Advertising on the Web

Advanced Search Techniques for Large Scale Data Analytics

Pavel Zezula and Jan Sedmidubsky

Masaryk University
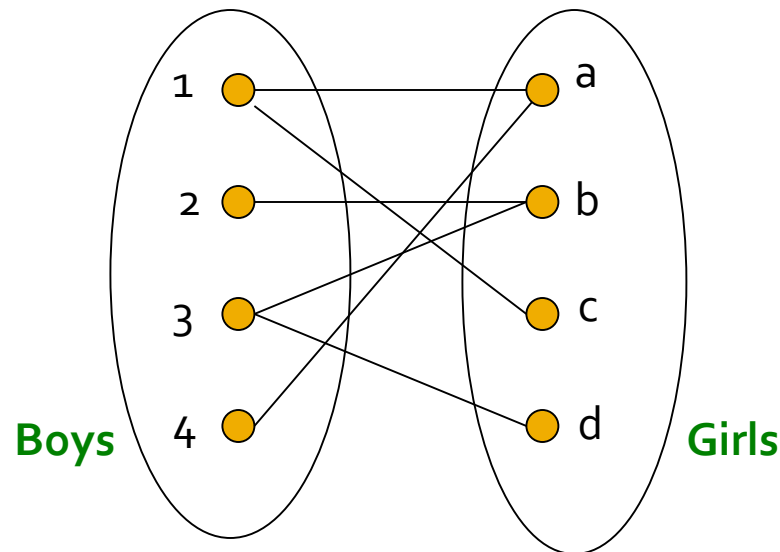
http://disa.fi.muni.cz

Online Algorithms

- ## Classic model of algorithms

  - You get to see the entire input, then compute some function of it

  - In this context, "offline algorithm"

- ## Online Algorithms

  - You get to see the input one piece at a time, and need to make irrevocable decisions along the way

  - **Similar to the data stream model**
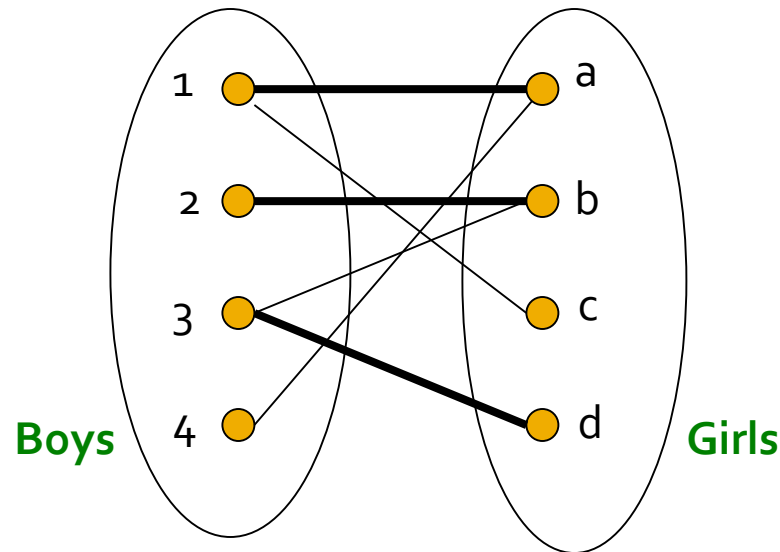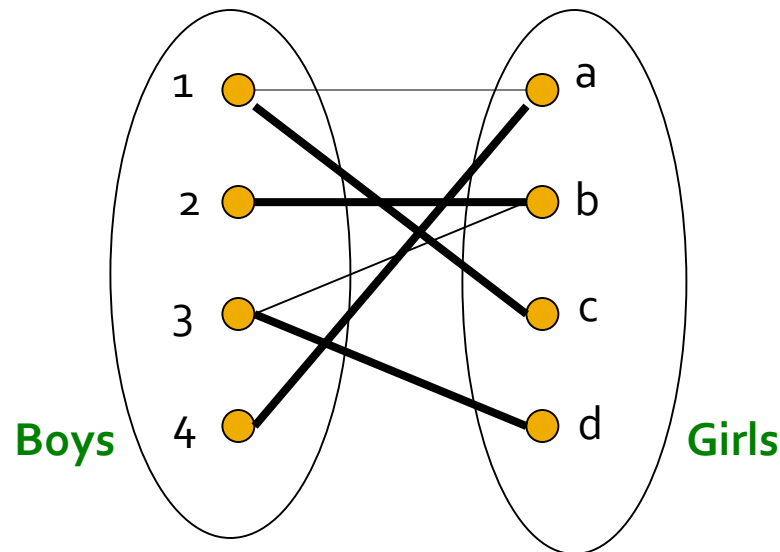
Online Bipartite Matching

Nodes: Boys and Girls; Edges: Preferences

Goal: Match boys to girls so that maximum number of preferences is satisfied

Example: Bipartite Matching

1 — a
2 — b
3 — c
4 — d

Boys   Girls

M = {(1,a),(2,b),(3,d)} is a **matching**
Cardinality of matching = |M| = 3

Example: Bipartite Matching

Boys 1 2 3 4 **Girls** a b c d

M = {(1,c),(2,b),(3,d),(4,a)} is a
perfect matching

Perfect matching … all vertices of the graph are matched
Maximum matching … a matching that contains the largest possible number of matches
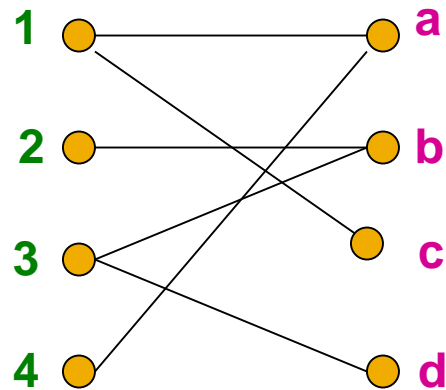
Matching Algorithm

- **Problem: Find a maximum matching for a given bipartite graph**
  - A perfect one if it exists

- There is a polynomial-time offline algorithm based on augmenting paths (Hopcroft & Karp 1973, see http://en.wikipedia.org/wiki/Hopcroft-Karp_algorithm)

- **But what if we do not know the entire graph upfront?**

Online Graph Matching Problem

- Initially, we are given the set **boys**
- In each **round**, **one girl's choices are revealed**
  - That is, girl's **edges** are revealed
- **At that time, we have to decide to either:**
  - Pair the **girl** with a **boy**
  - Do not pair the **girl** with any **boy**

- **Example of application:**
    Assigning tasks to servers

Online Graph Matching: Example

(1,a)
(2,b)
(3,d)

Greedy Algorithm

- **Greedy algorithm for the online graph matching problem:**

  - Pair the new girl with **any** eligible boy

    - If there is none, do not pair girl

- **How good is the algorithm?**

Competitive Ratio

- For input I, suppose greedy produces matching M_{greedy} while an optimal matching is M_{opt}

Competitive ratio =

$$min_{all\ possible\ inputs\ I}\ (|M_{greedy}|/|M_{opt}|)$$

(what is greedy's <u>worst</u> performance <u>over all possible</u> inputs I)

Analyzing the Greedy Algorithm

- Consider a case: $M_{greedy} \neq M_{opt}$
- Consider the set G of girls matched in M_{opt} but not in M_{greedy}
- Then every boy B <u>adjacent</u> to girls in G is already matched in M_{greedy}:

  - If there would exist such non-matched (by M_{greedy}) boy adjacent to a non-matched girl then greedy would have matched them

- Since boys B are already matched in M_{greedy} then
  (1) $|M_{greedy}| \geq |B|$

M_{opt}

M_{greedy}

1   a
2   b
3   c
4   d

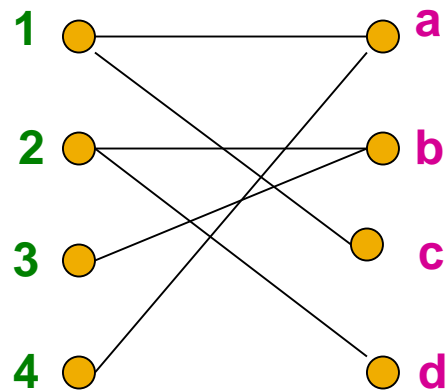$B=\{ \bullet \bullet \}$ $G=\{ \bullet \}$
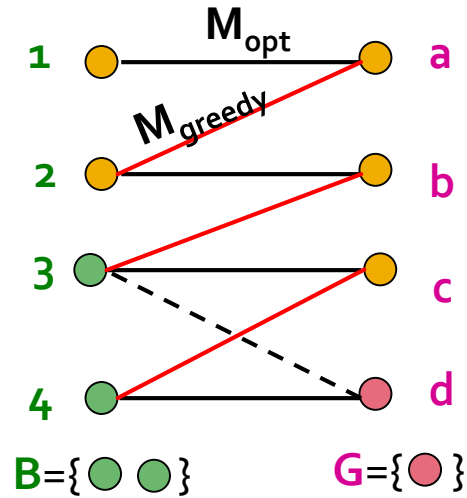
Analyzing the Greedy Algorithm

- **Summary so far:**
  - Girls G matched in M_{opt} but not in M_{greedy}
  - **(1)** $|M_{greedy}| \geq |B|$
- There are at least $|G|$ such boys ($|G| \leq |B|$) otherwise the optimal algorithm couldn't have matched all girls in G
  - So: $|G| \leq |B| \leq |M_{greedy}|$
- By definition of G also: $|M_{opt}| \leq |M_{greedy}| + |G|$
  - Worst case is when $|G| = |B| = |M_{greedy}|$
- $|M_{opt}| \leq 2|M_{greedy}|$ then $|M_{greedy}|/|M_{opt}| \geq 1/2$
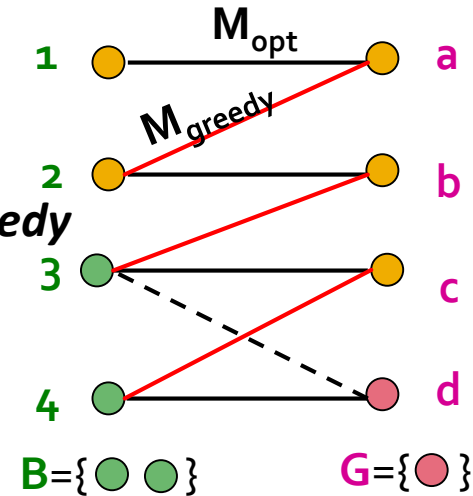
Worst-case Scenario

(1,a)
(2,b)

Web Advertising

History of Web Advertising

- **Banner ads** **(1995-2001)**
  - Initial form of web advertising
  - Popular websites charged *X*$ for every 1,000 "impressions" of the ad
    - Called "**CPM**" rate (Cost per thousand impressions)
    - Modeled similar to TV, magazine ads
  - From **untargeted** to **demographically targeted**
  - **Low click-through rates**
    - Low ROI for advertisers

CPM…cost per *mille*
Mille…thousand in Latin

Performance-based Advertising

- **Introduced by Overture around 2000**
  - Advertisers **bid** on **search keywords**
  - When someone searches for that keyword, the **highest bidder's ad is shown**
  - Advertiser is charged only if the ad is clicked on

- Similar model adopted by Google with some changes around 2002
  - Called **Adwords**

Ads vs. Search Results

Web 2.0

- **Performance-based advertising works!**
  - Multi-billion-dollar industry

- **Interesting problem:**
  What ads to show for a given query?
  - (Today's lecture)

- **If I am an advertiser, which search terms should I bid on and how much should I bid?**
  - (Not focus of today's lecture)

Adwords Problem

- **Given:**
  - **1.** A set of bids by advertisers for search queries
  - **2.** A click-through rate for each advertiser-query pair
  - **3.** A budget for each advertiser (say for 1 month)
  - **4.** A limit on the number of ads to be displayed with each search query
- **Respond to each search query with a set of advertisers such that:**
  - **1.** The size of the set is no larger than the limit on the number of ads per query
  - **2.** Each advertiser has bid on the search query
  - **3.** Each advertiser has enough budget left to pay for the ad if it is clicked upon

Adwords Problem

- A stream of queries arrives at the search engine: $q_1, q_2, ...$
- Several advertisers bid on each query
- When query q_i arrives, search engine must pick a subset of advertisers whose ads are shown

- **Goal: Maximize search engine's revenues**
  - **Simple solution:** Instead of raw bids, use the "**expected revenue per click**" (i.e., **Bid*CTR**)
- **Clearly we need an online algorithm!**

The Adwords Innovation

Advertiser	Bid	CTR	Bid * CTR
A	$1.00	1%	1 cent
B	$0.75	2%	1.5 cents
C	$0.50	2.5%	1.125 cents
		Click through rate	Expected revenue

Complications: Budget

- **Two complications:**
  - **Budget**
  - **CTR of an ad is unknown**

- **Each advertiser has a limited budget**
  - **Search engine guarantees that the advertiser will not be charged more than their daily budget**

Complications: CTR

- **CTR: Each ad has a different likelihood of being clicked**

  - **Advertiser 1** bids $2, click probability = 0.1

  - **Advertiser 2** bids $1, click probability = 0.5

  - **Clickthrough rate (CTR)** is measured **historically**

    - **Very hard problem: Exploration vs. exploitation**
    Exploit: Should we keep showing an ad for which we have good estimates of click-through rate
    or
    Explore: Shall we show a brand new ad to get a better sense of its click-through rate

Greedy Algorithm

- **Our setting: Simplified environment**
  - There is **1** ad shown for each query
  - All advertisers have the same budget **B**
  - All ads are equally likely to be clicked
  - Value of each ad is the same (=**1**)

- **Simplest algorithm is greedy:**
  - For a query pick any advertiser who has bid **1** for that query
  - **Competitive ratio of greedy is 1/2**

Bad Scenario for Greedy

- **Two advertisers A and B**

  - *A* bids on query *x*, *B* bids on *x* and *y*

  - Both have budgets of **\$4**

- **Query stream: *x x x x y y y y***

  - Worst case greedy choice: ***B B B B* _ _ _ _**

  - Optimal: **A A A A B B B B**

  - **Competitive ratio = ½**

- **This is the worst case!**

  - **Note:** Greedy algorithm is deterministic – it always resolves draws in the same way

BALANCE Algorithm [MSVV]

- **BALANCE** Algorithm by Mehta, Saberi, Vazirani, and Vazirani

  - **For each query, pick the advertiser with the largest unspent budget**

   - Break ties arbitrarily (**but in a deterministic way**)

Example: BALANCE

- **Two advertisers A and B**
  - **A** bids on query *x*, **B** bids on *x* and *y*
  - Both have budgets of **$4**

- **Query stream:** *x x x x y y y y*

- **BALANCE choice: A B A B B B _ _**
  - Optimal: **A A A A B B B B**

- **In general:** For **BALANCE** on **2** advertisers
Competitive ratio = ¾

Analyzing BALANCE

- **Consider simple case (w.l.o.g.):**
  - **2** advertisers, A_1 and A_2, each with budget **B** (≥ 1)
  - Optimal solution exhausts both advertisers' budgets

- **BALANCE must exhaust at least one advertiser's budget:**
  - **If not, we can allocate more queries**
    - Whenever BALANCE makes a mistake (both advertisers bid on the query), advertiser's unspent budget only decreases
    - Since optimal exhausts both budgets, one will for sure get exhausted
  - Assume BALANCE exhausts A_2's budget, but allocates *x* queries fewer than the optimal
  - **Revenue: *BAL = 2B - x***

Analyzing Balance
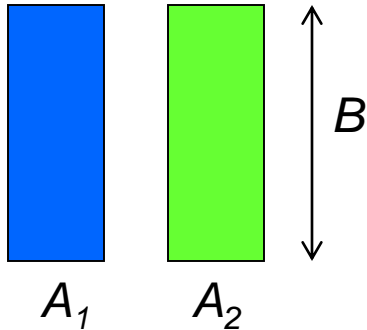
■ Queries allocated to A_1 in the optimal solution

■ Queries allocated to A_2 in the optimal solution

Optimal revenue = **2B**
Assume Balance gives revenue **= 2B-x = B+y**
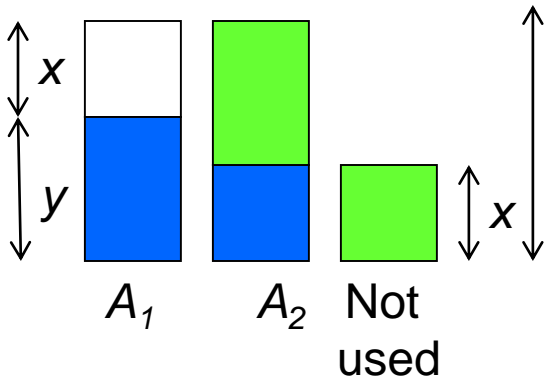
Unassigned queries should be assigned to A_2
(if we could assign to A_1 we would since we still have the budget)
Goal: Show we have **y ≥ x**

Case 1) ≤ ½ of A_1's queries got assigned to A_2
then $y \geq B/2$
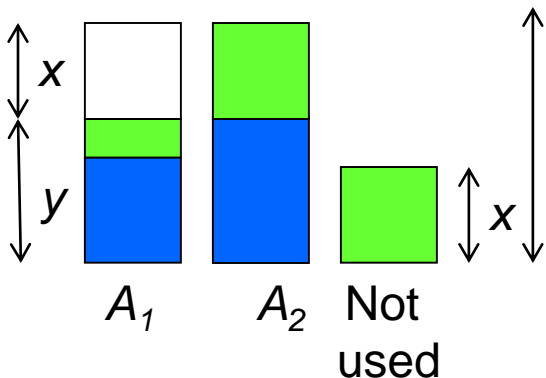
Case 2) > ½ of A_1's queries got assigned to A_2
then $x \leq B/2$ **and** $x + y = B$

Balance revenue is minimum for $x = y = B/2$
Minimum Balance revenue = $3B/2$
Competitive Ratio = 3/4

BALANCE exhausts A_2's budget

BALANCE: General Result

- **In the general case with *N* advertisers, worst competitive ratio of BALANCE is 1–1/e = approx. 0.63**

  - Interestingly, no online algorithm has a better competitive ratio!

Recommender Systems: Content-based Systems & Collaborative Filtering

Example: Recommender Systems
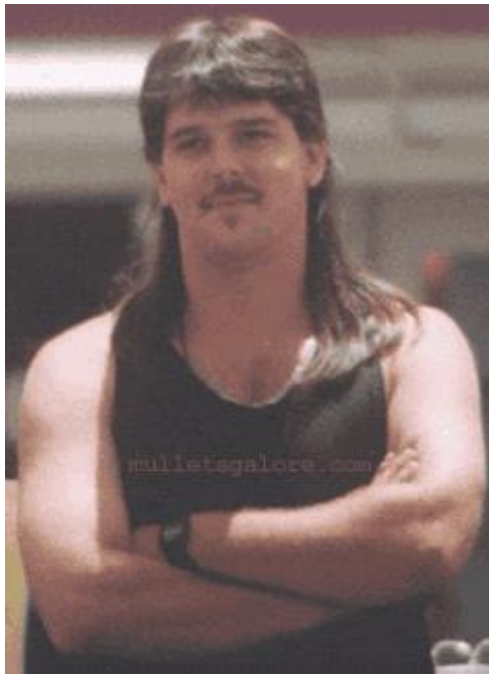
Customer X

- Buys Metallica CD
- Buys Megadeth CD

Customer Y

- Does search on Metallica
- Recommender system suggests Megadeth from data collected about customer **X**

Recommendations

Search **Recommendations**

Items

Products, web sites, blogs, news items, …

Examples:

amazon.com.

PANDORA

StumbleUpon

del.icio.us

NETFLIX
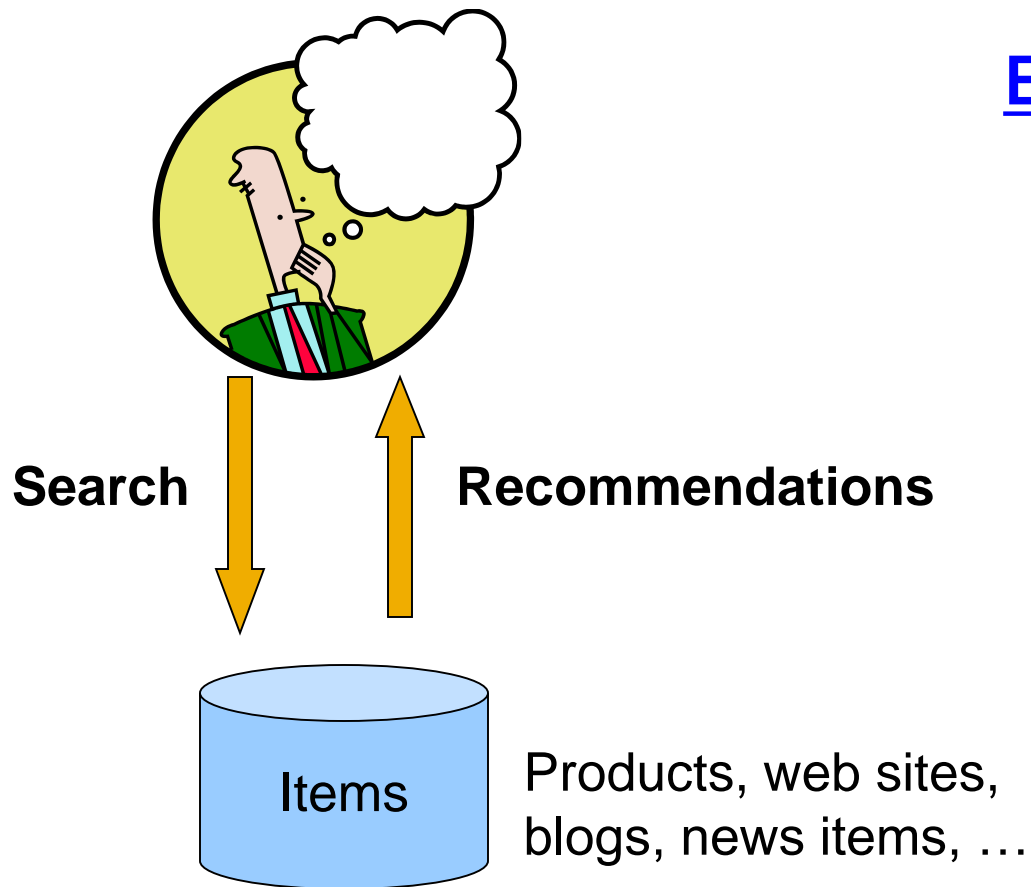
movielens
helping you find the *right* movies

last·fm
the social music revolution

Google News

You Tube

XBOX LIVE

From Scarcity to Abundance

- **Shelf space is a scarce commodity for traditional retailers**
  - Also: TV networks, movie theaters,…

- **Web enables near-zero-cost dissemination of information about products**
  - From scarcity to abundance

- **More choice necessitates better filters**
  - Recommendation engines
  - How **Into Thin Air** made **Touching the Void** a bestseller: http://www.wired.com/wired/archive/12.10/tail.html
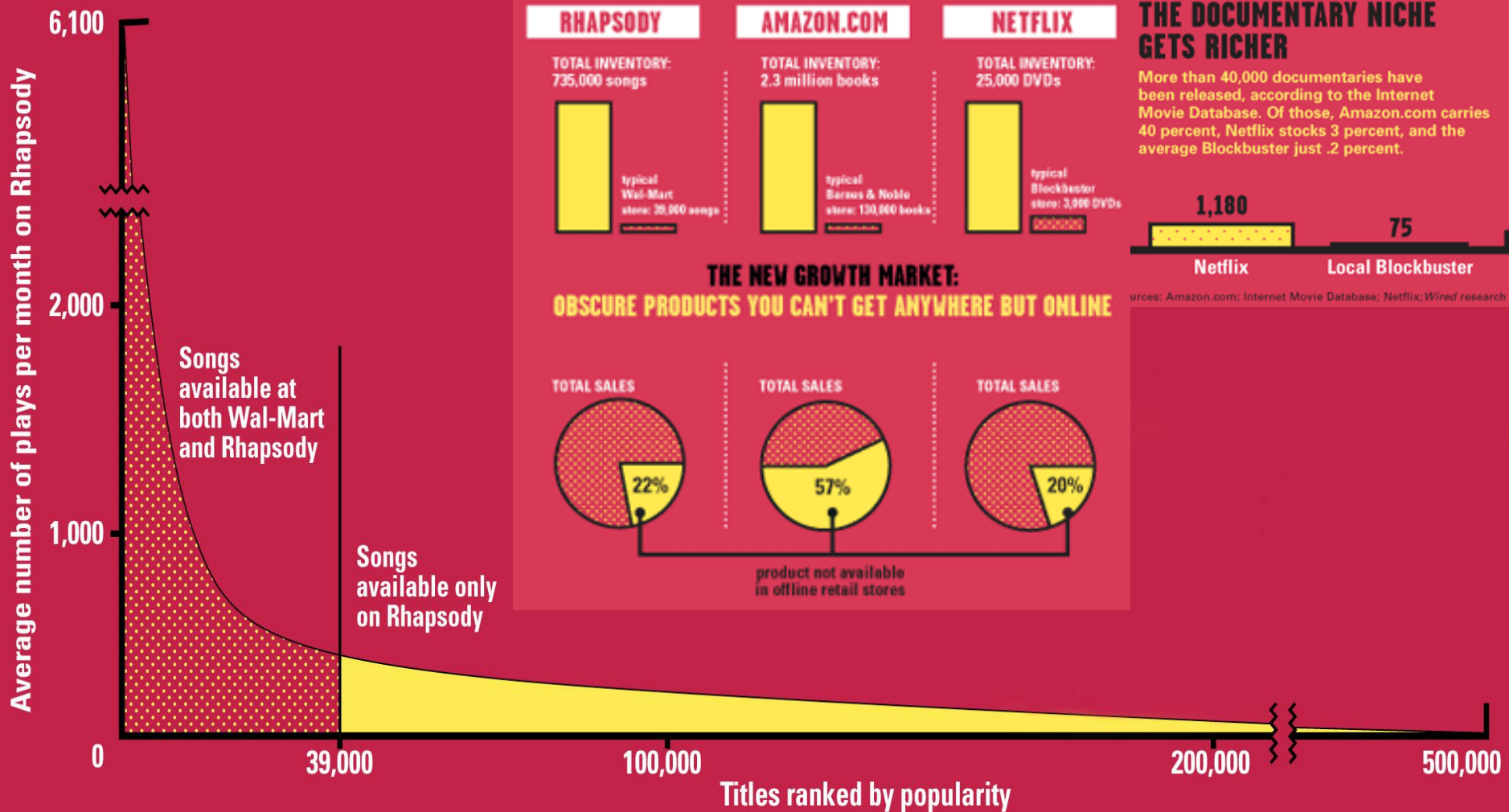
Sidenote: The Long Tail

Source: Chris Anderson (2004)

Types of Recommendations

- **Editorial and hand curated**
  - List of favorites
  - Lists of "essential" items

- **Simple aggregates**
  - Top 10, Most Popular, Recent Uploads

- **Tailored to individual users**
  - Amazon, Netflix, …

Formal Model

- *X* = set of **Customers**
- *S* = set of **Items**

- **Utility function** $u: X \times S \rightarrow R$

  - *R* = set of ratings

  - *R* is a totally ordered set

  - e.g., **0-5** stars, real number in **[0,1]**

Utility Matrix

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

Key Problems

- **(1) Gathering "known" ratings for matrix**
  - How to collect the data in the utility matrix

- **(2) Extrapolate unknown ratings from the known ones**
  - Mainly interested in high unknown ratings
    - We are not interested in knowing what you don't like but what you like

- **(3) Evaluating extrapolation methods**
  - How to measure success/performance of recommendation methods

(1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn't work well in practice – people can't be bothered

- **Implicit**
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?

(2) Extrapolating Utilities

- **Key problem:** Utility matrix *U* is **sparse**
  - Most people have not rated most items
  - **Cold start:**
    - New items have no ratings
    - New users have no history

- **Three approaches to recommender systems:**
  - **1)** Content-based
  - **2)** Collaborative
 - **3)** Latent factor based

Content-based Recommender Systems

Content-based Recommendations

- **Main idea:** Recommend items to customer *x* similar to previous items rated highly by *x*

Example:

- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, …
- **Websites, blogs, news**
  - Recommend other sites with "similar" content

Plan of Action

Item profiles

likes

recommend

build

match

Red
Circles
Triangles
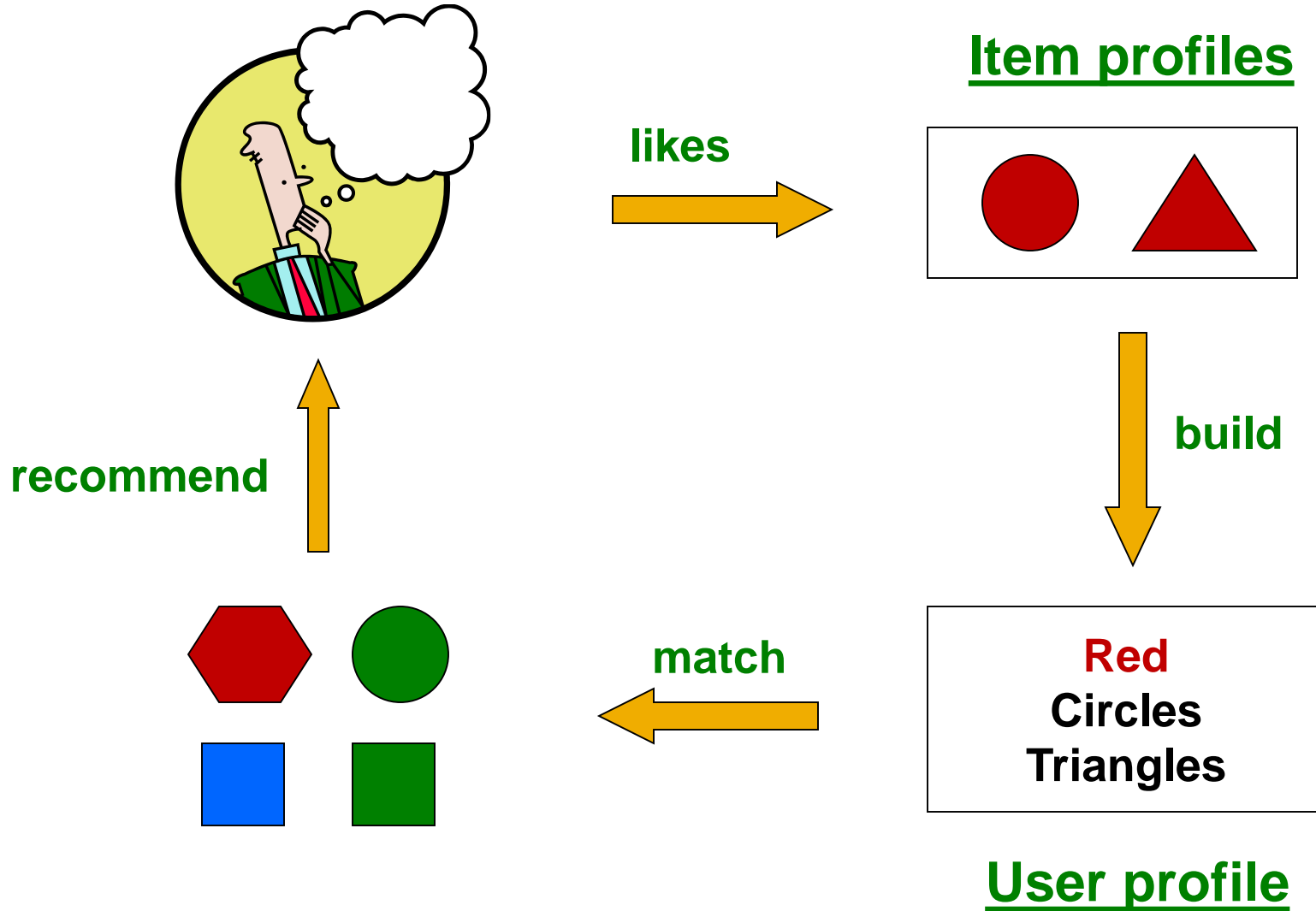
User profile

User Profiles and Prediction

- **User profile possibilities:**

  - Weighted average of rated item profiles

  - **Variation:** weight by difference from average rating for item

  - ...

- **Prediction heuristic:**

  - Given user profile **x** and item profile **i**, estimate

$$u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$$

Collaborative Filtering

Harnessing quality judgments of other users

Collaborative Filtering

- Consider user *x*

- Find set *N* of other users whose ratings are "**similar**" to *x*'s ratings

- Estimate *x*'s ratings based on ratings of users in *N*