the diversity of elements in the set, a simple modification, can also take element's quality into account.

- Let ϕ_i be a normalized feature vector describing element i, similarity of two elements $\phi_i^T \phi_j \in [-1, 1]$
- Let q_i be the quality measure of element i
- Let $L_{i,j} = q_i \phi_i^T \phi_j q_j$
- Let $S_{i,j} = \frac{L_{i,j}}{\sqrt{L_{i,i} L_{j,j}}}$

We have

$$P_L(Y) = \left(\prod_{i \in Y} q_i^2 \right) det(S_Y)$$

Intuitively, a set Y has higher probability if elements in it have higher quality (first term) and dis-similar (second term).

A basic DPP process does not restrict the size of the selected subset, in empirical applications very often it is desirable to select the most diverse and important subset of size k [52]. A k-DPP on a discrete set Γ is a distribution over all subsets $Y \in \Gamma$ with cardinality k. A k-DPP is obtained simply by conditioning on a standard DPP on the event that the set Y has cardinality k. A k-DPP probability P_L^k is:

$$P_L^k = \frac{det(Y_L)}{\sum_{|Y'|=k} det(Y_L')}$$

4.3.2 Our Approach

To demonstrate the idea of our approach, instead of using DPP without restricting the set output size, we use k-DPP to select the mode of app sets at different size k. To choose a set of k apps from all eligible ad apps, we need to prepare two inputs for DPP. One is the similarity between two apps and the importance of an app to a given user. In our proposed method, app similarity is computed through `app2vec` model introduced in Section 4.2. To get app importance $q(a, u)$ for a given user u, we use the average of an app's similarity to all currently installed and used apps A of this user.

$$q(a, u) = \frac{1}{n} \sum_{i=1}^{n} sim(a, A_i)$$

This is inspired by the classic, simple recommendation algorithm based on app similarity.

To optimize app-install ads click, our intuition is that ad apps that are relevant to the ones being used by users are useful to complement users ordinary usage scenarios. However, it is necessary to display a set of apps covering different categories to explore ad apps performance. Therefore, our proposed method is to select the k most important and diverse apps using k-DPP (recall Section 4.3.1) from all eligible ad apps (apps that being advertised). To assess the quality of selected ad apps in off-line experiments, we measure if the clicked apps are "contained" in the selected k ad apps. We use precision and recall introduced in Section 4.3.2 as evaluation metrics.

In the scenario of ad app consideration set selection, we want to generate ad app sets much smaller than the set of all eligible ad apps. To enforce the limit of k ad apps to be selected, we use a greedy algorithm inspired by [76] for k-DPP, which is described in Algorithm 1.

Comparison Models Now we list the four baseline models used to compare with our proposed model.

- **Random model** select k apps randomly from the set of available apps. This model serves as the baseline without any user modeling efforts.

- **Popularity model** builds apps global rank based on the number of users who use those apps, and top k apps are recommended to all users, so app j's rank score for all users is:

Algorithm 1 Greedy MAP for k-DPP

```
procedure SelectkDPP($L, S$)
    $Y \leftarrow \Phi, U \leftarrow \mathscr{Y}$
    while $U$ is not empty do
        $i^* \leftarrow \arg\max_{i \in U} \log\det(L_{Y \cup \{i\}})$
        if $|Y| \geq k$ then
            break
        end if
        $Y \leftarrow Y \cup i^*$
        $U \leftarrow \{i | i \notin Y, I(Y \cup i) \in S\}$
    end while
end procedure
```

$$r_{u,j} = |U_j|$$

where U_j is the users who installed app j.

- **Naive similarity model** measures the similarity between two apps using $Jaccard$ similarity,

$$Sim(i, j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$$

where U_i is the users who installed app i. Then the recommendation rank score of app j for a user is computed as weighted average of its similarity to all apps installed by user:

$$r_{u,j} = \frac{\sum_{i \in I_u} Sim(i, j)}{|I_u|}$$

where I_u is the set of apps installed by user u.

- **Matrix factorization model** used in our comparison is a low rank approximation model, where given a user-app matrix M, $m_{ij} = 1$ if user u_i uses app a_j. We want to find two matrices P and Q such that

$$M = P \times Q^T$$

with the goal to minimize the error:

$$e_{ij} = (m_{ij} - \sum_{k=1}^{K} p_{ik}q_{kj})^2$$

Evaluation Metrics Several evaluation metrics are applicable here, let the set of apps clicked by user u in the test data be $T(u)$, and the selected ad app consideration set for this user is $R(u)$, where $k = R(u)$. Then for recall and precision at consideration set size k are computed as:

- Recall measures how many apps user clicked in the test data are included in the consideration apps:
$$r@k = \frac{R(u) \cap T(u)}{T(u)}$$
- Precision is the average precision over all users:
$$p@k = \sum_{u} \frac{|R(u) \cap T(u)|}{n|R(u)|}$$

4.3.3 Experiments and Results

We evaluate our proposed method through the off-line experiments. We consider the ad apps that were displayed to a user as the whole pool of eligible apps to select consideration set, and our goal is to select a small subset of the ad apps such that the ad apps clicked by the user are contained in the selected apps. By varying k we want to measure the performance of consideration set at different sizes. In ad serving systems, this ad app consideration set can be used by ad selection algorithm to make ad serving decisions. Note that one can avoid the deterministic set of apps from greedy algorithm output by using sampling-based approach to approximate the mode of DDP's output sets.

We sampled 300K users[7] from February 2015 (after `app2vec` training data date) from the ad server log, who had impressions from more than 5 different ad apps. In our evaluations, for each method in comparison, we take top k ($1 \leq k \leq 5$) ad apps from all ads that were shown to the user, and compare the precision and recall.

[7] Anonymized user IDs from ad servers were used to evaluate method accuracy.

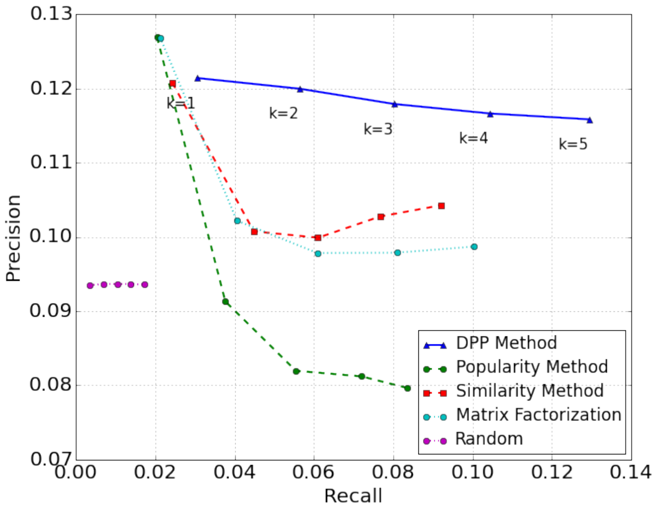


Figure 4.2: Performance comparison on ad click predictions for sampled users, for $k \in [1, 5]$. Each point corresponds to one value of k, and value of k increases from left to right.

Figure 4.2, shows the precision and recall comparison different top k's. Firstly, Random model serving as the baseline without any modeling efforts has the lowest recall and precision. As k increases, the precision stays almost the same due to the randomness of the selected ads. At $k = 1$, where we only select the top 1 predicted app, Popularity based prediction method performs the best, followed by Matrix Factorization method and Similarity method. This result makes sense since users are more or less aware of the popular apps (probably influenced by ads), so they are likely to click on the ad of the popular app. However when k is larger than 1, we can see that Similarity based method outperforms Popularity method, where similar apps to the ones used by the user are more relevant than most popular apps. Note that when $k > 2$, Popularity model has lower precision than Random model, it implicates that besides the very popular apps, the installation of most apps is not due to its global popularity. More importantly, when k is larger than 1, we can see that DPP method achieves better precision and more interestingly it gets a better recall. The performance decay, when increasing k is much slower than other methods. It implies that users tend to click on a broad span of app categories, so the diversified ad app sets are more effective than the classic methods.

4.4 app2vec in App Recommendation

The output from app2vec can be utilized to improve the performance of existing models. In this section, we showcase the application of app2vec model plus DPP in mobile app recommendation.

The app recommendation problem we want to solve is: when given a set of users U and the list of apps A_i used by each user u_i, how to generate a ranked list of k new apps for user u_i as recommendations. Such that the user is likely to use the recommended apps, and the apps ranked higher in the list should have higher probabilities to be used by the user.

In conventional collaborative filtering (CF) approach, one can construct a user-item matrix (M) with real or binary values for the matrix cell to indicate the strength of the signal, then factorize the user-item matrix to obtain low dimensional latent vectors for users (L_u) and items (R_v). In rating predictions, the values in M are the ratings, with the following object function in matrix factorization:

$$\min_{L_u,R_v} \sum_{r_{i,j}} (L_u \cdot R_v - r_{i,j})^2 + \lambda_u ||L||_2^2 + \lambda_v ||R||_2^2$$

where $r_{i,j}$ is item rating given by user i to item j.

4.4.1 DPP-based App Clustering

A common challenge shared by clustering algorithms that how to determine the proper number of clusters. Trial and error are one of the popular methods that one can vary the size of clusters then use some cluster quality checking metrics to find the best number of clusters.

One natural application of DPP on top of our app2vec model results is to cluster apps into semantic clusters [50, 15]. The two critical elements of DPP are similarity measurement between any pair of items and quality of items. In our scenario, app similarity can be easily computed from latent app vectors produced by the app2vec model, where the similarity between apps means the how related in usage context are the apps. As to app quality (importance), it can be measured by the number of users who installed the app as a proxy for its popularity.

We use the popular greedy algorithm proposed in [76] to find the most likely set of diverse and important apps, it is described in Algorithm 2.

Algorithm 2 Greedy MAP for DPP

```
procedure SELECTKDPP($L, S$)
    $Y \leftarrow \Phi, U \leftarrow \mathscr{Y}$
    while $U$ is not empty do
        $i^* \leftarrow \arg\max_{i \in U} \log\det(L_{Y \cup \{i\}})$
        if $\log\det(L_{Y \cup \{i^*\}}) < \log\det(L_Y)$ then
            break
        end if
        $Y \leftarrow Y \cup i^*$
        $U \leftarrow \{i | i \notin Y, I(Y \cup i) \in S\}$
    end while
end procedure
```

From the same day of `app2vec` training data, we filtered out apps that were used by less than 10 users. From the rest, we applied DPP to generate the most diverse and popular apps without limiting the result set size. The result set contains 197 apps and below are some examples of the found apps:

10 sample apps from the DPP result set: a weather app, a wallpaper app, a balloon shooting game (for kids), an app for a country's maps, a checker game app, a movie maker app, an adventure game, a navigation app, a news app, a financial account management app.

In next section, we propose a method to combine the DPP selected apps with the classic collaborative filtering method to boost app recommendation performance.

CF Combined With App Clustering In app recommendation task, the user feedback is implicit in that if an app is not used by a user, we do not know if it is because the user does not like the app or user does not get a chance to see the app. We take implicit feedback matrix factorization approach, which treats the task item prediction instead of rating prediction [47].

In addition to the classic matrix factorization based methods to approach recommendation problems, we propose taking advantage of the app similarity learned from `app2vec` model results to further improve the recommendation engine's performance. We create app clusters where the cluster centers are selected by DPP, which consumes app popularity (number of users) as app importance, and app similarity is computed based on apps' latent vectors.

The new objective function is:

Method	**Precision@5**	**Recall@5**	**Precision@10**	**Recall@10**
Popularity Model	0.039	0.095	0.029	0.142
Naive Similarity Model	0.057	0.143	0.042	0.204
MF	0.033	0.086	0.026	0.130
MF & App clusters	0.061	0.154	0.045	0.223

Table 4.6: App recommendation performance comparison.

$$\min_{L_u, R_v, W, W_u} \sum_{r_{i,j}} (L_u \cdot R_v + (W + W_u) \cdot \phi(a) - r_{i,j})^2 + \\ \lambda_u ||L||_2^2 + \lambda_v ||R||_2^2 + \lambda_w ||W||_2^2 + \lambda_{w_u} ||W_u||_2^2 \tag{4.1}$$

where W is the global weight vector for added features, W_u is the user-specific weight vector, which emphasizes personalized recommendation. $\phi(a)$ is the added app-related features. This modified objective function unifies matrix factorization and regression models, where the added side information eases the inherent cold-start problem of matrix factorization method for apps. To utilize the app clusters identified by `app2vec` and DPP, we let $\phi(a)$ be the nearest cluster center's ID.

4.4.2 Experiments and Results

To evaluate app recommendation performance, we sampled 100,000 users[8] from February 2015 (after `app2vec` training data date), where each user has at least 5 apps actively used in that period. For each user, 80% of the used apps are randomly selected to train matrix factorization model, and the rest are used to test model performance. The methods used in the comparison are the same ones used in Section 4.3.2.

Table 4.6 presents the performance comparisons of different approaches. Naive similarity-based model outperforms popularity model and matrix factorization model. It shows that users tend to use similar apps whereas most popular apps do not meet every user's need.

Our approach captures app similarity in usage pattern in addition to co-installation among users (captured by naive similarity model). Our proposed method combines matrix factorization and app clusters, generated based on app usage similarity and popularity, shows significant

[8] Anonymized user IDs from ad servers were used to evaluate method accuracy.

improvement. At $k = 5$ it has 7% precision lift and 7.7% recall lift over similarity-based model, and for $k = 10$ it has 7.1% precision lift and 9.3% recall lift over similarity-based model.

4.5 app2vec in App-install Ad Conversion Prediction

When an ad impression is available, ad selection algorithms need to estimate the probabilities of click and post-click conversion (app installation) for each eligible ad. A typical ad selection method is to rank ads by their expected revenue. For ad a_i, it is calculated as:

$$E(a_i) = b_i \times p(click) \times p(conversion|click)$$

Where b_i is the bid price for a click from the advertiser. The estimations of click and post-click conversion probabilities usually use information from impression context (*e.g.,*publisher meta-information, past performance), current user's profile (*e.g.,*demographics, geo-location, interests) and current ad (*e.g.,*type of ad, ad format, past performance, etc.). In practice, since ad conversions are very rare events, click prediction and post-click conversion prediction have separate models.

In this section, we focus on how to build user's app usage features to improve the post-click conversion prediction. One can use all apps as binary features in the model, but there are hundreds of thousands of apps in the markets and used by all users, these features will be very sparse, and rigorous regularization is probably needed to do feature reduction. Suppose a set of apps A are used by user u_i, if apps can be categorized into different types, one can compose user's app usage profile by the types of used apps, $C_{u_i} = \{c|c\ is\ type\ of\ a_j, a_j \in A\}$.

Then the question is how to categorize apps into different types. From our app2vec model, a vector representation of apps is learned from a large set of users app usage. Then apps can be clustered based on the distances computed based on their latent vectors. One can apply k-means clustering algorithm on the app vectors to cluster apps, and use the cluster membership as app type.

Experiment and Results

In our experiment, we use a logistic regression model to test the impact of models trained with user app usage profile created by different methods.

A logistic regression model is trained for each of the sampled 17 advertisers, where positive examples are users[9] who installed the advertised app after clicking on the ad, and negative examples are users who clicked on the ad but did not install. In our experiments, we sampled ten of thousands of positive examples for each advertiser, and down sampled negative examples to be about four times of positive examples (from our empirical experience this ratio yields better accuracy).

The baseline features used to train the models include features from the user (e.g., age, gender, categories of clicked ads, categories of converted ads), publisher (e.g., publisher id), ad (e.g., the category of the ad).

Here we want to compare the impact of users' app usage features created from different app categorization methods, as described below:

- Store category: apps' store categories are determined by app developers, who should presumably have the best understanding of the app and choose the most relevant app category.
- k-means: use k-means clustering algorithm to cluster apps based on the learned app vectors from `app2vec`. Here we let $k = 60$, which is about the amount of app categories in the app store.

For each advertiser, we conduct 3-fold cross validation to evaluate the model performance, where 66% of the data are used for model training, and the other 33% are used for accuracy evaluation. Figure 4.3 shows the conversion prediction models accuracy (AUC) comparison (with 95% confidence interval) by using different user app usage profiles. It is clear that adding user app usage features can improve baseline model's accuracy. Simply using app store categories achieves the smallest improvement. Maybe the predefined app store categories have ambiguity and developers of similar apps may assign their apps into different app store categories. However, the app similarity from `app2vec` results are from a huge number of users usage patterns, and we have shown the learned app similarities have high quality (see Table 4.4 in Section 4.2.2). Therefore, clustering apps into the same amount of categories as store categories generates better accuracy, with 4.7% AUC lift over the baseline.

[9] Anonymized user IDs from ad servers were used to evaluate method accuracy.

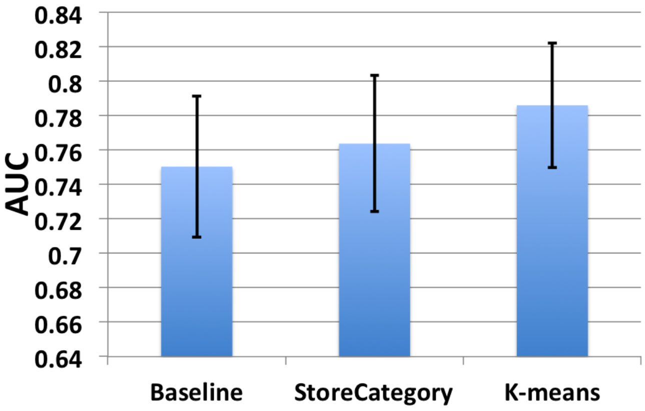


Figure 4.3: Install prediction model, AUC comparison using different methods to provide user used app category information. Error bar for 95% confidence interval is shown.

4.6 Discussion

There are several drawbacks of the discussed `app2vec` model in this work, (1). For rare apps which are used by small amount of users, this app usage based learning process suffers from lack of training data. In model training phase, these apps are filtered out by frequency threshold, since they only cover a small amount of cases in practice. One option is to utilize app store category information. We can conduction the vectorization process twice. First compose user app usage sequence just composed of app store categories, and learn latent vectors for app store categories. Then in second round, we can use the learned store categories as priors to initialize app vectors. In this case, for the apps that have small user bases, these priors from store categories may give them better chances to result in positions close to apps in the same store categories. (2). the quantitative evaluation of learned app latent vectors is expensive, and does not scale. This is a common problem for many follow-up work inspired by `word2vec`. One option is to apply the learned latent vectors in the end application (*e.g.,*classification task) and use impact on final application performance as feedback of the learned latent vectors quality. Then go back to the training process and adjust the parameters (*e.g.,*window size, context words weight, down sampling ratio). The use of domain knowledge will probably help save quality check efforts, without repeating the expensive feedback loops. For example, in our app vectorization case, we may use app store category to get coarse level quality check, where apps from the same store category should have smaller distances than apps from different store categories.

In this work, we focused on vectorization of mobile apps, and a future work is to learn user vectors based on the apps used and the learned app vectors. Then the user latent vectors can provide a dense set of user features to end applications. To do use vectorization, some directions can be explored are: (1). Learn a global vector to represent a user while learning individual apps' latent vectors, prior work paragraph2vec [35] already show significant performance improvement in content personalization. (2). After learning app vectors, we can form a tensor for users and the app vectors used by them. Then we can obtain users latent vectors through tensor factorization. However, the user vectorization model still faces the challenge of quality measurements, and how to get systematic measurements before applying in the end applications.

4.7 Related work

Our work is directly related to mobile application recommendation work. In [51] authors proposed that the context of user state and time can help significantly in recommending apps to users. [79] takes the user to user social network structure and social influence into account to predict app installation, instead of using the user to user graph constructed by commonly installed apps. Besides user interests, [61] considers the privacy preferences of a user to generate app recommendations. However for the cold-start problem, when there is no prior information about a user's preference, [60] proposed to use Twitter followers app preferences to make recommendations. In our study, we augment the start-of-art collaborative filtering approach with semantic relationships among apps from their usage contexts.

Our app2vec is inspired by word2vec work [74, 73]. The proposed concept of using shallow learning, but taking advantage of big data opens up research and applications in various areas. There is various follow-up work applying word2vec concept to other domains. [21] builds event level model of user's mobile activities, authors demonstrated the learned latent vector of events are effective in predicting what the next app user is going to use. [35] proposed a hierarchical model that by maintaining a document as the global context of words in the document, after the training process word vectors and document vector can be learned together. In our work, we treated each mobile app as a word and the app usage sequence from one user as a document. Through examples, we show that the learned app semantic similarity is meaningful.

In recommender system area, it has been recognized that it is important to recommend diverse content to users. In search results diversification efforts, [16] formulated the problem as selecting a set of result documents across different content categories such that the probability of at least one result document is useful to an average user is maximized. The similarity between documents is implicitly captured by the different document categories. In our study, the similarity between a pair of apps is modeled directly based on their usage contexts.

There is rich research work done on determinantal point process. Some focused on designing more efficient algorithms. [13] proposed sampling algorithms to approximate DPP in discrete and continuous spaces. [40] proposed a 1/4-approximation algorithm to find the most likely configuration. And some work focused on applying DPP in machine learning problems. [14] proposed Markov k-DPPS to select news articles for users over time, where the goal is to avoid focusing on a small set of items of users interest, given the dynamic nature of news articles. Different from prior work that used sampling based solutions to DPP and computes the average performance over a large number of samples, in the app-install ad selection scenario considered in our study, the items to select from are relatively stable, and it is more beneficial to approximate the most likely configuration to DPP.

4.8 Conclusion

In this paper, we introduce a new way to vectorize mobile app usage. This is inspired by deep learning technique in `word2vec` for word documents (in contrast to the conventional bag-of-words or collaborative filtering approaches), and this models mobile applications based on app usage contexts and captures semantics. Also, we showed to use Determinantal Point Processes (DPPs) in conjunction with our `app2vec` app vectors for three problems: (1). diverse ad selection, (2). mobile app recommendation, (3). app-install ad conversion rate prediction. Through extensive experiments with real data, we show the effectiveness of this combined technology. We believe that `app2vec` and DPPs will find many more uses in mobile app ad research.

Chapter 5

Large Scale Look-alike Audience Modeling

5.1 Look-alike Audience Targeting

Using look-alike audience in on-line advertising campaigns helps an advertiser reach users similar to its existing customers. It can be used to support many business objectives like targeting users who are similar to a list of past purchasers, web service subscribers, installers of particular apps, brand lovers, customers from customer relationship management (CRM) systems, ad clickers, supporters for a politician, fans of a sports team, etc.

The advantage of look-alike audience extension technology is that it can dramatically simplify the way to reach highly relevant people comparing to other targeting technologies in online advertising. An advertiser can just upload a list of users to generate customized audience without knowing any details about the user features used in an advertising system.

The input to a look-alike audience system is a list of user IDs (e.g., browser cookies, addresses, phone numbers or any other identifiers[1]) called "seeds". The seed users can be converters of advertiser's past ad campaigns, or the existing customers who have stronger purchasing power, etc. The output is a list of users who are believed to be look-alikes to the seeds according to certain look-alike model. Then advertisers can target to show ads to those identified look-alike audiences in ad campaigns. The efficacy of a look-alike audience system is measured by multiple business concerns:

- **Scalability** - What is the maximum size of the look-alike audience output? What are the upper and lower bounds of the seed amount that can lead to a reasonable size of look-alike audience?

[1] The original values of Personally identifiable information (PII), such as phone number, are not used. Instead, the hashed and anonymized user IDs are used in the system.

- **Performance** - How fast can the system generate a look-alike audience after advertiser uploading the seed list? How much return on investment (ROI) lift can look-alike audience achieve in a real ad campaign?
- **Transparency** - Is the system working as a black box or can it generate feedback and insights during an ad campaign setup? How fast can these feedbacks and insights be generated, in a few seconds or days?

In other words, there is no unique "best" design of a look-alike audience extension system given different requirements on scalability, performance and model transparency. The main contributions of our work [66] are summarized as follows:

- We present a large-scale look-alike audience extension system from Yahoo!, where the similar user query time is sub-linear to the number of query users. The core model of this look-alike system is based on graph mining and machine learning technique on pairwise user-2-user similarities. The system can score 3000+ campaigns, on more than 3 billion users in less than 4 hours.
- Through extensive experiments using real-world app installation campaigns data, we show that the recommended look-alike audience by our method can achieve more than 50% lift in app installation rate over other existing audience extension models.
- Furthermore, we also discuss the challenges and share our experience in developing large-scale look-alike audience extension system. We demonstrate that by using weighted user-2-user similarity graph, the app installation rate can be further improved by 11%.

5.2 Existing Look-alike Methods & Systems

The look-alikeness between 2 users is measured by their features. A user u_i can be characterized by a feature vector, $\mathbf{f}_i = (f_{i,1}, \cdots, f_{i,K})$. In an on-line advertising system, each feature $f_{i,k}$ could be continuous (e.g. time spent on a certain site), or in most situations, categorical or binary (e.g. use of a certain mobile app or not). Features can come from a user's attributes, behaviors, social interactions, pre-build audience segments and so on. Different companies may use different features to represent a user based on their businesses and understanding of users.

The dimension of a feature vector varies from thousands to millions. In this paper, we consider the case where $f_{ij} \in \{0, 1\}$, and continuous features are bucketized into multiple binary features if needed.

Although the business motivation is simple and straightforward, we share the same view of [90] that there is a significant lack of prior work of look-alike audience modeling in literature. To our best knowledge, look-alike systems being used in the on-line advertising industry can be categorized into three categories: simple similarity-based model, regression-based model, and segment approximation-based model.

5.2.1 Simple Similarity-based Look-alike System

A straightforward method to find look-alike users is to compare all pairs of seed users and available users in the system, then determine look-alikeness based on distance measurements. A simple similarity-based look-alike system can use direct user-2-user similarity to search for users that look like (or in other words, be similar to) seeds. The similarity between two users, u_i and u_j is defined upon their feature vectors $sim(\mathbf{f}_i, \mathbf{f}_j)$. Cosine similarity (for continuous features, Equation 5.1) and Jaccard similarity (for binary features, Equation 5.2) are two possible measures.

$$sim_{cosine}(\mathbf{f}_i, \mathbf{f}_j) = \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{||\mathbf{f}_i||||\mathbf{f}_j||} \tag{5.1}$$

$$sim_{Jaccard}(\mathbf{f}_i, \mathbf{f}_j) = \frac{\sum_{g=1}^{K} min(\mathbf{f}_{ig}, \mathbf{f}_{jg})}{\sum_{g=1}^{K} max(\mathbf{f}_{ig}, \mathbf{f}_{jg})} \tag{5.2}$$

The similarity between a given user u_i (with feature vector $\mathbf{f}_i$) and seed user set $S = \{u_j : \mathbf{f}_j\}$, could be calculated as the similarity of this user to the most similar seed user, i.e.

$$sim(\mathbf{f}_i, S) = \max_{\mathbf{f}_j \in S} sim(\mathbf{f}_i, \mathbf{f}_j) \tag{5.3}$$

A more complicated method is probabilistic aggregation of user-2-user similarity measures. This method only works with those user-2-user similarities that are strictly limited between 0 and 1. A pairwise similarity is treated as the probability of triggering a “look-alike” decision.

Therefore, the similarity from a user to seed set can be measured as:

$$sim(\mathbf{f}_i, S) = 1 - \prod_{\mathbf{f}_j \in S} (1 - sim(\mathbf{f}_i, \mathbf{f}_j)) \tag{5.4}$$

The simple similarity-based method is easy to implement on a small scale. It also has the advantage to leverage the information carried by all seeds in the user feature space. However, this method has a couple of the main concerns. The first is scalability because calculating pairwise similarities among a large number of users is not a trivial task. The brute force solution has computational complexity $O(kMN)$, for N candidate users and M seeds with k features on average per user. In a typical online advertising market, there are billions of candidate users and more than tens of thousands of seeds with hundreds of features per user. The second concern is different features of seed users are treated equally, and the model does not distinguish their power of identifying the most relevant candidate users. For example, the predicting power for ad clicks or conversions is not considered. Therefore, some less relevant candidate users may be regarded as look-alike users due to similarity from less important features.

5.2.2 Regression-based Look-alike System

Another type of look-alike audience systems for online advertising is built with Logistic Regression (LR) [84]. For a given user u_i who has a feature vector $\mathbf{f}_i$, a logistic regression model can be trained and model the probability of being lookalike to seeds as:

$$p(u_i \ is \ a \ lookalike \ to \ seeds | \mathbf{f}_i) = \frac{1}{1 + e^{-\mathbf{f}_i' \boldsymbol{\beta}}}$$

"Seeds" are usually treated as positive examples. Whereas, there are several options to select negative examples. The simplest negative example set can be a sample of the non-seed users. This is not the best choice since the potential ad converters may also be included in the non-seed users. Another choice is to process ad server logs and find out the users who have seen advertiser's ads in the past but did not convert. However, it suffers from the cold-start problem. If the seeds are from a new advertiser, or will be used for new ad campaigns, there are no users who have seen the ads. When the number of features is large, feature selection is conducted by imposing an L_1 regularization to the loss function to reduce the number of non-zero values in the final model. In practice, machine learning packages, such as Vowpal Wabbit, provide a flexible and scalable solution for training logistic regression on a large scale (e.g., with millions of features) on Hadoop clusters and predicting such probabilities for a massive amount of users.

The advantage of this modeling methods is that information carried by seeds are compressed into a model (e.g. a $\boldsymbol{\beta}$) and there is no need to remember the feature vectors of seeds for scoring users to be a look-alike or not. Apart from the huge benefit, there are some potential caveats for applying regression modeling methods to the look-alike problem. LR is a linear model, although a lot of conjunctions, cross or other dependencies can be embedded into the features, it may not be perfect and efficient for clustering users. We implement an LR based look-alike system for comparison with our proposed system.

5.2.3 Segment Approximation-based Look-alike System

Another type of look-alike system is based on user segment approximation, where user segments can be user characteristics such as user interest categories. For example, if a user likes NBA games, this user is considered to belong to *Sports* segment. In comparison to other methods, a segment approximation-based system uses pre-built user segments as user features. The general idea of segment approximates based method is to find out top segments that are "shared" by as many seed users as possible.

Here we discuss one the most recent published segment approximation based method from Turn Inc, [90]. In their approach, each user is represented as a bag of segments, $u_i = \{c_{i1}, \cdots, c_{iK}\}$. The optimization goal for look-alike audience extension is, given an advertiser's provided segment set C (i.e., the segments that appear in seed users), to recommend new users with a segment set C', satisfying following three properties:

$$sim(aud(C), aud(C^{'})) > \alpha \tag{5.5}$$

$$perf(aud(C^{'})) - perf(aud(C)) > \beta \tag{5.6}$$

$$|aud(C \cup C^{'})| \gg |aud(C)| \tag{5.7}$$

where $aud(C)$ is the set of users who have any of the segments in C, $perf(aud(C))$ is the performance of users $aud(C)$ regarding ad click-through-rate or conversion-rate. Intuitively, the above criteria mean that: (1). the extended audience set $aud(C^{'})$ is "similar" enough (in terms of user overlap) to the seed audience $aud(C)$ (Eq. 5.5); (2). The performance of the extended audience set $aud(C^{'})$ should be better compared to the seed audience set (Eq. 5.6); (3). And the size of the extended audience set $aud(C^{'})$ should be much larger than the seed audience (Eq. 5.7).

There can be many segments that satisfy the above properties so a scoring function is needed to distinguish the importance of segments. [90] discussed a greedy approach, which is an application of the set cover algorithm, and a weighted criteria-based algorithm. For any newly recommended category (or segment) c_{new}, they calculate a score by combining the three functions (Eq. 5.5, 5.6, 5.7) above:

$$score \propto sim(aud(c_{new}), aud(C)) \times perf(aud(c_{new})) \times nov(aud(c_{new})|aud(C))$$

where $nov(aud(c_{new})|aud(C))$ (corresponds to Eq.5.7) measures the proportion of users from this new category that are not in seed users. Based on this score, the authors sort all existing user segments and recommend top-k segments to the campaign for user targeting (for more details, refer to [90]).

Segment approximation works well when the pre-built segments quality is high and have good coverage. It is particularly the case for big branding advertisers, but maybe less useful for small advertisers. The pre-build process also introduces another computation pipeline which may delay the look-alike audience generation. We implemented the algorithm described above, as another method for large-scale comparisons.

5.3 Graph-Constraint Look-alike

In this section, we present the details of our developed graph-constraint look-alike system, which takes advantages of both simple similarity and regression-based methods. First, a global user-2-user similarity graph is built, which enables us to limit candidate look-alike users to the nearest neighbors of seeds. Then the candidate users are ranked by their feature importance specific to ad campaigns.

Training and scoring look-alike audience per ad campaign against all users is a low-efficiency process. A typical look-alike audience size is 1 to 10 million, which is about 0.1% to 1% in a 1 billion user pool. Scoring a lot of irrelevant users wastes a significant amount of computation resources. Also, training to get feature weights against the 99% irrelevant users faces a challenging user sampling problem. We propose to generate look-alike audience in two phases, namely global graph construction and campaign specific modeling.

5.3.1 Phase I: Global Graph Construction

In this phase, a user-2-user similarity graph is built for all available users in the system, where an edge between two users indicates their Jaccard similarity (Eq. 5.2). The goal is to find look-alike candidates for a particular set of seeds using this global graph. The core formulation of our approach is a pairwise weighted user-2-user similarity, defined as:

$$sim(\mathbf{f}_i, \mathbf{f}_j) = \frac{\mathbf{f_i}'\mathbf{A}\mathbf{f_j}}{||\mathbf{f_i}|| \times ||\mathbf{f_j}||} \tag{5.8}$$

where $\mathbf{f_i}$, $\mathbf{f_j}$ are feature vectors of users u_i and u_j. The weight matrix $\mathbf{A}$ can incorporate the importance of linear correlations for both individual features and pairwise feature combinations. For example, a pairwise weighted user-2-user similarity is a weighted cosine similarity if the weight matrix $\mathbf{A}$ is diagonal. A non-diagonal matrix $\mathbf{A}$ can model pairwise feature combinations. Higher order feature combinations are rarely observed to be useful in a large and sparse feature space. Therefore, our feature modeling is only limited to individual and pairwise features to avoid over-engineering a look-alike system. The scale of the weight matrix is determined by the total number of features which is relatively smaller in comparison to the total number of users. This property allows us to build a specific weight matrix for each ad campaign to leverage more tuning power on features.

The challenges of building the global user-2-user similarity graph live in both the high complexity of user pairwise similarity computation ($O(N^2)$) and similar user query ($O(N)$). To approach these challenges, we use the well-known Locality Sensitive Hashing (LSH) to put similar users into the same clusters by reading user data in one pass and enable similar user query in sub-linear time. Here we briefly introduce the background of LSH.

MinHash Locality Sensitive Hashing (LSH). Using LSH [85], each user is processed once in two steps. The first step is using a set of hash functions to transform a user feature vector into a similarity-preserving signature, which consists of a much smaller amount of values than original user feature vector. Then the second step is to assign the user to clusters based on the signature. Signature values are grouped to create clusters of users who are similar to each other (above a predefined similarity threshold) with a bounded probability. At query time when given an input user, we can hash this user to find out the clusters it falls into and the users in those clusters are the candidate similar users. Both cluster construction and the retrieval of similar users are per user based process without pairwise user operations. Therefore, LSH dramatically simplifies the simple similarity-based look-alike system. Below we use MinHash LSH as an example to discuss the details of those two steps:

- **MinHash.** Hashing technique is used in LSH to reduce user feature space while preserving their original similarity with high probability. Different choices of hashing schemes can be applied, such as using random projection to approximate cosine similarity and using MinHash to approximate Jaccard similarity etc. In our case users' Jaccard similarities are computed on their binary features, so MinHash is used:

  $$h_{min}(\mathbf{f}_i) = \arg \min_{x \in (1,\ldots,K),\ f_{i,x}=1} h(x)$$

  where $\mathbf{f}_i$ is a K-dimensional feature vector of user u_i, x is the index of features in $\mathbf{f}_i$. Hash function $h(x)$, $[h : (1, \ldots, K) \to R]$ maps the index of a feature into a random number. A function $h_{min}(\mathbf{f}_i)$ is then defined as the feature index that has the minimal hash value, and $h_{min}(\mathbf{f}_i)$ is called the hash signature. It can be proven that MinHash preserves Jaccard similarity (refer to Chapter 3 of [85]), which means that the probability of two users having the same MinHash value equals to the their Jaccard similarity:

  $$sim_{Jaccard}(\mathbf{f}_i, \mathbf{f}_j) = P\left(h_{min}(\mathbf{f}_i) = h_{min}(\mathbf{f}_j)\right)$$

  In other words, if two users' Jaccard similarity is r, and a total of H MinHash functions are used to hash the two users independently. Then the two users are expected to have $H \cdot r$ signatures identical.

- **LSH.** Signatures generated by H Minhash functions form an H dimensional vector. Signatures for all N users form an N-by-H matrix, known as signature matrix [85]. As $H \ll K$, the signature matrix can be considered as a dimension reduction result from the raw user-feature matrix, which is a N-by-K matrix.

  Using the H-dimensional signatures, LSH method provides various flexible ways to cluster similar users into buckets for retrieval purposes. The most popular approach is the "AND-OR" schema, where H signature dimensions are partitioned into b bands. Each band consists of r signature dimensions, where $b \times r = H$. Users are clustered into the same bucket only if they have the same r-dimensional signature within a band. A user can only fall into one bucket in a band. Therefore each user has b buckets in total. If a seed user is found to fall into bucket b_i, all users in b_i can be retrieved as its look-alike candidates, which is a small amount of users compared to the whole user population. The retrieved candidate users can be then scored for exact ranking if needed. The values of b and r are usually determined empirically based on application data similarity distribution, and the desired the item-2-item similarity threshold.

The LSH technique introduced above can be used to reduce the complexity of the system implementation. After conducting MinHash LSH, each user u_i has a list of bucket ids generated by the hash signatures. Candidate look-alike users for a campaign can then be generated by looking up the hash signatures of all the seeds and merging their similar users fetched from the global graph. A reasonable good size of candidates varies from $5\times$ to $10\times$ of the final look-alike audience. A too small candidate set may have risks of missing good look-alike users that may not be selected to the candidate list. A too large candidate set increase the cost of computations in Phase II. When the number of candidate users fetched from the graph is small (e.g., due to a small amount of seeds), one can use regression or segment approximation-based methods to ensure the desired size of look-alike audience. In the rest of this paper, we only consider the cases that sufficient number (e.g., more than 10 million) of candidate users can be fetched.

5.3.2 Phase II: Campaign Specific Modeling

In this phase, the goal is to train a simple campaign-specific model and refine the audience candidates from Phase I. The candidates are scored by the campaign-specific feature weights, and top ranked users are selected as the final look-alike audience.

Different ad campaigns may care about different user features. A retail store's ad campaign may care more about users' location rather than their education. An insurance ad campaign cares more about users' financial status than their social status. An optimized weight matrix (Eq. 5.8) for an ad campaign can ignore the irrelevant features and help generate look-alike audience that can lead to a better return-on-investment for ad dollars.

Intuitively a relevant feature should have more power to distinguish seeds from other users. There are many off-the-shelf methods and algorithms for feature selection, feature ranking, and modeling labeled data. The popular ones are simple feature selection and linear model, e.g. logistic regression (with regularization), or decision trees. Here, we discuss a simple feature selection method as an example because the simple method is a non-iterative feature selection method which can be run very efficiently. Also, it gives a reasonably good performance in a sparse user feature space in comparison to linear models and other complicated methods. However, other methods are still highly recommended if computational cost and response time are not the bottlenecks.

A simple feature selection method calculates a diagonal weight matrix A, where $A_{i,j} = 0,\ for\ i \neq j, 1 \leq i, j \leq L$ and $A_{j,j}$ is the importance of the feature j calculated from a set of seeds S and a set of comparing users U. Let us denote the proportion of users in seeds that has feature j as $p_j = \frac{\sum_{f_i \in S} f_{i,j}}{|S|}$, and the proportion of users in the comparing user set U that has feature j as $q_j = \frac{\sum_{f_i \in U} f_{i,j}}{|U|}$.

The seeds set S is provided by advertisers. But there are multiple options for composing user set U, such as the users who have seen ad impressions from the advertisers but did not click or convert, or the whole pool of candidate users. The benefit of the latter method is that the computed q_j can be shared across all campaigns to improve system efficiency.

There are several options for evaluating feature importance through univariate analysis, such as mutual information [23], information gain [54] and information value (IV) [91]. The advantage of these methods is that they assume independence of features such that features can be evaluated independently of each other in distributed manner. Based on our experience, these different methods yield similar results. Therefore, in this paper, we take IV as the example implementation, which measures the predictive power of a feature. The importance of the binary feature[2] j is calculated as:

$$A_{j,j} = \begin{cases} (p_j - q_j)\log\left(\frac{p_j(1-q_j)}{(1-p_j)q_j}\right), & \text{if } p_j > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

The directional filter $p_j > 0.5$ is optional that enforces only positive features will be selected, which means those features are more significant in seed users than in other users. Negative features can be used to discount a user's rank, and it is one of our future exploration directions. The positive features for a particular campaign can be ranked by their IV values where features with higher values have more predictive power. Furthermore, those top-ranked features can be displayed to advertisers to provide transparency of the model.

To further motivate our proposed IV based method, we can look at the predicted hyper-plane for classification in logistic regression (with standardized features) [46]:

$$\sum_{i=0}^{K} \hat{\beta}_i x_i = C$$

Where C is a constant. Further, the probability of being positive or negative-labeled can be obtained. If the features, x_i's are binary, those estimated $\hat{\beta}_i$'s are (roughly) a measure of the impact of the log-odd ratio given a presence/absence of the feature $f_{i,j}$. In such situation (i.e. features are binary), the estimated $\hat{\beta}_i$'s are of the same scale and is a measure of the importance of the corresponding feature. In other words, a ranking of the absolute value of these $\hat{\beta}_i$'s corresponds to a ranking of the feature importance.

Following almost exactly the same logic mentioned above for logistic regression, the user scoring method falls into the following framework:

[2] In our system, we only consider binary features, where continuous features are bucketized to create binary features as well.

$$s(f_i) = \sum_{j=0}^{K} \beta_j f_{i,j} + \sum_{g,j=0}^{K} \beta_{jg} f_{i,j} f_{i,g}$$

where β_j's are a measure of the variable importance; β_{jg} are a second order measure, a measure of the interactions of the variables. One realization of the above campaign-specific scoring framework is based on IV values, the score of a user for a specific campaign c is,

$$s(f_i, c) = \sum_{1 \le j \le K} A_{j,j} f_{i,j}. \tag{5.9}$$

Note that the above scoring (Eq. 5.9) is unbounded. Although in our final audience recommendation, we only need to use this score to rank candidate audience and recommend a specific amount of audience, it will be useful to rank the audience and let advertisers control and leverage the size of the recommended audience and the "quality" of the audience. If one wants to derive a "normalized" score, i.e. to create a mapping from the raw score to a score with range in $[0, 1]$.

$$s() : s(f_i) \mapsto [0, 1]$$

A candidate for this is to use a sigmoid function $s(t) = \frac{1}{1+e^{-\beta t}}$. It is possible to use this normalized score to determine a threshold that top-N users can be selected without sorting.

5.3.3 System Design and Pipeline

Our graph-constraint look-alike system is running in production at Yahoo! and generating thousands of look-alike audience sets. The sketch of the system is depicted in Figure 5.1, which contains four major components:

Construct global user graph. The system periodically builds global user graph based on user-2-user similarity. When global graph builder (component 1 in Figure 5.1) is triggered to run, it loads configured user features from user profile service for each user, and calls our in-house LSH package to hash users into different buckets. If two users' similarity is above our configured threshold, they may be found hashed into the same bucket in multiple bands of the min-hash table. After processing all users, a map-reduce job is run to group users by bucket ids, such that users fall into the same bucket can be stored together to enable efficient user query.

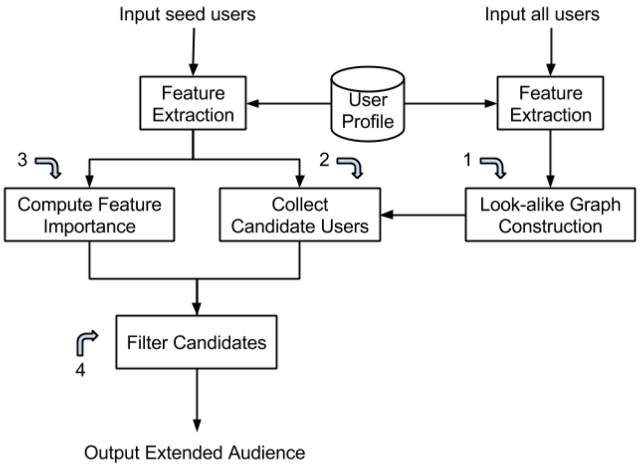


Figure 5.1: System Architecture and Pipeline.

Collect candidate users. Look-alike audience extension system takes a list of seed user ids as input. Component 2 hashes all seed users features and uses the hash signatures to find out the buckets those seed users fall into. Users in those buckets are all merged into the candidate user set. At the end of seed extension process, user profile service is queried for candidate users, and the output user id with feature data are saved on HDFS.

Compute feature importance. To compute feature importance, component 3 reads global user count and seed user count for each feature. Then it calculates feature importance and saves the results on HDFS. The global feature distribution can be shared by all campaigns to make the feature importance learning very efficient.

Score and recommend audience. As the last procedure of look-alike audience extension system, component 4 reads two data sets from HDFS: top K feature list and their importance score, and candidate users and their features. It iterates through all candidate users and scores users by the campaign specific feature importance. As a final step, all candidate users are ranked, and top N users ids are sent to user profile service to update user profiles for campaign targeting.

5.4 Experiments and Results

In this section, we evaluate look-alike models using back-play tests on real mobile app install campaigns data, where the goal of the campaigns is to get new app installers while meeting the advertiser's target cost-per-install.

Evaluation Methodology

In app install campaigns, advertisers (usually the app developers) want to get more users to install their apps while meeting their target cost per install. In our evaluations, we focus on the user targeting side and measure how accurate the targeted audience is for specific ad apps. The problem of which app ad should be served to the targeted user at impression time is out of the scope of this paper, and it should be handled at downstream by ad selection algorithms. As running online campaigns for experiments is costly, the performance of a look-alike model can be evaluated by using off-line back-play tests.

Depending on their objectives, advertisers have many choices to put together a seed user list to use a look-alike system. For example, an advertiser may select users who made in-app purchases in the recent past as seeds and use look-alike to find similar users who may potentially also make in-app purchases. Another example is that an advertiser can put together some existing installers as seeds and aim to find similar users who may install the app. In our experiments, we try to simulate the second scenario using the following steps:

1. Fix a date t_0, collect the installers from t_0 to t_1 (e.g., $t_1 - t_0 = \ 1\ month$) as seed users for an app.

2. Use a look-alike audience extension system to extend seed users into a look-alike audience with size n.

3. Calculate the performance (app installation rate) of recommended audience in the test period $(t_1, t_2]$ (e.g., $t_2 - t_1 = \ 2\ weeks$).

4. Compare different look-alike systems' performance by varying audience size n.

Data and Evaluation Metrics

User features. If app developers use the SDK[3] provided by Yahoo!, app installation and use are tracked and recorded. We also derive user app usage features (e.g., a user is likely to be interested in which categories of apps) from this data to construct user feature vectors. The

[3] https://developer.yahoo.com/flurry/

global graph is built on a user profile snapshot right before the testing period, which includes more than a billion mobile users. The user features include apps installed, app categories, app usage activities, and app meta information. These same features are also used for the simple similarity-based method and logistic regression method in comparison. When we compare with segment-approximation based method, user app interest categories (e.g., Sports, Adventure Games, etc.) are used as the segments.

Seeds. From Yahoo! advertising platform, we sampled 100 apps which have more than 10,000 converters per month. For each app, the seeds were app installers in May 2015. The testing examples are active users in the first two weeks of June, where new installers of the test apps are positive examples and non-installers as negative examples.

Performance metric. We use installation rate (IR) to measure the percent of recommended audience (A) that install the target app (P) in the testing period. It is computed as: $IR(P, A) = \frac{|I|}{|A|}$, where I is the set of new installers from A. Note that in conventional conversion rate computation the denominator is the number of impressions, whereas here the denominator is the number of unique recommended users. The motivation is that we want to capture whether the recommended audience is more likely to install the ad app, without considering ad serving time decisions.

Experiment Results To compare the installation rate (IR) of different methods, we compute IR at various recommended audience sizes. For graph-constraint model and the logistic regression model, each candidate audience has a score from the model so that we can rank all the users and take the top-n scored users for evaluation. The candidate audience output from Phase I of our graph-constraint method already have similarity to seeds larger than the pre-configured threshold. Therefore, for the simple similarity-based method, we randomly sample n users from the candidate set to avoid pair-wise user similarity computation between seeds and candidate users. For segment-approximate based model, there is no rank among the candidate users, and we randomly sample n users from the candidate set.

Figure 5.2 shows the weighted average of installation rates for all four methods. The axises are normalized by constants to comply with company policy. The evaluations were conducted on audience size in the scale of millions. GCL is our proposed graph-constraint look-alike model (section 5.3.2). LR is logistic regression model (section 5.2.2), where seed users are

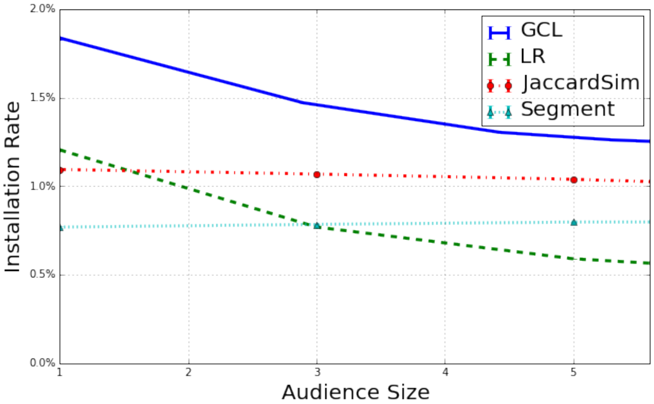


Figure 5.2: Weighted Average Installation Rate Comparison on 100 Apps. Axises are scaled by constants.

positive examples and non-seed users are used as negative examples (the same way to compose training data as the GCL model). Since the majority of users are non-seed users, so downsampling was done on negative samples. $JaccardSim$ is the simple user similarity based method (section 5.2.1). $Segment$ is the segment approximation method (section 5.2.3). We can see from the plot that GCL model performs best among the models in comparison, as audience size increases the installation rate decreases. $JaccardSim$ model's performance remains relatively stable across different sizes of audiences, which reflects the average installation rate of users similar to seeds. When audience size is small, top ranking users identified by GCL performs more than 50% better than the average similar users.

At smaller recommended audience sizes, the LR model performs better than $JaccardSim$ model and $Segment$ based model. It seems the top features play a key role. When the audience size increases the LR model performance goes down dramatically. This may be due to the sparsity of user features (e.g., apps installed), when none of the apps plays a significant role in the LR model, its effect is down to random user picking. As the audience size gets larger and there is more noise in the larger candidate set.

Figure 5.3 shows 4 example apps installation rate comparisons from different categories. Since segment based method does not rank users, so we take all the recommended audience by segment based method and select the same amount of top audiences from the logistic regression model and our graph-constraint look-alike model. For all methods in comparison, they work better for entertainment and action game apps than social apps. Our hypothesis is that users

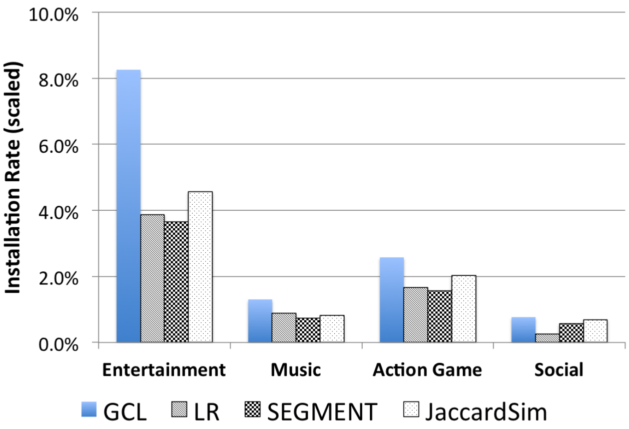


Figure 5.3: Installation Rate Comparison of 4 Example Apps. Axises are scaled by constants.

usually do not frequently install new social apps, and they stably use the apps where their friends are, so it is hard to get relevant features to predict what additional social apps they will install. On the opposite, users are more likely to explore new entrainment apps (new sources of entertainment), and play new action games when the already installed games are done or got stuck. Therefore, there are more user signals related to those types of apps, and look-alike models can pick up those signals to make better predictions.

5.5 Discussion

Our look-alike system is a running product at Yahoo!, where the production pipeline can score 3000+ campaigns (using about 3000 mappers on 100 nodes), on more than 3 billion users (with millions of features in total) within about 4 hours. In empirical on-line campaigns, these generated look-alike audiences are driving up to 40% improvements in conversion rates and up to 50% reduction in cost-per-conversion compared to other targeting options like demographic or interests[4]. It is challenging to develop an efficient and effective large-scale look-alike system, in this section we share some experiences in tuning a look-alike system and running on-line look-alike audience targeting campaigns.

Weighted LSH. Intuitively, users with similar app usage behaviors should be more similar than users who use apps quite differently even if they have the same apps installed. For example,

[4] https://advertising.yahoo.com/Blog/GEMINI-CUSTOM-AUDIENCE-2.html

given three users u_1, u_2 and u_3, and they all have a *sports* app installed. u_1 and u_2 use this app more than ten times a day, while user u_3 uses this app only once a week. For these three users, intuitively u_2 should be more similar to u_1 than u_3. To encode this key observation, we compute *tf-idf* weight for each feature of a user, where each user is treated as a document, and the intensity of a feature is treated as the frequency of a word.

MinHash LSH discussed in Section 5.3.1 approximates Jaccard similarity between two sets of items. To extend LSH to encode weighted Jaccard similarity, several prior works proposed different methods [33, 48, 67]. One straight-forward method is: suppose w_i is weight of feature i, and w_i is integer. One can generate w_i replicas of feature i to replace the original feature, then proceed with LSH on the transformed feature sets. It can be shown the MinHash LSH on the transformed item sets approximates weighted Jaccard similarity.

In our scenario, the feature weights are real values. So the approaches discussed above is not directly applicable. We adopted the approach proposed in [33] to do a transformation on the original MinHash value to encode feature weight: $f(X_i) = \frac{-\log X_i}{w_i}$, where w_i is the weight of feature f_i, X_i is the original MinHash value normalized to $(0, 1)$. Intuitively, when feature weight w_i is larger, the hashed value $f(X_i)$ is smaller, so feature f_i is more likely to be selected as the signature, and two users are more likely to have the same hash function signature if both of them have this feature with higher weights.

We use this weighted LSH scheme to build a weighted global graph, and the campaign specific modeling is the same as Section 5.3.2. Figure 5.4 compares the performance of weighted graph-constraint look-alike model with the non-weighted model on the same 100 ad apps (see Section 5.4). We can see that when considering user feature weights, audience generated from weighted graph consistently has about 11% lift in installation rate.

Seed size effect. The number of seed users input to look-alike system will affect the number of direct neighbors (or similar direct users) that can be found. By varying the size of the input seed set, we can empirically observe the trade-offs between recommended audience size and expected installation rate performance. Figure 5.5 show the 100 campaigns weighted average performance at different sizes of seed sets. A small number $n = 1000$ is used as the base value, and we sample $k \times n$ seeds for each of the 100 ad apps. We can see that as the number of seeds increases:

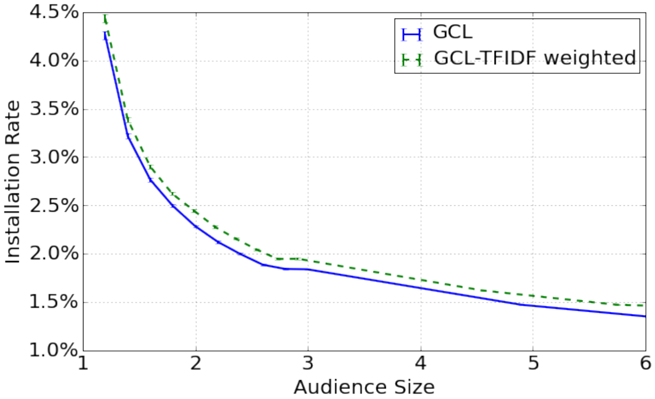


Figure 5.4: Installation rate improvement achieved by weighted global-graph. Audience sizes are in millions. Axises are scaled by constants.

- For a fixed amount of recommended audience, larger seed size generates better performance. This is because when seed user size is small, noise is stronger than signal.

- The maximum size of recommended audience increases, but performance decreases. This is expected in that the more seed users, the more neighbor users can be found in the graph. But meantime more noise are introduced into the model, hence the performance at the maximum number of users decreases.

- The improved performance and the maximum number of recommended audience do not change much when seed user size reaches $40n$. This value could be used to make suggestions to advertisers as how large the seed size should be to have more freedom in performance and audience reach trade-off. But a rule of thumb is that more seeds are better regarding both installation rate and audience size.

However, an optimal choice of the seed size depends on many factors, such as campaign, users features, etc.

Audience reach. Look-alike audience expansion system is to help advertisers find a set of audiences who look like the provided seeds, the audience size is much smaller than the overall targetable audience. This imposes two challenges: (1). It is beneficial for advertisers to target on the smaller set but with high conversion potential users, saving budget from exploring the whole audience population. However, it may not benefit the marketplace where advertisers bid to show ad impression to users. If all advertisers only target on their specific set of users,

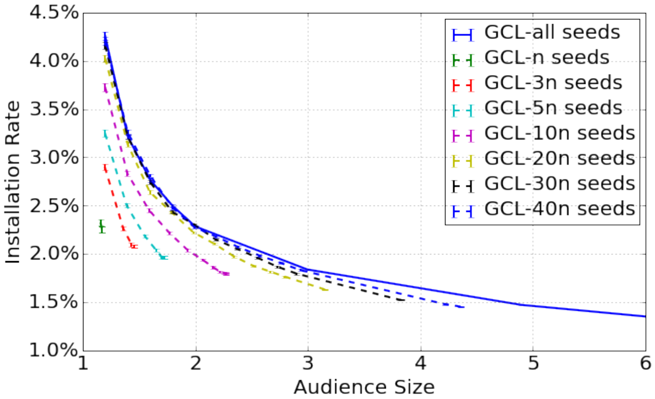


Figure 5.5: Different seeds set size for 100 apps. $n = 1000$, and audience sizes are in millions. Axises are scaled by constants.

the competition on the ad impressions from a user is reduced. Hence, the bid price may drop. The challenge is how to maintain the revenue of the marketplace. One possibility is to charge advertisers additional fees for use of look-alike audience targeting if the cost-per-conversion still meet the goals of advertisers; (2). Lookalike audience targeting campaigns usually have better conversion rate performance comparing to other campaign targeting methods. However, the high-quality look-alike audience size is limited. If advertisers want to increase the spend on look-alike audience targeting campaigns, a trade-off has to be made between look-alike audience size and quality. In online advertising, ad platforms may give suggestions about the expected performance at different audience sizes and leave the decision to advertisers to choose the desired audience size.

Clickers vs. converters. Clickers are the users who click on the ad while converters are the users who finally convert on an ad (with or without clicking on the ad). Finishing a car insurance online quote, for instance, is an example of conversion to a car insurance ad. In online advertising, users who click on ads may behave differently from users who convert on ads. In look-alike audience expansion system, advertisers have the freedom to upload seed user ids (e.g., browser cookies), or specify the rule to collect users as seeds (e.g., users who visit a particular web page). When the seeds provided to look-alike systems are past ad campaign converters or users who completed some actions on the advertiser's website, then the extended audience look like those converters but may not necessarily look like clickers. In online campaigns that target look-alike audience, the ad click through rate (CTR) may be lower than campaigns that use other targeting methods, although the conversion rate may be higher. This is a concern for cost-per-click price type where ad-network's revenue is at risk. In such cases, look-alike system could differentiate the seed users by introducing weights on seeds, where a seed has higher weight if the seed generated both ad click and ad conversion whereas a seed has lower weight if the seed only converted on the ad.

5.6 Related Work

The most related prior work is from Turn's audience extension system [90], the authors proposed a systematic method to extend the advertiser provided audience using weighted criteria algorithm. The algorithm balances the quality of extended audience using three key metrics namely user similarity, size of new users and the past performance of extended users. In that work, users are extended on predefined segments, so the quality of existing segments in the targeting system affect the extended audience performance. In our work, the audience extension is at the user level, where a model is learned on user features. [19] describes a large-scale system to classify users for campaigns, where the goal is to predict the likelihood of a user to convert to an existing campaign. The system we propose is generic and can be applied to any set of seed users, which can be generated with different goals.

Behavioral targeting is an area that focuses on inferring users interests from past behaviors, with the assumption that users' past interests can help improve prediction and recommendation tasks. [18] introduced a large-scale distributed system to infer users' time varying interests. In recommender systems, [49] combined taxonomies and latent factors to predict users purchase behavior; [17] proposed a content based latent factor model to infer user preferences by combining global and individual users preferences. Using the trending deep learning technique, [34] proposed to use hidden conditional random field model to model users online actions, such as click and purchase. These prior work are orthogonal to our proposed system, where the learned user interests and preferences can be converted to user feature vector as input the look-alike system for further improvements.

To deal with a large number of objects, especially when objects pairwise distances is important, hashing techniques are heavily used to improve efficiency. The locality sensitive hashing [92] used in our proposed look-alike system was also applied to many areas, such as used the technique to detect duplicate images [33, 53]; extract topics from a collection of documents [41]; web page clustering [45], genomic sequence comparison [26]; 3D object indexing [69]. The original LSH algorithm treats the features of objects equally. However, intuitively the features do not have equal importance when measuring the distances between objects. Therefore, there are prior works focused on how to factor in feature weights in the algorithm [33, 48, 67]. In our look-alike system, LSH is used as the initial step to cluster users at a coarse level. After user filtering at this step, campaign specific model ranks the candidate audience which has a larger impact on the final performance.

Ad-networks are tracking hundreds of millions anonymous online users (*e.g.,*identified by browser cookies), and the daily activity data could be in scales of tens of terabytes. Therefore, even before modeling user for various applications, the data storage and processing cost is not ignorable. As data warehouse computing becomes available, advanced server technology can be used to improve overall performance and reduce the capital and operational cost [55, 56, 57, 58, 59].

5.7 Conclusion

In this work, we present a large-scale look-alike audience extension system, where the similar user queries time is sub-linear. The core model of this look-alike system is based on graph mining and machine learning technique on pairwise user-2-user similarities. Through extensive evaluations on app installation campaigns, we show that the identified look-alike audience can achieve more than 50% lift in app installation rate over other state of the art audience extension models. We also discuss the challenges and share our experience in developing look-alike audience extension system, and running online look-alike targeting campaigns. One of our future work is to leverage the real-time campaign feedback signals into the continuous optimization process of look-alike campaigns.

Chapter 6

Future Work

Although user targeting has been known to the industry for more than a decade, among academics there has been a lack of systematic study of how users are targeted by ads and how ads impact the online user experience. In our user profile driven ad crawling study, we investigated the interactions among content, ads, and online user profiles. The study was conducted under considerably informal conditions, and even the user profile, at the center of this study, is a much simplified version of reality. We need better models to simulate users online behavior. In fact, each user activity is going to impact user presentation in ad platforms and in subsequent ads. A more careful study is required in order to understand such dynamics. Finally, in such ad crawling studies, the traffic generated by ad crawler is going to impact the performance of ad campaigns and campaign targeting strategies. It is therefore important to find ways to minimize the impact on the market.

For many types of ads, profile information is key in targeting and in building prediction models. The problem is how to represent the pieces of user information in models. In our context-based app vectorization work, user features learned from the app vectors found useful applications in app-install advertising such as in ad selection and conversion prediction tasks. However, the concept of vectorization of entities is not limited to mobile apps. Query terms, web pages, even users can be vectorized together based on the activities users generate over time. In such shared latent space, it will be useful to explore the relationships amongst users, ads, and content.

While online user activities are often difficult to grasp, the situation is even more complex in the case of ad clicks and ad conversions. Although ad platforms provide the common language of user interest profiles, which give advertisers the freedom to specify the audiences they wish to target, it is often quite challenging to determine the optimal targeting criteria. Look-alike audience models present a form of black box to advertisers, with user similarities calculated according to all available user information within the ad platform. However, while a look-alike audience extension system works well with existing users, it misses a lot of opportunities to identify new users.

In general, user targeting provides advertisers a way to identify users at a fine granularity without exceeding their exploration budgets. However, this goal conflicts with the marketplaces revenue model which states that the price of impressions depends on the competition in ad impression auctions. A way must be found to balance the competing interests of advertisers and the ad platforms.

Apart from online advertising, as discussed in this thesis, a multitude of areas benefit from user modeling as well, such as social networks, online content personalization, online gaming, and online shopping. As technology advances, new forms of consuming online materials will continue to emerge. However, the need to understand online user behavior and model user online representations will remain. At this point, the relevant questions are: What new properties will users reveal on the new platforms, what new methods will enable better user modeling, and what new applications will user models be applied to?

References

[1] App install addiction shows no signs of stopping. flurrymobile.tumblr.com/post/115194583975/app-install-addiction-shows-no-signs-of-stopping. 2014/12/17.

[2] App install addiction shows no signs of stopping. *http://flurrymobile.tumblr.com/post/115194583975/app-install-addiction-shows-no-signs-of-stopping*. 2014/12/17.

[3] Facebook statistics. http://newsroom.fb.com/company-info/. Checked on 2016/04/14.

[4] Google i/o by the numbers: 1b android users, 900m on gmail. http://www.cnet.com/news/google-io-by-the-numbers-1b-android-users-900m-on-gmail/. 2015/05/28.

[5] The mobile app-install ad is driving a boom in mobile ad spend - and not just among game makers. *http://www.businessinsider.com/the-mobile-app-install-ad-is-driving-a-boom-in-mobile-ad-spend-and-not-just-among-game-makers-2015-1*. 2015/06/15.

[6] Seven years into the mobile revolution: Content is king again. http://flurrymobile.tumblr.com/post/127638842745/seven-years-into-the-mobile-revolution-content-is. 2015/08/26.

[7] Shopping, productivity and messaging give mobile another stunning growth year. http://flurrymobile.tumblr.com/post/115194992530/shopping-productivity-and-messaging-give-mobile. 2015/01/06.

[8] Smartphones: So many apps, so much time. http://www.nielsen.com/us/en/insights/news/2014/smartphones-so-many-apps--so-much-time.html. 2014/07/01.

[9] The u.s. mobile app report. *http://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report*. 2014/08/21.

[10] word2vec. https://code.google.com/p/word2vec/. Accessed: 2015/03/30.

[11] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[12] V. K. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang. Vivisecting youtube: An active measurement study. In *INFOCOM, 2012 Proceedings IEEE*, pages 2521–2525. IEEE, 2012.

[13] R. H. Affandi, E. Fox, and B. Taskar. Approximate inference in continuous determinantal processes. In *Advances in Neural Information Processing Systems*, pages 1430–1438, 2013.

[14] R. H. Affandi, A. Kulesza, and E. B. Fox. Markov Determinantal Point Processes. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, 2012.

[15] A. Agarwal, A. Choromanska, and K. Choromanski. Notes on using determinantal point processes for clustering with applications to text clustering. *arXiv preprint arXiv:1410.6975*, 2014.

[16] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

[17] A. Ahmed, B. Kanagal, S. Pandey, V. Josifovski, L. G. Pueyo, and J. Yuan. Latent factor models with additive and hierarchically-smoothed user preferences. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 385–394. ACM, 2013.

[18] A. Ahmed, Y. Low, M. Aly, V. Josifovski, and A. J. Smola. Scalable distributed inference of dynamic user interests for behavioral targeting. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 114–122. ACM, 2011.

[19] M. Aly, A. Hatch, V. Josifovski, and V. K. Narayanan. Web-scale user modeling for targeting. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 3–12. ACM, 2012.

[20] A. Arasu and H. Garcia-Molina. Extracting Structured Data from Web Pages. In *In Proceedings of ACM SIGCOM*. ACM, June 2003.

[21] R. Baeza-Yates, D. Jiang, F. Silvestri, and B. Harrison. Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 285–294. ACM, 2015.

[22] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan. Adscape: Harvesting and analyzing online display ads. In *Proceedings of the 23rd international conference on World wide web*, pages 597–608. ACM, 2014.

[23] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5(4):537–550, 1994.

[24] I. A. Board. IAB internet advertising revenue report. 2015 first six months results. http://www.iab.com/wp-content/uploads/2015/10/IAB_Internet_Advertising_Revenue_Report_HY_2015.pdf, October 2015.

[25] A. Broder, M. Najork, and J. Wiener. Efficient URL Caching for World Wide Web Crawling. In *In Proceedings of the World Wide Web Conference (WWW)*, May 2003.

[26] J. Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.

[27] C. Castelluccia, M. Kaafar, and M. Tran. Betrayed by Your Ads! In *In Privacy Enhancing Technologies*, pages 1–17. Springer, 2012.

[28] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.

[29] C. Change, M. Kayed, M. Girgis, and K. Shaalan. A Survey of Web Information Extraction Systems. *In IEEE Transactions on Knowledge and Data Engineering*, 18(10), October 2006.

[30] G. Chatzopoulou, C. Sheng, and M. Faloutsos. A first step towards understanding popularity in youtube. In *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*, pages 1–6. IEEE, 2010.

[31] Y. Chen, D. Pavlov, and J. F. Canny. Large-scale behavioral targeting. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 209–218. ACM, 2009.

[32] X. Cheng, J. Liu, and C. Dale. Understanding the characteristics of internet short video sharing: A youtube-based measurement study. *Multimedia, IEEE Transactions on*, 15(5):1184–1194, 2013.

[33] O. Chum, J. Philbin, A. Zisserman, et al. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, volume 810, pages 812–815, 2008.

[34] N. Djuric, V. Radosavljevic, M. Grbovic, and N. Bhamidipati. Hidden conditional random fields with distributed user embeddings for ad targeting. In *IEEE International Conference on Data Mining*, 2014.

[35] N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic, and N. Bhamidipati. Hierarchical neural language models for joint representation of streaming documents and their content. In *Proceedings of the 24th International Conference on World Wide Web*, pages 248–255. International World Wide Web Conferences Steering Committee, 2015.

[36] F. Duarte, F. Benevenuto, V. Almeida, and J. Almeida. Geographical characterization of youtube: a latin american view. In *Web Conference, 2007. LA-WEB 2007. Latin American*, pages 13–21. IEEE, 2007.

[37] D. Eichmann. The RBSE Spider-balancing Effective Search Against Web Load. In *In Proceedings of the 1st World Wide Web Conference (WWW)*, 1994.

[38] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao. Youtube everywhere: Impact of device and infrastructure synergies on user experience. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 345–360. ACM, 2011.

[39] P. Gill, V. Erramilli, A. Chaintreau, B. Krishnamurthy, K. Papagiannaki, and P. Rodriguez. Follow the money: Understanding economics of online aggregation and advertising. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 141–148, New York, NY, USA, 2013. ACM.

[40] J. Gillenwater, A. Kulesza, and B. Taskar. Near-optimal map inference for determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 2735–2743, 2012.

[41] S. Gollapudi and R. Panigrahy. Exploiting asymmetry in hierarchical topic extraction. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 475–482. ACM, 2006.

[42] S. Guha, B. Cheng, and P. Francis. Challenges in Measuring Online Advertising Systems. In *In Proceedings of the ACM SIGCOMM Internet Measurement Conference*, pages 81–87. ACM, 2010.

[43] S. Guha and A. McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009.

[44] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring price discrimination and steering on e-commerce web sites. In *Proceedings of the 2014 conference on internet measurement conference*, pages 305–318. ACM, 2014.

[45] T. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. 2000.

[46] J. M. Hilbe. *Practical Guide to Logistic Regression.* CRC Press, 2015.

[47] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[48] S. Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 246–255. IEEE, 2010.

[49] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, J. Yuan, and L. Garcia-Pueyo. Supercharging recommender systems using taxonomies for learning user purchase behavior. *Proceedings of the VLDB Endowment*, 5(10):956–967, 2012.

[50] B. Kang. Fast determinantal point process sampling with application to clustering. In *Advances in Neural Information Processing Systems*, pages 2319–2327, 2013.

[51] A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer. Climbing the app wall: enabling mobile app discovery through context-aware recommendations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2527–2530. ACM, 2012.

[52] A. Kulesza and B. Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

[53] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2130–2137. IEEE, 2009.

[54] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, volume 33, pages 81–93, 1994.

[55] C. Li, I. Goiri, A. Bhattacharjee, R. Bianchini, and T. Nguyen. Quantifying and Improving I/O Predictability in Virtualized Systems. In *IEEE/ACM International Symposium on Quality of Service (IWQoS)*, pages 1–12, June 2013.

[56] C. Li, P. Shilane, F. Douglis, D. Sawyer, and H. Shim. Assert(!Defined(Sequential I/O)). In *6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 14)*, Philadelphia, PA, June 2014. USENIX Association.

[57] C. Li, P. Shilane, F. Douglis, H. Shim, S. Smaldone, and G. Wallace. Nitro: A Capacity-Optimized SSD Cache for Primary Storage. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 501–512, Philadelphia, PA, June 2014. USENIX Association.

[58] C. Li, P. Shilane, F. Douglis, and G. Wallace. Pannier: A Container-based Flash Cache for Compound Objects. In *Proceedings of the 16th International Middleware Conference (ACM/IFIP/USENIX Middleware 15)*, pages 50–62. ACM, 2015.

[59] C. Li, S. Zou, and L. Chu. Online Learning Based Internet Service Fault Diagnosis Using Active Probing. IEEE ICNSC, 2009.

[60] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua. Addressing cold-start in app recommendation: Latent user models constructed from twitter followers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 283–292. ACM, 2013.

[61] B. Liu, D. Kong, L. Cen, N. Z. Gong, H. Jin, and H. Xiong. Personalized mobile app recommendation: Reconciling app functionality and user privacy preference. 2015.

[62] B. Liu, A. Sheth, U. Weinsberg, J. Chandrashekar, and R. Govindan. Adreveal: improving transparency into online targeted advertising. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, page 12. ACM, 2013.

[63] M. Liu, R. Cai, M. Zhang, and L. Zhang. User Browsing Behavior-driven Web Crawling. In *In Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, October 2011.

[64] Q. Ma, S. Muthukrishnan, and M. Sandler. Frugal streaming for estimating quantiles. In *Space-Efficient Data Structures, Streams, and Algorithms*, pages 77–96. Springer, 2013.

[65] Q. Ma, S. Muthukrishnan, and W. Simpson. App2vec: Vector modeling of mobile apps and applications. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2016.

[66] Q. Ma, M. Wen, Z. Xia, and D. Chen. A sub-linear, massive-scale look-alike audience extension system. In *Proceedings of the 5th International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, 2016.

[67] M. Manasse, F. McSherry, and K. Talwar. Consistent weighted sampling. *Unpublished technical report) http://research. microsoft. com/en-us/people/manasse*, 2010.

[68] J. MARSHALL. Forrester: US Online Display Ad Spending will Nearly Double by 2019. http://blogs.wsj.com/cmo/2014/10/06/forrester-us-online-display-ad-spending-will-nearly-double-by-2019/, October 2014.

[69] B. Matei, Y. Shan, H. S. Sawhney, Y. Tan, R. Kumar, D. Huber, and M. Hebert. Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1111–1126, 2006.

[70] O. McBryan. GENVL and WWWW: Tools for Taming the Web. In *In Proceedings of the 1st World Wide Web Conference (WWW)*. Geneva, 1994.

[71] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[72] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Crowd-assisted search for price discrimination in e-commerce: First results. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 1–6. ACM, 2013.

[73] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[74] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[75] S. Nath. Madscope: Characterizing mobile in-app targeted ads. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 59–73. ACM, 2015.

[76] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[77] L. Olejnik, T. Minh-Dung, C. Castelluccia, et al. Selling off privacy at auction. In *Annual Network and Distributed System Security Symposium (NDSS). IEEE*, 2014.

[78] C. Olsten and M. Najork. Web Crawling. *In Foundations and Trends in Information Retrieval*, 4(3), 2010.

[79] W. Pan, N. Aharony, and A. Pentland. Composite social network for predicting mobile apps installation. In *AAAI*, 2011.

[80] S. Pandey and C. Olsten. User-Centric Web Crawling. In *In Proceedings of the World Wide Web Conference (WWW)*, May 2005.

[81] L. Partners. Display LUMAscape. http://www.lumapartners.com/lumascapes/display-ad-tech-lumascape/, 2016.

[82] J. S. Pedro, S. Siersdorfer, and M. Sanderson. Content redundancy in youtube and its application to video tagging. *ACM Transactions on Information Systems (TOIS)*, 29(3):13, 2011.

[83] B. Pinkerton. Finding what people want: Experiences with the WebCrawler. In *In Proceedings of the 2nd World Wide Web Conference (WWW)*, volume 94, pages 17–20. Chicago, 1994.

[84] Y. Qu, J. Wang, Y. Sun, and H. Holtan. Systems and methods for generating expanded user segments, Feb. 18 2014. US Patent 8,655,695.

[85] A. Rajaraman, J. D. Ullman, J. D. Ullman, and J. D. Ullman. *Mining of massive datasets*, volume 1. Cambridge University Press Cambridge, 2012. Chapter 3.

[86] S. Rallapalli, Q. Ma, H. H. Song, M. Baldi, S. Muthukrishnan, and L. Qiu. Modeling the value of information granularity in targeted advertising. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):42–45, 2014.

[87] J. J. Ramos-Muñoz, J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz, and J. M. López-Soler. Characteristics of mobile youtube traffic. *Wireless Communications, IEEE*, 21(1):18–25, 2014.

[88] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 12–12. USENIX Association, 2012.

[89] F. Roesner, T. Kohno, and D. Wetherall. Detecting and Defending Against Third-party Tracking on the Web. In *In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI)*, pages 12–12, Berkeley, CA, USA, 2012. USENIX Association.

[90] J. Shen, S. C. Geyik, and A. Dasdan. Effective audience extension in online advertising. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2099–2108. ACM, 2015.

[91] N. Siddiqi. *Credit risk scorecards: developing and implementing intelligent credit scoring*, volume 3. John Wiley & Sons, 2012.

[92] M. Slaney and M. Casey. Locality-sensitive hashing for finding nearest neighbors (lecture notes). *Signal Processing Magazine, IEEE*, 25(2):128–131, 2008.

[93] X. Wu, A. G. Hauptmann, and C.-W. Ngo. Practical elimination of near-duplicates from web video search. In *Proceedings of the 15th international conference on Multimedia*, pages 218–227. ACM, 2007.

[94] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th international conference on World wide web*, pages 261–270. ACM, 2009.

[95] H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian. Exploiting enriched contextual information for mobile app classification. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1617–1621. ACM, 2012.

[96] H. Zhu, H. Xiong, Y. Ge, and E. Chen. Mobile app recommendations with security and privacy awareness. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 951–960. ACM, 2014.

[97] M. Zink, K. Suh, Y. Gu, and J. Kurose. Watch global, cache local: Youtube network traffic at a campus network: measurements and implications. In *Electronic Imaging 2008*, pages 681805–681805. International Society for Optics and Photonics, 2008.