# ICML'11 Tutorial: Recommender Problems for Web Applications

Deepak Agarwal and Bee-Chung Chen

*Yahoo! Research*

# Other Significant Y! Labs Contributors

- ## Content Targeting
  - Pradheep Elango
  - Rajiv Khanna
  - Raghu Ramakrishnan
  - Xuanhui Wang
  - Liang Zhang

- ## Ad Targeting
  - Nagaraj Kota

# Agenda

- Topic of Interest
  - Recommender problems for **dynamic**, **time-sensitive** applications
    - Content Optimization (main focus today), Online Advertising, Movie recommendation, shopping,…

- Introduction (20 min, Deepak)

- Offline components (40 min, Deepak)
  - Regression, Collaborative filtering (CF), …

- Online components + initialization (70 min, Bee-Chung)
  - Time-series, online/incremental methods, explore/exploit (bandit)

- Evaluation methods + Multi-Objective (10-15 min, Deepak)

- Challenges (5-10 min, Deepak)

# Three components we will focus on today

- Defining the problem
  - Formulate objectives whose optimization achieves  some long-term goals for the recommender system
    - E.g. How to serve content to optimize audience reach and engagement, optimize some combination of engagement and revenue ?

- Modeling (to estimate some critical inputs)
  - Predict rates of some positive user interaction(s) with items based on data obtained from historical user-item interactions
    - E.g. Click rates, average time-spent on page, etc
    - Could be explicit feedback like ratings

- Experimentation
  - Create experiments to collect data proactively to improve models, helps in converging to the best choice(s) cheaply and rapidly.
    - Explore and Exploit (continuous experimentation)
    - DOE (testing hypotheses by avoiding bias inherent in data)

# Modern Recommendation Systems

- Goal
  - Serve the right item to a user in a given context to optimize long-term business objectives

- A scientific discipline that involves
  - Large scale Machine Learning & Statistics
    - Offline Models (capture global & stable characteristics)
    - Online Models (incorporates dynamic components)
    - Explore/Exploit (active and adaptive experimentation)
  - Multi-Objective Optimization
    - Click-rates (CTR), Engagement, advertising revenue, diversity, etc
  - Inferring user interest
    - Constructing User Profiles
  - Natural Language Processing to understand content
    - Topics, "aboutness", entities, follow-up of something, breaking news,…

# Some examples from content optimization

- ## Simple version
  - I have a content module on my page, content inventory is obtained from a third party source which is further refined through editorial oversight. Can I algorithmically recommend content on this module? I want to improve overall click-rate (CTR) on this module

- ## More advanced
  - I got X% lift in CTR. But I have additional information on other downstream utilities (e.g. advertising revenue). Can I increase downstream utility without losing too many clicks?

- ## Highly advanced
  - There are multiple modules running on my webpage. How do I perform a simultaneous optimization?

Recommend search queries

Recommend packages:
  Image
  Title, summary
  Links to other pages

Pick 4 out of a pool of *K*
  *K* = 20 ~ 40
  Dynamic

Routes traffic other pages
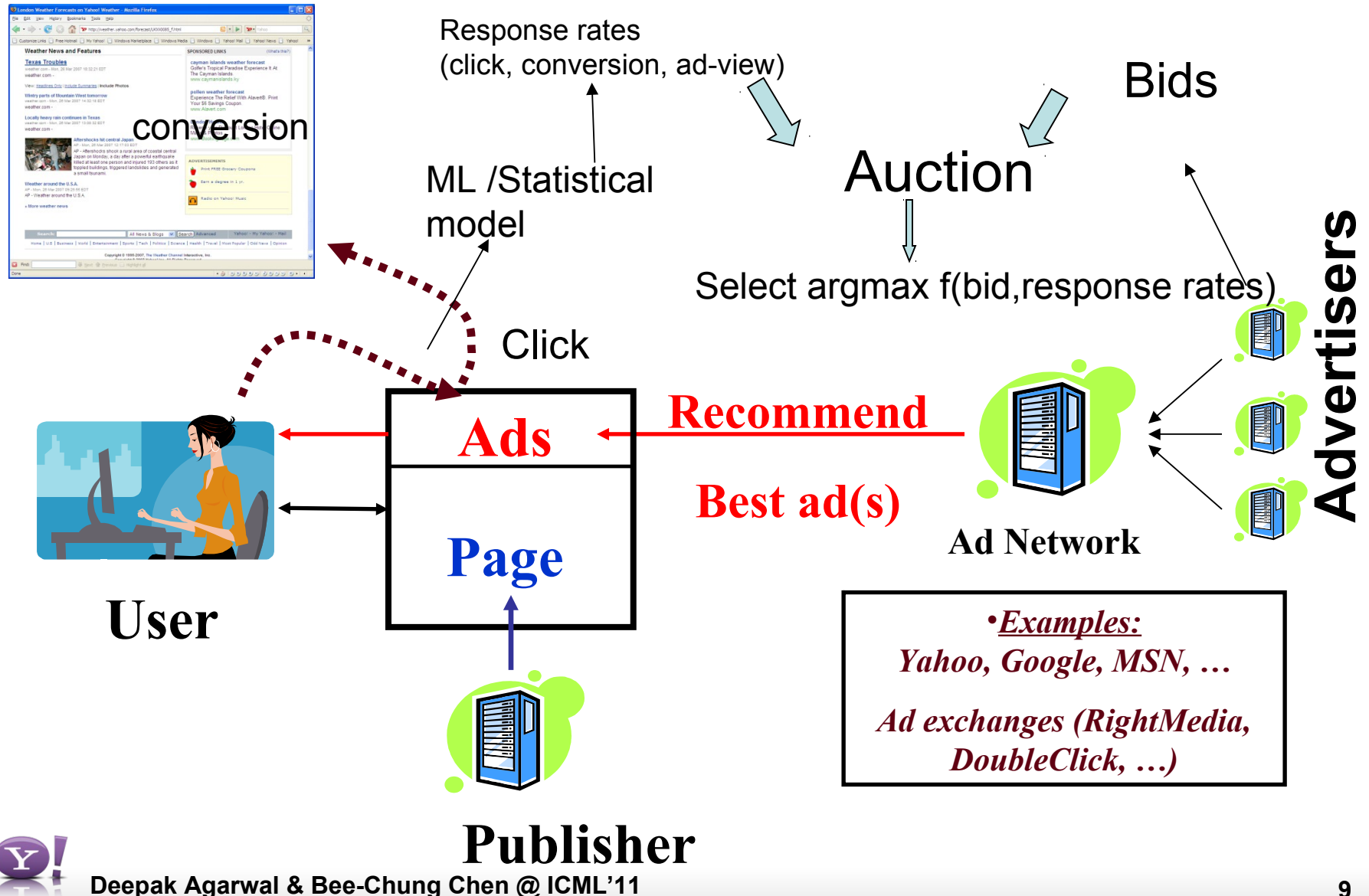
Recommend applications

Recommend news article

# Problems in this example

- Optimize CTR on multiple modules
  - Today Module, Trending Now, Personal Assistant, News
  - Simple solution: Treat modules as independent, optimize separately. May not be the best when there are strong correlations.

- For any single module
  - Optimize some combination of CTR, downstream engagement, and perhaps advertising revenue.
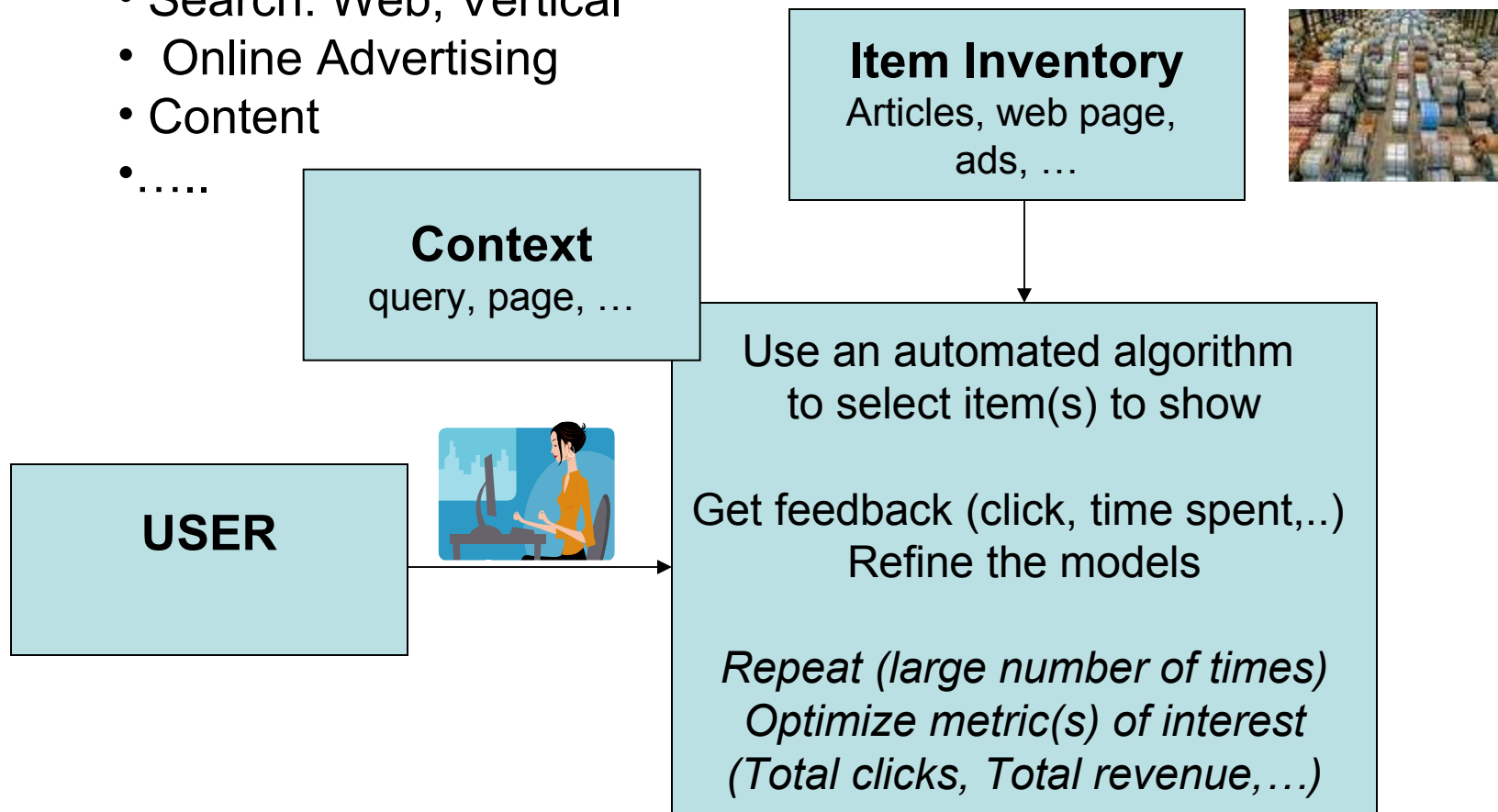
# Online Advertising



Response rates
(click, conversion, ad-view)

Bids

conversion

ML /Statistical model

Auction

Select argmax f(bid,response rates)

Click

Ads

**Recommend**

**Best ad(s)**

Page

Ad Network

Advertisers

User

•*Examples:*
*Yahoo, Google, MSN, …*

*Ad exchanges (RightMedia, DoubleClick, …)*

**Publisher**

# Recommender problems in general

- Example applications
  - Search: Web, Vertical
  - Online Advertising
  - Content
  - …..

**Item Inventory**
Articles, web page, ads, …

**Context**
query, page, …

**USER**

Use an automated algorithm to select item(s) to show

Get feedback (click, time spent,..) Refine the models

*Repeat (large number of times) Optimize metric(s) of interest (Total clicks, Total revenue,…)*

# Important Factors

- **Items:** Articles, ads, modules, movies, users, updates, etc.

- **Context:** query keywords, pages, mobile, social media, etc.

- **Metric** to optimize (e.g., relevance score, CTR, revenue, engagement)
  - Currently, most applications are single-objective
  - Could be multi-objective optimization (maximize $X$ subject to $Y$, $Z$,..)

- **Properties of the item pool**
  - Size (e.g., all web pages vs. 40 stories)
  - Quality of the pool (e.g., anything vs. editorially selected)
  - Lifetime (e.g., mostly old items vs. mostly new items)

# Factors affecting Solution (continued)

- **Properties of the context**
  - Pull: Specified by explicit, user-driven query (e.g., keywords, a form)
  - Push: Specified by implicit context (e.g., a page, a user, a session)
    - Most applications are somewhere on continuum of pull and push

- **Properties of the feedback on the matches made**
  - Types and semantics of feedback (e.g., click, vote)
  - Latency (e.g., available in 5 minutes vs. 1 day)
  - Volume (e.g., 100K per day vs. 300M per day)

- **Constraints specifying legitimate matches**
  - e.g., business rules, diversity rules, editorial Voice
  - Multiple objectives

- **Available Metadata** (e.g., link graph, various user/item attributes)

# Predicting User-Item Interactions (e.g. CTR)

- Myth: We have so much data on the web, if we can only process it the problem is solved
  - Number of things to learn increases with sample size
    - Rate of increase is not slow
  - Dynamic nature of systems make things worse
  - We want to learn things quickly and react fast

- Data is sparse in web recommender problems
  - We lack enough data to learn all we want to learn and as quickly as we would like to learn
  - Several Power laws interacting with each other
    - E.g. User visits power law, items served power law
      - Bivariate Zipf: Owen & Dyer, 2011

# Can Machine Learning help?

- Fortunately, there are group behaviors that generalize to individuals & they are relatively stable
  - E.g. Users in San Francisco tend to read more baseball news

- Key issue: Estimating such groups
  - Coarse group : more stable  but does not generalize that well.
  - Granular group: less stable with few individuals
  - Getting a good grouping structure is to hit the "sweet spot"

- Another big advantage on the web
  - Intervene and run small experiments on a small population to collect data that helps rapid convergence to the best choices(s)
    - We don't need to learn all user-item interactions, only those that are good.

# Predicting user-item interaction rates



Feature construction
Content: IR, clustering, taxonomy, entity,..
User profiles: clicks, views, social, community,..

Offline
( Captures stable characteristics
at coarse resolutions)
(Logistic, Boosting,….)

Initialize

Online
(Finer resolution
Corrections)
(item, user level)
(Quick updates)

Explore/Exploit
(Adaptive sampling)
(helps rapid convergence
to best choices)
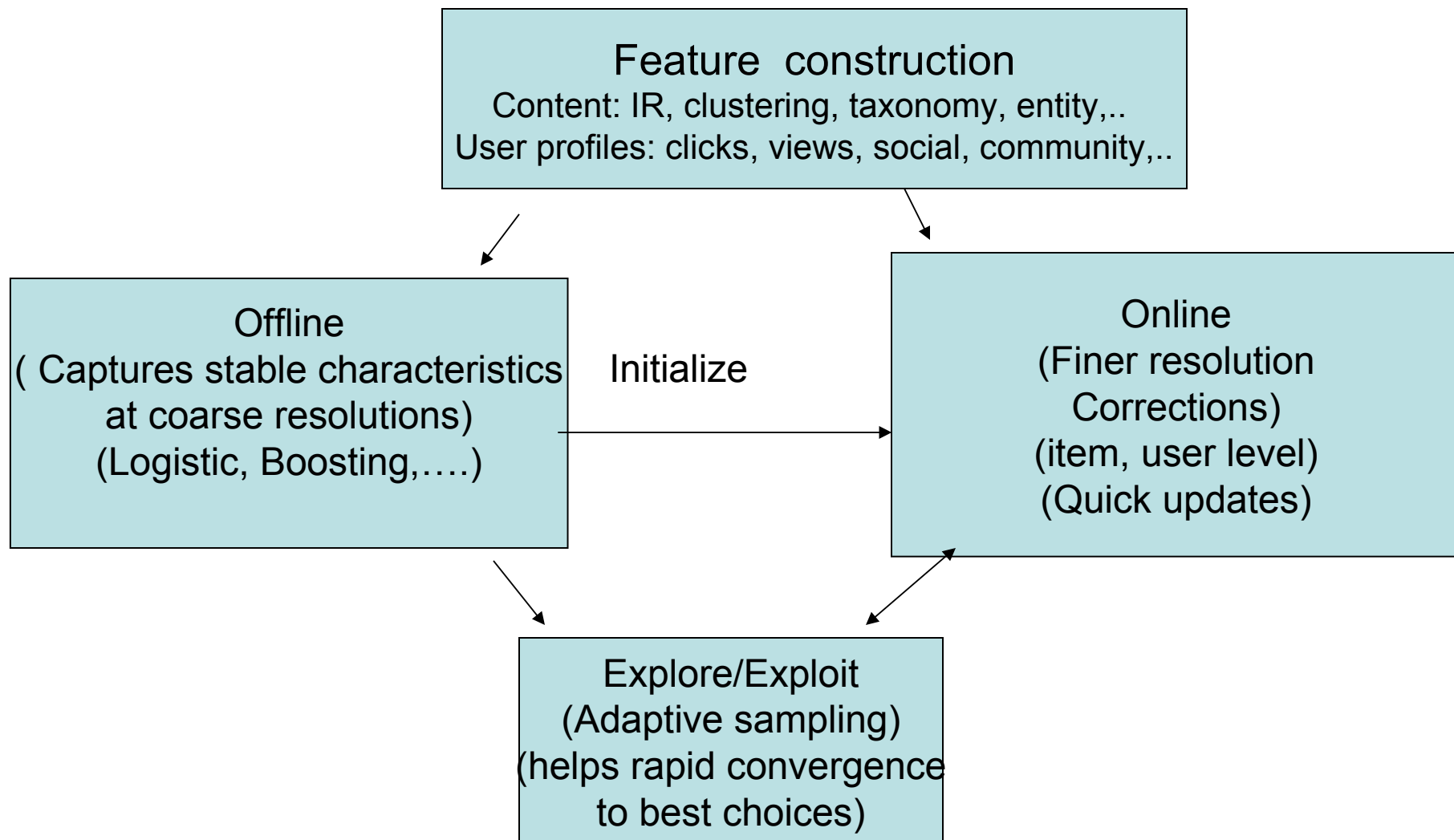
# Post-click: An example in Content Optimization



Recommender  +  EDITORIAL

content

Clicks on FP links influence downstream supply distribution

SPORTS

NEWS

OMG
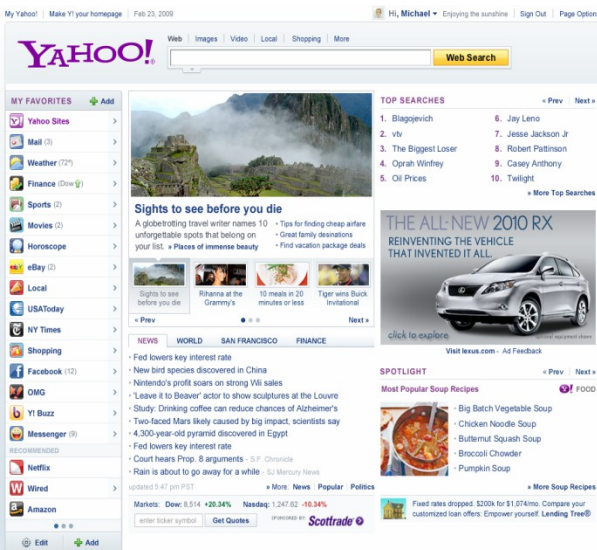
FINANCE

AD SERVER

DISPLAY ADVERTISING Revenue

Downstream engagement

(Time spent)

- ## What do we want to optimize?

- Current: Maximize clicks (maximize downstream supply from FP)

- But consider the following
  - Article 1: CTR=5%, utility per click = 5
  - Article 2: CTR=4.9%, utility per click=10
    - By promoting 2, we lose 1 click/100 visits, gain 5 utils

- If we do this for a large number of visits --- lose some clicks but obtain significant gains in utility?
  - E.g. lose 5% relative CTR, gain 40% in utility (revenue, engagement, etc)
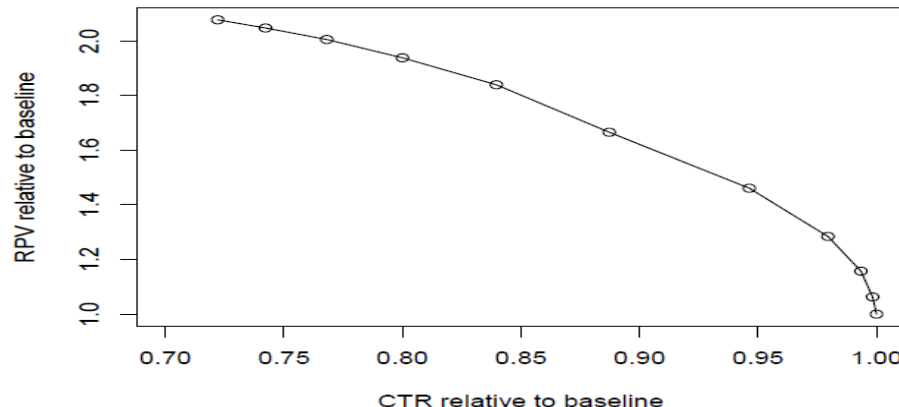
**Example Application:
Today Module on Yahoo! Homepage**

Currently in production, powered by some methods
discussed in this tutorial

Recommend packages:
  Image
  Title, summary
  Links to other pages

Pick 4 out of a pool of $K$
  $K = 20 \sim 40$
  Dynamic

Routes traffic other pages

# Problem definition

- Display "best" articles for each user visit

- Best - Maximize User Satisfaction, Engagement
  - BUT Hard to obtain quick feedback to measure these

- Approximation
  - Maximize utility based on immediate feedback (click rate) subject to constraints (relevance, freshness, diversity)

- Inventory of articles?
  - Created by human editors
  - Small pool (30-50 articles) but refreshes periodically

# Where are we today?

- ## Before this research
  - Articles created and selected for display by editors

- ## After this research
  - Article placement done through statistical models

- ## How successful ?

"Just look at our homepage, for example. Since we began pairing our content optimization technology with editorial expertise, we've seen click-through rates in the Today module more than double. ----- Carol Bartz, CEO Yahoo! Inc (Q4, 2009)

# Main Goals

- ## Methods to select most popular articles
  - This was done by editors  before

- ## Provide personalized article selection
  - Based on user covariates
  - Based on per user behavior

- ## Scalability: Methods to generalize in small traffic scenarios
  - Today module part of most Y! portals around the world
  - Also syndicated to sources like Y! Mail, Y! IM etc

# Similar applications

- Goal: Use same methods for selecting most popular, personalization across different applications at Y!
- Good news! Methods generalize, already in use

# Next few hours

|  | Most Popular Recommendation | Personalized Recommendation |
|---|---|---|
| **Offline Models** |  | Collaborative filtering (cold-start problem) |
| **Online Models** | Time-series models | Incremental CF, online regression |
| **Intelligent Initialization** | Prior estimation | Prior estimation, dimension reduction |
| **Explore/Exploit** | Multi-armed bandits | Bandits with covariates |