RTB Optimization for Online Advertising

Sponsor: GumGum (Santa Monica, CA)

Industry Mentor: Ken Weiner

Introduction

GumGum is an online advertising platform. Our company distributes ads through a number of ad channels. One important distribution channel is Real-Time Bidding (RTB) where ad impressions are sold via programmatic instantaneous auction.

When we get an opportunity to serve an ad, GumGum sends RTB bid requests to DSP's (demand side platforms). Each DSP has milliseconds to submit a bid with a price based on information in the bid request (browser, user, site, page, image, etc.) and a floor price. GumGum computes the floor price by considering the cost of that impression (what we pay the owner of the website) and a margin that we can adjust. Once we collect all the bids submitted for that impression we determine the winner.

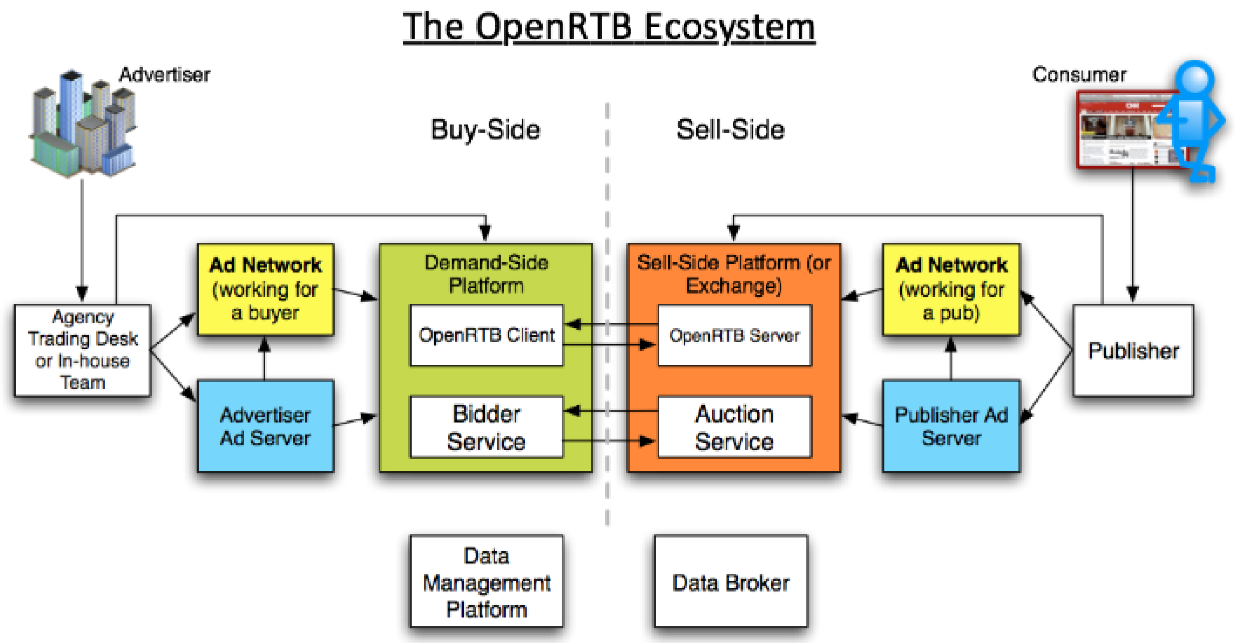
We would like to improve the way we are setting the floor price in order to maximize revenue per impression. This may involve improving the existing algorithm (see next section) or come up with a completely different approach to RTB optimization.

Real-Time Bidding (RTB)

RTB allows online advertising companies to run their inventory on a per-impression basis. To do so, they have to enter a programmatic instantaneous auction. In more detail [1]:

A typical transaction begins with a user visiting a website. This triggers a bid request that can include various pieces of data such as the user's demographic information, browsing history, location, and the page being loaded. The request goes from the publisher to an ad exchange, which submits it and the accompanying data to multiple advertisers who automatically submit bids in real time to place their ads. Advertisers bid on each ad impression as it is served. The impression goes to the highest bidder and their ad is served on the page. This process is repeated for every ad slot on the page. Real time bidding transactions typically happen within 100 milliseconds from the moment the ad exchange received the request.

The following figure shows all involved parties and their interactions:



GumGum is an RTB ad exchange on the sell side (orange box in the above figure). We collect bid request data, submit the request to potential bidders, collect their bids and determine the winning bid.

Auction, bids and floor price

Every bid request includes a floor price. This is the minimum bid accepted and consists of the amount that the publisher would like to receive for displaying an ad on their page, plus a margin constituting revenue for GumGum. Determining the optimal floor price for each bid request is at the heart of this project.

After collecting the various bids we perform what is known as a 2nd price auction where the highest bid wins but pays a clearing price that is 1 cent higher than the 2nd highest bid price (or the floor price if there are no other competing bids).

Current approach: Multi-armed bandit

GumGum currently uses an epsilon greedy multi-armed bandit algorithm (MAB) to optimize floor prices for a segment that includes a zone/country combination. MAB models attempt to find a balance between acquiring new knowledge (“exploration”) and optimizing decisions based on existing knowledge (“exploitation”). The uncertainty of change in DSP behavior make MABs an attractive choice.

Formally MABs consist of a set of real distributions

$$B = \{R_1, \ldots, R_K\}$$

where each R reflects the potential rewards associated with pulling lever *k*. The goal is to minimize the so-called *regret* associated with pulling the wrong levers:

$$\rho = T\mu^* - \sum_{t=1}^{T} \widehat{r}_t$$

which is the difference between expected and actual rewards.

The epsilon greedy MAB variation attempts to find the ideal balance between exploration and exploitation by applying a known optimal value a certain number of times (often around 90% of all trials) while trying out different values during the remaining trials.

MABs are just one of many possible approaches to RTB optimization. We welcome our team to focus on optimizing our existing MAB model, but we also encourage them to explore other approaches. Our only non-negotiable requirement concerns the algorithm's speed. Given the fast nature of programmatic RTB any proposed algorithm shall not take more than eight milliseconds per bid request.

Provided data

GumGum stores RTB data in log files. Each line contains a JSON string which consisting of three main entries.

Ad event data

This section lists user data such as their computer's environment variables, their country, the time of browsing, etc. For instance:

KEY	EXPLANATIO N	EXAMPLE
id	unique identifier	
ua	user agent	Mozilla/5.0 (Linux; U; Android 4.2.2; es-es; Luno Build/OrangeLuno) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30
ip	user IP address	213.143.48.97
cc	user country	ES
ct	user city	Barcelona
isp	user's ISP	Orange Espana

lt	local time	Wed May 27 04:01:17 +0200 2015

Bid request data

This section lists all parties to whom the bid request is sent, along with detailed information regarding the ad space (web page and image).

KEY	EXPLANATION	EXAMPLE
bidder	name of the bidder	
verticalid	web page content type	
bidfloor		2.45000002
impressions	details regarding ad type, number of slots, etc	
site	details about site, such as URL, important page keywords, categories, etc	

Bid results

This section lists bids received during the auction and identifies the winning bid.

KEY	EXPLANATION	EXAMPLE
bidder	name of the bidder	
price	price submitted by bidder	
winner	boolean value indicating if submitted bid won the auction	

GumGum is adhering to the OpenRTB specifications [6] for its ad exchange. It provides documentation for most of the data points used in our log files.

We are planning on providing a complete data set from May 2015. The data will be available as uncompressed log files with a size of about 1.5TB.

Testing

Due to the nature of RTB auctions we cannot use a historical dataset to test our team's algorithm. Instead we will allow the team to submit to us algorithms which we will run as an A/B test in our production environment along with our current algorithm. An algorithm's performance relative to our current system as well as previously submitted algorithms should help our team to measure their progress.

Special Requirements

Programming language: At GumGum we use Groovy and Java as our main programming languages. However we will accept solutions in any programming language.

Computer Hardware: We will provide our team with sufficient computational resources from Amazon's AWS cloud services, such as a virtual computing environment (EC2) and storage. We will provide the academic mentor with all necessary information to access the resources prior to June 22nd.

Software Packages: Our team may use any open source software package.

Desired Outcomes

We expect our team to

- acquire full understanding of the problem space
- provide at least one solution that may improve our current RTB algorithm; we encourage our team to try multiple solutions
- test their proposed algorithm(s) in a real-case environment that we will provide
- provide detailed documentation of their proposed solution(s), including commented code

Recommended Reading

1. Introduction to Real Time Bidding: http://en.wikipedia.org/wiki/Real-time_bidding
2. Introduction to Multi-Armed Bandit Algorithm: http://en.wikipedia.org/wiki/Multi-armed_bandit
3. White, John Myles: Bandit Algorithms for Website Optimization. (You can download this book for free here: http://it-ebooks.info/book/1341/)
4. Introduction to Second Price Auction: http://en.wikipedia.org/wiki/Vickrey_auction
5. Myerson, Rogers: Optimal Auction Design (1961): http://www.eecs.harvard.edu/~parkes/cs286r/spring07/papers/myerson.pdf
6. OpenRTB specifications: http://www.iab.net/media/file/OpenRTB-API-Specification-Version-2-3.pdf