YarcData™
Getting to **Eureka!** faster™

YarcData

Ramesh Menon

VP Solutions

YarcData

May 7, 2014

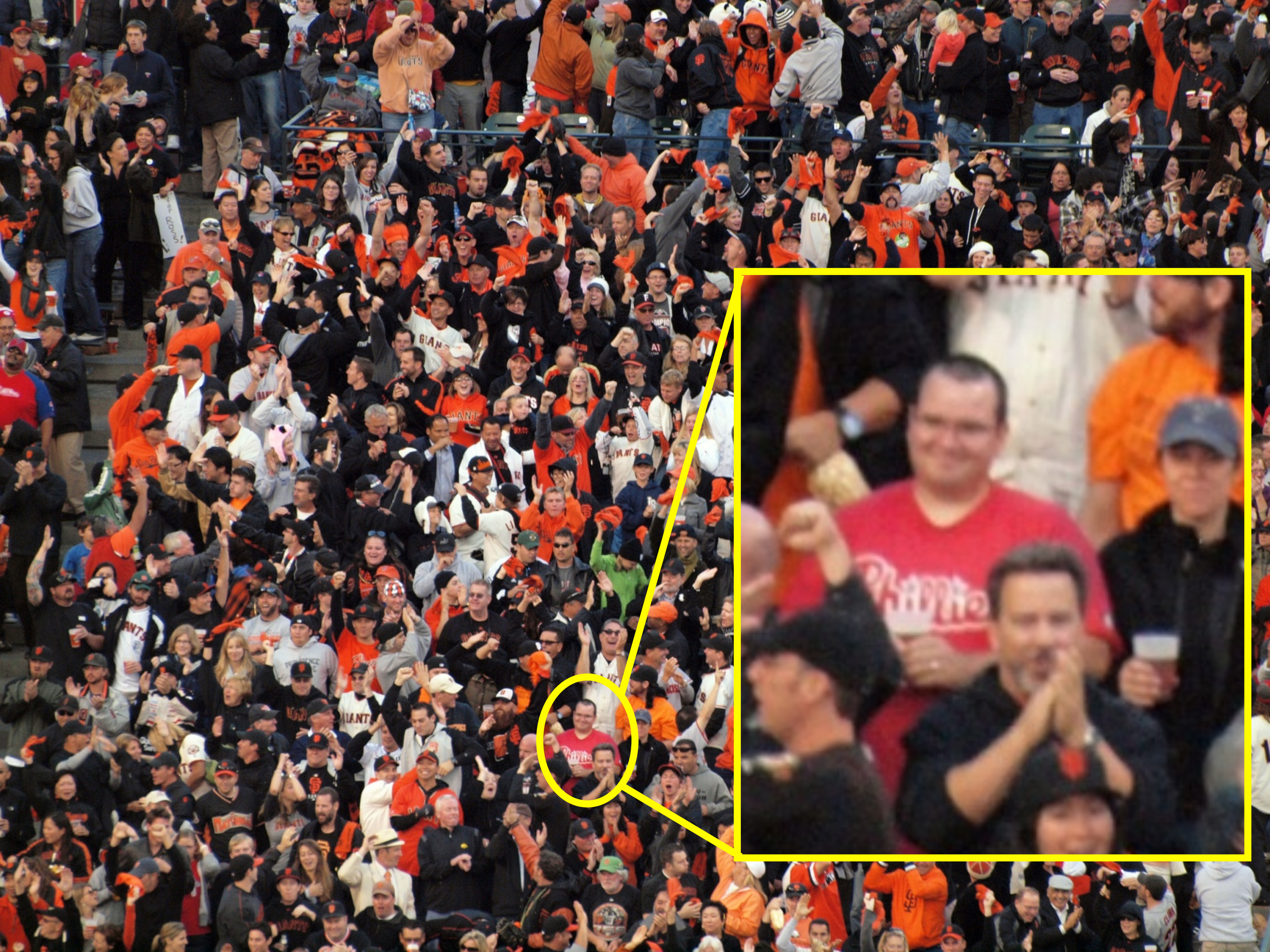
Search Problems…

- How many fans are there?
- What demographics?
- What apparel? By color/style/size?
- What concessions?

…

Maybe even…

- Fan density issues for safety, ticketing?
- Based on lighting and face direction, where in the park was this shot taken?

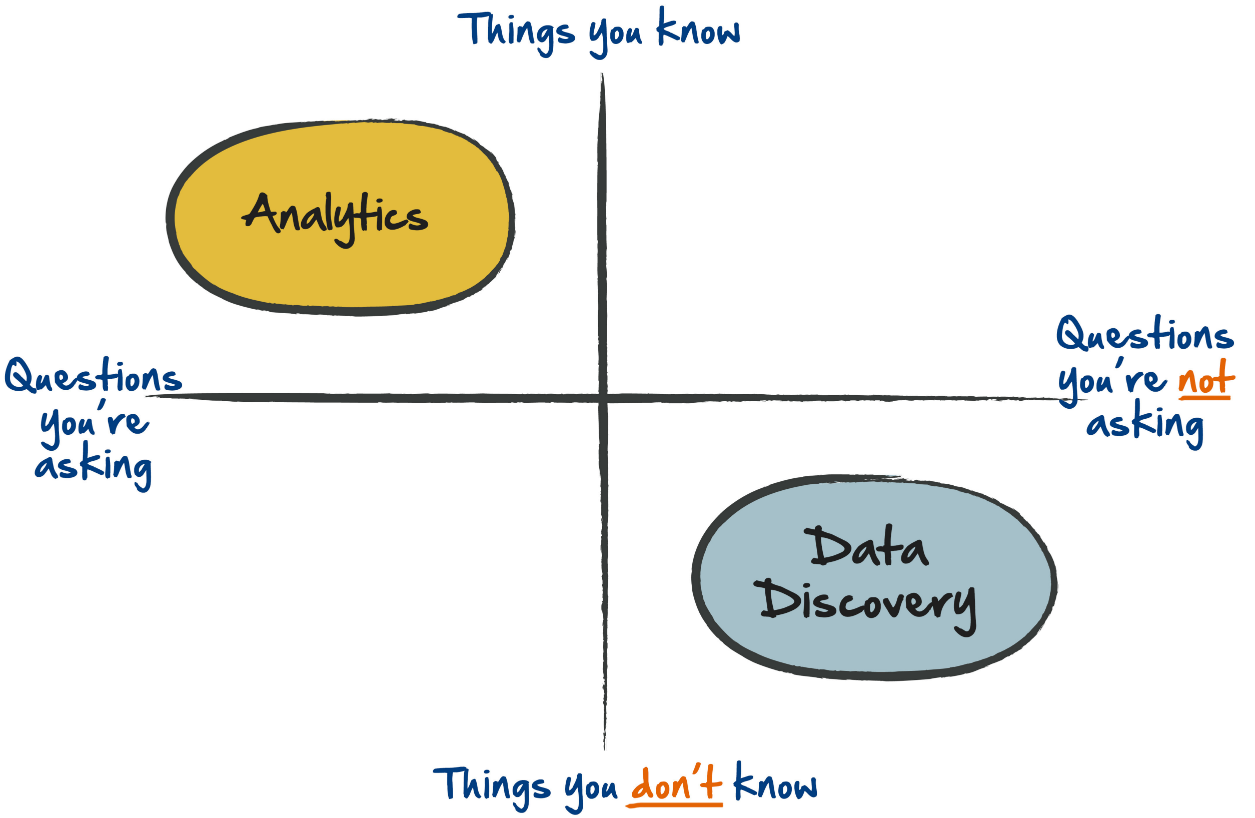

GIANTS
Phillies

Search: Scalably and Efficiently analyzing all sorts of metrics on lots of Giants fans you expect to find

Discovery: What is the implication of the one Phillies fan you didn’t expect to find?

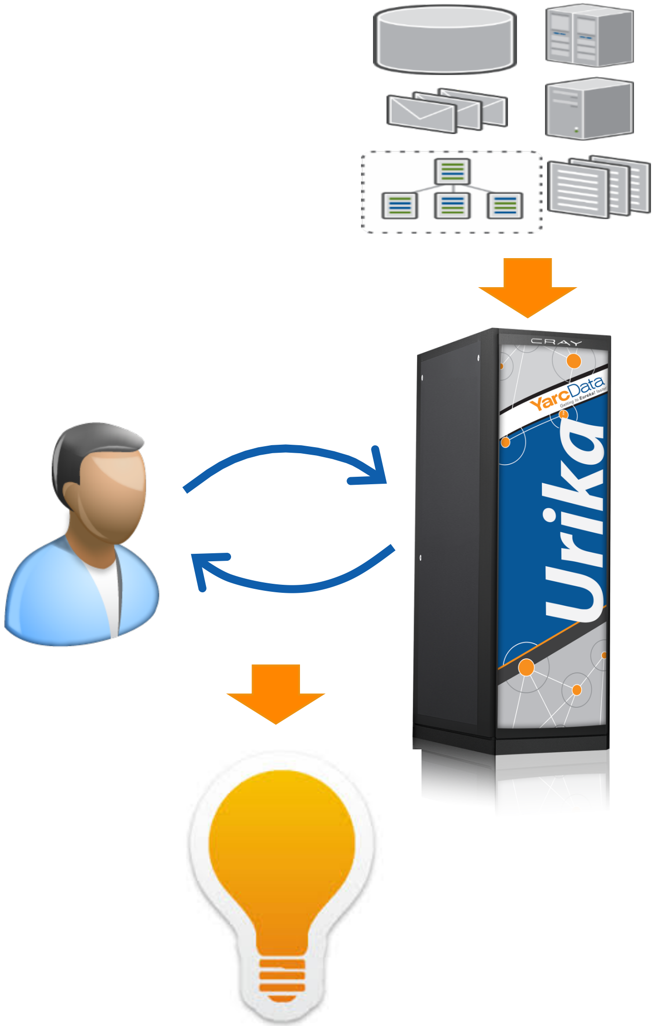
Data Discovery: The Real Promise of Big Data…

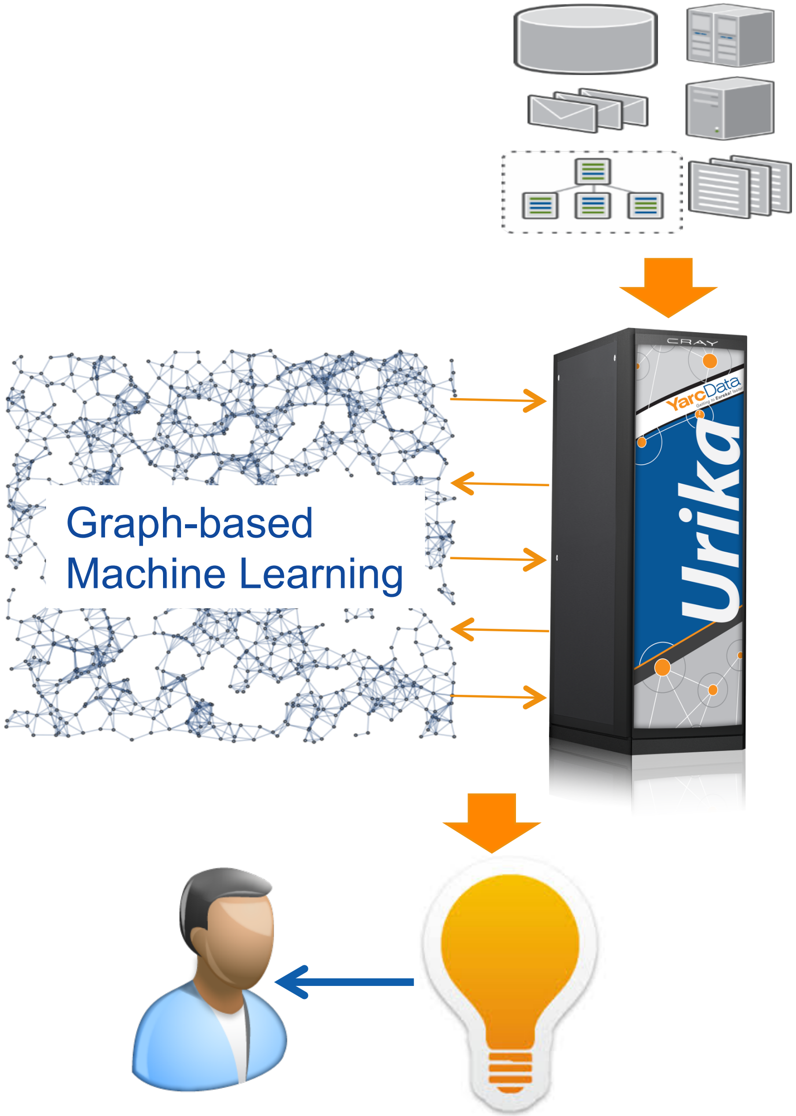
"Take all these different data sources and put them together and then help me find something about the data that I don't already know…"



YarcData Getting to Eureka! faster

Common Approach to Discovery using Urika

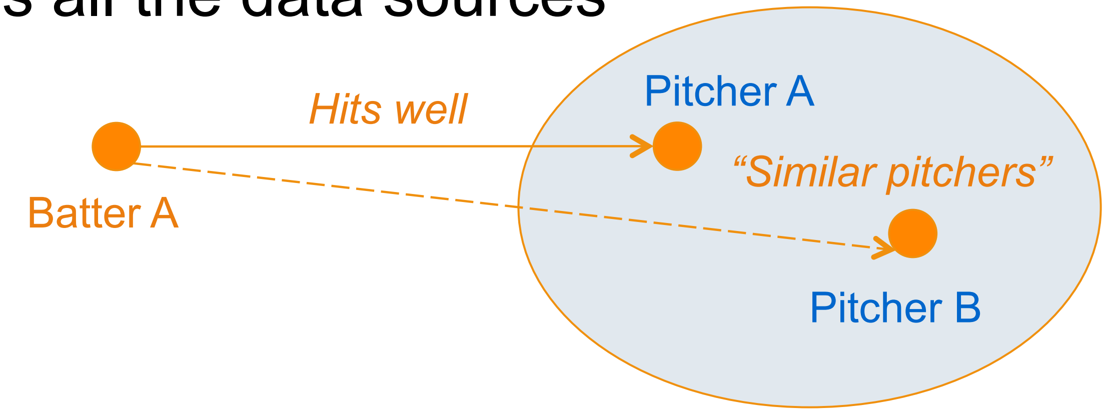




Graph-based Machine Learning

Customer Use-Case

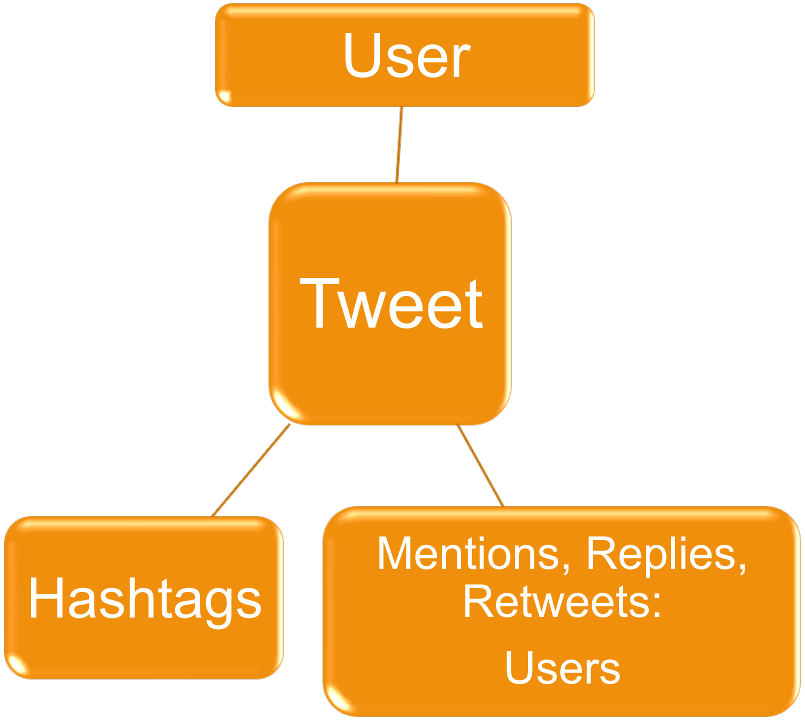
- The Objective
  - How to evaluate player performance in a given situation?
  - Batter against a particular pitcher, stage of game, location of game, weather…
- The Conundrum: Sample size of data for each variable is ***too small*** to make meaningful conclusions (1:1 data, new batter-pitcher combos, full career)
- Urika approach: ***Predict*** performance by discovering patterns across all the data sources



Benefits: Development patterns, Minor league projections, Injury detection

Another Customer Use-case - Social Media Analytics

- Who are the “influential” tweeters on topics related to certain hi-tech products and services



Who follows who?

- Twitter data feed doesn’t provide that information
- So we have to figure it out

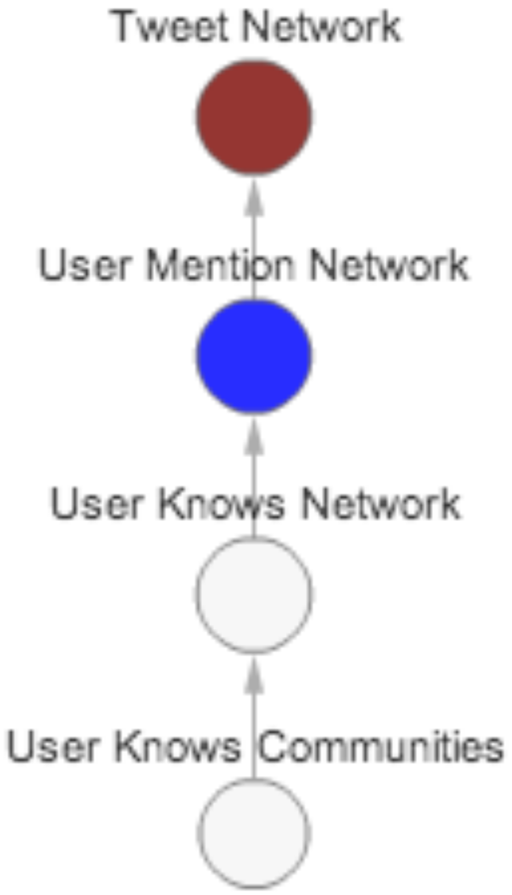
Overview of Process

- From the Twitter data

  - Construct a network of Users

  - Construct a more distinct network of Users

  - Then run an algorithm to detect communities



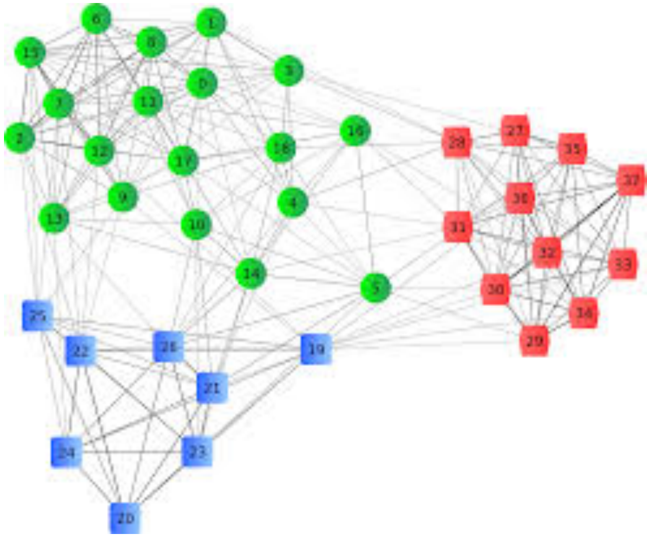
YarcData
Getting to *Eureka!* faster

What do these Discovery examples have in common?

- Clustering – Classification of data into groups based on similarity

- Community Detection is computationally challenging
  - But ideally suited for Urika

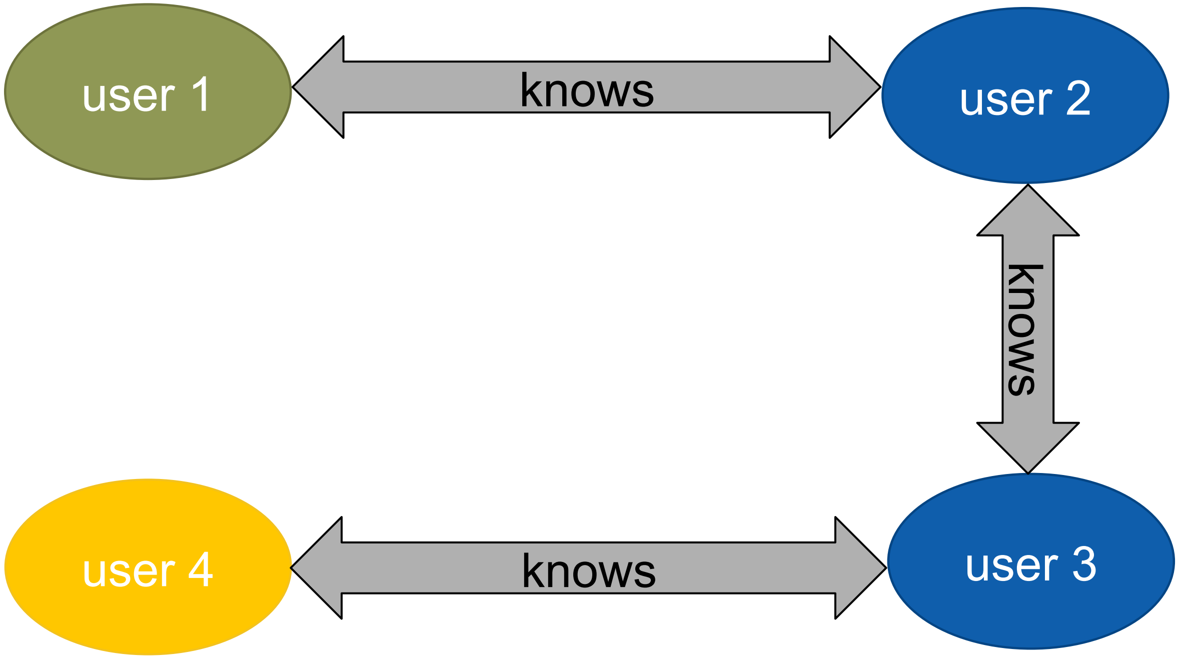
- Specific implementation
  - Label Propagation



YarcData Getting to Eureka! faster

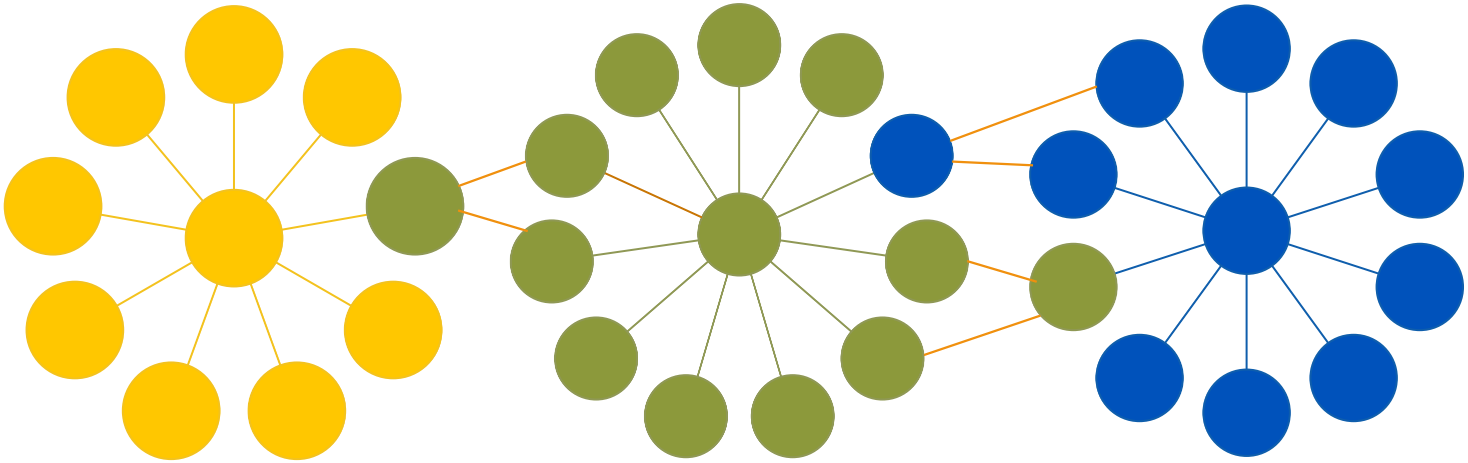
Label Progation: Iteration

- For each user, count the most popular community of its neighbors, and assign it to that one
  - After multiple iterations, users will join into bigger communities
  - Eventually, they converge at a stable membership, where users settle on the most popular



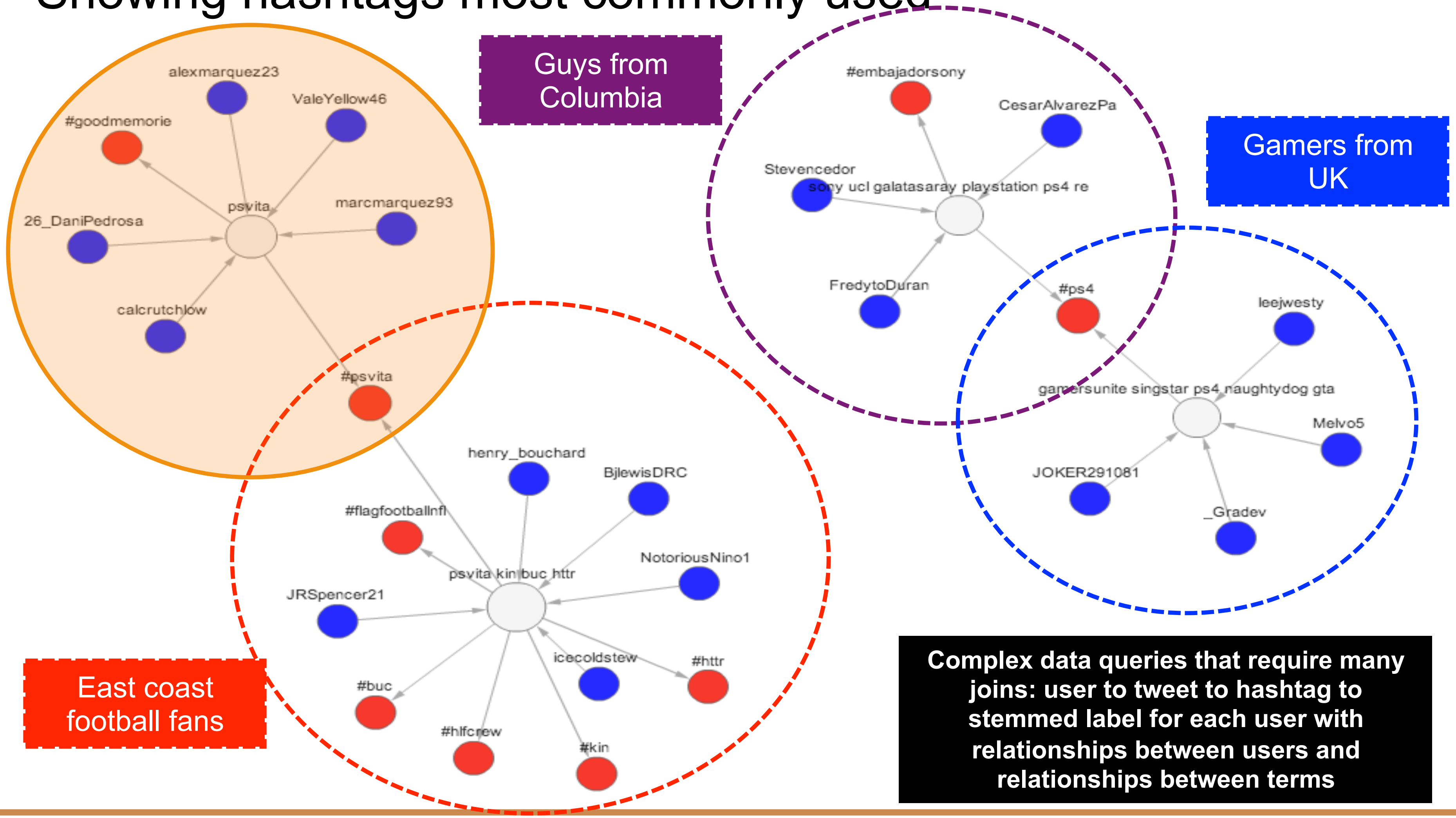
Label Progation: Iteration - Size

- Each iteration is a very large calculation
  - For each user, say 100 million
  - Find most popular community among neighbors
    - average 10 neighbors (users they know)
    - find neighbors, group by community, count, max, and tie break
  - Iteration will run multiple times before convergence



Example: User-Knows-User Communities (Bipartite)

- Showing hashtags most commonly used



Guys from Columbia

Gamers from UK

East coast football fans

Complex data queries that require many joins: user to tweet to hashtag to stemmed label for each user with relationships between users and relationships between terms

Getting to *Eureka!* faster

User Roles within Community

- User role based on retweet activity
  - Indicated by color scale

🟩 Green: rebroadcaster

🟩 retweets others

🟨 Yellow: even

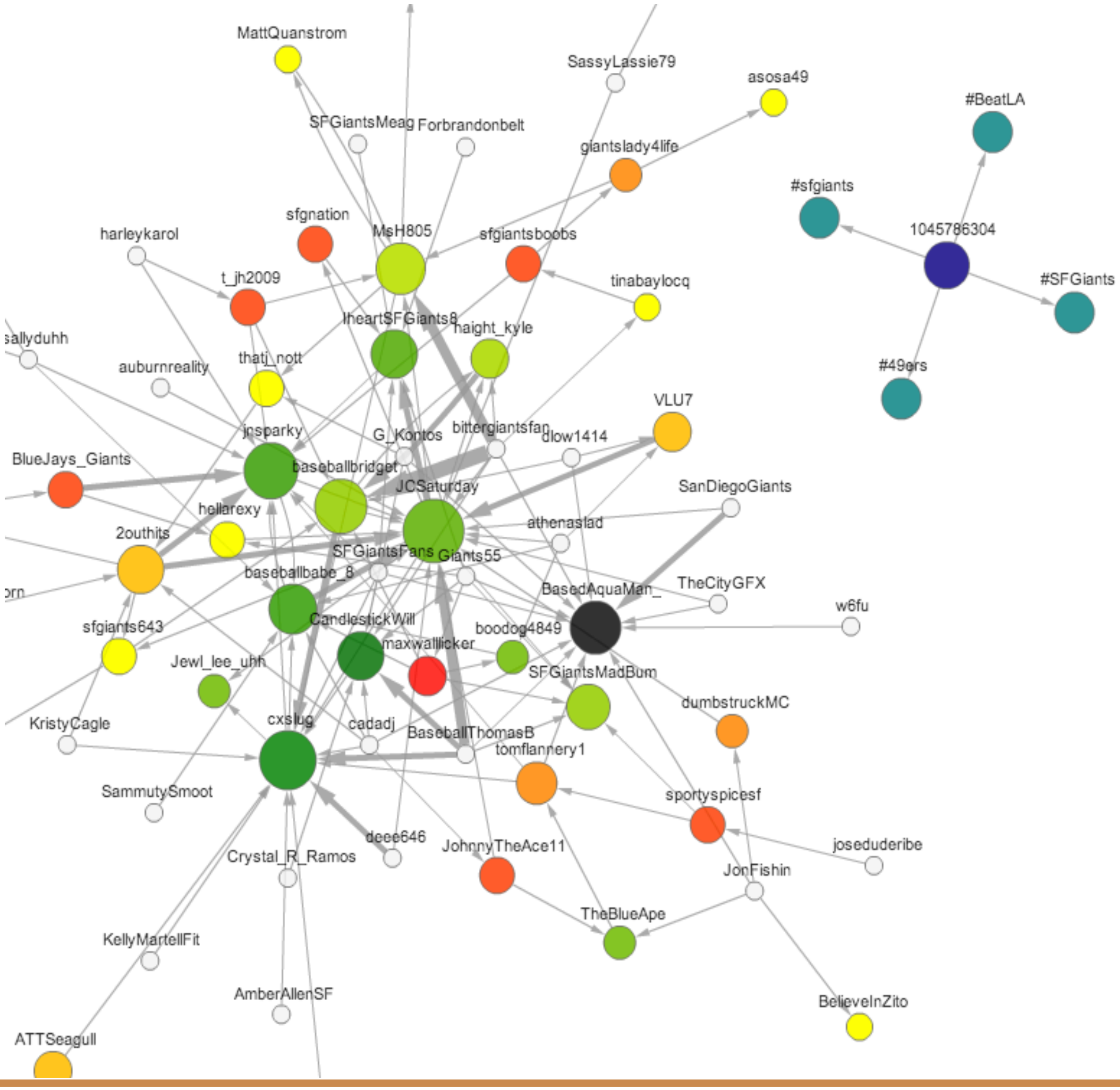
🟧

🟥 Red: source

Is retweeted by others

○ White: little RT activity

- Arrows
  - Direction of RT flow, read “retweeted by”
  - Size: amount of RT



Label Propagation: implementation

- Implemented as Sparql queries and inserts
  - All heavy lifting is executed in Urika as standard Sparql queries
  - The queries, steps and iteration are driven by a client-side program (python)

- *Extract Features*
- *Create Similarity Network*
- *Create Clusters Based on Network*
- Review and Evaluate Clusters
  - Automated/Objective
  - SME Validation
  - Applicability to Business Case Problem
- Extract additional features if needed
- Create new similarity networks
- *Make output consumable to end user*
- Convert to production

SPARQL enables "wild-card joins", easy to write queries

Graphs are best to do relationship networks

RDF easily supports multiple predicate types

- Reduces cost of joins since this is implicit
- Ontologies make it easy to build links
- Computation cost for the above is very high if you don't have massive shared memory and massive multi-threading

YarcData Getting to **Eureka!** faster

CD Applications

- Customer Analytics
  - Finding similar customers
  - Financial Services, Telecom, Media, Retail etc.
- Defense/Security
  - Uncover groups of actors
  - Ties between actors (better targeting)
  - Organization structure (roles within communities)
- Recommender Systems
  - Collaborative Filtering:
    - People buy items that people like them, people related to them, people that influence them buy
  - Content Based Filtering:
    - People buy things that are similar to objects ones that they've purchased
- Healthsciences
  - Patient Cohort Identification

YarcData
Getting to **Eureka!** faster™

Thank You!