UNIVERSITY OF CALIFORNIA
SANTA CRUZ

REPUTATION SYSTEMS IN LABOR AND ADVERTISING MARKETPLACES

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Maria Daltayanni

March 2015

The Dissertation of Maria Daltayanni
is approved:

Professor Luca de Alfaro, Chair

Professor Patrick Mantey

SVP Turn Inc Ali Dasdan

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Reputation Systems in Labor and Advertising Marketplaces

by

Maria Daltayanni

Reputation systems in most marketplaces involve two parties, reviewers and reviewees, e.g., employers and employees, buyers and sellers, consumers and advertisers. In this bipartite system, reviewers explicitly or implicitly assign reviewees ratings of their quality, value, or performance, e.g., job competence, product value, level of interest drawn. These ratings are aggregated by the system through a certain mechanism to give reviewees a score for their "reputation" in the community, an indication of their trustworthiness or reliability according to their reviewers. This reputation score can then be used by other people in "choosing" a reviewee (e.g., hiring a worker, buying a product). On the other hand, bias scores are computed for the reviewers, based on the reputation of the items they have rated. This bias can then be used when our interest is in representing the reviewer and his judgements.

In this work, we study the existing reputation systems, and propose new ones, in two different marketplaces: labor and advertising.

In the labor context, we study how reputation systems contribute to smart hiring. In this process, the employer posts a job in the marketplace to receive applications from interested workers. After evaluating the suitability of applicants for the job, the employer hires one or more of them via an online contract. Once the job is completed and the contract ended, the employer can provide the worker with a rating, which becomes visible on the worker's online profile. This explicit feedback can guide future hiring decisions because it (supposedly) indicates the worker's true ability.

However, reputation systems based on end-of-contract ratings have several shortcomings, such as data sparsity, review bias and skewness, and latency in acquiring review signals. Our study builds reputation mechanisms that use Bayesian updates to combine employers' implicit feedback signals of actual application evaluation, e.g., hiring, interviewing, and rejecting, in a link-analysis approach and thus address such shortcomings while yielding better signals of worker quality to inform hiring decisions.

In the advertising context, we study how reputation systems benefit smart targeting. Audience selection is one of the determining factors in bidding success in all forms of advertising. Typically, advertisers select their audiences either by manually targeting rules or by running machine-learning models on users' past performance data. Here, the reputation setting is based on the decision of a user (reviewer) to convert to an ad posted by an advertiser (reviewee).

However, traditional reputation systems based on explicit user preferences (clicks, actions, or conversions) on the advertised content (opinion about the advertised product or advertiser) mainly accumulate such opinions, explicit or implicit (such as search queries or website visits), by representing them in user profiles. This system has several shortcomings, such as poor representation of interest, along with data sparsity and latency in acquiring user review signals.

Therefore, we create a reputation system that estimates the quality of the items preferred by users, on top of which items they prefer, to accurately represent user interest while automating the audience selection process. This system also produces immediate results for the cold start problem of new campaigns by accounting for crowd insights from a pool of similar auctioneers, in a privacy-preserving environment.

Overall, the proposed reputation systems show how challenges regarding transparency,

availability, and privacy are overcome in the two marketplaces, and provide a general framework for addressing cold start in typical reputation environments.

To my husband,

Panagiotis

Acknowledgements

I would like to thank the people who played a critical role for the completion of this dissertation.

First, I would like to thank my academic advisor, Professor Luca de Alfaro, for his continuous support, respect and encouragement during my studies. I always appreciated his deep insights in my work, and his positive attitude. I learned many things from him by his example, e.g., how to use my skills to get things done in research as well as in real life. His belief in me helped me become independent at work and face research uncertainty.

I would also like to thank Ali Dasdan, who co-advised me during the last year of my PhD that I spent at Turn Inc. Ali gave me the opportunity to deal with real problems from industry and under his guidance I learned how to handle real-world datasets and model real-world challenges as research problems. He is an enthusiastic and visionary individual, that has created an excellent work and team environment at Turn.

Professor Pat Mantey provided me with his support and fruitful comments in both my preliminary exam and my dissertation. Professors Alkis Polyzotis and Phokion Kolaitis helped me feel like home in my first steps in UCSC and adapt to the graduate studies environment.

My undergraduate advisor, Professor Yannis Ioannidis, introduced me to database systems and data mining through his excellent teaching, and who guided me in my first research steps. Moreover, my professors, Manolis Koubarakis, Stathes Hadjiefthymiades, Dimitrios Gunopulos introduced to me computer science as a field worthy to devote effort and time.

I would like to thank all of the friends who have been close to me and have shared small or big parts of my journey in research. George Garbis was my collaborator in many of my undergraduate projects. Joel and Karla Barajas were great examples of patience and optimism as they

were facing the double challenge of being both graduate student and parent. Many thanks to my friends in Greece, Eleni Papageorgiou, Ermina Papadopoulou, Konstantina Myta, Maria Messini, who have been believing in me in all of my academic and non academic steps, and my friends, Ioannis Koltsidas, George Papadimitriou, Yannis and Vivi Antonellis, Foivos and Marilena Karachalios, Kostas and Vasiliki Krikellas, Christos and Maria Koufogiannakis, Stefania and Alkis Sellis, Giorgos Karakonstantakis, Kostas and Eirini Morfonios, Petros and Liza Karasakalidis, for their support during my graduate years. I also want to thank my lab-mates, Enela Pema, Lenia Dritsoula, Michael Shavlovsky, Vassilis Polychronopoulos, Pigi Kouki for sharing ideas and challenges in research life.

I would also like to thank my relatives and especially John and Nancy, Antonis and Youla, Antonis and Evangelia and Therapon. I feel their prayers and I think about them every day and night. I want to specially thank my sister, Hara. We grew up together and shared the same room until I moved to the US. Although she is now distant in space, she is always close in my mind and I will always keep loving, thinking and worrying about her. I would specially like to thank my parents in law Evangelos and Diamando. They offered me the love they had for their children, and they got a place in my heart next to my parents. Special thanks to my parents, Vassilis and Pinio; they would always support me and offer me their unconditional love. I owe to them so much for what I am today and they will always serve as role examples in my life.

I will always thank my son, Evangelos, who, through the most innocent and inspiratory way, taught me in one day how to set priorities, and work on everything in a balance. He and my second son, who is coming next month, have helped me to constantly recall that everything in life is a blessing. Finally, I want to deeply thank my husband, Panagiotis. He is my soulmate, a true gift in my life, the one who reflects all what I have dreamt of. He has shared with me the biggest blessings

of my life, and he has been helping me become a better person since the first day I met him. I will always thank him for being the one, who has been inspiring me and sharing the joy and hardships of my journey in life.

Chapter 1

Introduction to Reputation Systems

1.1 Evolution of Reputation Systems

Over the last few years the emergence of social media on the Internet has broadened the sharing of people's opinions within private and public online communities. People share opinions about products or services they purchase, doctors they consult, hotels where they stay and pretty much every aspect of their life where they have to choose among various alternatives. Their opinions are usually publicly available and they provide crowdsourced descriptions of the advantages and the disadvantages of the items they refer to. Such descriptions are quite important in helping other individual make decisions. As an example, let us suppose that a person is looking for a place to have lunch in a city, such as Menlo Park, CA. Considering that the person would be unfamiliar with the city, without Internet access, he would have to resort to finding friends who are familiar with the area to ask them for recommendations. In the case where no one can provide an opinion, the person would have to first arrive at the city before asking local people about their recommendation for a nice place to eat. As all of the local people are unknown and the person in question does not have

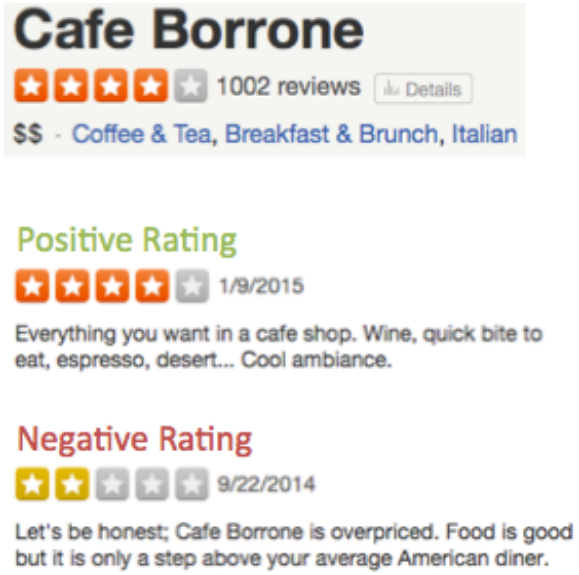

Figure 1.1: Yelp Example of Online Restaurant Reviews

any incentive to distinguish who to trust, he would have to collect a few opinions before deciding which place to select for lunch. If 3 out of 5 people recommend a certain restaurant, it is highly likely that the particular restaurant would be chosen over other suggestions.

Since social media services have become available on the Internet, accessing information about different places, for example restaurants, has become significantly easier. The online search path is somewhat similar to our previous example, look for places that have received a rating and a review by a large amount of people, decide which reviews to trust, and choose one place that accumulates a good overall score, both from friends and from unknown people. The most difficult aspect in this example is obtaining ***trust***. Hence, the reputation of the restaurant is very important to satisfy the information need of the potential customer in order to make the final choice. An example from Yelp, a popular all purpose reputation network, is shown in Figure 1.1.

A lot of issues arise when it comes to reliability of the reputation signal itself. For exam-

ple, a reviewer may always rate restaurants negatively due to the fact that he generally has negative predispositions. Therefore, his reviews would not be helpful, as they would reflect his ***bias*** rather than his true restaurant experience. Furthermore, unless a restaurant is popular and well-known, there won't be a sufficient number of people with the incentive to provide an online rating. For example, at Yelp.com there are often only 1-2 online reviews for a significant percentage of a city's restaurants. Such ***sparsity*** in the reputation signal would not allow a person to get significant information about the true quality of rated items. Even more, as they are concerned about the opinion of the general public, restaurant owners may provide special incentives to customers such as discounts and offers, in order to obtain positive feedback entries. They might even ask friends or colleagues to positively rate their business online in order to increase their overall reputation score. In that case, the reputation of a restaurant would mostly reflect an aggregation of ***spam/noisy*** entries rather than true quality.

In this thesis we study mechanisms that express quality of rated items via reputation scores, facing the issues of bias, sparsity and spam. The restaurant example in the previous section introduces the notion of reputation based on a recent scenario, where technology is prominent in people's lives. Nevertheless, reputation is not a new notion, it can be noted as an existing concern as far back as 1603 A.D,, in "Othello", one of Shakespear's popular theatrical plays. In this play, Othello is a general of the Venetian army with Cassio, his loyal lieutenant, and Iago, the trusted but unfaithful ensign, as his underlings. Iago attempts to undermine Cassio's reputation by spreading a rumor that Cassio had an affair with Othello's wife. When Cassio learns about the rumor he becomes particularly concerned about his reputation, saying "O! I have lost my reputation. I have lost the immortal part of myself, and what remains is bestial." Iago responds by saying: "As I am an

honest man, I thought you had received some bodily wound; there is more offense in that than in reputation. Reputation is an idle and most false imposition; often got without merit, and lost without deserving: you have lost no reputation at all, unless you repute yourself such a loser." Cassio's expressed concern shows the importance of reputation which people perceive as part of one's ***identity***. Iago's response shows that at the same time people realize that reputation does not always reflect true quality; hence someone can define his reputation by his own deeds, even if his true quality may be much higher or lower. The case of Cassio's reputation links to the problem of spam in reputation, which needs to be seriously addressed in online reputation systems. Consequently, in this thesis, we aim to represent the notion of true quality through reputation scores that we compute for online entities, such as workers or advertisements.

In recent years, the advent of Web 2.0 made reputation ubiquitous. As a result, several web services have gained great benefit from reputation systems. In particular, when considering ***search*** engines, the PageRank [14], [51] algorithm has brought order to a previosly slightly chaotic web by ordering the pages based on their in-links and out-links. PageRank implements a reputation system that allows weighs each page's quality according to its connections with other web pages and their quality, which in turn has greatly improved search accuracy performance. In ***e-commerce***, companies such as eBay, [1], Epinions [2],, Bizrate [3], and Trustpilot [4] introduce product review systems which help online users decide on a purchase. In ***programming communities***, such as Advogato [5], Stack Overflow [6] and Freelance marketplaces (Elance-oDesk [7], Freelancer [8]), attack resistant

[1] http://www.ebay.com
[2] http://www.epinions.com
[3] http://www.bizrate.com
[4] http://www.trustpilot.com
[5] http://www.advogato.org
[6] http://stackoverflow.com
[7] https://www.elance-odesk.com
[8] https://www.freelancer.com

trust metrics were developed. Users certify each other in a kind of peer review process and subsequently this information is used to avoid the abuses that commonly plague open community sites. Conversely, these reputation concepts have strong connections with PageRank. Similar reputation systems are used by online ***image hosting*** services such as Imgur [9], where the images content is curated in real time by a dedicated community through commenting, voting and sharing. In the ***internet security area***, systems like TrustedSource [10] were developed to provide reputation scores for Internet identities, such as IP addresses, URLs, domains, and email/web content. Reputation is also an important element in ***Wikis***, for the purpose of increasing contribution quantity and quality. In examples like Wikipedia [11] (a web-based, free-content encyclopedia project based on a model of openly editable content), reputation systems such as WikiTrust [12] have been developed in order to assess the credibility of content and author reputation by using automated algorithms.

In the online ***social networking and news*** area, companies like Reddit [13] and Digg [14], support systems where registered community members can submit content, such as text posts or direct links and then users vote submissions "up"(digging) or "down"(burying) to organize the posts and determine their position on the site's pages. This voting system reflects a reputation mechanism about news pages and posts. Similarly, in ***question-and-answer sites***, such as Quora [15] and Yahoo! Answers [16], up- and down- voting is also supported. In these sites questions are created, answered, edited and organized by the community of registered users, and up- and down- voting regulates the quality of both questions and answers. This approach not only computes reputation about user

[9] http://imgur.com
[10] http://trustedsource.org
[11] https://www.wikipedia.org
[12] http://wikitrust.soe.ucsc.edu
[13] http://reddit.com
[14] http://digg.com
[15] http://www.quora.com
[16] https://answers.yahoo.com

responses, but also gives users the incentive to respond more responsibly, accurately and frequently in order to accumulate high reputation across the platform. At the same time, reputation mechanisms have been developed for online ***email services***, from simple anti-spam techniques which flag emails as dangerous or safe, to more intelligent approaches for reputation lookup of emails, such as the service of RapLeaf [17]. Another example of online reputation is that of ***personal reputation***, as in the example of CouchSurfing [18] (for travelers), a hospitality exchange and social networking website. The website provides a platform for members to "surf" on couches by staying as a guest at a host's home, host travelers, or join an event. The traveler's experience rating becomes an input about the host's reputation and can be used by future travelers to make decisions on host selection and trust. Other ***non governmental organizations*** (NGOs), such as GreatNonProfits [19], GlobalGiving [20] offer services where users can read reviews about non profit organizations. ***Translation services*** companies like BlueBoard at ProZ [21], offer professional reputation of translators and translation outsourcers. Finally, Yelp [22] the well known ***all purpose reputation system***, publishes crowd-sourced reviews about local businesses.

1.2 Reputation in Online Marketplaces

In this section we present a basic framework for describing reputation systems in the context of online marketplaces. An online marketplace is an online platform that facilitates transactions between sellers and buyers. While various types of marketplaces exist we will introduce three types

[17] http://intelligence.towerdata.com
[18] https://www.couchsurfing.com
[19] http://greatnonprofits.org
[20] http://www.globalgiving.org
[21] http://www.proz.com/blueboard
[22] http://www.yelp.com

which will be used as running examples in our framework:

(a) In a ***product*** marketplace such as Amazon.com [23], sellers present and offer their products online, while users can browse products, register to the system and purchase the preferred products at the offered prices. The marketplace fulfills the transaction and item shipping, while also offering other additional services.

(b) In a ***labor*** marketplace such as Elance-oDesk [24], two parties participate; employers and workers. Employers post job requirements and conditions. Workers apply for the job, propose a price, and offer additional information about themselves as well as reasons why they are the best fit for the posted job. Finally the employer selects which worker(s) to hire, usually through interviews which will help uncover additional information about the worker to aid in the task of selecting the most appropriate worker for the job.

(c) In an ***advertising*** marketplace such as Turn[25], advertisers wish to promote products or services and they create advertisements for that purpose. Afterwards they make a decision on ad display and user targeting among the billions of online users (cookies) that are online.

1.2.1 Actors

There are two types of ***actors*** in a marketplace reputation system: the ***reviewers*** and the ***reviewees***. Reviewers post their personal feedback/opinion regarding an item whose quality they wish to rate based on their experience. Reviewees receive feedback from reviewers about their performance quality given the reputation domain. For example, in a product marketplaces like

[23] http://www.amazon.com
[24] http://www.elance-odesk.com
[25] http://www.turn.com

Amazon, the reviewer is the user who purchases a product and returns to the marketplace to post a review about the product usage value or price, that essentially describes his investment satisfaction. The reviewee is the product purchased by the reviewer. In a labor marketplaces, the reviewer-reviewee relationship is bidirectional. When the employer is the reviewer he returns with a review about the worker who accomplished the job, based on his satisfaction about the work. The reviewee is the worker, and he accumulates an overall score based on multiple employers' ratings in multiple jobs where he was hired. In the other direction a worker can also be a reviewer as he can also rate the employer after the job is finished. In this case the employer is the reviewee and he accumulates an overall score based on the ratings of all the hired workers. In an advertising marketplaces, the online user who views the displayed ad is the reviewer. The review can be positive if the user either clicks the ad positively or enrolling / purchasing the related product/service. The review is negatively if the displayed ad produces no action. The reviewee is the advertiser who, based on the users response, will not only gain a return on his investment to show the ad through the ad publishers, but will also obtain a reputation about his targeting ability, that is, how successfully he identifies the right users for the right advertised content.

1.2.2 Reputation Mechanism

In addition to actors, to fully define a reputation mechanism, we also need to provide answers to following three questions:

- *How do we **elicit** reputation signals?* In every reputation system we need to define the signals or ratings that are used as input to the system. These signals can either be explicit, i.e., the answers to questions such as "did you find this search result useful?", or implicit, i.e.,

clicking or not clicking a particular search result. In the product marketplace example, useful signals to gather are either 1-to-5 star ratings of products or text comments that are written by consumers. In labor marketplaces, explicit signals that indicate workers'/employers quality are either star ratings or the text comments provided by the employers/workers. Implicit signals can be the hiring actions of the employers. In the advertising marketplaces example, signals that need to be followed to identify user interest are ad clicks and purchase actions.

- *How do we **aggregate** reputation signals to produce reputation scores?*: There are usually too many base signals to provide a concise summary of the reviwee. Thus, a reputation mechanism needs to include an function or algorithm that can aggregate the base signals and provide a reputation score. In product marketplaces, the average rating is often used to aggregate the reputation signals into one score. In labor marketplaces, weighted average ratings are considered, where the weighting is based on the monetary value of the job, that is the amount of money offered to the worker in total, throughout his collaboration with the employer at the job. Finally, in advertising marketplaces, several techniques are used to measure user interest, such as click-through-rate (CTR), that is how many clicks were given to an ad compared to how often it was viewed by users, and more.

- *Whats is the **use** of the reputation scores?* Although reputation scores have different uses in different marketplaces, they are usually drivers of reviewer's future decisions. Depending on the use case, the reputation mechanism should choose the appropriate base signals and aggregation function to effective respond to the situation. In a product marketplace example, reputation can become very useful towards deciding whether to purchase (or not purchase) a specific product. In labor marketplaces, reputation can significantly aid employers in their

decision which worker to hire for a new job, given the worker's accumulated reputation score at application time. In advertising marketplaces, reputation of advertisers and ads can help advertisers decide which ads are expected to pay off their investment, and which users to target.

In this thesis, we elaborate on a) eliciting new reputation signals and devising novel reputation aggregation algorithms in labor marketplaces, b) devising novel aggregation mechanisms for ads reputation, c) formulating the targeting problem in ads marketplaces as a reputation problem and using reputation information to solve this problem.

1.3 Organization

This thesis is divided into two parts. In the first part of the thesis (Chapter 2), we focus on reputation systems in online labor marketplaces. In these marketplaces an employer posts a job to receive applications from interested workers. After evaluating the match to the job, the employer hires one (or more workers) to accomplish the job via an online contract. At the end of the contract, the employer can provide the hired worker with a rating that becomes visible in the worker online profile. This form of explicit feedback guides hiring decisions of future employers, since it is indicative of the worker's true ability. In this thesis, we initially discuss the shortcomings of existing reputation systems that are based on end-of-contract ratings. Then we propose a new reputation mechanism that uses implicit quality judgments provided by the employer at application evaluation time rather than at the end of contract. To aggregate these signals, in Chapter 2 we propose an aggregation mechanism that is inspired by Elo ratings used to score chess players. The proposed approach addresses the shortcomings of the existing 5-star rating system, and it yields

better signals for predicting worker hires.

In the second part of the thesis (Chapter 3) we study a reputation problem in an advertising marketplace. In particular we focus on the problem of selecting the right audience for an advertising campaign which is one of the most time-consuming and costly steps in the advertising process. To target the right users, an advertiser needs to either perform user studies to identify population segments of interest or to devise sophisticated machine learning models trained on data from past campaigns. In this section we study how a Demand Side Advertising Platform (DSP) company like Turn[26] or Rocketfuel[27] can leverage the data it collects from various advertisers and user actions to recommend target user segments for both new and existing advertisers. We propose a reputation system for advertisers and users that on one side quantifies the ability of advertisers to select the right audience and on the other side analyzes the propensity of users to respond to advertising messages. We also present a decision-tree-based algorithm that uses the reputation system and advertiser similarity to yield transparent targeting rules that are easy for the advertiser to interpret and refine. Finally, we perform an extensive experimental evaluation on a real dataset from Turn that showcases the benefits of our approach for both new and existing advertiser campaigns.

At the end of each of the aforementioned parts we present other work related to the chapter topic. Finally in Chapter 4 we present our closing remarks and discuss potential future work.

[26] http:://www.turn.com
[27] http://www.rocketfuel.com

Part I

Reputation Systems in Labor Marketplaces

Chapter 2

WorkerRank: Using Employer Implicit Judgements to Infer Worker Reputation

2.1 Introduction

In online labor marketplaces, such as Elance-oDesk [1] and Freelancer [2], two parties are involved; employers and workers. Employers post job openings and candidate workers apply to them, based on their qualifications, skills and interests. The employers review the applicants' online resumes, and interview few applicants to decide hiring. The worker reputation, i.e., the ratings that the worker has received in his past jobs in the platform, is one of the most important considerations for the employer hiring decision, since it reveals how other employers evaluate the worker true ability in real job scenarios. Although the reputation information is a useful signal, reputation scores are usually *skewed* towards high ratings [33], because employers care about the impact of their

[1] http://www.elance-odesk.com
[2] http://www.freelancer.com

feedbacks on the workers' future opportunities for jobs in the marketplace. The skewed distribution of ratings makes them less helpful in identifying very competent workers.

The reputation signal is also very *sparse*, since a worker needs to apply, get hired and complete few jobs to obtain a representative reputation score. Usually, an unknown rating implies that we have no explicit information about the employer's preference for the worker. In that case we need to build a model to predict the unknown information, or, alternatively make inferences from the employer's behavior[4].

To address the limitations of the existing reputation systems in labor marketplaces, we present *WorkerRank*, a new reputation system that leverages employers' implicit judgements at the application evaluation moment, rather than the employer's explicit feedback at the job completion moment. Although the implicit judgements are more noisy than the explicit ones, they are more broadly available, since the number of applications is usually one to two orders of magnitudes higher than the number of hires. Moreover, the implicit actions of the employers are not revealed and, consequently, the employers do not bias their judgements towards high ratings (as happens when they aim to avoid the negative impact on the workers). As a result, the obtained ratings are not skewed.

We consider an employer decision to hire worker A, thus ranking A above some other candidate B, as an input that "A won over B" in a match. The employer decisions can thus be interpreted as a set of match outcomes. There are many algorithms ([26], [29], [32], [56]) that can be used to aggregate match outcomes. Our reputation system builds upon the Elo ratings system[26] that is widely used to evaluate chess players. In particular, we assign each worker an initial rating and we treat the applicants to a job opening as the participants in a chess tournament. Applicants

that get hired get their scores increased and those who are rejected get their scores decreased. The extent of the increase or the decrease depends upon the ratings of the other applicants, i.e., the better the rejected applicants are, the more the rating of the hired worker increases. Similarly, the worse the hired applicants are, the more the ratings of the rejected applicants decrease.

To deal with the noise of implicit judgements, we assign each employer a score that quantifies the agreement of his decisions with the observed quality of the workers. We then use the obtained scores to weigh the employer judgements. For example, if an employer tends to take decisions that are very different from the rest of the employers, his score will be low and his hiring decisions will have a small impact on the worker ratings. The rest of the paper is organized as follows. In Section 3.2 we present some notation and in Section 2.3 we introduce WorkerRank, the new proposed reputation system. We evaluate the new reputation approach on a real-world dataset from oDesk in Section 3.4. Our results show that the new reputation system not only provides information for far more workers in the marketplace, but it also serves as a better discriminatory signal for hiring decisions. In Section 2.5 we discuss some related work and we conclude in Section 3.6.

2.2 Notation

We represent the labor marketplace data with a directed bipartite graph $G = (U, V, A)$ (Figure 2.1); U is the set of jobs posted by employers within a specific time period; V is the set of workers who applied to the posted jobs (see Figure 2.1). Edge $(v, u) \in A$ represents the application of worker $v \in V$ to job $u \in U$. Edge $(u, v) \in A$ represents the employer action on the the worker's application. We consider the following six employer actions:

- *hire*, the employer hires the worker;

- *interview*, the employer contacts the worker to obtain a better understanding of his skills, but the worker is not hired;
- *shortlist*, the employer shortlists the worker for future consideration, but the worker is not invited for interview;
- *ignore*, the employer reviews the worker online resume, but he takes no action on it;
- *hide*, the employer reviews the worker resume and he "hides" the applicant without notifying him; and
- *reject*, the employer reviews the worker resume and notifies him that he will not be considered for the job.

Note that if an employer is never presented with a worker's info, no action is logged in our dataset. Among the six actions, we consider the first three as positive indications of the worker ability, and the last three as negative. We also assume that the employer actions indicate a ranking on the applicant perceived ability to accomplish the job in the following decreasing order: hire $>$ interview $>$ shortlist $>$ ignore $>$ hide $>$ reject. For example, a worker that is selected to be interviewed, is considered to be a better fit for the job than a worker who is ignored. The goal of this paper is to compute a score $r(v)$ for each worker v that is informative of his ability to accomplish the jobs that he applies to. Score $r(v)$ is considered informative if the relative difference between $r(v)$, $r(v')$ for workers v and v' is predictive for the relative ranking of v, v' in the future jobs that they apply.

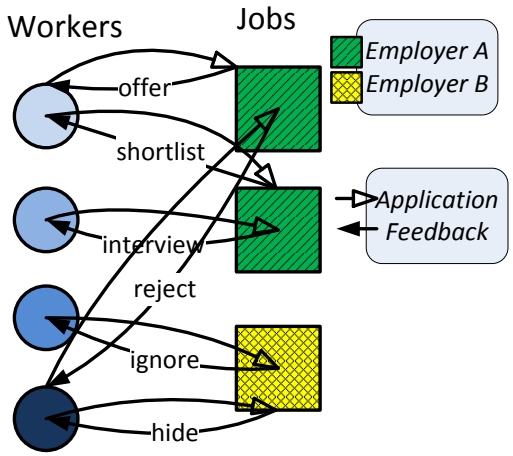

Figure 2.1: Bipartite graph between workers and jobs posted by employers

2.3 Proposed Reputation System

In this section we describe a reputation system that builds upon the employer decisions on the worker applications. In Section 2.3.1 we provide our generic approach and in Section 2.3.2 we show how we can improve our scores by leveraging job specific information. Finally, in Section 2.3.3 we discuss how we can combine our reputation system scores with the end-of-contract ratings to obtain a hybrid reputation system.

2.3.1 WorkerRank

The WorkerRank reputation system assigns *reputation* scores $r(v)$ to each worker $v \in V$. Along with reputation scores, WorkerRank also computes an *importance* score $b(u)$ for each opening $u \in U$ that reflects how a job is important in terms of how objective its employer is when judging candidates. The scores are computed via a reputation calculation process on the application graph G, using the Elo constants for t_{elo}, K, as shown in Algorithm 1.

Comparing Performances: The intuition of simulating jobs by tournaments, and looking at the worker performances at each job in pairwise manner, offers a dynamic way for comparison; first, higher label workers clearly win lower label opponents (for example, hire wins over interview, shortlist wins over hide). Also, draws in positive label workers provide useful information about how their qualities compare. While a draw between two hired workers is an instance of equality between their qualities, the same does not necessarily hold in the case of draws between negative label workers (such as two rejected candidates). That is because in certain cases, rejecting or hiding a candidate may reflect the fact that the employer had offered only a limited number of positions hence he had to reject some good quality candidates. Other similar exceptions may apply too. Hence in our algorithm we exclude draws among negative label workers.

Algorithm: In step 1 we initialize reputation $r(v)$ of each worker v to 1.0 and importance $b(u)$ of each opening to 1.0. Then, in steps 3 - 13 we consider each job as a tournament and in steps 5 - 13 we update the worker scores by considering every pair (v, v') of worker applications at a job u to be a game in the job tournament with possible outcome of matches:

$$t(v, v', u) = \begin{cases} 0, & \text{if } v \text{ lost against } v' \text{ at job } u \\ 0.5, & \text{if } v \text{ came to draw with } v' \text{ at job } u \\ 1, & \text{if } v \text{ won against } v' \text{ at job } u \end{cases} \tag{2.1}$$

At step 3 we initialize the outcome variables $T_{v,u}, X_{v,u}$ to 0 for each candidate v who applied to job u. At step 5, we compute $T_{v,u}$ as the sum of the actual points that v scored in job u against the other opponent candidates. At step 6, we compute $X^i_{v,u}$ as the sum of expected points that v would earn at time i against each opponent candidate $v' \neq v$ at job u, according to Elo's formula[26]:

$$t^i_{elo}(v, v', u) = \frac{1}{1 + 10^{(r^i(v') - r^i(v))/400}}, \forall v' : (v', u) \in A \tag{2.2}$$

For example, consider workers v_1, v_2, v_3 and v_4 who applied to a job; v_1 gets an offer, v_2 is

Algorithm 1 WorkerRank: Compute Workers Reputation Scores and Jobs Importance Scores

Input: Graph $G = (U, V, A)$

Output: Reputation scores $r(v)$ for workers $v \in V$, importance scores $b(u)$ for jobs $u \in U$

Initialize $i = 0$; $r^0(v) = 1, \forall v \in V$; $b^0(u) = 1, \forall u \in U$

for $u \in U$ **do** ▷ For each job tournament in chronological order

  $T_{v,u} = 0, X^i_{v,u} = 0, \forall v : \exists (v, u) \in A$

  for $v, v' : (v, u) \in A, (v', u) \in A, v \neq v'$ **do** ▷ For each worker-applicant pair

    $T_{v,u}$ += $t(v, v', u)$, and $T_{v',u}$ += $t(v', v, u)$

    $X^i_{v,u}$ += $t^i_{elo}(v, v', u)$, and $X^i_{v',u}$ += $t^i_{elo}(v', v, u)$

  end for

  $i \leftarrow i + 1$

  for $v : (v, u) \in A$ **do** ▷ For each worker who applied to job u

    $\delta^i(v, u) \leftarrow b^{i-1}(u) \cdot (T_{v,u} - X^{i-1}_{v,u})$

    $r^i(v) \leftarrow r^{i-1}(v) + \frac{K}{\binom{n}{2}} \cdot \delta^i(v, u)$ ▷ Update reputation

  end for

  $b^i(u) \leftarrow \frac{n^i_c(e,u) - n^i_w(e,u)}{n^i_c(e,u) + n^i_w(e,u)}$ ▷ Update job importance

end for

interviewed but never hired and v_3 and v_4 are rejected without interview. Each applicant participates in 3 games versus the other applicants. Worker v_1 wins all three games versus v_2, v_3 and v_4, since he received an offer which is the most positive employer judgement. Worker v_2 loses against v_1 but wins over v_3 and v_4. Finally, each of the workers v_3 and v_4 loses in the games against v_1 and v_2 but they draw when they face each other. The worker points in this job are 3 for v_1, 2 for v_2 and 0.5 for

either of v_3 and v_4. At step 10, the rating update $\delta^i(v, u)$ of worker v due to his application to job u at the i-th time step is calculated as follows:

$$\delta^i(v, u) = b^{i-1}(u)(T_{v,u} - X_{v,u}^{i-1}) \tag{2.3}$$

where $b^{i-1}(u)$ is the importance score of job u from the $i-1$-th step, $T_{v,u}$ is the sum of the actual points that v scored in job u and $X_{v,u}^{i-1}$ is the sum of points he was expected to score based on his rating and the ratings of the other applicants from the previous step. Time steps advance at each new job occurrence. Note that the score update is multiplied by the importance of the job, $b(u)$, so that employer bias is taken into account, as shown in Equation 2.5. Also note that the more the applicants in an opening, the more the expected points, since the update includes a summation term for each applicant. What is more, the higher the difference between the rating of worker v and the ratings of the other applicants of job u, the more the points that v is expected to score. This is particularly useful since application success of an applicant is not independent from the application success of the remaining candidates at a particular job. Finally, to obtain the rating of worker v at the i-th time step, at step 11, we add the average of his partial rating updates (Equation 2.3) to his rating from the previous time step, $r^{(i-1)}(v)$:

$$r^i(v) = r^{i-1}(v) + \frac{K}{n-1} \cdot \delta^i(v, u) \tag{2.4}$$

The K-factor which represents the maximum possible adjustment per game is set to 32, and is normalized by dividing over $n-1$, where n is the number of the job applicants. Normalization is completed by the summation over all pairs of candidates. Normalization is important, since without it, we would have dramatic inflation/deflation of scores at jobs with high application rate. Popular jobs usually require generic skills, hence inflated scores would not necessarily imply that

the quality of their hired workers is higher. A worker's reputation is updated according to the average win/loss score incurred in comparisons with other applicants to each job. This average ensures that applicants gain according to their relative position in the applicant ranking, rather than the number of applicants. Note that since δ can also be negative, applicants to many jobs being rejected by many employers is not necessarily a good strategy. After calculating the worker ratings, at step 13 we compute the importance scores of jobs $b(u)$. The intuition in Formula 2.5 is to give more credence to rating updates (Equation 2.4) that come from jobs posted by unbiased employers. The importance score is high for employers who make decisions that respect the worker ratings and it is low for employers who do not. Notice that after the occurrence of a small amount of job tournaments, the worker ratings is an outcome taken out of the aggregation of all employers judgements (step 11). At the same time, Elo scoring is based on a self-correcting rating system[1]. Hence the following formula reflects importance of a job as a measure of judgement deviation between the job's respective employer and the rest employers:

$$b^i(u) = \frac{n_c^i(e,u) - n_w^i(e,u)}{n_c^i(e,u) + n_w^i(e,u)} \tag{2.5}$$

$e \in E$ denotes employer (in the set of employers E) who posted job $u \in U$, $n_c(e,u)$ denotes the number of applicant pairs that were "correctly" ranked by employer e in job u and $n_w(e,u)$ denotes the number of applicant pairs that were "wrongly" ranked. We regard a pair of applicants as correctly ranked if the employer prioritizes the applicant with the highest reputation score. For example, if applicants v and v' have ratings $r(v) = 1$ and $r(v') = 2$ and employer e hires v' and rejects v, then the pair is considered to be correctly ranked.

2.3.2 Skill WorkerRank

The approach described in Section 2.3.1 predicts a reputation score for each worker based on application data. That score is computed in a global scope over all jobs where workers have applied. Although global scores provide a signal for the quality of workers, they may not be as powerful to discriminate among similar score workers and guide hiring with accuracy. For example, consider candidates v_1, v_2 in the example of Figure 2.2. The skills listed under each job reflect the skill-set required for the job. Also assume that v_2 is better in java than v_1, however v_1 is better overall than v_2, which can be due to the fact that v_1 has applied to more jobs where he received offers. At this point our reputation system would prioritize v_1 in the candidates list, missing the fact that v_2 is better in java.

Our goal is to achieve higher accuracy at predicting worker quality and prioritize candidates appropriate for the particular job. As mentioned above, besides the information regarding which worker applied to which job, there is further information regarding what skills each job requires and what skills each worker claims in their profile description. Given this information, we use WorkerRank to derive scores for candidates in a skill-wise fashion, such that eventually we learn how good each worker is at each particular skill. Then, if a job requires a skill, we may rank candidates according to their reputation scores at that particular skill and suggest the top ranked ones for getting hired. In the used example, workers v_1, v_2 who both claim to be experts in java (and they apply to job u_2 which requires java) will obtain a java score that will show their quality in that skill. The expectation is that ranking v_2 on top will lead to successful hiring decision.

Skill-wise reputation algorithm: We consider set of skills S, where $S_u \subseteq S$ denotes the skills required for job $u \in U$ and $S_v \subseteq S$ denotes the skills claimed by worker $v \in V$. Also, we

consider bipartite graph $G_S = (U \times S, V \times S, A_S)$ similar to the definition in Section 2.3.1, where:

- each worker node v is replaced by set of pair (worker, skill) nodes (v, s), one node for each skill claimed by the worker
- each job node u is replaced by set of pair (job, skill) nodes (u, s), one node for each skill required for the job
- each (worker, job) edge (v, u) is replaced by set of pair ($\{$worker, skill$\}$, $\{$job, skill$\}$) edges, $(\{v, s\}, \{u, s\})$, one edge for each skill that the worker claims and the job requires.

Then we run Algorithm 1 on G_S. The reputation and importance scores are derived skill-wise, such that we obtain a set of reputation scores $r(v, s),$ where$(v, s) \in V \times S$ and a set of importance scores $b(u, s),$ where$(u, s) \in U \times S$. Obtaining reputation scores for each $\{$worker, skill$\}$ pair provides information about the performance of the worker at the particular skill. Figure 2.2 describes an example for graphs G and G_S, with candidates v_1, v_2, v_3 applying to jobs u_1, u_2. Workers v_1, v_3 apply to job u_1 (v_1 receives an offer) and v_1, v_2 apply to job u_2 (v_2 receives an offer). The skills required for job u_1 are $\{$python, django$\}$ and the skills required for job u_2 are $\{$java$\}$. Worker v_1 claims to have skills $\{$python, java django$\}$, v_2 claims $\{$java$\}$, and worker v_3 claims $\{$python, django$\}$.

Correlation between skills and hires: Looking at the application data (enriched with skills information) we observe that in most cases jobs require more than one skills. In that case, we need to decide a ranking for candidates based on the intersection of their scores on a set of different skills. That ranking will reflect their suitability for the multi-skill requiring job. In our example where job u_1 requires python and django, we need to rank candidates according to their quality in python and their quality in django. However, the python-score may be more informative about

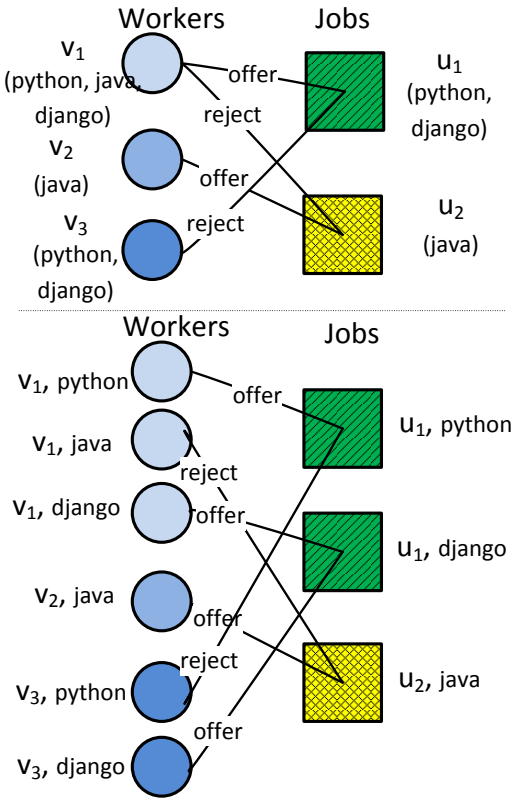

Figure 2.2: Skill-wise bipartite graph

hiring than the django-score. For example, it may more beneficial to hire a candidate with a high python-score than hire one with a high django-score. Hence it is important to measure how each skill contributes towards hiring, before we rank candidates according to skills.

In order to combine a set of scores for each worker across the set of skills required for a job, we allow for a weighted average over the worker's skill-wise scores. We use logistic regression to compute coefficients for skills as features, where we use the binary outcome of the application (hire/no-hire) as the response variable. Coefficients for skill scores will eventually show how informative each skill is about the quality of the worker, measured by the worker's potential of getting hired.

In Algorithm 2 we aggregate skill-wise scores into a single reputation/importance score for each worker/job respectively. The input of the new algorithm is the set of scores derived by Algorithm 1 for the sets of {worker, skill}, {job, skill} pairs. The output of Algorithm 2 is the final reputation score for each worker and importance score for each job, after examining workers' quality across their skill-set and tuning it according to the significance of each skill.

In step 1 we consider the set of features F, where one's reputation score $r(\cdot, s)$ at a particular skill $s \in S$ is regarded as a feature. We use $f(s) = r(\cdot, s)$ to denote feature regarding reputation score at skill s. For example, if $s =$ python, then any (denoted by '$\cdot$') worker's score in python, $r(\cdot, \text{python})$, is a feature. Recall that the coefficients pertain to how each skill score of a worker contributes towards his getting hired. In step 2 we consider response variable y to be the binary hiring outcome $y \in \{hire, no\text{-}hire\}$. Then in step 3 we run logistic regression (LR) on the set of features F with response variable y and we obtain coefficients $w(f)$ for each skill-score variable $f \in F$. Finally, in steps 5 - 6 we aggregate the input skill-wise scores using weights across

Algorithm 2 Combine Skill-wise Scores into Reputation

Input: Set of skill-wise scores $r(v, s)$, $b(u, s)$, where $(v, s) \in V \times S$, $(u, s) \in U \times S$

Output: Reputation scores $r(v)$ for workers $v \in V$, Importance scores $b(u)$ for jobs $u \in U$

1: Consider the set of feature variables $F \leftarrow \cup_{s \in S} f(s)$, where each feature variable corresponds to the skill repuation $f(s) \leftarrow r(\cdot, s), \forall s \in S$

2: Consider response variable $y \leftarrow$ hiring outcome, where $y \in \{hire, no\text{-}hire\}$

3: Learn coefficients $w(f) \leftarrow LR(F, R), \forall f \in F$

4: **for** $v \in V$ and $u \in U$ **do**

5: $\quad r(v) \leftarrow \dfrac{\sum_{s \in S} r(v, s) \cdot w(s)}{\sum_{s \in S} w(s)}$

6: $\quad b(u) \leftarrow \dfrac{\sum_{s \in S} b(u, s) \cdot w(s)}{\sum_{s \in S} w(s)}$

7: **end for**

skills learned at step 3. The output of the algorithm is a single reputation score for each worker (step 5) and a single importance score for each job of the application data (step 6).

2.3.3 Hybrid Model

While it is interesting to compare implicit judgements against explicit judgements in order to infer a quality measurement for workers, we expect that a hybrid model which combines both, shall yield better results. In this section we use rank aggregation to combine WorkerRank ranking with feedback ranking into an optimal listing of workers such that we predict true ranking (as specified by employer judgements) with higher accuracy.

In particular, we use the weighted rank aggregation method described in [53]. In this approach the function performs rank aggregation via a Cross-Entropy Monte Carlo algorithm. The algorithm searches for a desired list which is as close to the provided ordered lists as possible. In

our implementation we use the Spearman distance to measure the correlation of the ordered lists of implicit and explicit reputation rankings. The convergence criterion used is the repetition of the same minimum value of the objective function in a number of consecutive iterations.

2.4 Experimental Results

To evaluate the proposed reputation system, we compare WorkerRank with baseline schemes in terms of a) the sparsity of the signal in the marketplace, b) the time needed to obtain a signal for new workers, and c) discriminatory power for hiring decisions. During the evaluation, WorkerRank is compared against a baseline collaborative filtering approach which uses data of *implicit* reputation to rank workers (as a reminder, WorkerRank also runs on implicit reputation data). In addition, we compare WorkerRank against the current ranking approach that is based on *explicit* reputation data.

2.4.1 Setting

Dataset: We use a sample of real-world application data, along with explicit reputation scores provided by oDesk[3]. The oDesk dataset spans the time period of 53 weeks between January 2013 through January 2014 and it contains approximately 10M applications submitted by 0.5M workers to 1.1M job openings posted by 0.2M employers. Note that we do not account for unseen applications (case where employer has taken no action). In table 2.1 we provide some statistics regarding the dataset. In the experiments we use a sample of the applications submitted during the testing period. Tables 2.2 and 2.3 show a real job posting example. This example includes information about the job title, category, description, required skills, candidate applicants along

[3] http://www.odesk.com

with their declared skills, hire decision, reputation scores learned via WorkerRank and via skill-based WorkerRank. We observe overlapping skills among the job required skills and the skill-sets declared by the candidates. In Figure 2.4 we show the distribution of the ratings data. As studied in [33], we encounter skewness towards the high rating values.

Baseline - Explicit Data: A baseline approach currently used in the marketplace is to represent the quality of workers based on *explicit* data; in particular, it collects the explicit star ratings assigned to each worker by the employers according to their performance on accomplished jobs, and aggregates them in an average ratings score. Then the workers quality is estimated according to the average employers judgements. The average employers rating $q(e, v)$ is used to rank candidate workers $v \in V$ who apply at a new job posting of employer $e \in E$, as follows:

$$q(e, v) = \frac{1}{N} \sum_{e':(v,u') \in A} \sum_{u' \in U_{e'}} q(e', v) \tag{2.6}$$

where $(v, u') \in A$ denotes application of worker v to job u' posted by employer e', $U_{e'}$ is the set of jobs posted by e', $q(e', v)$ is the explicit ratings score in $[1, 5]$ assigned to v by e', and $N = \sum_{e'} |U_{e'}|$ is the number of accomplished jobs, posted by employers e'. Note that employers provide explicit reviews only to workers who have accepted their offer and upon completion of the job.

Baseline - Implicit Data: In certain cases, reputation systems are used to provide rating scores in recommendation problems. Since WorkerRank is applied on implicit reputation data represented on a hiring actions bipartite graph, we use a collaborative filtering score scheme as one of the baseline approaches, viewing WorkerRank as a collaborative filtering approach based on implicit data. Our baseline for representing quality of workers based on *implicit* reputation data, is using collaborative filtering as follows. We consider employers e, e' similarity $sim(e, e')$ and we collect the sets of jobs at which workers received an offer within the study training period. Then

Table 2.1: Dataset Statistics

	Training	Testing
Application dates	$03/01/2013-$ $03/01/2014$	$03/02/2014-$ $18/02/2014$
#Jobs	$1,151,859$	$1,446$
#Workers	$477,464$	$21,642$
#Employers	$232,014$	$1,405$
#Applications	$9,214,557$	$34,054$
Avg # cands / job	24.1	22.5
Min # cands / job	11	11
Max # cands / job	50	50
Median # cands / job	23	20

for each employer we count the total number of offers assigned to each worker by similar employers in the training set and we estimate worker quality score according to their hire rate, weighed by employer similarity. For employer e who posts a new job u, we compute recommended score for candidates v as the hire rate in jobs posted by employers e', neighbors of e, weighed by their similarity, $sim(e, e')$. Denoting employer implicit actions (offer, interview, reject, and more) by $q_{im}(e', v)$, we consider the following baseline implicit reputation score:

$$q_{im}(e, v) = \frac{1}{\sum_{e'} |sim(e, e')|} \sum_{e':(v,u)\in A} \sum_{u\in U_{e'}} sim(e, e') q_{im}(e', v) \tag{2.7}$$

where $q_{im}(e', v)$ takes values in $\{0$ if v was not hired, 1 else$\}$. *Similarity* between employer e and neighbor e', $sim(e, e')$, is defined by their cosine similarity based on their ratings on workers that e and e' have co-rated. Then e receives recommendations from the neighbor e' judgements [4].

In the testing phase we rank candidates by collaborative implicit hire rates and by explicit ratings reputation to recommend the top ranked workers for the new job openings. Then we compare the two baseline performances with WorkerRank performance.

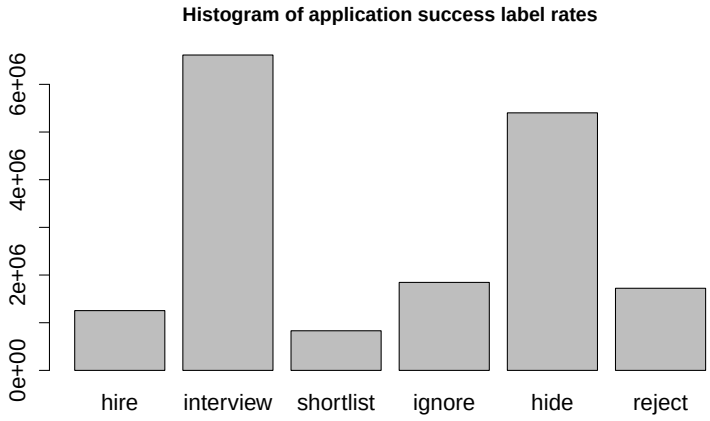

Figure 2.3: Histogram of application success label rates

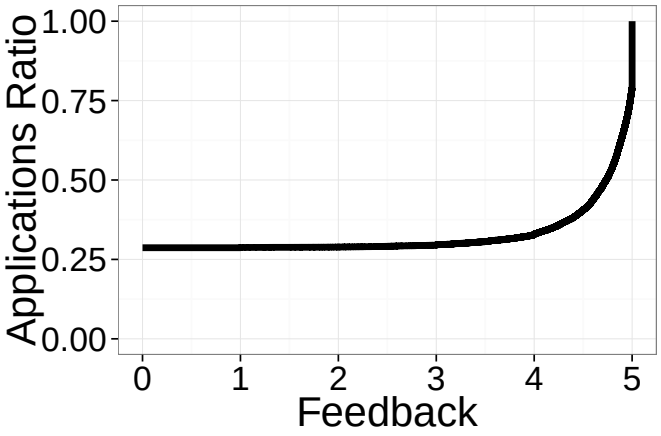

Figure 2.4: Data Skewness

Table 2.2: Job Posting Example (Applicants shown in table 2.3)

Title	*Wordpress Developer*
Category	*Web Development*
Required Skills	*wordpress, css, php*
Description	*1.Need Wordpress theme developed. Slider, logo, left sidebar menu, copyright.* *2.Must be experienced Wordpress developer.* *3.Must know CSS, php, Wordpress, html.* *4.Will provide visual design guide.*

Table 2.3: Skill-based reputation versus global reputation scores

Cand.	Success	Skill 1	Skill 2	Skill 3	Skill 4	Skill5	reput	skill-reput
v_1	offer	css3	php	wordpress	html	css	2.54	2.28
v_2	**offer**	css3	php	wordpress	html5	css	1.42	**4.67**
v_3	offer	web dev	sof-dev	–	–	–	1.39	2.05
v_4	**reject**	css3	php	wordpress	html5	ajax	**4.82**	3.40
v_5	reject	gr.design	vis-c++	wordpress	web des	illustr	1.38	1.06
v_6	reject	css3	php	javascript	html	jquery	1.05	2.45

Table 2.4: Performance Improvement: WorkerRank vs Baselines

	Explicit	Implicit		% Improv	
	Rating Based(RB)	Coll.Filt.(CF)	WorkerRank(WR)	WR vs CF	WR vs RB
MAP	0.24	0.23	0.32	**+39.1%**	**+33.3%**
AUC	0.59	0.52	0.65	**+25.0%**	**+10.1%**
Coverage	59.4%	–	80.8%	–	**+50.8%**

2.4.2 Coverage

First, we show that since WorkerRank's results become available at the time of application, the coverage of workers for whom we obtain reputation signal is higher compared to the coverage obtained from explicit ratings. In particular, we run WorkerRank over the applications of the first 52 weeks of the dataset. During this time period we also keep track of the feedback ratings that the workers receive after the job is completed. Then we report the number of applications of the 53-rd week for which there is a WorkerRank score versus the applications for which there is an employer feedback score. Our results show that out of 88, 294 applications in the 53-rd week, we have WorkerRank scores for 79, 083 (89.6%), while we have feedback scores only for 52, 471 (59.4%). The increase in the marketplace application coverage is +50.8%. We present these results in table 2.5. Note that the above measurements account for both active and inactive applications.

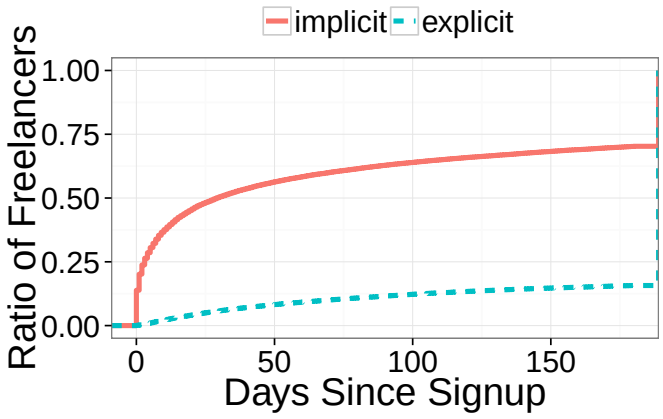

Figure 2.5: Cold Start: WorkerRank vs Explicit Feedback

2.4.3 Cold Start

Second, we show that WorkerRank is faster in acquiring signal for new workers joining the system, compared to the explicit ratings approach. Since the online marketplaces grow fast, the identification of new competent workers is very significant for their healthy development. For all workers who joined the platform during the last 3 months of our study period, we calculate the percentage of workers for whom we obtain reputation signals within X days. X is varying from 1 to 120 days. As presented in Figure 2.5, the WorkerRank scores are available for more than 60% of the new workers within 3 months of their joining the platform and the percentage ratio grows to 75% after another 4 months. On the contrary, there are less than 10% of new workers who received feedback at the end of their first 3 months in the platform and this percentage does not exceed 18% at the end of the additional 4-month period.

Figure 2.6: MAP in predicting the hiring outcome

2.4.4 Ranking Precision

Third, we show that WorkerRank outperforms baseline approaches accuracy in ranking workers by quality, thus yielding a more reliable system for ranking candidates in new job openings.

WorkerRank vs Implicit vs Explicit Baselines:

MAP: We use Mean Average Precision (MAP) to evaluate rankings produced by the baseline approaches and the WorkerRank scores. The truth rankings each method is compared against, are specified by employer true scores that they implicitly assigned to candidates through their hiring actions in past jobs. As shown in Figure 2.6, by using WorkerRank, employers encounter approximately 1 good worker for every 3 workers in the ranked list. That performance is compared against 1 good worker for every 5 (or, 4 respectively) workers that the employer encounters by using the baseline collaborative filtering ranking (or, explicit ratings, respectively). Overall, the new algorithm improves the chances of identifying good workers in the top results by 39.1% (33.3%, respectively).

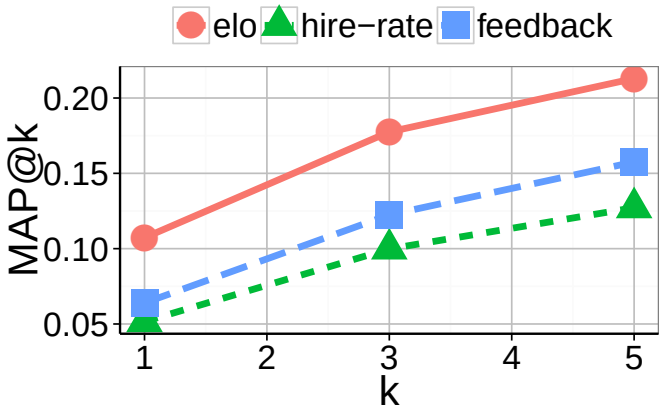

Figure 2.7: MAP@k in predicting the hiring outcome

Lift: To evaluate the quality of WorkerRank scores, we compare them against the explicit reputation scores as signals for taking hiring decisions. We use the data of the first 52 weeks of our dataset to calculate the WorkerRank scores, and we then use these scores as predictors for the hiring outcomes of the applications submitted during the 53-rd week. In particular, we rank all of the applications by the WorkerRank scores of the applicants. Then we calculate the hiring *lift* in the top x percent of the applications as follows:

$$lift(x) = \frac{\text{hiring probability in the top-}x\%\text{ applicants}}{\text{hiring probability across all applicants}} \tag{2.8}$$

Lift shows the performance of WorkerRank versus the performance of a random scoring of applicants. Similarly, we calculate the lift for the explicit ratings scores and we present the results in the bar-plot of Figure 2.8. The plot has five triplets of bars and each triplet looks at a different percentage value of the top ranked workers, $x \in \{0.25, 0.35, 0.5, 10, 25\}$. Left bars look at the Elo ratings obtained by WorkerRank, center bars look at the explicit feedback ratings, while right bars reflect random worker ranking. The height of each bar shows the lift value for the corresponding

scheme and for the corresponding x value. For example, the first left bar from the left shows that the top-0.25% of applicants as ranked by the Elo ratings are 2.66 times more likely to be hired than a random applicant. We observe that the lift of the explicit feedback scores is flat at 1.1 for all x values, since the top 25% of the applications correspond to workers with perfect 5-score rating. As a result, the existing reputation system does not provide a sufficient signal to discriminate high quality workers. On the other hand, WorkerRank Elo ratings yield an increasing lift as we limit the percentage of the top-$x\%$ applications that we consider. The Elo lift is already higher than the feedback lift for $x = 25\%$ and it exceeds 2.5 as we limit x to the top 0.25% of the applications. Note that for an average of 36 applications per job, that percentage implies that the top 9 candidates are 2.66 times more likely to be hired than any random candidate, as opposed to the almost equal likelihood that explicit feedback yields (1.1 times more likely than random) .

Performance across Job Types: We also illustrate how WorkerRank outperforms the baseline approaches when studied in category subsets of the datasets. Figure 2.9(a) shows how the algorithms perform when applied on the different segments of jobs, Knowledge Processing Outsourcing (KPO) and Information Technology (IT). Figure 2.9(b) shows how the algorithms perform when applied on different type jobs (fixed-price (fp) versus hourly-rate (hr)). An interesting illustration is that of Figure 2.4.4, which shows how the different approaches behave on different job categories. For example, employers in software engineering jobs appear to provide more precise feedback about the quality of workers they have collaborated with, whereas in sales and marketing feedback ratings do not reflect the overall quality of workers as accurately.

Hybrid: WorkerRank and Explicit Reputation: In Figures 2.11(a) and 2.11(b) we show how the hybrid approach performs at MAP and MAP@k: for $k \in [1, 5]$, compared to the two

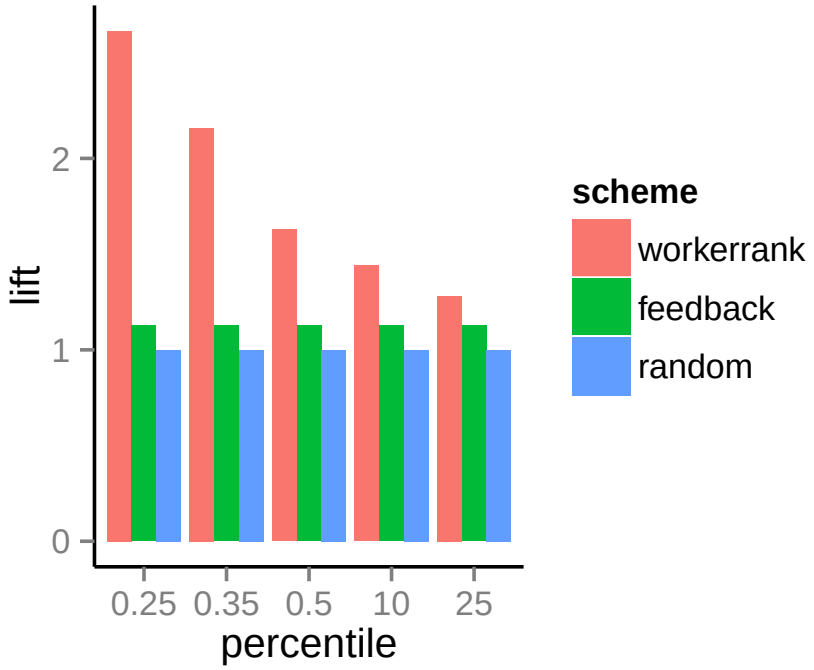

Figure 2.8: Lift in predicting the hiring outcome

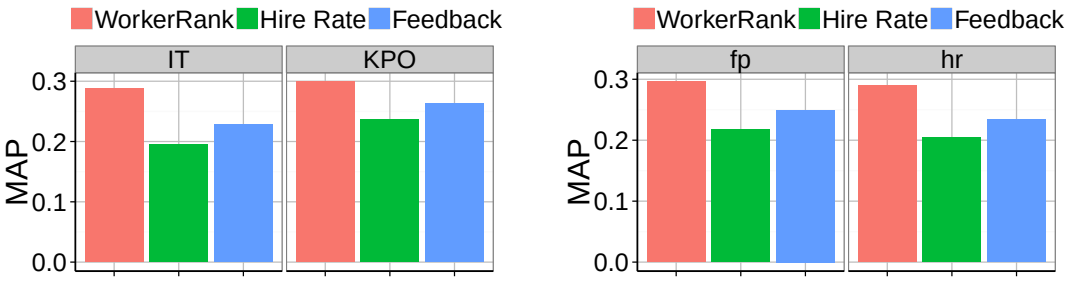

Figure 2.9: MAP across different job segments and types

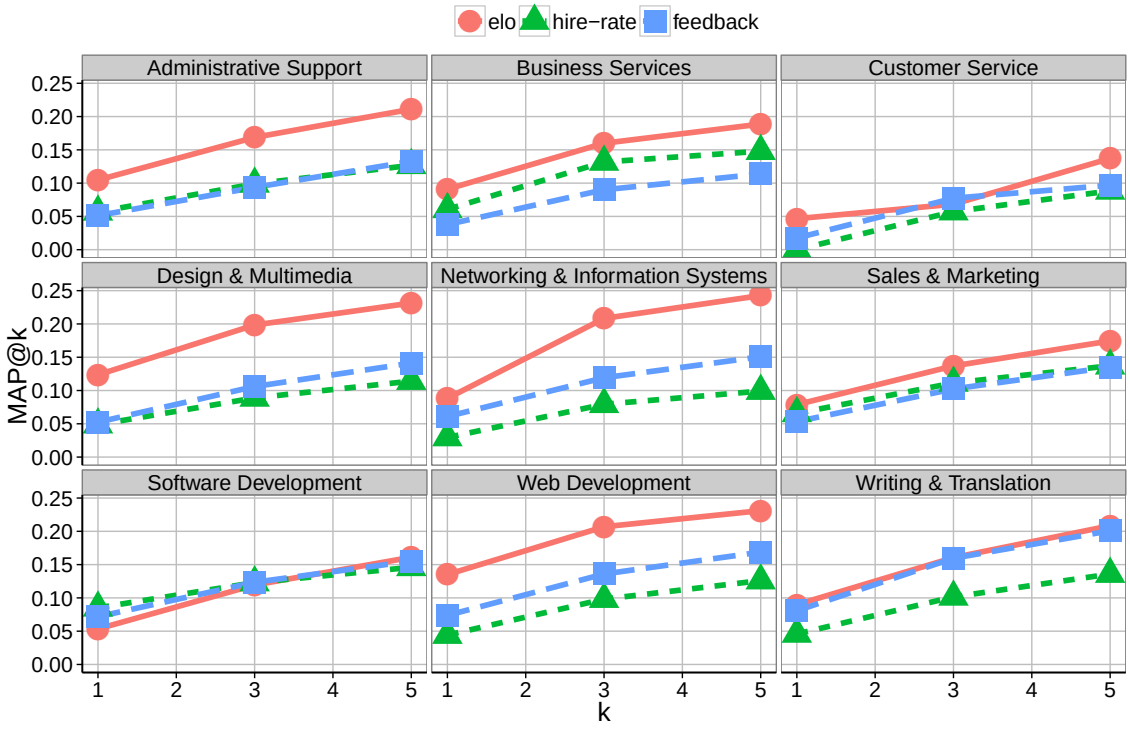

Figure 2.10: MAP across different job categories

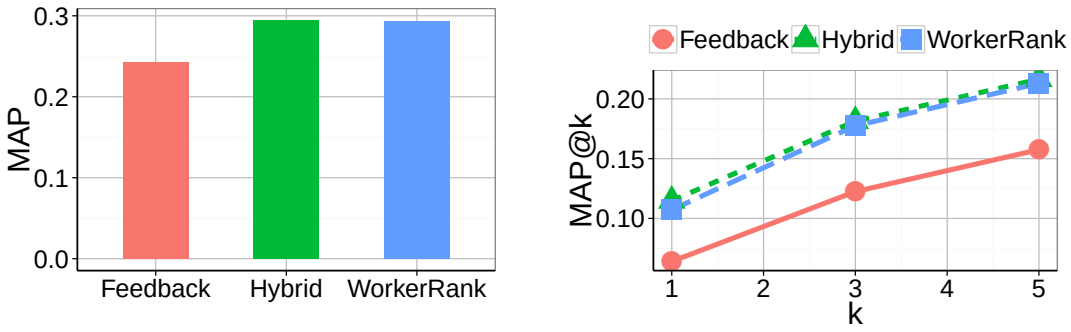

Figure 2.11: (a) MAP@Inf, (b) MAP@k for hybrid model in predicting the hiring outcome

approaches it combines; WorkerRank and explicit reputation. The hybrid model appears to slightly improve WorkerRank, the best of two approaches, although the improvement is not as high as it was in the comparison between implicit versus explicit reputation in Figures 2.6 and 2.7. It is interesting to mention that in all cases the hybrid model improves WorkerRank and explicit feedback, except for the hourly rate jobs and the Web Development category, where WorkerRank marginally outperforms the hybrid model and significantly outperforms the explicit reputation model.

Finally, during rank aggregation which derives the hybrid model, we tested a few weighting combinations to prioritize the influence of one of the two rankings (WorkerRank or explicit). Equal weights on the two rankings appears to be the best combination that makes the hybrid model behave optimally. The Figures shown for the hybrid model performance assume equal weight on the two lists.

Skill-wise WorkerRank vs WorkerRank: In Figure 2.12 we show how skill-wise WorkerRank performs at MAP compared to the plain WorkerRank approach. Since skills are available for approximately 65% of the jobs, we use a subset of the dataset in this experiment. As mentioned earlier, skill-wise WorkerRank produces specialized scores for workers, pertaining specifically to

Table 2.5: Performance Improvement: Hybrid vs WorkerRank

	WorkerRank	Hybrid	% Improvement
	WR	HB	HB vs WR
MAP@1	0.14	0.15	**+9.1%**
MAP@3	0.23	0.24	**+5.6%**
MAP@5	0.26	0.27	**+4.8%**
MAP@Inf	0.32	0.35	**+3.5%**

Figure 2.12: MAP in skill-wise WorkerRank

the skills that potential jobs require. The results show that ranking workers by skill-wise scores is more accurate than ranking them based on the WorkerRank scores. In particular, the improvement shown in the MAP Figure provides a better system to rank workers. However in certain cases skill-wise scores do not appear experimentally to differ from WorkerRank scores. That is because for several jobs a single skill is specified instead of a set of skills. This is one of the reasons why skill scores do not add significant knowledge to our estimations about workers quality. We apply logistic regression to combine skill-wise scores in a weighted fashion such that the intersection of our knowledge about the ability of the workers on multiple scores is incorporated.

2.5 Related Work

The problem tackled in this paper overlaps with four research fields; graph link analysis, building online reputation systems, game competition match analysis and predicting high-quality response in query-answering.

Graph link analysis research is related to our work, since we represent our data using a bipartite graph and we perform link analysis to examine the job applications of workers along with the respective employer feedback (edge weight). Several approaches have been proposed about ranking graph nodes in a network, such as PageRank [51], [14] and HITS [38], while Donato et. al. extend the study of HITS in [13] and Zhang et. al. in [65] study how PageRank and HITS perform when applied on the Java forum domain. Finally, Mishra et.al. [47], and Lescovec et. al. in [44] and [43] , present their node scoring methods with the presence of both positive and negative edge weights. Note that in our approach we also implicitly make use of negative information about applications, such as the "ignore", "hire" and "reject" feedback responses by the employer. However we only account for the relativity among different feedback labels, that is, who won over whom, hence we do not face restrictions of edge positivity.

Research on online reputation systems is directly related to our work, as we build a reputation system for workers in the labor marketplace, and we derive additional heuristic de-bias scores for employers. In [3] and [2] Adler et.al. tackle the problem of measuring the quality of contributions in Wikipedia, while Tan et.al. expand on this problem in [58]. Kokkodis et.al. in [39] discuss how to address data sparseness in building labor marketplace reputation systems. In [22], Dellarocas summarizes online reputation mechanisms and challenges they face in terms of usage and evaluation. What is more, Archack in [9] discusses how reputation challenges strategic behavior of

contestants in the TopCoder marketplace, while Chen in [16] describes their de-biasing mechanism for building a reputation system in a comments rating environment. TwitterRank [63] is another reputation system which aims to build reputation scores such that they incorporate a measure of influence for the Twitter users. Finally, in our past approaches in [20] and [18], we discuss the usage of link analysis using weighting schemes in order to build reputation systems.

It is interesting to reference a few game competition works such as the Elo method [26] that we are using in our current approach in order to predict the expected hire probability of each worker given our prior knowledge about their opponent's performance and their own performance. Elo is using a Bayesian update scheme to score chess game players based on past matches activity and update their scores by their expected performance in future tournaments. Glickman in [29] presents an improved approach, which keeps updating the mean and variance of the player scores such that confidence information is also carried along with the player's quality estimation. Methods tackling further improvement of match updates are proposed, such as TrueSkill [32] which tackles multi-player and multi-team challenges, while Nikolenko et.al. further improve TrueSkill's challenges of multiway ties and variable team size. Finally, Sismanis in [56] proposes a re-visit on the Elo method which incorporates tournament recency and other parameters in the tournament analysis in order to avoid over-fitting of the player ratings.

Moreover, query answering methods are referenced since they tackle the challenge of predicting quality of a response content such as question answers and social media content; that challenge is similar to our work's goal of predicting worker quality scores. Several approaches have been proposed aiming to identify quality in social media content such as Agichtein et. al. [5] and Bian et. al. [11]. Shah et.al. [55], Suryanto et.al. [57], Jurczyk et. al. [35] and Anderson et. al. [8]

study quality of answers in question answering, while Tsaparas et.al. [59], study quality of online review systems, such as in Yelp or Epinions. Finally, Chen et. al. [15] study de-biasing approaches to set votes more informative in question-answering systems towards higher quality in answer and expert ranking.

Finally, Kokkodis et al. [41] formulate the problem of hiring in work marketplaces as a binary classification problem, where the target variable is the hiring decision. An informative reputation mechanism like the one we present in this paper can be a very predictive signal in such classifiers.

2.6 Conclusions

The results of our experiments show that WorkerRank improves ranking of candidates compared to baseline approaches, since its reputation scores reflect worker quality more accurately. What is more, WorkerRank solves the basic problems encountered in explicit reputation systems (unreliable employer ratings, limited coverage of worker scores, cold start problem for new workers with no history information). Our future work includes research on weighting schemes as discussed in[20] and modeling implicit actions on the marketplace website.

Part II

Reputation Systems in Advertising Marketplaces

Chapter 3

Auto-Segmentation: A Reputation Based Algorithm for Audience Selection Recommendations

3.1 Introduction

"Whom should I show my ad to?" This is the typical question that every advertiser has to answer before starting a new advertising campaign [42]. The advertisers convey their audience selection requirements with demand side platform (DSP) companies like Turn, Rocketfuel, or others via boolean expression on user *demographics*, i.e., all males in California, or *technographics*, i.e., all users that use the Chrome Web Browser. Then the DSPs handle the optimal targeting and bidding on advertising opportunities within the audience space defined by the advertiser selection. While the targeting and bidding processes are usually machine learning driven, audience selection

is traditionally set manually by the advertisers. Advertisers come initially with certain rules based on market research or past data; then they allow the campaigns to run and refine their audience selection preferences according to the observed results.

Although the use of explicit boolean constraints ensures the advertiser control and transparency in the audience selection, the "manual" setting part of the process is usually a source of inefficient choices. While most advertisers research the market and they can adequately describe the audience that are more likely to engage with their campaigns in words, they usually lack the data or the skills to express their preferences as boolean expressions over user demographics and technographics [28]. For example, an advertiser may be aware that the users of interest are non-tech-savvy individuals. However, he is probably unaware that a way to describe a superset of such users is by selecting people that use an outdated version of the default browser in the operating system they use, i.e., Internet Explorer on Windows or Safari on Mac. A suboptimal audience selection negatively affects the success of an advertising campaign in various ways: an audience selection that is too narrow may dramatically limit the reach of a campaign, while a very broad selection will make the optimal bidding problem intractable.

Several attempts have been made to bridge the gap between the advertiser domain specific knowledge and the language perceived by the advertiser systems. In most of them, the solutions include the use of proprietary data such as user browsing behavior, search history or emails [10, 36, 7, 30, 61] and as such they can only be leveraged by companies that have access to such data. Even if the access to the proprietary data was granted, these solutions would not be useful to advertisers that want to retain control over the audience selection with transparent rules, since they usually provide a black-box function that determine in a binary fashion whether a user belongs to the ad

audience.

In this work we propose a mechanism for audience selection recommendations that uses data that are typically avaiable to a DSP: ad impressions, user conversions and user demographic and technographic data. The output of our method is binary expressions on user data that are transparent and easy to communicate with advertisers. The basic components of our solution are a reputation system that we build upon users and advertisers and decision tree learning on past user conversion data.

The proposed reputation system uses link analysis on the bipartite graph between users and advertisers and it yields: (a) an individual score per user that reflects the advertisers aggregated belief about his targeting value, that is, whether the advertiser should target them or not, and (b) a score per advertiser reflecting his targeting performance in identifying converters. Our reputation system builds upon the HITS [38] algorithm, that considers a web page as an important hub if it includes out-links to important authorities and considers a web page as important authority if is has many in-links from important hubs. In our context, we are interested in finding how many advertisers a user was targeted by, along with the advertiser's quality measured by his success in identifying converters in the given content category. The content taxonomy may introduce a level of noise, which we partially address by removing fraud and taxonomy inconsistencies.

Given the reputation system, we derive audience selection rules by learning decision trees on user historical conversion data. In the learning process we define the target variable to be proportional to advertiser Return On Investment (ROI) and we use as features user data and repuation scores. Figure 3.1 shows a decision tree that regresses on user demographic (age, gender, and more) and technographic (operating system, browser type, net connection speed, and more) features. Our

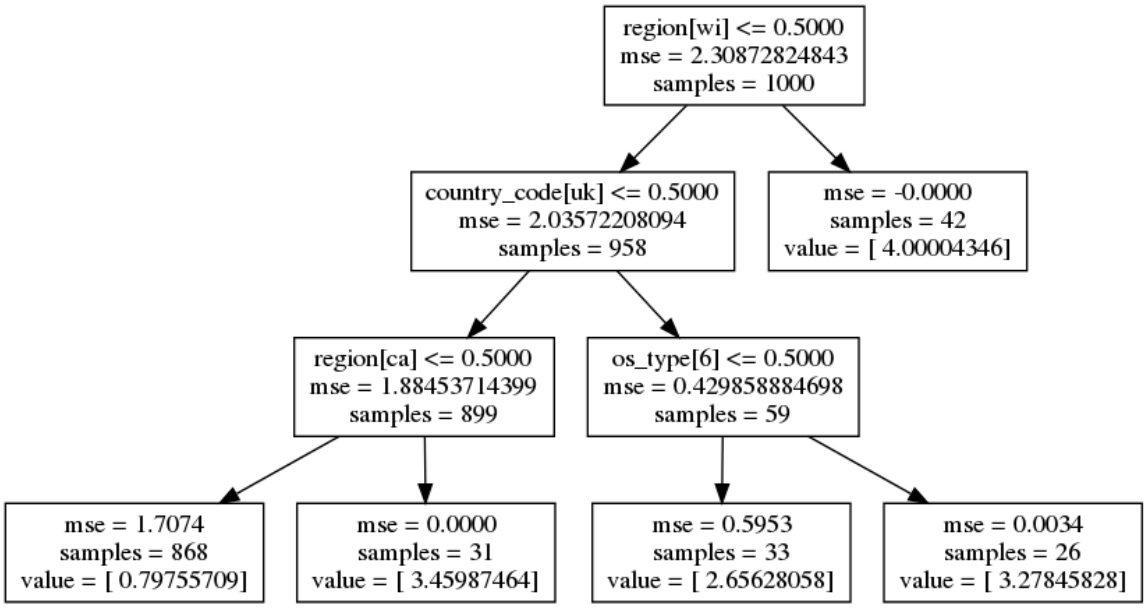

Figure 3.1: A sample regression tree. Example rule: region $\neq$ 'WI' and country code = 'UK' and os type = 6(mac), mean(ROI) = 3.27

method can also yield *negative* recommendation criteria, that is, which users the advertisers are advised *not* to target.

In our experiments, we evaluate our proposed audience selection mechanism in a real-world dataset provided by Turn. The results show that new advertisers that adopt our recommendations can enjoy an increase of up to 450% in conversion rate compared to advertisers that stick to their own rules. Advertisers with existing campaigns can also use our algorithm to refine their audience selection preferences and identify segments that have up to 10x higher conversion rate than the average of their campaigns. Finally, we show that advertisers can potentially increase the conversion rates of their ads if they adopt our negative selection recommendations.

The remainder of the paper is organized as follows. In Section 3.2 we present the notation and problem statement of this work. In Section 3.3 we describe the *Auto-Segmentation* algorithm, our proposed algorithm for audience selection. In Section 3.4 we present a set of experiments which

Table 3.1: Advertiser Categories

Category	Subcategory 1	Subcategory 2
Telecommunications	Wireless	TV
Autos	Cars	Commerical Trucks

illustrate the effectiveness of our models over baselines. In Section 3.5 we describe related work in audience selection and reputation systems. Finally, in Section 3.6 we conclude this study.

3.2 Notation and Problem Statement

We consider sets of *line items* L, *advertisers* A and *users* U. Each advertiser $a \in A$ owns one or more line items, denoted as $l_a \in L_a \subseteq L$, that he advertises to users. Advertisers belong to a specific content *category* $q \in Q$, where Q is a set of categories that describes the content of the advertiser's business domain, such as autos or telecommunications. A sample of categories is shown in table 3.1. In fact, advertisers are categorized across a taxonomy, but for the scope of this work we consider the lowest level in the taxonomy as a distinct category. For example, if node "credit cards" has two children, "loans" and "mortgages", we consider the two categories, "credit cards - loans" and "credit cards - mortgages".

Users *click* or *view* a line item; we define click and view functions:

$c(u, l) = \mathbb{1}(\text{user } u \text{ clicked line item } l)$, $v(u, l) = \mathbb{1}(\text{user } u \text{ viewed line item } l)$, where $\mathbb{1}$ is the indicator function. We denote users who clicked line item l as $U_l = \{u \in U : c(u, l) = 1\}$ and users who viewed line item l with $V_l = \{u \in U : v(u, l) = 1\}$. Users perform *actions* after viewing and clicking a line item, which may reflect product purchases, subscriptions, friend invitations, and more. We denote number of actions of user u during time period dt after viewing/clicking ad of line item l, as $\alpha(u, l, dt) \in \mathbb{N}$. Interesting quantities related to actions are the action rate, computed

Table 3.2: User features

Field	Domain	Type
age	[19,99]	numeric
gender	{male, female}	nominal
oper.system	{ Windows, ..., Mac }	nominal
browser type	{ Chrome, ..., Firefox }	nominal
country	{Albania,...,Zimbambwe }	nominal

as percentage of a user's actions over clicks or impressions, or the action density, computed as the percentage of actions within a time period over the actions performed during a wider study period. These are defined and used in Section 3.3.3 to describe segment feasibility and optimality. Note that although time parameter is naturally involved in the actions definition, it is omitted from the impression (view) and click definitions, for simplicity.

User *features* F are defined as functions of users, with $f \in F$ and $f : U \to D_f$, where $d_f \in D_f$ is a value in the domain of values D_f for a particular feature f. Based on the feature, $D_f = \mathbb{R}$ if f = income, $D_f = [19, 99]$ if f = age, $D_f = \{\text{male, female}\}$ if f = gender, and so on. Examples of features are shown in table 3.2.

Rule-set $\mathcal{R}$ is defined as the set of possible feature key-value pairs along with conjunctive and disjunctive expressions of them. Keys are related with values with any of the comparison operators $op = \{<, =, \neq, >, \leq, \geq\}$ where applicable. Note that we use only the $=, \neq$ operators for nominal fields. Given the above, we define rule-sets as formulas $\mathcal{R}$ for which the following hold:

$$f \in F, d_f \in D_f, \bowtie \ \in op \to f \bowtie d_f \in \mathcal{R}$$
$$\phi_1, \phi_2 \in \mathcal{R} \to \neg\phi_1, \phi_1 \vee \phi_2, \phi_1 \wedge \phi_2 \in \mathcal{R} \tag{3.1}$$

Also, we consider $\mathcal{P}(U)$, powerset of users U, that is the set of all possible subsets of U. User *segment* $S \in \mathcal{P}(U)$ is defined as the set of users described by rule $r \in \mathcal{R}$: $S = \{u : r(u) = \text{True}\}$. *Reach* of segment S with respect to a line item $l \in L$, is the size of the segment, that is, the number

of users targeted for the line item's campaign,

$$\text{reach}(S, l) = |V_l|_{u \in S} = |\{u \in S : v(u, l) = 1\}| \tag{3.2}$$

Return of a campaign is defined as the gain (campaign's gross profit) from the investment minus the cost of investment, as specified by the budget planned by the advertiser for a line item. *Cost* of a campaign includes targeting cost, data cost and several other fees. Targeting cost refers to the cost for reaching users, mainly through the Demand Side Platform (DSP) such as Turn, and the publisher. Data cost pertains to the cost of consuming user feature data for targeting; since most of the user attributes are offered by third-party providers, their availability incurs additional fees for the advertiser. For simplicity, we do not provide further details about how these quantities are broken down. In this work we consider CPA campaigns, where the advertiser pays for each observed user action (the type of actions are specified by the advertiser).

Return on investment (ROI) function $g : U_L \times L \to \mathbb{R}$ is defined on the set of line items $l \in L$ clicked by users $u \in U_L$ within a given study period,

$$g(u, l) = \frac{return}{cost} \tag{3.3}$$

Note that in the above definition ROI is computed in user level, that is, return pertains to user action gain. Also, g is a function of line items, users, and *time*; however in the scope of this paper we assume that the time period of our study is fixed and the candidate audiences will not change during the experiments.

Using the simplified expression, we consider the following problem. *Given a history of user activity, and a set of initial features about users, derive rules to describe user segments that optimize return over cost, given that their reach size meets a minimum threshold.* History data of

activity involve a set of users (clickers $U_L = \bigcup_{l \in L} U_l$ or viewers V_L) targeted for a set of line items L that are owned by one or more advertisers in A, along with the ROI of each user - line item interaction, $g(u, l)$, $\forall u \in V_L$ (or U_L), $\forall l \in L$. The input space granularity may be that of an entire advertisers category q, in case we are interested in recommendations for a new line item $x \notin L$ owned by advertiser in category q, or that of one or more campaigns of a particular advertiser. The features data involve values $f(u) \bowtie d_f, \forall f \in F, d_f \in D_f, u \in U_L$, of demographic, technographic, and other features of users available from their behavior history. The rules to be derived are described as rules $r \in \mathcal{R}$ that describe feasible segments of users $S_r \subseteq U$.

Although g takes values in $\mathbb{R}$, ROI maximization is considered with respect to the maximum budget available for line item x, along with return in the case of bid win as specified by the market with the current bid prices during the time of study. (Feasibility is defined in Section 3.3.3.)

With return on investment optimization, we aim at identifying segments of users who have high probability of conversion (action) upon viewing x 's ad. At the same time, we prioritize large size segments such that reach is considerable for the advertiser. Finally, we introduce the pass/fail testing of feasibility and negativity about segments, which reflects whether the segments to recommend are meaningful for the advertiser. For example, an airline that flies only in Europe would not be as interested to learn the best user segments in the US.

The optimization aims at satisfying three major priorities for advertisers; high ROI, high user reach, and high confidence. The first two are discussed above. Confidence is regarded with respect to reliability in prediction behind any targeting recommendation. That may evolve from the number of examples used to learn our predictions, from the prediction accuracy itself, as well as from the reputation quality of the examples being used. For example, if a user has converted in

the past, was that a random occurrence, or was it a meaningful response that reflects longer term human interest? Since at the moment of recommendation we have not seen the actual return for the new line item x yet, the confidence component is captured by $E[g(u, x)]$, the expectation about the unobserved ROI for x. As we discuss in the algorithm Section, we address confidence by ensuring considerable number of examples in the input, and by introducing a reputation system about users and advertisers to capture reliability in the user interest and user targeting signals.

3.3 Auto-Segmentation

In our proposed approach, we devise an algorithm that derives a set of rules that describe a set of optimal segments (in terms of user conversion) for the given input data. The applications of Auto-Segmentation in this paper are a) campaign refinement and b) cold start. In the first case, the goal is to produce refined user segments based on the advertisers existing segment selections, with respect to user conversion; the input of the algorithm is the set of line items of a particular advertiser along with his targeted user segments, the user features, and the ROI performance in each user-line item relation. In the second case, the goal is to recommend rules that describe optimal segments for a new campaign line item for which only business context is know, that is, information about its type and content (for example, line item of an airlines company that flies in Europe); the input of the algorithm is the new line item, the line items of advertisers in the same business category/taxonomy of the new line item owner, along with the user segments selected by the relevant advertisers, user features and ROI performance between user-line item.

In the first stage (steps 1 - 2), we look at the user preferences (line items clicked or to which the user converted) along with the return on investment as an implicit indication about the

interest strength of the user. We compute quality of these preferences using a reputation system that is based on a weighted version of the HITS algorithm [38] that accounts for ROI-based weighting along with popularity of items. Along with line item reputation, the reputation system also computes a bias/reliability score for users, to reflect a measure of quality for their judgements / click decisions. Overall, the reputation system is used to identify consistent and reliable responders to popular line items in the category. In particular, we build a bipartite graph from the input elements and we run the weighted HITS algorithm, $wHITS$, as described in Section 3.3.1.

In the second stage (step 3), we learn candidate user segments, by using weighted decision trees on the input. User features are used as variables and value domains are used as possible split points, while the target is set as the user-line item level ROI, weighted by the crowd computed reputation and reliability scores. Tree decision learns the distribution of the weighted ROI success metric as a function of the user features; then the highest metric score leaves are used as candidate segments for recommendation. The decision tree algorithm $DecTree$ is a variation of the CART [31] algorithm, as described in Section 3.3.2.

In the third stage (steps 4 - 5), we apply segment selection out of the suggested candidate segments from the second stage. Candidate segments are filtered by an optimality function that determines segment performance in terms of the used success metric (such as conversion rate). In addition, negative recommendations are derived in this stage, where segments advised *not* to target are provided as well. $SelectSegments$ is described by Algorithm 6 in Section 3.3.3.

3.3.1 User Reputation

In the first stage, we create a reputation system that estimates the quality of the items preferred by users, on top of which items they prefer, to improve accuracy in representing user

Algorithm 3 Auto-Segmentation

Input: Advertisers A, line items L, users U_L, ROI $g(U_L, L)$, features $f(U) \bowtie d_f$, [new line item $x \notin L$ in cold start case]

Output: Rules $\{r \in \mathcal{R}\}$, segments $\{S_r\}$ [for new line item x]

1: $G = (U, L, E) \leftarrow$ build graph from $A, L, U_L, g(U, L)$

2: $z(U), b(L) \leftarrow$ wHITS(G) ▷ Reputation

3: $\{s_i\} \leftarrow$ DecTree$(U, [f(U) \bowtie d_f$ and $z(U), b(L)], g(U, L))$ ▷ Candidate Rules Extraction

4: $\{s_{best}\} \leftarrow$ SelectSegments$(\{s_i\})$ ▷ Segment Selection

5: **return** s_{best}

interest. A signal about a converter's reliability comes from the user's target activity. Activity is allowed for with respect to, at what frequency does the user respond (popularity) when targeted, and what is the gain obtained for the advertiser (ROI) from the user's response. Since in this work we account for ROI with focus on CPA campaigns, gain refers to user actions. To derive a *conversion reliability* score for the users, we make use of the above signal as follows.

We devise a link analysis algorithm on a graph that connects users with their clicked line items, owned by a set of advertisers. An edge between a user and a line item holds the information that the corresponding advertiser targeted that user and the information of whether the user responded, along with his interest strength. The latter is measured by the total return from the response over to the total cost for the advertiser targeting this user. We use the ROI weight on the edge also to cover the information about the *targeting value* of the advertiser. Then we apply a mutual recursion computation of scores between line items and users as shown in steps 4 - 5 of Algorithm 4. That recursion attributes to the user not only the number of his clicked line items weighed by the ROI that followed from the corresponding actions; it also attributes to the user the quality of the line

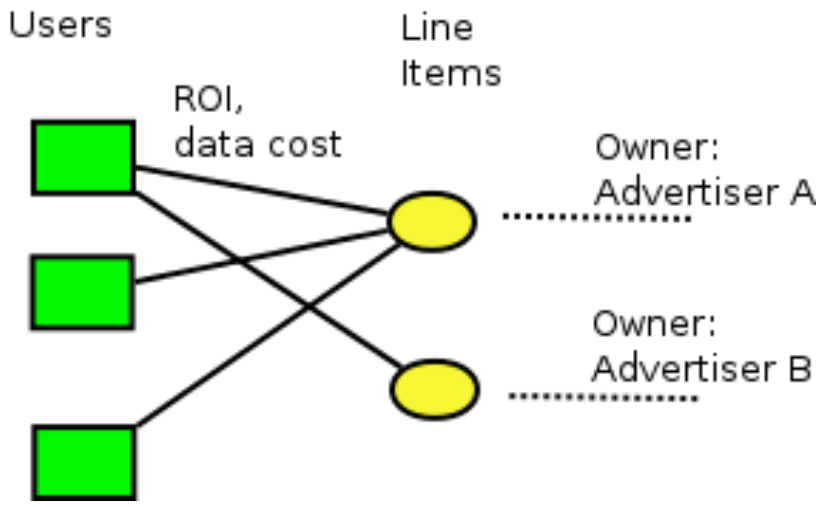

Figure 3.2: Reputation Bipartite Graph between users and advertisers

items that he chose, in terms of their popularity. If a line item was chosen by many users, that indicates that the ad was successful, and also that the targeting was successful, that is the right people were reached for a product of their interest. If, in addition, the total ROI for the related actions of those clickers was high, including the original user, that validates the monetary gain from the line item - user targeting relation.

Formally, we consider bipartite graph $G = (L, U_L, E)$, as shown in Figure 3.2, where L is the set of line item nodes owned by advertisers A within category q, U_L is the set of user nodes, for users who clicked or viewed the line items in L, and E is the set of edges between L and U, denoting the targeting relation between them; an edge $e = (l, u)$ exists if user u clicked on line item l posted by an advertiser $a \in A$. Each edge is accompanied with a weight that characterizes the success of the targeting relation, measured by the ROI of the action following the click, $g(u, l)$.

We run $wHITS$, a weighted version of the HITS algorithm [38] where we update the scores of each node by the linked nodes scores weighted by the ROI relation between line item and

user nodes. In particular, if a user has clicked on a line item and that click was followed by an action, the degree of success for that action measured for the advertiser is taken into account. In this way, the scores of user nodes reflect not only the user's activity popularity, that is how many line items they clicked on, but also the value of their click as measured by the advertiser. In the long run, a user with high score ends up representing a reliable converter, that is, whose clicks and actions are rather not random events but they imply true interest in the line items clicked. At the same time, a line item with high score represents a popular component of the targeting value of its advertiser. Note that convergence is quaranteed as the quantities in steps 4, 5 are positive [38].

Algorithm 4 wHITS

Input: $G = (L, U_L, E)$

Output: Reputation scores $z(u), \forall u \in U_L$, $b(l), \forall l \in L$

1: Initialize $z(u) = 1, \forall u \in U_L$; $b(l) = 1, \forall l \in L$

2: **repeat**

3: **for** $u \in U_L, l \in L$ **do**

4: $z(u) \mathrel{+}= g(u, l) \cdot b(l)$

5: $b(l) \mathrel{+}= z(u)$

6: **end for**

7: Normalize $z(u), \forall u \in UL$ and $b(l), \forall l \in L$

8: **until** convergence

3.3.2 Rule Extraction

In the second stage of the algorithm, we learn candidate user segments for the given input, by using weighted decision trees. In particular, we build a decision tree based on a variation of the

CART algorithm [31] using as features the user demographic, technographic and behavioral features along with the computed reputation and reliability scores of users and line items from the previous stage, obtained from Algorithm 4.

In traditional CART, for different feature-value pairs we successively split the user space into two partitions, aiming for optimal splitting at each level. Optimal split is considered in terms of node impurity computed by the error on the average ROI of the ads clicked by the users in each partition. Then we iterate until a minimum number of samples is left on each sub-region. Below we show our varied version of the algorithm, with the addition of weights in the splitting process. We consider set of users $u \in U$ as the set of inputs, and user features $f \in F$ as the set of variables with splitting points d_f. We also add reputation scores of line items and reliability scores of users as *new features* whose domain of splitting points is the range of scores computed from Algorithm 4. We also consider response variable $\gamma(u)$, where $\gamma(u)$ is the average ROI of the ads that user u has clicked on.

$$\gamma(u) = \underset{l:c(u,l)=1}{\text{avg}} [g(u,l)] \tag{3.4}$$

Then, for each splitting variable f and split point d, we consider regions $R_1(f,d) = \{u : f(u) \leq d\}$ and $R_2(f,d) = \{u : f(u) > d\}$, if f is numeric, or $R_1(f,d) = \{u : f(u) = d\}$ and $R_2(f,d) = \{u : f(u) \neq d\}$, if f is nominal. We select the best $\langle f', d' \rangle$ pair for which:

$$(f', d') = \underset{f,d}{\text{argmin}} \left[\sum_{u \in R_1(f,d)} (\gamma(u) - \bar{\gamma}(R_1(f,d)))^2 + \sum_{u \in R_2(f,d)} (\gamma(u) - \bar{\gamma}(R_2(f,d)))^2 \right] \tag{3.5}$$

In the above expression, we set $c_1 = \bar{\gamma}(u)$ and $c_2 = \bar{\gamma}(u)$ and we use $\gamma(u)$ which reflects the weights of the line items, $b(l), \forall l \in L$. The process is repeated until a minimal number of

samples is reached on each of the sub-regions.

To give an example of splitting across user features, consider pair $\langle$age, 35$\rangle$, which divides the space of users into users whose age is less than 35 (class k_1 of 19-34 year old users) and users with age equal to or higher than 35 (class k_2 of 35-99 year old users). In an ideal scenario, region R_1 would contain users with total $\gamma = 0$, while R_2 would contain users with total $\gamma = 100$, assuming that the max value or ROI is 100. That would split the users in pure nodes of low and high ROI values, which would easily suggest candidate region R_2 as a segment to consider for recommendation. Such a performance depends on the data though, hence extracting top ROI segments from the tree regressor does not suffice to represent suitable segments. In Section 3.3.3 we describe our approach for selecting segments to recommend in a personalized way for advertisers.

A benefit from using regression trees in the above model is the fact that trees automatically yield segments where high ROI is concentrated across the multiple user selections that the advertisers in category q have made. In particular, the best performance leaf nodes become good segment candidates among which we can select recommendations for the new ad, x. What is more, the structure of trees automatically yields the rule sets to describe the users of the suggested segments, by taking one or more paths across its nodes. For example, a candidate rule set in the example shown in Figure 3.1 would be region $\neq$ 'WI' and country code = 'UK' and os type = 6 (Mac). A single path in the tree consists of feature-value pairs related with conjunction (*and*), as in the example, while multiple tree paths may be combined in a disjunction (*or*) relation to broaden the users included for the recommended targeting.

Algorithm 5 DecTree

Input: $U, f(U) \bowtie d_f, g(U, L), z(U), b(L)$

Output: Candidate user segments $\{s_i\}$

for all $u \in U$ **do** Compute $\gamma(u)$ according to Equation 3.4

end for ▷ Average user ROI

for all (splitting variables f, splitting points d_f) **do**

    Find (f', d') that minimizes Equation 3.5

end for

$R_1, R_2 \leftarrow$ Divide user space in 2 regions according to (f', d')

$\{s_{lft}\} \leftarrow$ DecTree$(R_1, f(R_1) \bowtie d_f, g(R_1, L), z(R_1), b(L))$

$\{s_{rgt}\} \leftarrow$ DecTree$(R_2, f(R_2) \bowtie d_f, g(R_2, L), z(R_2), b(L))$

if $|R_1| \leq$ min_samples **then return** s_{lft}

end if

if $|R_2| \leq$ min_samples **then return** s_{rgt}

end if

Boosting for Feature Selection

The set of features under consideration for producing rules for optimal segments play an important role. Trees are powerful in representing the structure of the data and supporting complex functions, however their predictive power is limited. In order to derive a high accuracy model that recommends reliable targeting rules, we learn a set of gradient boosting tree models on a set of data partitions. Boosting trees yield optimal predictive accuracy by training multiple single tree weak predictors and aggregating single predictions for best performance. Then, from all boosted trees we retain feature importances computed based on the Gini index. Finally, to produce a single rule set which reflects the contribution of each user attribute to targeting, we build a single decision tree on the input features indicated by the boosted models. The prediction accuracy of the final tree is improved, while we also obtain the corresponding rules for the user targeting.

3.3.3 Segment Selection

In this section we describe our approach for selecting segments out of the pool of candidate segments derived from the decision tree model. Usually advertisers determine their targeting based on two criteria; user response and hard constraints. The performance criterion dominates the literature interest, where the probability of user response is studied. The second criterion pertains to custom hard constraints set by the advertisers to achieve brand advertising or for individual campaign interest goals. For example, a European airline company may always want to target all Europeans even if 90% of their ad responses come only from UK citizens. Although hard constraints are not easy to describe, unless specifically specified by the advertiser, they could be studied, for example, by looking at common targeted users across several line items of an advertiser. In this

work, our goal is to recommend segments with optimal expected response performance, leaving hard constraints as secondary priority. First, we describe the criteria that determine *feasibility* and *optimality* of segments, and next we propose a method to recommend both which segments to include in targeting based on these criteria, and which segments to exclude from targeting that the advertiser is probably already using.

Segment Feasibility

With user segment *feasibility*, we aim to capture a confidence level about the recommended users response expectation. We set two main criteria to define feasibility; segment applicability and user activity.

First, segment recommendation has to be applicable in the advertiser's domain; for example, recommending a truly high response segment of US users to an airline company that flies only in Europe is not very meaningful, as the expectation is that these users will depart from/land in the US. Hence the European company would only be interested in case it had an alliance with some US-flying airline that extends its network. To address applicability, we prioritize segments which overlap to some extent with the existing targeting selections of the advertiser. Note that our focus is in adjusting the targeted population of an advertiser towards best performance, rather than extending it with similar audiences. Second, the time parameter is important for recommending segments. For example, a user who recently booked an airline ticket, will probably not be interested in buying another one to the same destination within the next months. Hence, usually active users with long inactivity by the study time, are more eligible to get recommended as they are expected to respond relatively soon. We define feasibility as follows:

Definition *feasible(s,l)*: Segment $s \subseteq U$ that is candidate for recommendation at time t to advertiser a who posted line item l and had past targeted user segments $s_a \in S_a \subseteq \mathcal{P}(U)$, is called *feasible*, if $s \bigcap \{s_a\} \neq \emptyset$, and $\frac{\sum_{u \in s} \alpha(u,l,[t-\sigma,t])}{\sum_{u \in s} \alpha(u,l,[0,t])} \leq \epsilon$, given that $reach(s_a) > \eta$ and $reach(s) > \eta$, for $\eta \in \mathbb{N}$, $\epsilon \in [0,1]$, $\sigma \in [0,t]$ constant parameters.

Feasibility of segment s with respect to line item l is defined as the indicator function with condition $feasible(s,l)$, according to the above definition. Note that $\frac{\sum_{u \in s} \alpha(u,l,[t-\sigma,t])}{\sum_{u \in s} \alpha(u,l,[0,t])}$ represents action density with regards to line item l within time period $[t-\sigma,t]$. Also, η is determined by the traffic observed within the category of the advertiser, and σ represents the level of *recency* under study. The recency threshold is determined by the line items content; for example, recency in airline tickets may refer to months, while recency in autos may refer to years. Also note that feasibility partially covers the advertisers hard constraints, along with user response performance, since the overlap with past defined segments entails some confidence about the advertisers interest in the recommended users. Unfortunately, it is very hard to extract or simulate the hard constraints of the advertisers targeting and use them for extensive feasibility testing, since this data is not provided by the advertiser nor any other party.

Segment Quality

With user segment *quality* we aim to capture the quality of recommendation, with regards to the expected user response. In particular, we regard quality segments as those which include the users that are expected to show the highest response among all the clickers or viewers of the advertiser's line item ads. Formally, we use the following definition:

Definition *quality(s,l)*: Segment $s \subseteq U$ that is candidate for recommendation at time t to advertiser a who posted line item l and had past targeted user segments $s_a \in S_a \subseteq \mathcal{P}(U)$, is a *quality* segment, if $E[cta(s)] > \text{avg}_{\,s_a \in S_a} E[cta(s_a)]$ and $\sum_{u \in s} \alpha(u, l, [0, t]) \geq \zeta$, given that $reach(s_a) > \eta$ and $reach(s) > \eta$, for $\eta \in \mathbb{N}$, $\zeta \in \mathbb{N}$, constant parameters.

Quality of segment s with respect to line item l is defined as the indicator function with condition $quality(s, l)$, according to the above definition. Note that $E[cta(s)]$ reflects the expected action conversion rate (known as Click-Through-Action) of segment s, that is the ratio of actions over clicks by the time of recommendation, t, $cta(s) = \frac{\sum_{u \in s} \alpha(u,l,[0,t])}{\sum_{u \in s} c(u,l,[0,t])}$. Parameter η is determined by the traffic observed within the category of the advertiser. Parameter ζ reflects action *frequency* as an infimum of actions that must be observed from the recommended users by the time of recommendation t. Again, the frequency threshold is determined by the content of the line items in the category; for example, frequency in airline tickets may refer to dozens, while frequency in bath products may refer to hundreds. Quality covers response performance, however hard constraints may not be covered; for example, in case the advertiser includes some users for brand advertising, the conversion rate within those users sub-segment is not expected to be high.

Who not **to target**

Along with recommending which users to target (”good” segments), it would be of great value to the advertiser to get an insight also about which users to *stop* targeting (”bad” segments). It is often useful to know which user segments perform worse than the average selections within an advertiser’s target groups, or within the category’s overall target population, so that future campaigns can be adjusted towards higher conversion. In our context, ”bad” user segments reflect users

who constantly do not respond to ads of that particular advertiser, or to ads of line items in the advertiser's category (for instance, autos line items). Hence, since our recommendation mainly optimizes user response performance, we tackle *negative* recommendations from that perspective. The advertiser may find negative targeting recommendations useful, or they may choose not to remove any targeted users as those may correspond to users that satisfy the advertiser's hard constraints. For example, if New Zealand local population do not respond to airline offers of a company that flies to New Zealand, the company may always want to be targeting that population, with the expectation that when they choose to travel, they will prefer that airline.

In this section we study quality of segments in category level, that is which segments perform worse among users targeted within a given category, or in advertiser level, that is which segments perform worse among users targeted by an individual advertiser. In the former case, we propose negative recommendation rules 3.3.1 and 3.3.2 and in the latter case we propose rule 3.3.3. In both cases the recommendation is applied on the last stage of our algorithm. Consider set of users S who have clicked on line items of category q in the past, owned by n advertisers $a \in A$. Also consider segments $s_a \subseteq \mathcal{P}(S)$ that the advertisers have targeted in the past. For new advertiser in the category (that is, who has not targeted users in S yet) who wants to find which users not to target for a new line item campaign, choose "bad" segments among m candidate segments $s_i, 1 \leq i \leq m$, based on the following recommendations:

Recommendation 3.3.1 (Category-wise) *Do not target s_i, if $cr(s_i) < cr(S)$.*

Recommendation 3.3.2 (Category-wise) *Do not target s_i, if $cr(s_i) < \text{avg}_{1 \leq j \leq n} cr(s_{a_j})$.*

For advertiser in the category who has targeted users from S in the past and who wants to find which users not to target for a new line item campaign, choose "bad" segments among m candidate

segments $s_i, 1 \leq i \leq m$, based on the following recommendations:

Recommendation 3.3.3 (Advertiser-wise) *Do not target s_i, if $s_a \cap s_i \neq \emptyset$ and $cr(s_i) < cr(s_a)$.*

Note that most often, advertiser a creates more than one segments, however for simplicity and without loss of generality, we use s_a to denote any segment created by advertiser a. Finally, along with "bad" segments definition, we also consider confidence level for the negative recommendation, based on the ratio of the conversion rates under comparison in each of the above recommendations.

Algorithm 6 SelectSegments

Input: Line item x, candidate segments $\{s_i\}$

Output: Best user segment s_{best} for x

1: **for all** segments $s_i \in \{s_i\}$ **do**

2: $\lambda(s_i) \leftarrow feasible(s_i, x)$

3: $\mu(s_i) \leftarrow quality(s_i, x)$

4: $\nu(s_i) \leftarrow$ if any of the recommendations 3.3.1, 3.3.2, 3.3.3 is True

5: **end for**

6: **return** $s_{best} = \text{argmax}_{s_i} \lambda(s_i) \cdot \mu(s_i) \cdot (1 - \nu(s_i))$

3.4 Experiments

In this section, our goal is to show that the proposed Auto-Segmentation algorithm solves the advertisers' cold start problem and that its solution for the campaign refinement problem outperforms baseline approaches. First, we show how our segment recommendations solve the cold start problem of new campaigns for which only their business context is known (such as their content cat-

egory or taxonomy) by deriving better performing segments based on a category-wide input dataset, as opposed to advertisers original selections. Second, to test how auto-segmentation contributes to campaign refinement via automatic rule derivation, we compute conversion rates of three types of segments; baseline segments defined by the original advertiser's selections for a campaign, segments recommended by our basic model computed on the campaign's data, and segments recommended by our reputation-based model on the same data. The basic version of our model implements steps 3 to 5 of algorithm 3, using only basic features about users (demographic, techno-graphic and behavioral), which mainly describe his online past behavior. The reputation system based version of our model implements the entire algorithm and includes reputation and reliability scores as new features for the rule extraction. These scores represent the signal about advertiser's user targeting; their contribution shows the value of integrating advertiser with user oriented signals. Finally, we show how negative recommendations benefit targeting in an offline training-testing experiment. Knowing which users not to target within a candidate segment saves investment, time and data cost resources for the advertiser.

3.4.1 Dataset and Metrics

In our experiments we use a dataset that corresponds to 15 days of data obtained from Turn Inc. After removing click fraud using external vendors and internal methods, the dataset includes $660M$ impressions, $170M$ distinct users, $7.5K$ line items, 150 advertiser categories, $1M$ actions. We use 20 organic user features, including demographic, locale, technographic (such as operating system and browser), and behavioral features (such as sites visited) and we use return on investment (ROI) as target. Table 3.3 shows the feature importances computed as the (normalized) total reduction of the mean squared error node split criterion across the tree nodes that is brought

Table 3.3: Feature importances across categories

Category	Features (Importances)
airlines	region:MA (0.12), region:FL (0.09), city:Buffalo (0.08)
autos	os type:6 (0.3), os type:4 (0.15), age(0.07)
bath prod.	age (0.17), browser type:4 (0.12), gender (0.10)
hotels	age (0.19), country code: US (0.15), region:CA (0.05)
insurance	os type:1 (0.13), browser type:3 (0.08), age (0.07)
fem. clothing	os version:7 (0.15), age(0.12), city: Philadelphia (0.05)

by that feature (Gini importance). A general observation is that of different attribute importances across different advertiser categories. For example, for category female clothing, age appears to be the most important feature, while for category autos, operating system and age appear as the most important features.

To measure the performance of segmentation, we use a custom conversion metric for each segment defined as

$$cr(s) = \frac{\sum_{u \in s} \gamma(u) \cdot \mathbb{1}(\gamma(u) > 1)}{\sum_{u \in s} \gamma(u)} \tag{3.6}$$

where $\mathbb{1}()$ is the indicator function. The above expression captures the percentage of impressions in which the return on investment is positive over all impressions. Positive return on investment occurs when the total return from an advertiser's investment on an ad for a particular line item, is (equal or) greater than the total cost that the advertiser had to pay based on the type of his campaign for advertising the product. The most common types of campaigns are the per action payment basis, and the per click payment basis. In the current study we only account for the former. The intuition behind defining the above custom conversion rate lays on the fact that the actual gain for the advertiser occurs only when the user actually converts to an advertised product, either by signing up or purchasing or providing data such as in survey ads. In the current study our goal is to optimize user engagement. Other types of advertising, such as brand advertising, would require

other approaches for testing.

3.4.2 Cold Start

Advertisers manually select an initial audience for a new campaign, based on their first estimations about user interest (plan phase). Then they start the campaign (execute phase) and after observing user behavior for a short time, they refine targeting based on recent activity high response groups (analyze phase). They repeat this process in several iterations until their targeting achieves the desired performance in terms of return on investment. Since that costs investment expenses and time, in this experiment we show how Auto-Segmentation provides audience recommendations which outperform the average expected performance of targeting selected in the cold start of a new campaign.

For a given advertiser category, such as ”autos”, we collect all advertisers campaigns and we run the Auto-Segmentation algorithm on the input of user activity; on the output segments we compute conversion and we compare it against conversion of the original segments targeted across the line items owned by advertisers in the category. In Figure 3.3 we show the percentage of average conversion rate improvement by the suggested segments versus the average conversion rate of original segments selected by advertisers within a particular period. We illustrate the results for several categories. Note that for the baseline we use the mean conversion rate of existing segments within a 15-day period, assuming that this represents the expected performance of a new campaign targeting in cold start. Both for anonymity but also because early targeting data availability is limited for most categories, we approach early segment performance by using the average performance of the segments. We notice that the recommended segments outperform the original segmentation selections for all categories.

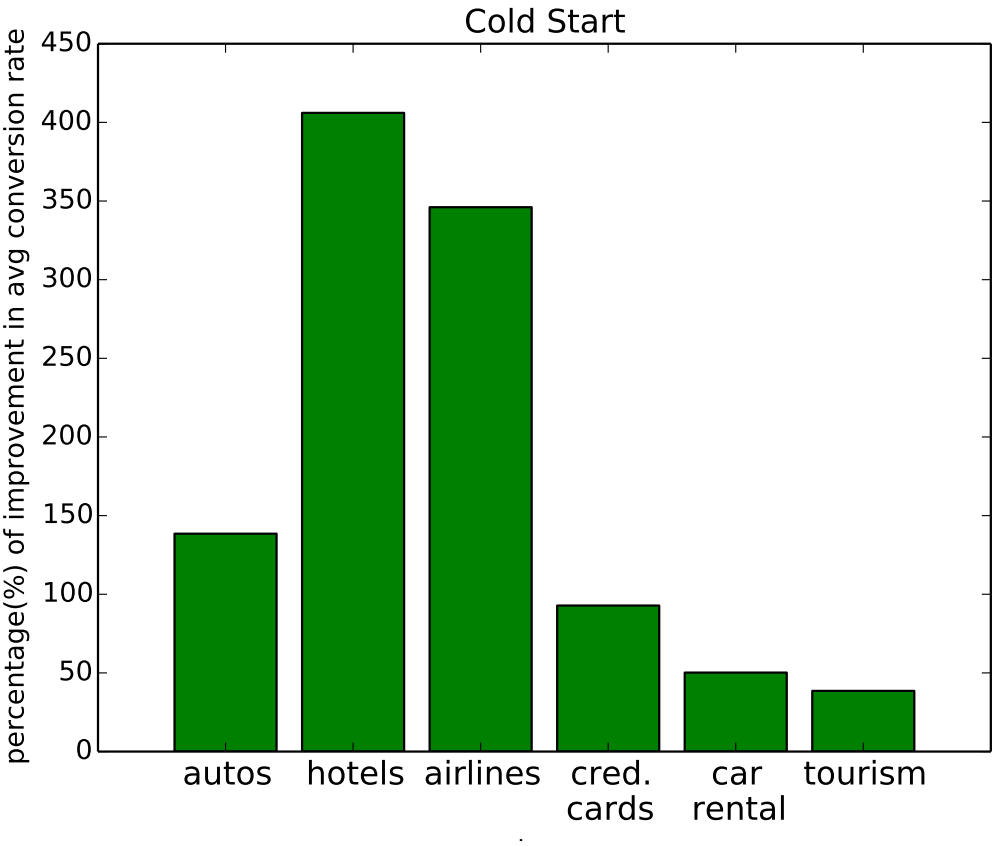

Figure 3.3: Cold Start

3.4.3 Campaign Refinement

To show the quality of Auto-Segmentation recommended segments, we compare their conversion rates against rates on segments originally selected by advertisers. In particular, we split user data into training and testing sets; then we learn segments based on the training data and we compute conversion of the testing population that corresponds to each segment. Along with each segment's conversion rate, we also compute its reach, that is, the amount of users reached when each segment is targeted. Since advertisers are particularly interested in reach along with conversion, we compute conversion rate performance as reach is being increased, by taking weighted average of $reach \times cr$ across segments, described in the following: First, we sort the M recommended segments $s_1, s_2, \ldots, s_M$ by decreasing conversion rate, that is, $cr(s_1) > cr(s_2) > \ldots cr(s_M)$. Then to find conversion rate $cr(N)$ of the first N users reached, we take the segment s_n in which users fall. We find the $n-th$ segment s_n by $n = \arg_n\{(\sum_{i=1}^{n} s_i) < N\}$. Then we compute the weighted average:

$$\frac{[\sum_{i=1}^{n-1} cr(s_i) \cdot reach(s_i)] + [cr(s_n) \cdot (N - \sum_{i=1}^{n-1} reach(s_i))]}{N}$$

where N is the number of users reached

We choose a popular advertiser (we keep their information anonymous for privacy), and we run AutoSegmentation on one of his campaign's data, that is $650K$ impressions for a 15 day-long period. About $4K$ impressions have $\gamma(u) > 1$ for users $u \in U$, which means that the return on investment is greater than the total cost for the advertiser. Our algorithm runs with this campaign's data as input, and it produces a set of sub-segments with optimal conversion performance. In order to show the value of using advertiser reputation and user reliability scores as features during rule extraction, we show the performance of both our reputation model, that is, including the scores

among the features (named as "reputation" in the plot), and the performance of our model using only the basic demographic, techno-graphic and behavioral features (named as "basic-model" in the plot).

We compare the model segments against the segments originally formed by the advertiser and we display our results in Figure 3.4. Figure 3.4 illustrates the performance of $cr(N)$ as reach N increases (log scale figure shown in Figure 3.4). Note that in these figures, reach and conversion are computed on the entire data-set of impressions of users, that is all viewers, clickers and action takers are included. We notice that for the first $1,000$ users reached, the conversion rate of the recommended segments ranges between 0.15 and 0.24 for our basic model and between 0.32 and 0.5 for the reputation model, while for the next $24,000$ users it ranges between 0.08 and 0.25 for the basic model and between 0.22 and 0.5 for the reputation model. The rates of the next $500,000$ users vary between 0.01 and 0.05. On the other hand, the advertiser's original segments do not reach higher conversion rate than 0.025 for the entire population of users targeted in this campaign. These results show that our model improves conversion significantly for the users targeted. Similar results are extracted when this experiment is performed in other campaigns.

3.4.4 Negative Recommendations Contribution

Besides recommending which users to target it is also useful to denote which users is not advisable to target, since they are not expected to convert. In this experiment we learn "bad" segments, that is segments with low conversion rates, on a training set of line items, and we recommend that advertisers do not target these users in a testing set of line items. Then we compare the conversion performance of the original segments as selected by the advertisers, against conversion performance as it would be if the "bad" users suggested by the negative recommendations were

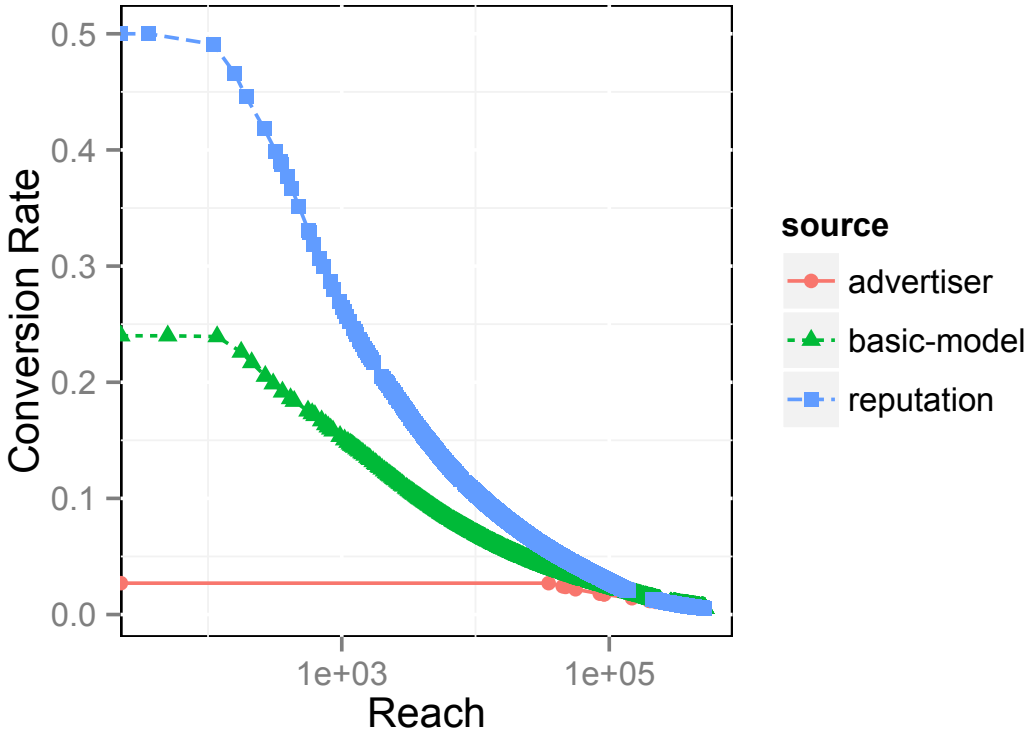

Figure 3.4: Cumulative conversion rate as reach increases (log scale)

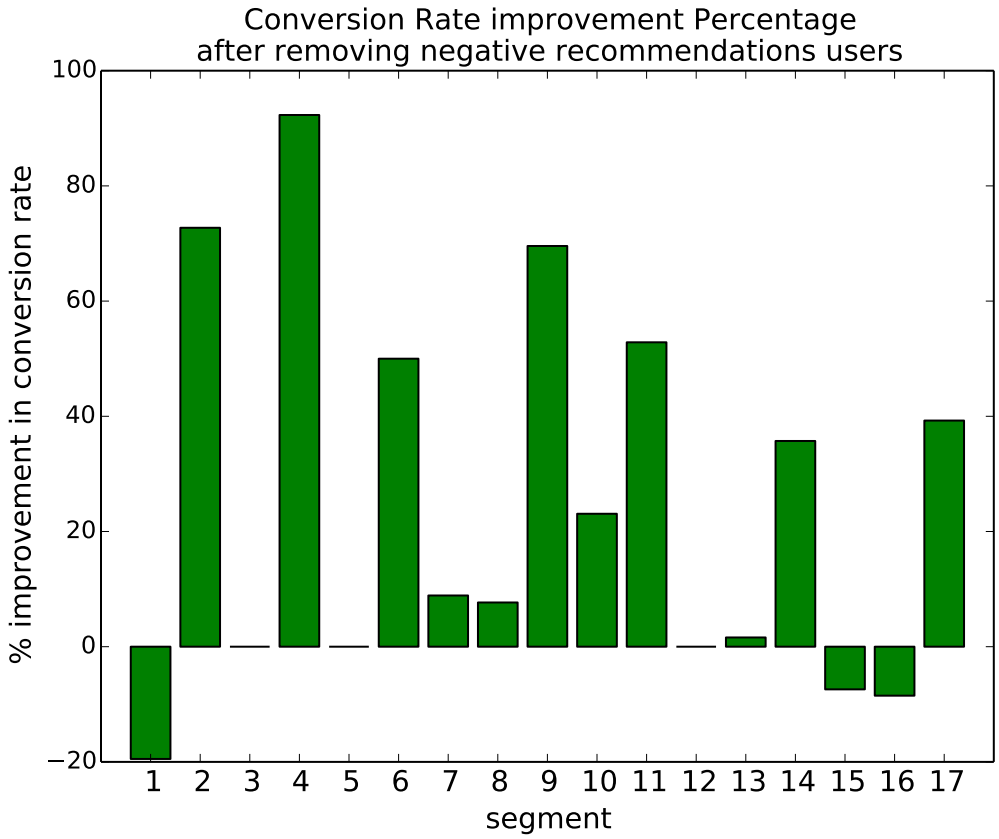

Figure 3.5: Negative Recommendations Effect: Conversion Rate improvement Percentage after removing negative recommendation users

removed. The results in Figure 3.5 show that the rates are improved by 7% on average across the segments.

3.5 Related Work

The current work mainly overlaps with two areas of related work; audience selection and reputation systems.

In audience selection, Pandey et.al. [52] present a description of targeting approaches and how focusing on conversion instead of clicks benefits targeting quality in behavioral targeting.

According to them, in audience selection related work, the three typical approaches for identifying the best users to target (ad targeting) are property targeting, user-segment targeting and behavioral targeting (BT). In property targeting, users expected to visit a particular page are targeted with the placement of particular ads on that page. In user-segment targeting user with common demographic features are targeted, such as age and gender, such that a meaningful user group is defined (young adults, for instance). In BT, users past history of online behavior is examined, including search queries, email responses, and browsing activities and users highly probable to convert are then targeted. In the first two cases groups of users are formed, while in BT users are targeted in individual level. Tyler et.al. [61] solve audience selection as a ranked retrieval problem. Fuxman et. al. [28] present an audience selection method that focuses on modeling user interests to infer targeting. Archak et.al. [10] describe ad factors based aggregation of user information that the advertiser can use to extract deeper insights about the effects of their ads. Bilenko et. al. [12] present a user personalized advertising model that build a user profile under the user's privacy control. Provost et. al. [54] suggest extracting quasi-social networks from browser behavior on user-generated content sites, for the purpose of finding good audiences for brand advertising. Kanagal et.al. [36] propose a focused matrix factorization model to learn user preferences towards specific campaign products, while also exploiting information about related products. Also, Aly et. al. [7] build a web-scale user modeling platform for optimizing display advertising targeting. Finally, Grbovich et. al. identify users to target based on advertisers expectation about user behavior using (manually defined rules [30].

Research on reputation systems is related to our work, as we build a reputation system for advertisers, and we derive reliability scores for users. In reputation systems, several works propose systems that represent the quality of the involved parts and methods to compute reputation scores

along with bias. In [19] and [20], Daltayanni et. al. propose WorkerRank, a reputation system to score workers and employers in an online labor marketplace. This work is also based on bipartite relations, similar to the approach in the current study. In [39] and [40], Kokkodis et. al. address data sparseness in building reputation systems in labor marketplaces. In [63], Weng et. al. build reputation scores such that they represent an influence measure for Twitter users. In [16], Chen et. al. discuss how to de-bias reputation in a comments rating environment. Finally, the works of Adler et. al. in [2] and [3] study reputation in the Wikipedia environment and they achieve to measure the quality of contributions. We also make a reference to recommender system works such as that of Adomavicius [4] who gives an overview of traditional recommender systems algorithms and categorizes them in content-based, collaborative, heuristic-based, model-based, or hybrid.

3.6 Conclusions

Audience selection is a hard problem that advertisers usually do not have enough data to solve; the available user behavior data is too large and too sparse and there are not enough informative signals to use in order to constrain the user space and select the best users suitable for an advertising campaign. In this study, we showed how a DSP that has data from many advertisers and users can help advertisers solve the above problem. We proposed Auto-Segmentation, a novel approach to combine the signals that we take from users and advertisers, using them within the context of a reputation system to automate user segmentation. We showed experimentally how we can use auto-segmentation for audience selection; first, we showed how it contributes to recommending segments of optimal conversion to new advertisers, significantly improving by $40 - 450\%$ the performance for new campaigns that face the cold start problem. Second, we showed how the

recommended segments can replace the existing ones thus contributing to refining the advertisers' campaigns and achieving better conversion rates. In our next steps we would explore how to use the advertiser targeting signal for other advertising problems such as bidding optimization.

Chapter 4

Conclusions & Future Work

4.1 Conclusions

The social evolution of the Web has paved the way for the introduction of a new generation of reputation systems that can leverage the unprecedented scale of ratings and reviews that are generated on a daily basis. While a decade ago, the opinions of a handful of friends were sufficient for the selection of a dining venue, users today feel confident about the quality of a restaurant only if its reputation is validated by a multitude of trusted users/reviewers. The magnitude of the available data has created new opportunities for reputation systems. For example, noisy signals that offer negligible information at a small scale can now be aggregated to provide valuable insights. At the same time dealing with ratings on a scale pose challenges to existing reputation algorithms. For example, while dishonest raters that occur 0.1% of the time can be safely ignored by a small scale reputation system, they can possibly represent millions of raters in a large-scale reputation system which subsequently requires for their behaviors to be modeled.

In this thesis we presented large-scale reputation systems for two different types of mar-

ketplaces. In Chapter 2 we introduced WorkerRank for labor marketplaces, a novel reputation system that scores workers by aggregating implicit judgements from employers. In a massive online labor marketplace the WorkerRank score plays the role of the word-of-mouth reputation of a worker in a small professional environment. In Chapter 3 we presented how to model the user responses to displayed ads as ratings in the context of a reputation system. Such responses are usually quite rare, since they happen in 0.01% of ad impressions. Despite the small response rate, the ads have tangible impact since the number of ad impressions can range from multiple millions to multiple billions. We then showed how to use the reputation system as part of Auto-Segmentation, an audience selection recommendation algorithm for advertisers.

In both types of marketplaces, the reputation systems that we presented improved the decision process of their users. In the case of labor marketplaces, the ultimate user of the worker reputation scores is an employer who is in the process of making a hiring decision among various workers that have expressed interest in his job posting. Our experiments with a real-world dataset from oDesk shows that our reputation system can improve the hiring prediction accuracy by 2 to 5 times when compared to the star rating system that is currently used in this marketplace. In the case of the advertising marketplaces, we employ the developed reputation system in our framework for audience selection recommendations. The user advertising scores improve the accuracy of our proposed targeting approach two-fold versus the approach without reputation scores.

Our experimental results throughout this thesis demonstrate the scalability of the proposed algorithms. In addition to asymptotic time analysis and applications to synthetic data, we showcase the effectiveness of our algorithms to real-world problems. In case of oDesk.com, we applied our reputation system to an application graph between employers and workers that has millions

of nodes and tens of millions edges. We did not make a comparison of the performance of our reputation system versus some baseline, but versus the existing star ratings system that is used in the marketplace. In the case of Turn, we trained the audience selection recommendation algorithm on a dataset with billions of ad impressions and millions of clicks. Finally, we compared the performance of our recommendations to the audience selections that were made by real advertisers.

4.2 Future Work

Our thesis has laid the foundations for several directions of future work. In particular, various ideas that we propose as part of WorkerRank can be applied to other domains. For example, the use of implicit signals that a user provides as part of his evaluation process is also relevant in other domains such as the evaluation of search results. While workers compete to get hired in a particular job, search results "compete" to draw the user's attention and satisfy his information need. In the same way that the hiring of a particular worker can provide us with information about his relative superiority to other applicants, the click on a certain search result conveys information about its relevance with respect to the rest of the results. In the future we plan to study a reputation system of searchers and web pages by considering the interaction with the search results as ratings.

We are also interested in studying how the use of reputation systems like WorkerRank can affect the application behavior of workers. Currently, workers tend to apply to as many jobs as time permits to maximize the chance of getting hired or receiving multiple offers. In a real life situation where they are aware that applications that receive unfavorable treatment may negatively affect their reputation scores, they would probably be more careful about where they apply. We are interested in studying the problem in game theoretic context where rational workers pick the jobs to apply not

only to maximize their probability of getting hired, but also to avoid the risk of doing harm to their reputation scores.

Finally, there is interesting follow-up applications to the work we have presented for advertising marketplaces. Audience selection is only the first step in the advertising process and in this thesis we showed how to improve its accuracy by using the advertisers/users reputation system. In the future we plan to use the reputation scores we find as features in the models that perform real-time bidding and quantify the possible impact there. Given the results we have in audience selection, we expect that the bidding process can also benefit from our approach.

Bibliography

[1] Elo rating system - Wikipedia, the free encyclopedia.

[2] B. Thomas Adler and Luca de Alfaro. A content-driven reputation system for the wikipedia. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 261–270, New York, NY, USA, 2007. ACM.

[3] B. Thomas Adler, Luca de Alfaro, Ian Pye, and Vishwanath Raman. Measuring author contributions to the wikipedia. In *Proceedings of the 4th International Symposium on Wikis*, WikiSym '08, pages 15:1–15:10, New York, NY, USA, 2008. ACM.

[4] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[5] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 183–194, New York, NY, USA, 2008. ACM.

[6] Timo Aho, Bernard Zenko, Saso Dzeroski, and Tapio Elomaa. Multi-target regression with rule ensembles. *Journal of Machine Learning Research*, 13:2367–2407, 2012.

[7] Mohamed Aly, Andrew Hatch, Vanja Josifovski, and Vijay K. Narayanan. Web-scale user modeling for targeting. In *Proceedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, pages 3–12, New York, NY, USA, 2012. ACM.

[8] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Discovering value from community activity on focused question answering sites: A case study of stack overflow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 850–858, New York, NY, USA, 2012. ACM.

[9] Nikolay Archak. Money, glory and cheap talk: Analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on topcoder.com. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 21–30, New York, NY, USA, 2010. ACM.

[10] Nikolay Archak, Vahab S. Mirrokni, and S. Muthukrishnan. Mining advertiser-specific user behavior using adfactors. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 31–40, New York, NY, USA, 2010. ACM.

[11] Jiang Bian, Yandong Liu, Ding Zhou, Eugene Agichtein, and Hongyuan Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 51–60, New York, NY, USA, 2009. ACM.

[12] Mikhail Bilenko and Matthew Richardson. Predictive client-side profiles for personalized ad-

vertising. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 413–421, New York, NY, USA, 2011. ACM.

[13] Allan Borodin, Gareth O Roberts, Jeffrey S Rosenthal, and Panayiotis Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Transactions on Internet Technology (TOIT)*, 5(1):231–297, 2005.

[14] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[15] Bee-Chung Chen, Anirban Dasgupta, Xuanhui Wang, and Jie Yang. Vote calibration in community question-answering systems. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 781–790, New York, NY, USA, 2012. ACM.

[16] Bee-Chung Chen, Jian Guo, Belle Tseng, and Jie Yang. User reputation in a comment rating environment. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 159–167, New York, NY, USA, 2011. ACM.

[17] Paolo D'Alberto and Ali Dasdan. Non-parametric information-theoretic measures of one-dimensional distribution functions from continuous time series. In *SDM*, pages 685–696. SIAM, 2009.

[18] Maria Daltayanni and Luca de Alfaro. Recommending workers in the labor marketplace. *In*

Workshop: Data Design for Personalization: Current Challenges and Emerging Opportunities. In conjunction with WSDM 2014.

[19] Maria Daltayanni, Luca de Alfaro, and Panagiotis Papadimitriou. Workerrank: Using employer implicit judgements to infer worker reputation. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 263–272, New York, NY, USA, 2015. ACM.

[20] Maria Daltayanni, Luca de Alfaro, Panagiotis Papadimitriou, and Panayiotis Tsaparas. On assigning implicit reputation scores in an online labor marketplace. In *EDBT*, pages 724–725, 2014.

[21] C. de Kerchove and P. van Dooren. The PageTrust Algorithm: How to rank web pages when negative links are allowed? In *SIAM: Data Mining Proceedings*, pages 346+, 2008.

[22] Chrysanthos Dellarocas. The digitization of word of mouth: Promise and challenges of online feedback mechanisms. *Manage. Sci.*, 49(10):1407–1424, October 2003.

[23] Chrysanthos Dellarocas. Reputation mechanisms. In *Handbook on Economics and Information Systems*, page 2006. Elsevier Publishing, 2006.

[24] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286, 2010.

[25] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation revisited.

[26] Arpad E. Elo. *The rating of chessplayers, past and present*. Arco Pub., New York, 1978.

[27] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Mach. Learn.*, 73(2):133–153, November 2008.

[28] Ariel Fuxman, Anitha Kannan, Jessie Li, and Panayiotis Tsaparas. Enabling direct interest-aware audience selection. In *International Conference on Information and Knowledge Management*. ACM, October 2012.

[29] Mark E Glickman. The glicko system. 1995.

[30] Mihajlo Grbovic and Slobodan Vucetic. Generating ad targeting rules using sparse principal component analysis with constraints. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 283–284, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.

[31] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[32] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskilltm: A bayesian skill rating system. In Bernhard Schoelkopf, John Platt, and Thomas Hoffman, editors, *NIPS*, pages 569–576. MIT Press, 2006.

[33] Nan Hu, Jie Zhang, and Paul A. Pavlou. Overcoming the j-shaped distribution of product reviews. *Commun. ACM*, 52(10):144–147, October 2009.

[34] Won-Seok Hwang, Ho-Jong Lee, Sang-Wook Kim, and Minsoo Lee. On using category experts for improving the performance and accuracy in recommender systems. In Xue wen

Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors, *CIKM*, pages 2355–2358. ACM, 2012.

[35] Pawel Jurczyk and Eugene Agichtein. Discovering authorities in question answer communities by using link analysis. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 919–922, New York, NY, USA, 2007. ACM.

[36] Bhargav Kanagal, Amr Ahmed, Sandeep Pandey, Vanja Josifovski, Lluis Garcia Pueyo, and Jeffrey Yuan. Focused matrix factorization for audience selection in display advertising. In *ICDE*, pages 386–397, 2013.

[37] Maurice Kendall and Jean D. Gibbons. *Rank Correlation Methods*. A Charles Griffin Title, 5 edition, September 1990.

[38] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[39] Marios Kokkodis and Panagiotis G. Ipeirotis. Have you done anything like that?: predicting performance using inter-category reputation. In Stefano Leonardi, Alessandro Panconesi, Paolo Ferragina, and Aristides Gionis, editors, *WSDM*, pages 435–444. ACM, 2013.

[40] Marios Kokkodis, Panagiotis Papadimitriou, and Panagiotis G. Ipeirotis. Hiring behavior models for online labor markets. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 223–232, 2015.

[41] Marios Kokkodis, Panagiotis Papadimitriou, and Ipeirotis Panagiotis. On hiring decisions in online labor markets. In *WSDM*, 2015.

[42] Sébastien Lahaie, David M Pennock, Amin Saberi, and Rakesh V Vohra. Sponsored search auctions. *Algorithmic game theory*, pages 699–716, 2007.

[43] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 641–650, New York, NY, USA, 2010. ACM.

[44] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1361–1370, New York, NY, USA, 2010. ACM.

[45] Yue Lu, Panayiotis Tsaparas, Alexandros Ntoulas, and Livia Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 691–700, New York, NY, USA, 2010. ACM.

[46] Frank McSherry and Marc Najork. Computing information retrieval performance measures efficiently in the presence of tied scores. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 414–421, Berlin, Heidelberg, 2008. Springer-Verlag.

[47] Abhinav Mishra and Arnab Bhattacharya. Finding the bias and prestige of nodes in networks based on trust scores. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 567–576, New York, NY, USA, 2011. ACM.

[48] mturk. Mechanical turk.

[49] Sergey I. Nikolenko and Alexander Sirotkin. A new bayesian rating system for team competitions. In Lise Getoor and Tobias Scheffer, editors, *ICML*, pages 601–608. Omnipress, 2011.

[50] oDesk. https://www.odesk.com.

[51] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[52] Sandeep Pandey, Mohamed Aly, Abraham Bagherjeiran, Andrew Hatch, Peter Ciccolo, Adwait Ratnaparkhi, and Martin Zinkevich. Learning to target: What works for behavioral targeting. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1805–1814, New York, NY, USA, 2011. ACM.

[53] Vasyl Pihur, Susmita Datta, and Somnath Datta. Weighted rank aggregation of cluster validation measures. *Bioinformatics*, 23(13):1607–1615, July 2007.

[54] Foster Provost, Brian Dalessandro, Rod Hook, Xiaohan Zhang, and Alan Murray. Audience selection for on-line brand advertising: Privacy-friendly social network targeting. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 707–716, New York, NY, USA, 2009. ACM.

[55] Chirag Shah and Jefferey Pomerantz. Evaluating and predicting answer quality in community qa. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 411–418, New York, NY, USA, 2010. ACM.

[56] Yannis Sismanis. How i won the "chess ratings - elo vs the rest of the world" competition. *CoRR*, abs/1012.4571, 2010.

[57] Maggy Anastasia Suryanto, Ee Peng Lim, Aixin Sun, and Roger H. L. Chiang. Quality-aware collaborative question answering: Methods and evaluation. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 142–151, New York, NY, USA, 2009. ACM.

[58] Chun How Tan, Eugene Agichtein, Panos Ipeirotis, and Evgeniy Gabrilovich. Trust, but verify: Predicting contribution quality for knowledge base construction and curation. In *WSDM*, 2014.

[59] Panayiotis Tsaparas, Alexandros Ntoulas, and Evimaria Terzi. Selecting a comprehensive set of reviews. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 168–176, New York, NY, USA, 2011. ACM.

[60] Grigorios Tsoumakas, Eleftherios Spyromitros Xioufis, Aikaterini Vrekou, and Ioannis P. Vlahavas. Multi-target regression via random linear target combinations. *CoRR*, abs/1404.5065, 2014.

[61] Sarah K. Tyler, Sandeep Pandey, Evgeniy Gabrilovich, and Vanja Josifovski. Retrieval models for audience selection in display advertising. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 593–598, New York, NY, USA, 2011. ACM.

[62] Ellen M. Voorhees and Dawn M. Tice. The trec-8 question answering track evaluation. In *TREC*, 1999.

[63] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM.

[64] Bernard Zenko and Saso Dzeroski. Learning classification rules for multiple target attributes. In Takashi Washio, Einoshin Suzuki, Kai Ming Ting, and Akihiro Inokuchi, editors, *PAKDD*, volume 5012 of *Lecture Notes in Computer Science*, pages 454–465. Springer, 2008.

[65] Jun Zhang, Mark S. Ackerman, and Lada Adamic. Expertise networks in online communities: Structure and algorithms. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 221–230, New York, NY, USA, 2007. ACM.