facebook

facebook

People you may know

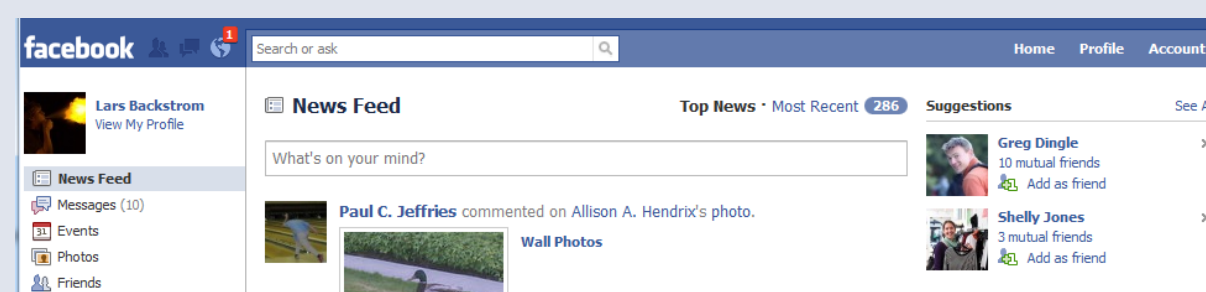
Lars Backstrom
07/12/2010

Agenda

1. Who to suggest?
2. Static, offline predictions
3. Dynamic, online reranking
4. Performance/Wrap-Up

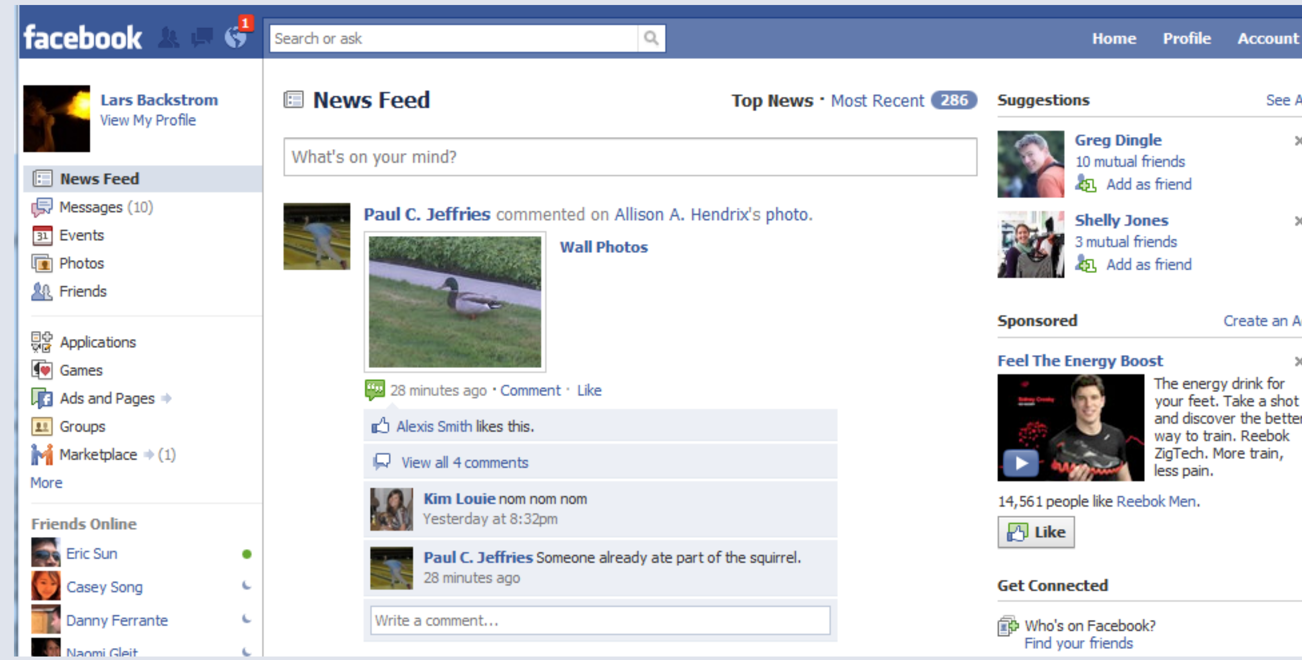
Helping people find friends on FB

- Recommendation has proven itself in many contexts
  - Amazon, NetFlix, etc. all have sophisticated systems
- Like them, we can increase value to users by making good suggestions
  - People with more friends use the site more, get more out of it
- Unlike those systems (collaborative filtering) our's must take social context into account



People you may know

- Top 1-2 suggestions shown on homepage of Facebook
  - See all link leads to more suggestions
  - Many more friend adds from home than 'see all' page.
- 'Xing' a user removes that person from list permanently
  - Pulls in next suggestion
- Accounts for a significant chunk of all friending on Facebook

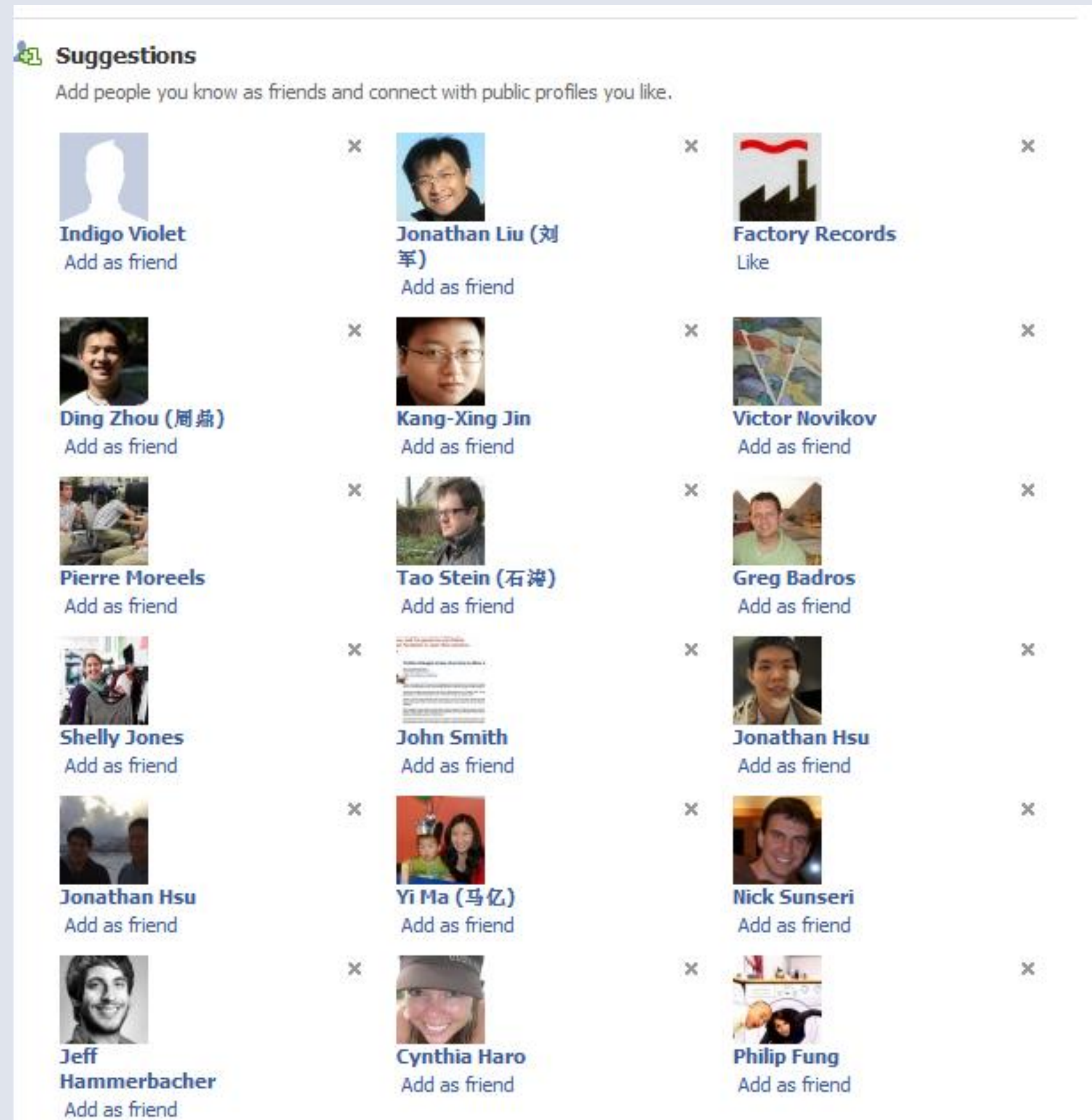


People you may know

- Top 1-2 suggestions shown on homepage of Facebook
  - See all link leads to more suggestions
  - Many more friend adds from home than 'see all' page.
- 'Xing' a user removes that person from list permanently
  - Pulls in next suggestion
- Accounts for a significant chunk of all friending on Facebook

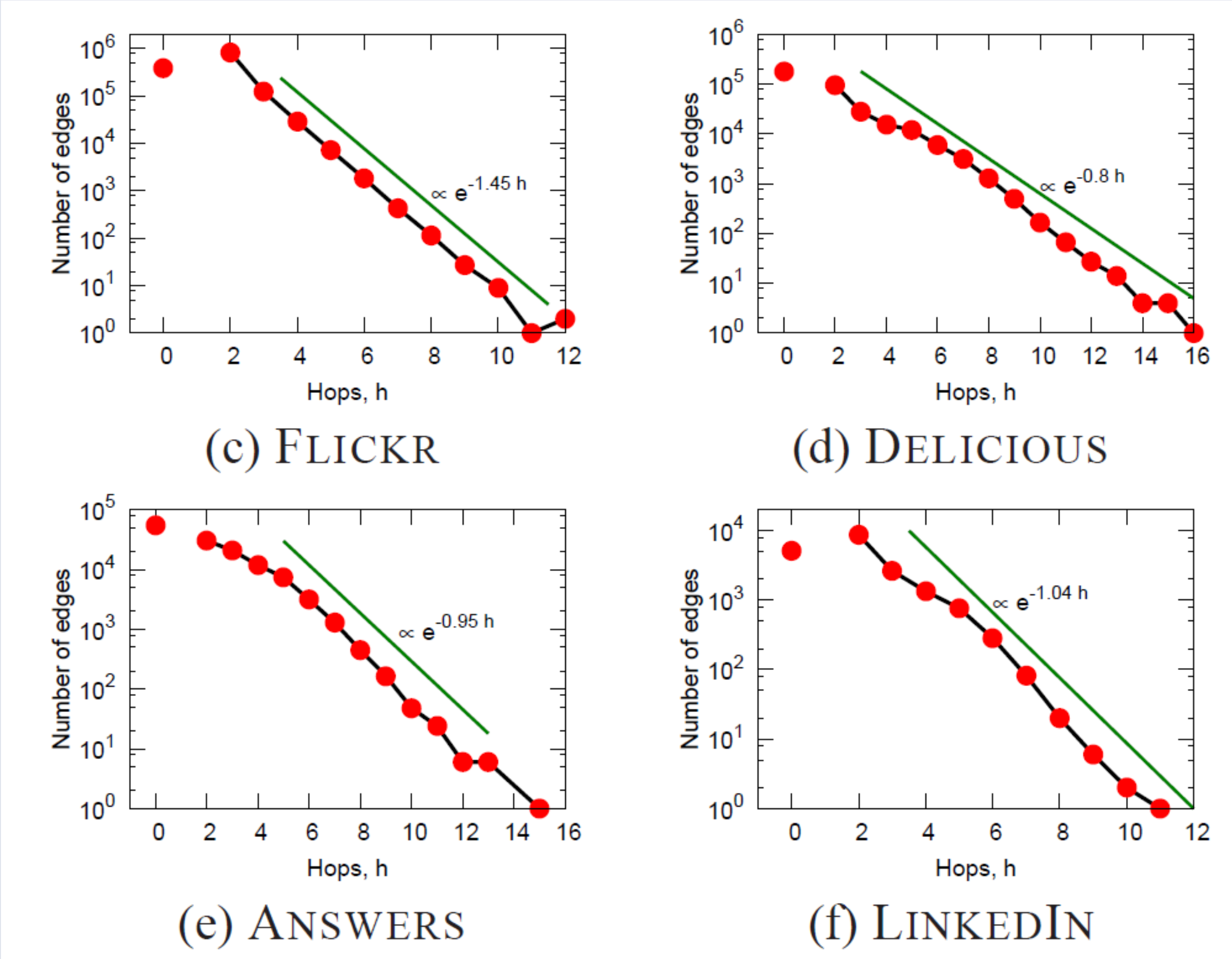
People you may know

- Top 1-2 suggestions shown on homepage of Facebook
  - See all link leads to more suggestions
  - Many more friend adds from home than 'see all' page.
- 'Xing' a user removes that person from list permanently
  - Pulls in next suggestion
- Accounts for a significant chunk of all friending on Facebook



How to make suggestions

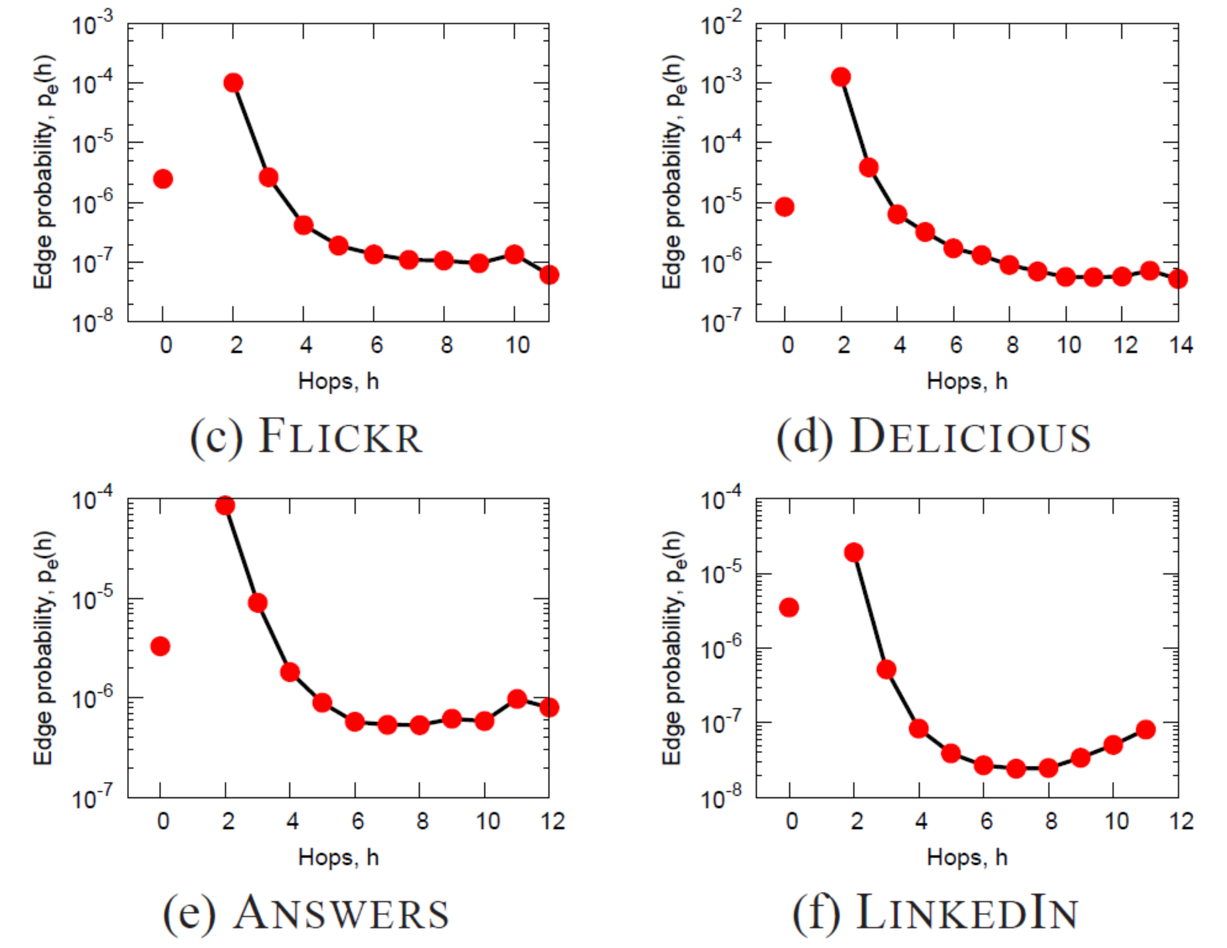
- Most friendships go to friends-of-friends
  - Previous work shows over 5x more friendships to FoFs (2-hops) than 3+ hop users (Lescovec et. al '08)
  - 92% of new friendships on FB
- From a practical point of view, doing more than FoF is impossible
  - Average user has over 130 friends
    - 130*130 = 17K FoFs
    - 130^3 = 2.2M FoFoFs
  - Power users have up to 5K friends



(c) FLICKR (d) DELICIOUS (e) ANSWERS (f) LINKEDIN

How to make suggestions

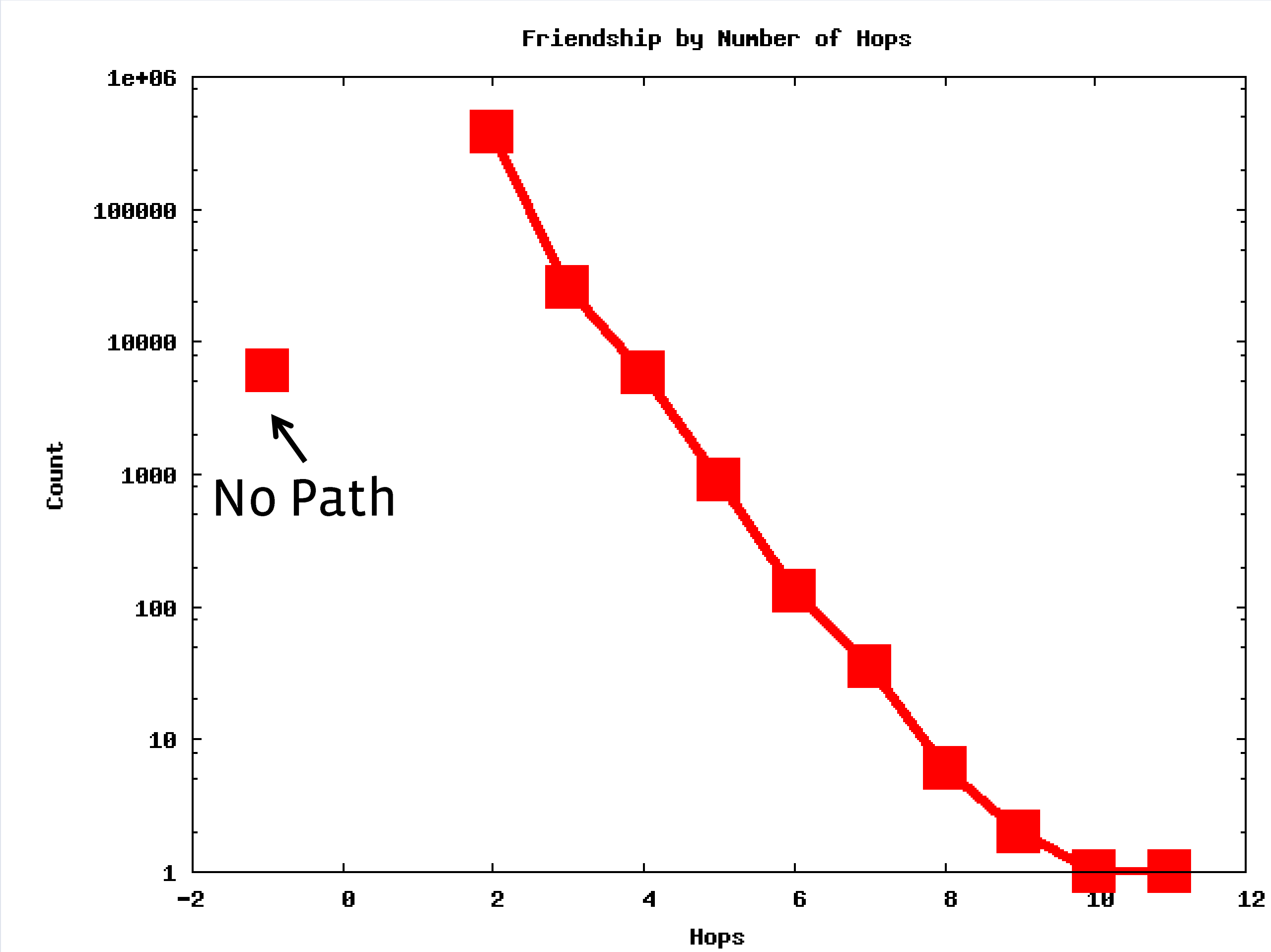
- Most friendships go to friends-of-friends
  - Previous work shows over 5x more friendships to FoFs (2-hops) than 3+ hop users (Lescovec et. al '08)
  - 92% of new friendships on FB
- From a practical point of view, doing more than FoF is impossible
  - Average user has over 130 friends
    - 130*130 = 17K FoFs
    - 130^3 = 2.2M FoFoFs
  - Power users have up to 5K friends



(c) FLICKR
(d) DELICIOUS
(e) ANSWERS
(f) LINKEDIN

How to make suggestions

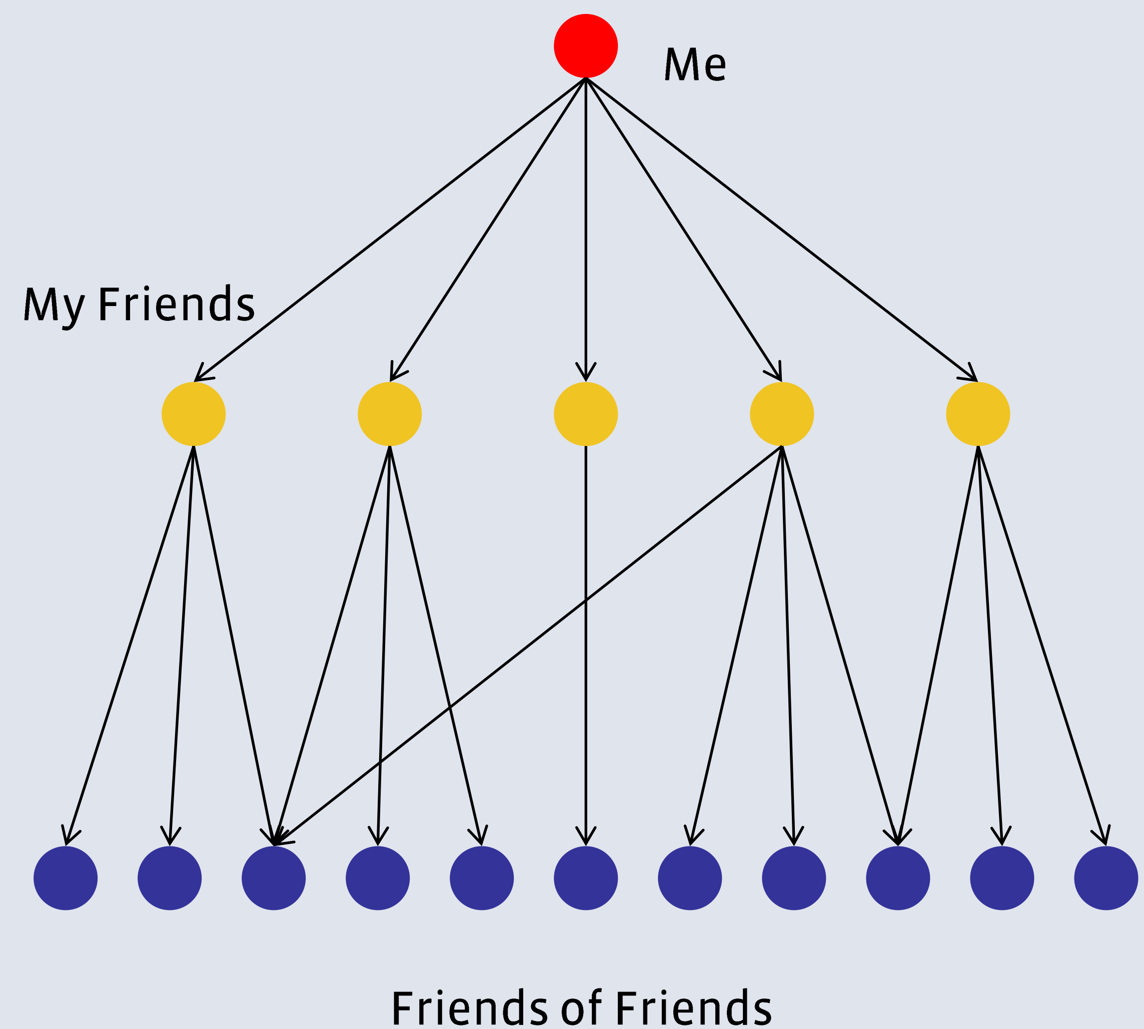
- Most friendships go to friends-of-friends
  - Previous work shows over 5x more friendships to FoFs (2-hops) than 3+ hop users (Lescovec et. al '08)
  - 92% of new friendships on FB
- From a practical point of view, doing more than FoF is impossible
  - Average user has over 130 friends
    - 130*130 = 17K FoFs
    - 130^3 = 2.2M FoFoFs
  - Power users have up to 5K friends



Facebook

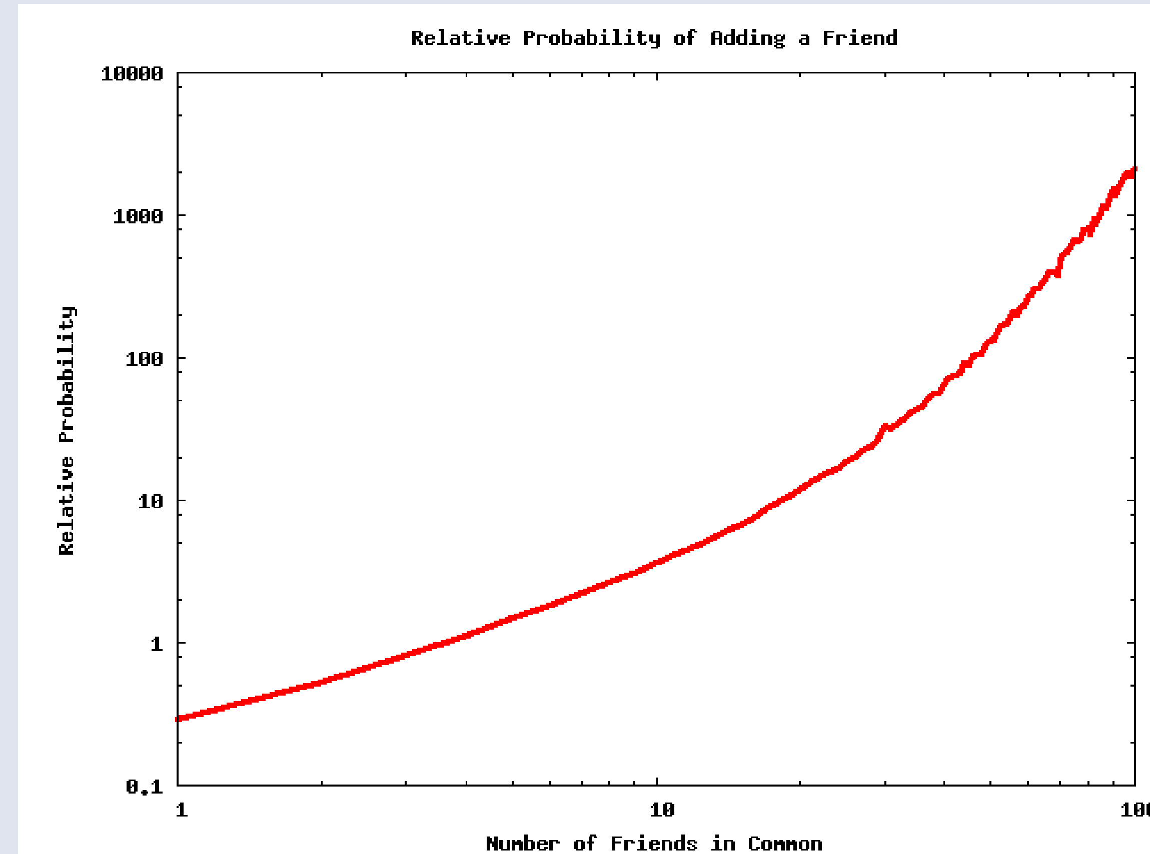
Suggesting Friends of Friends

- Problem Statement:
  - Given a source user, find the best FoFs to suggest
- Challenges:
  - A typical user has tens of thousands of FoFs (about 40K on average, 99th percentile 800K!)
    - What features will help us pick from these
    - How can we combine network and demographic features



Friends in Common

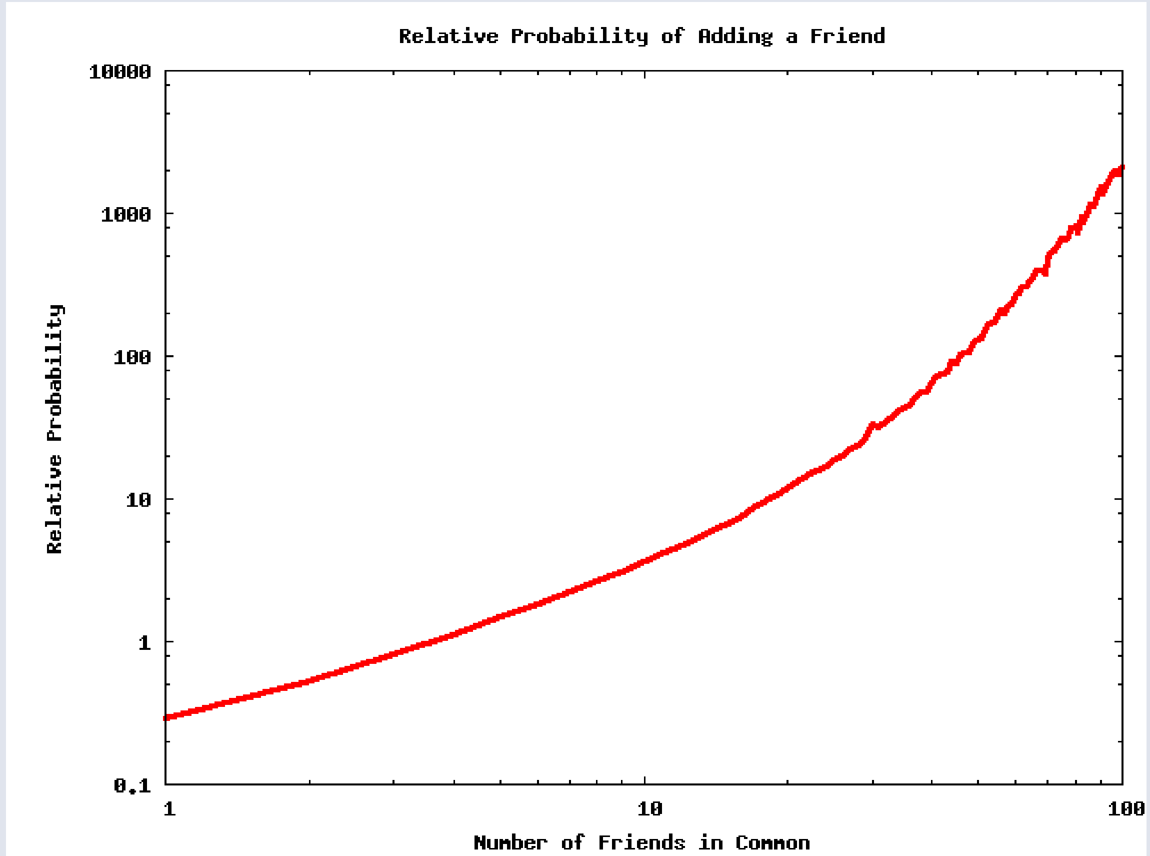
- Number of friends in common is a good start
  - Two people are 12x more likely to become friends with 10 mutual friends than 1
- Other social network features are also helpful
  - For example, if your good friend just made a new friend, that is a good suggestion

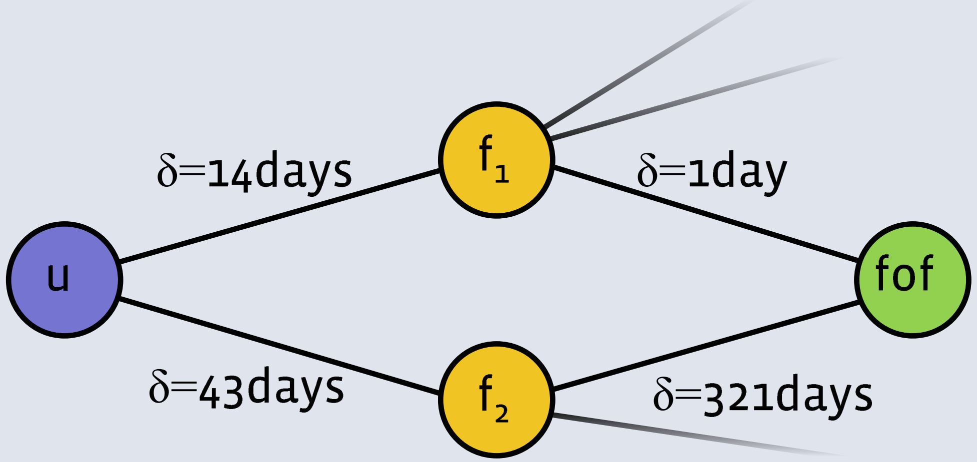


Friends in Common

- Number of friends in common is a good start
  - Two people are 12x more likely to become friends with 10 mutual friends than 1
- Other social network features are also helpful
  - For example, if your good friend just made a new friend, that is a good suggestion
- We can combine network properties:
  - $\delta_{u,v}$ gives the time since edge creation

$$v(fof) = \sum_{f_i} \frac{(\delta_{u,f_i} \cdot \delta_{f_i,fof})^{-0.3}}{\sqrt{friends_{f_i}}}$$

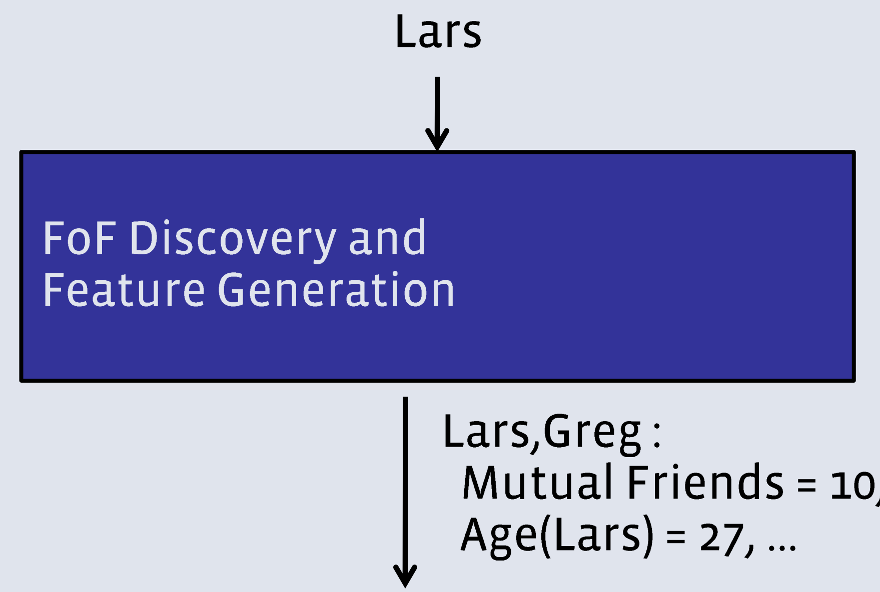




System Overview

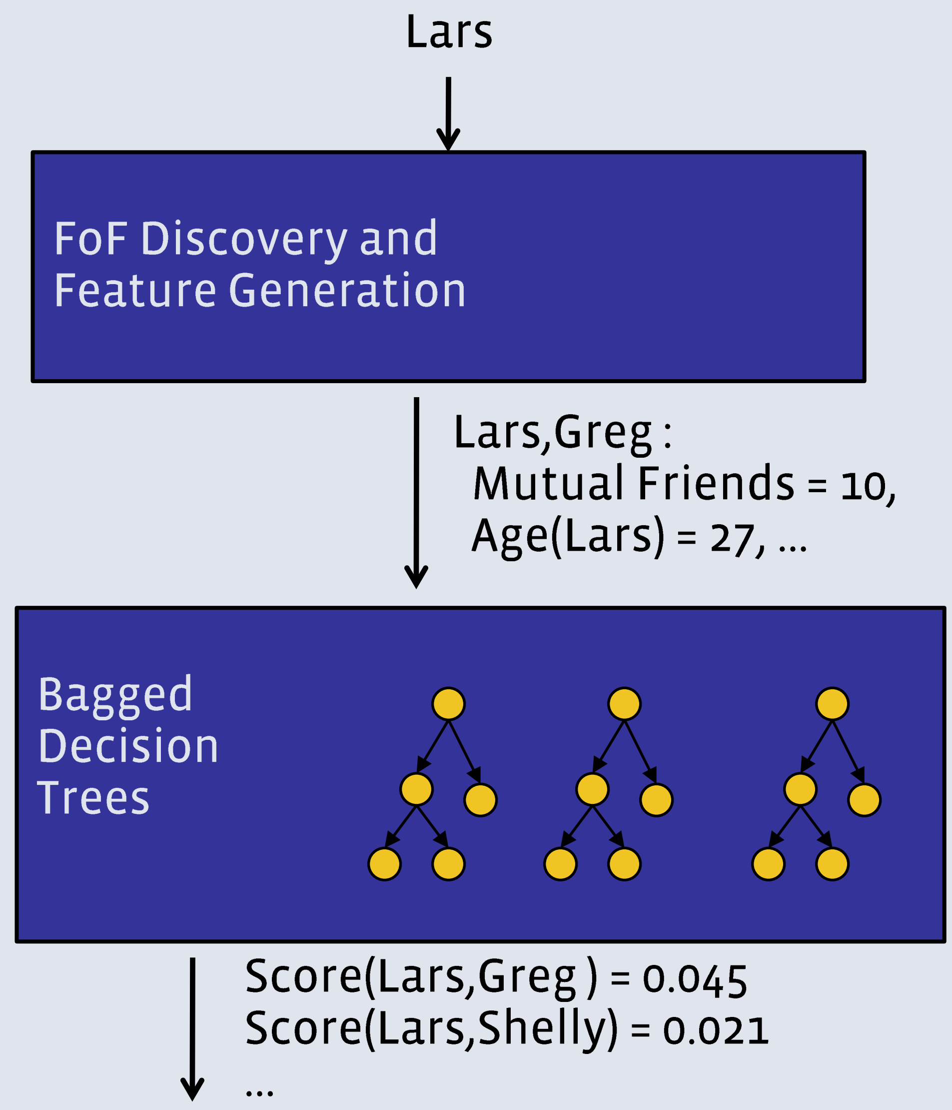
System Overview

- System examines all FoFs



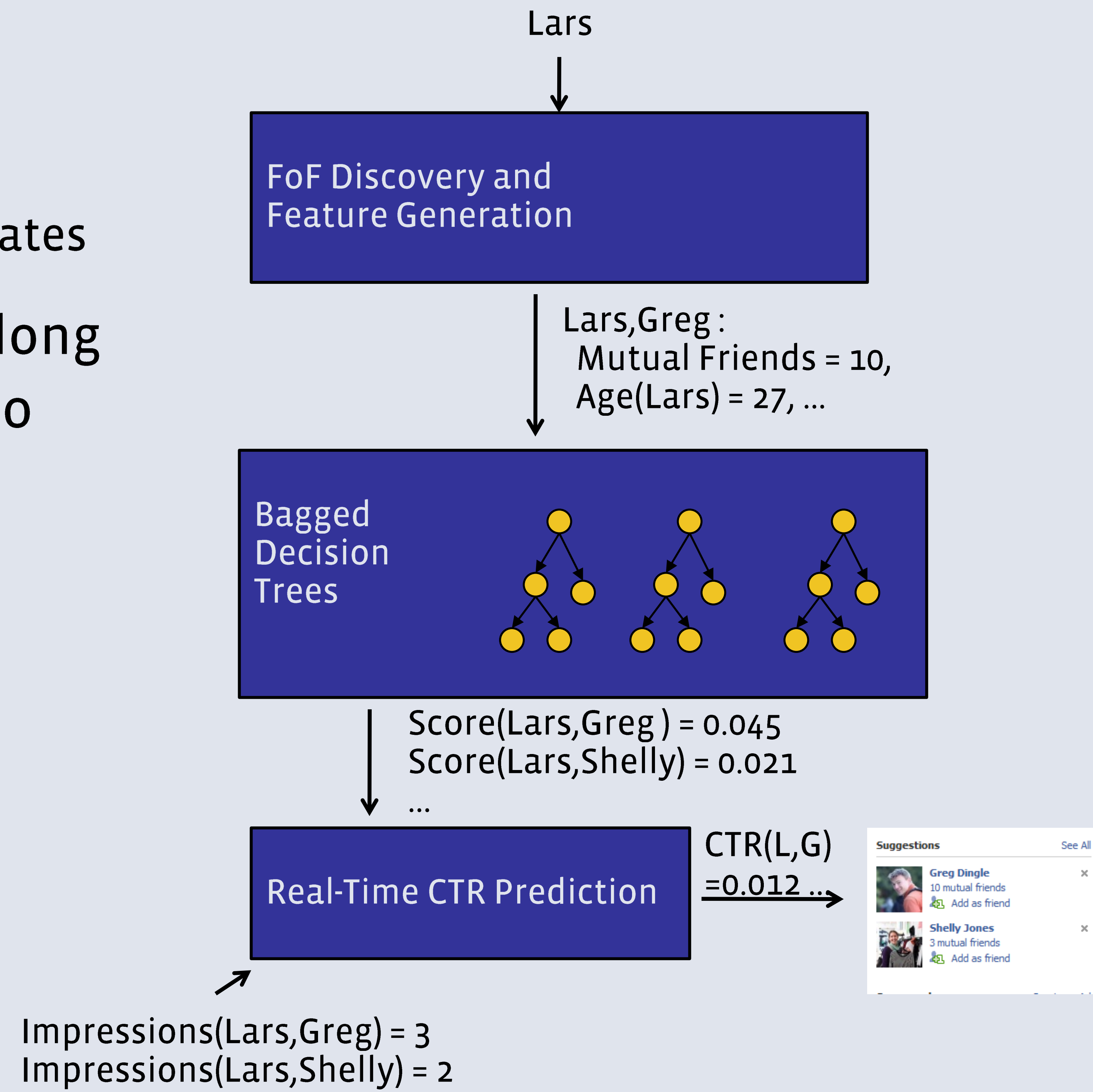
System Overview

- System examines all FoFs
  - Generates list of top 100 candidates



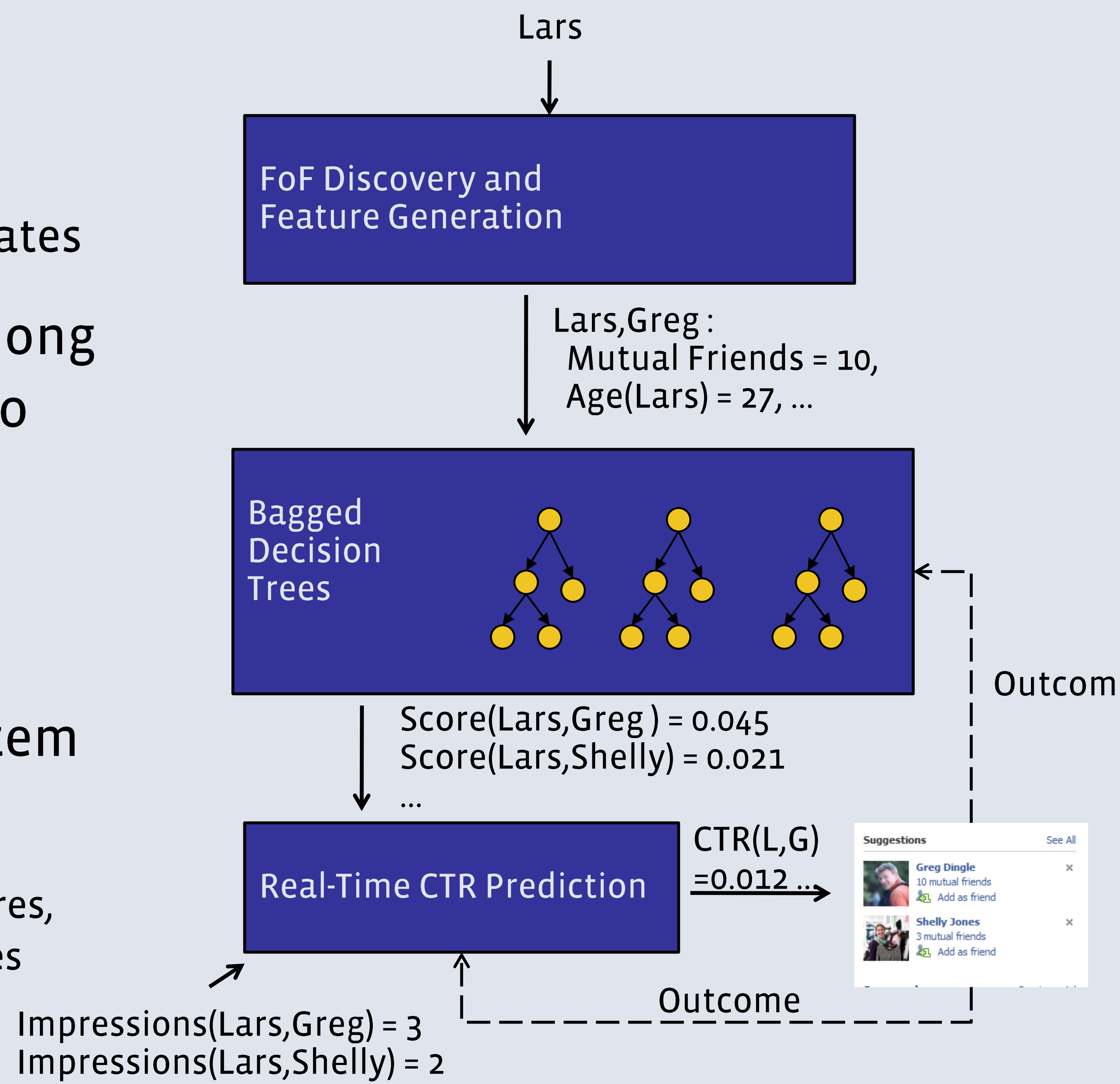
System Overview

- System examines all FoFs
  - Generates list of top 100 candidates
- Scores are stored and used along with cheaply available data to predict real-time CTRs
  - Candidates are re-ranked and displayed on each impression



System Overview

- System examines all FoFs
  - Generates list of top 100 candidates
- Scores are stored and used along with cheaply available data to predict real-time CTRs
  - Candidates are re-ranked and displayed on each impression
- Results are fed back into system for retraining
  - Real-time model depends on input scores, must be retrained when offline changes

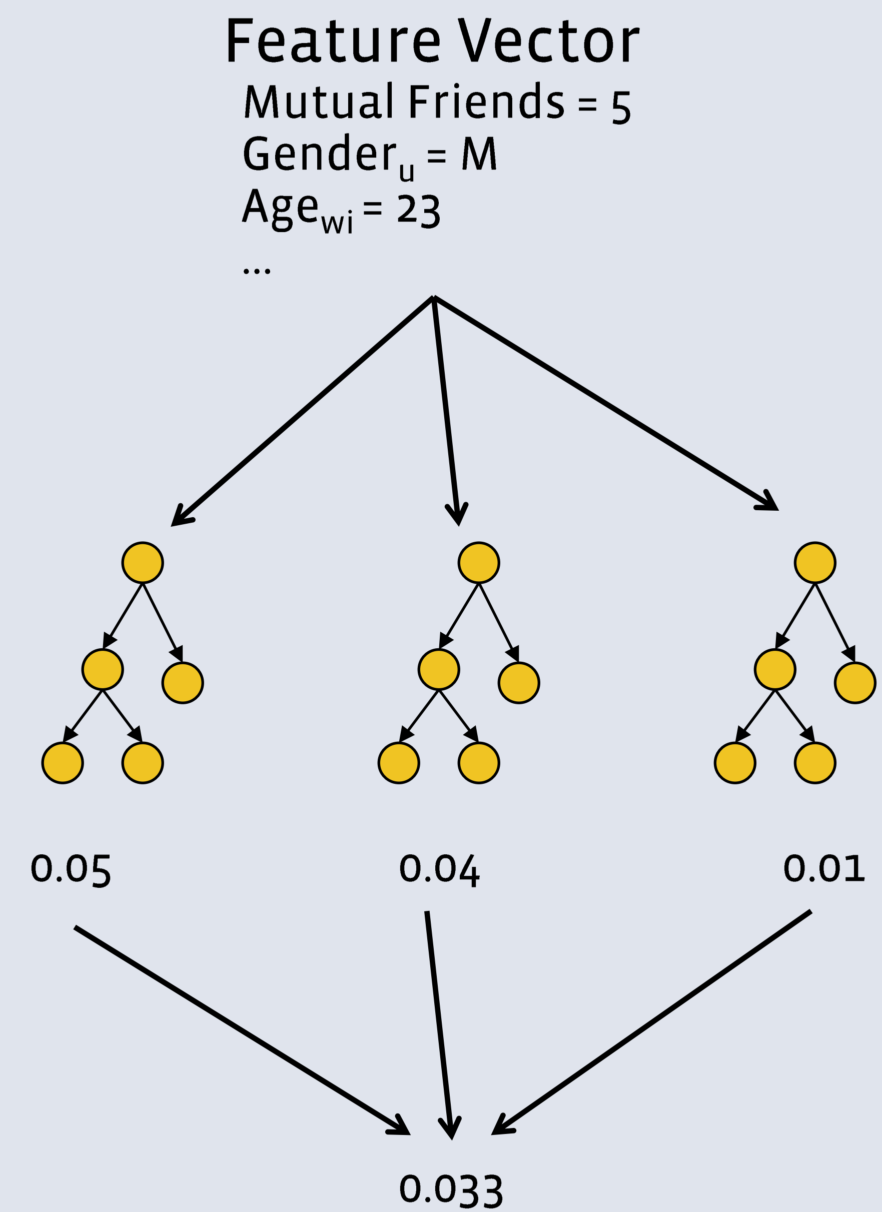


Agenda

1. Who to suggest?
2. Static, offline predictions
3. Dynamic, online reranking
4. Performance/Wrap-Up

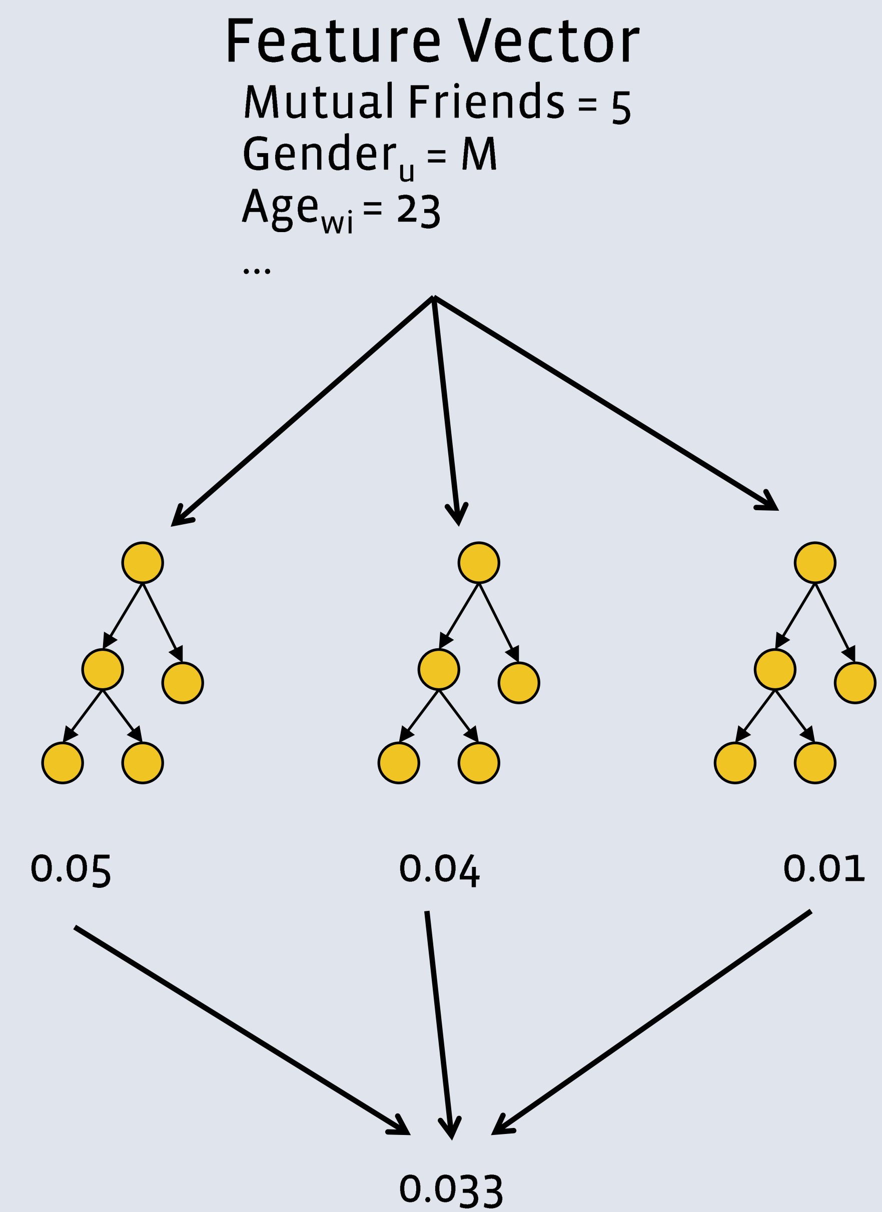
Making Static Predictions

- Use traditional machine learning
  - For a user u, consider all FoFs $w_1, \ldots, w_k$
  - For each pair (u, w_i) generate a bunch of features
    - Mutual friends, time discounted mutual friends, new mutual friends, etc.
    - Also incorporate features of just u and w_i
      - Age, gender, country, total friends, time on FB, etc.
  - We use bagged decision trees (the average of many decision trees)
    - Training data comes from past PYMK
    - Only train on 'first impressions'

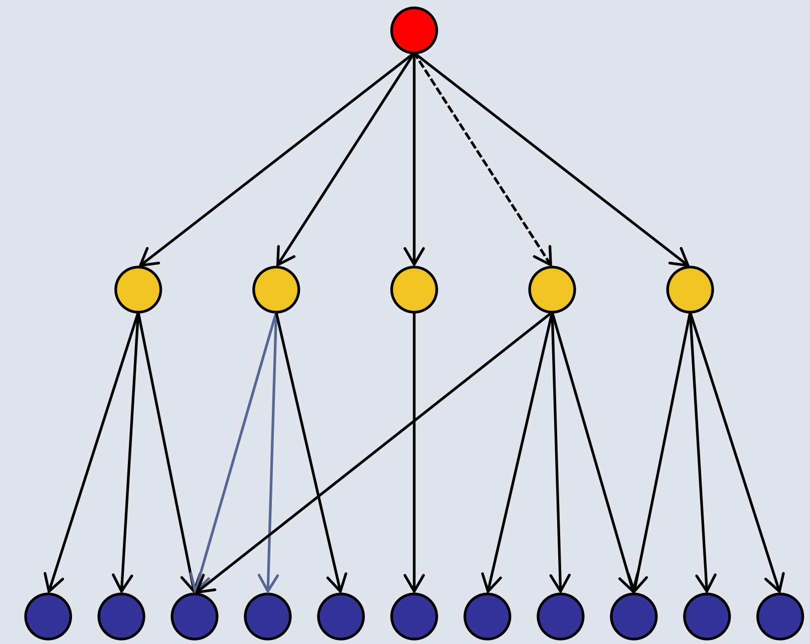


Making Static Predictions

- Out of all features, time discounted mutual friends are most important
- Total friends of user, suggestion also very important
  - For instance, having 3/3 mutual friends better than 3/200
- Demographic information also used, but secondary
  - Age, gender, country

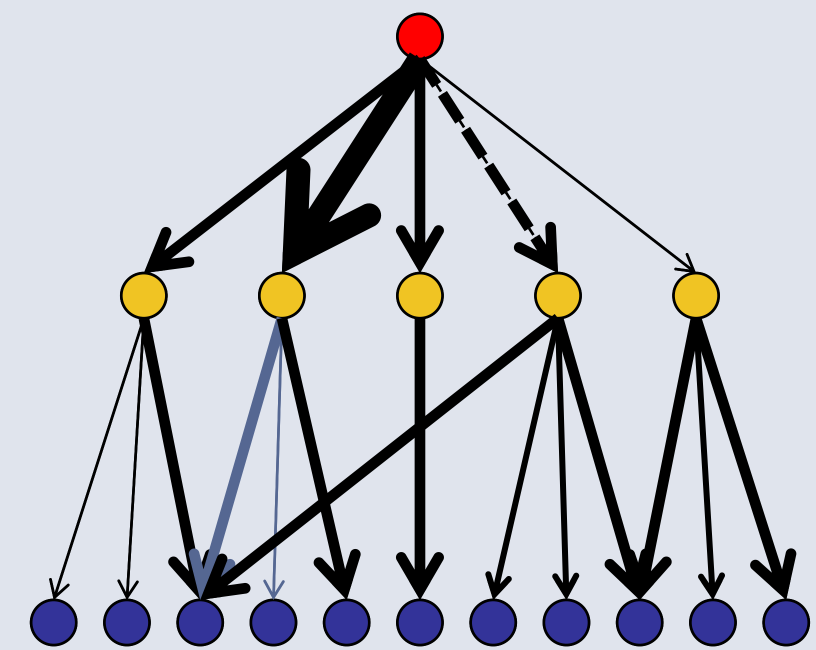


Friend of Friend Features



- Two types of features
  - Weighted Friend-of-Friend
    - Actual FoFs, Pending FoFs, Time Weighted FoFs, Coefficient Weighted FoFs
  - Demographic features
    - Age, country, Facebook age, gender, friend count, etc.
    - Because average person has 40K FoFs, these must be local, and hence are not sharded, but are duplicated on every machine.
- Most important features for prediction
  1. Time discounted mutual friends: $v(fof) = \sum_{f_i} \frac{(\delta_{u,f_i} \cdot \delta_{f_i,fof})^{-0.3}}{\sqrt{friends_{f_i}}}$
  2. Number of friends
  3. Country and Facebook age of source user

Friend of Friend Features



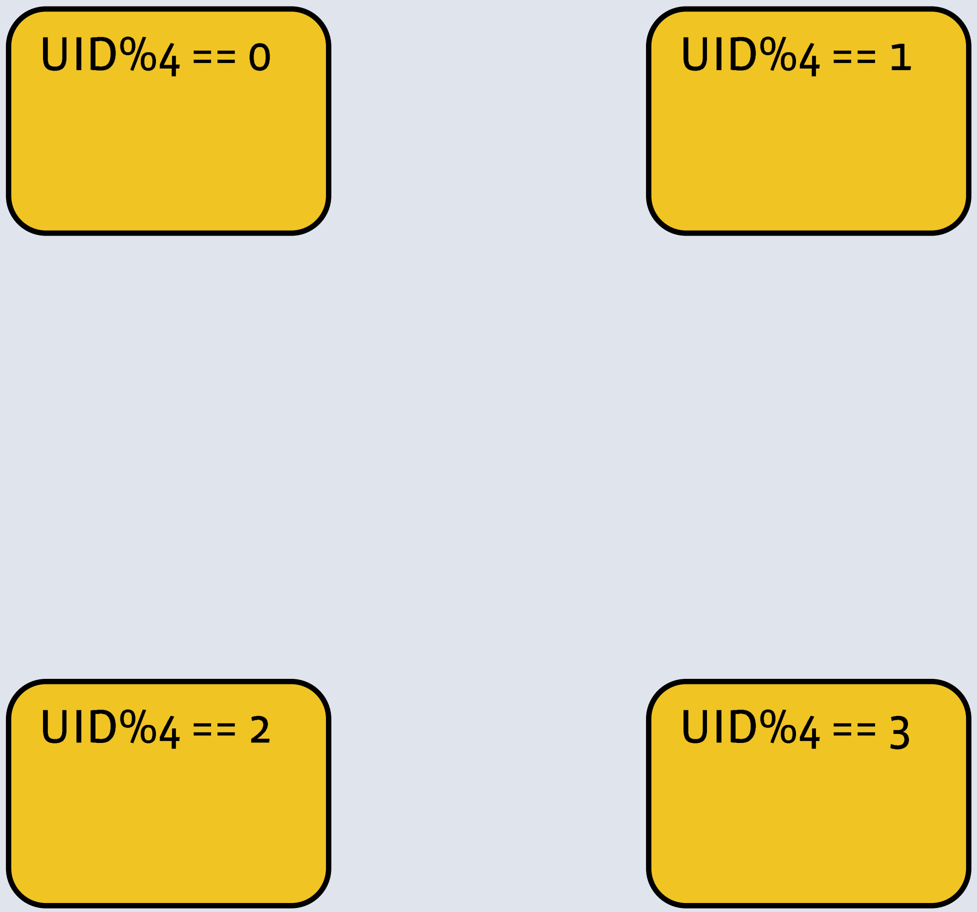
- Two types of features
  - Weighted Friend-of-Friend
    - Actual FoFs, Pending FoFs, Time Weighted FoFs, Coefficient Weighted FoFs
  - Demographic features
    - Age, country, Facebook age, gender, friend count, etc.
    - Because average person has 40K FoFs, these must be local, and hence are not sharded, but are duplicated on every machine.
- Most important features for prediction
  1. Time discounted mutual friends: $v(fof) = \sum_{f_i} \frac{(\delta_{u,f_i} \cdot \delta_{f_i,fof})^{-0.3}}{\sqrt{friends_{f_i}}}$
  2. Number of friends
  3. Country and Facebook age of source user

Doing this is expensive!

- The average user has 40K FoFs
- There are over 400M users
- 40K * 400M = 16 Trillion!
- Multiple racks (40 machines) with 72GB memory each
  - Each machine holds a fraction of the social graph in memory (it's far too big for one machine)
  - Even so, we only compute new suggestions once every ~2 days
- To ensure the best suggestions for new users, we generate for them more often

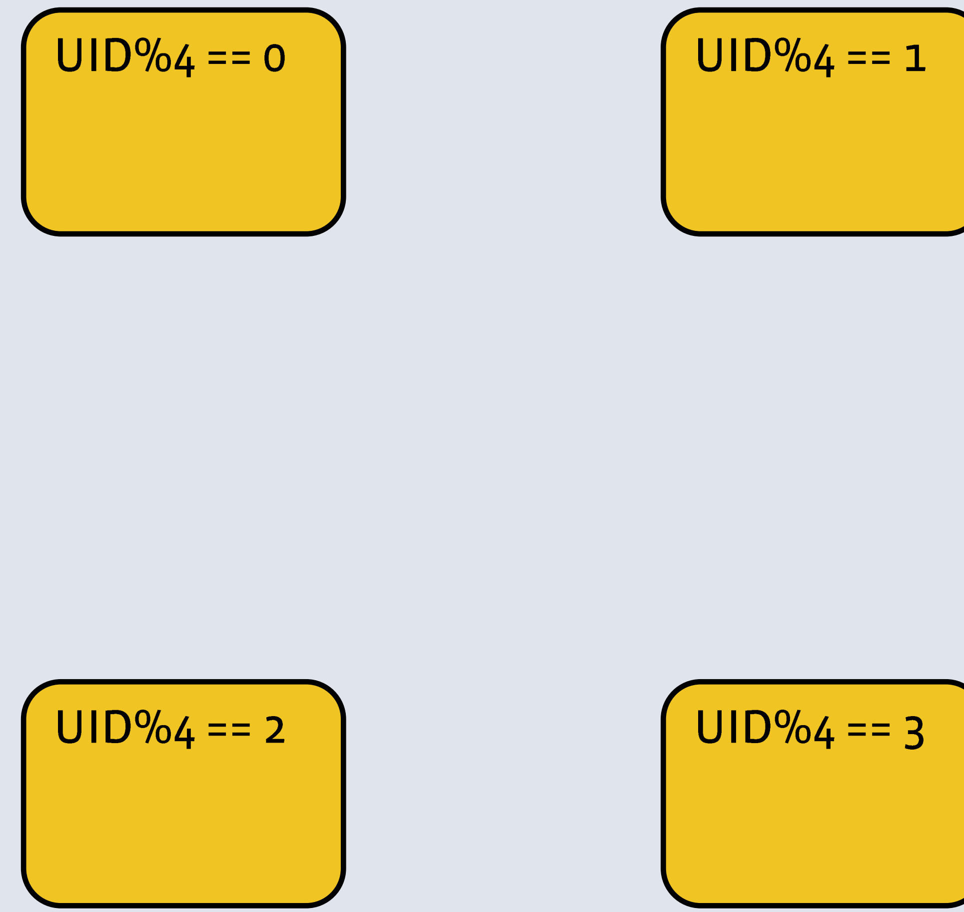
Suggestions Generation

- Social graph sharded among 40 machines
  - Includes annotations on edges: creation time, direction, coefficient
- Request goes directly to machine with user's friendlist
  - That machine splits the friend list and requests the FoFs from rest of tier
- Results are aggregated and ranked
  - Top 100 returned



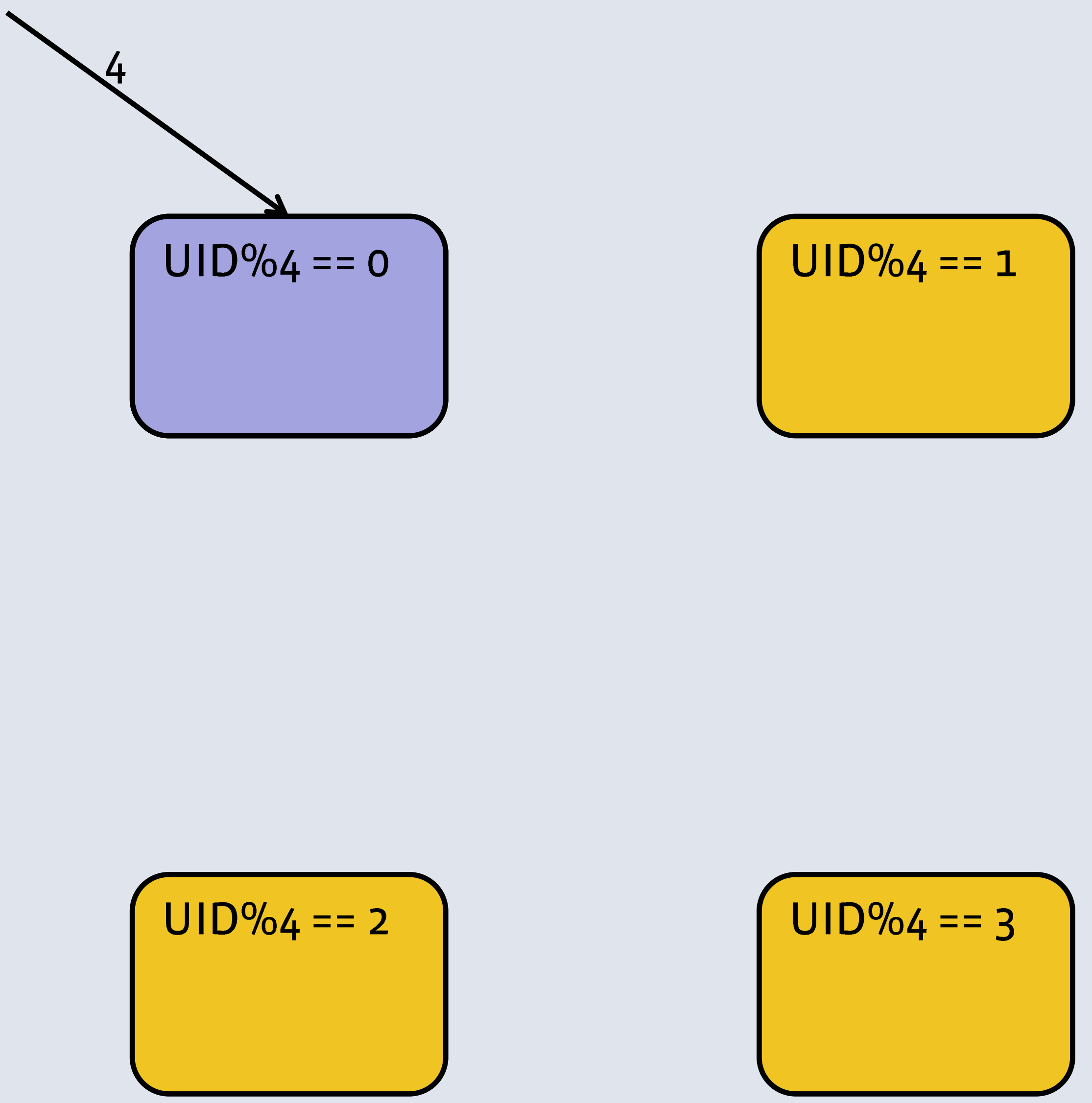
Suggestions generation

- Simple example with 4 machines



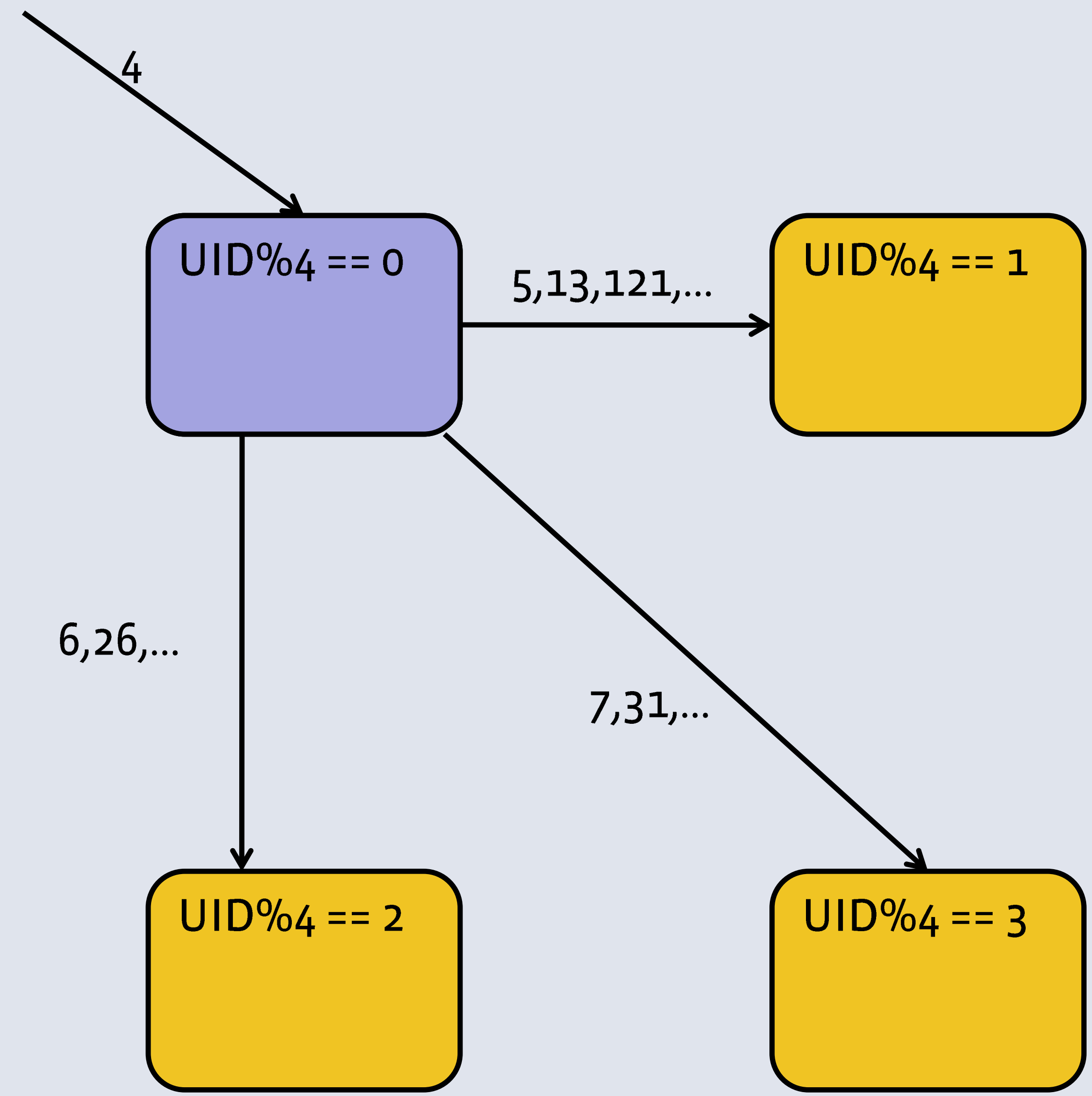
Suggestions generation

- Simple example with 4 machines
- User 4 requests PYMK
  - User 4 is friends with 5,6,7,13,26,31,121,…



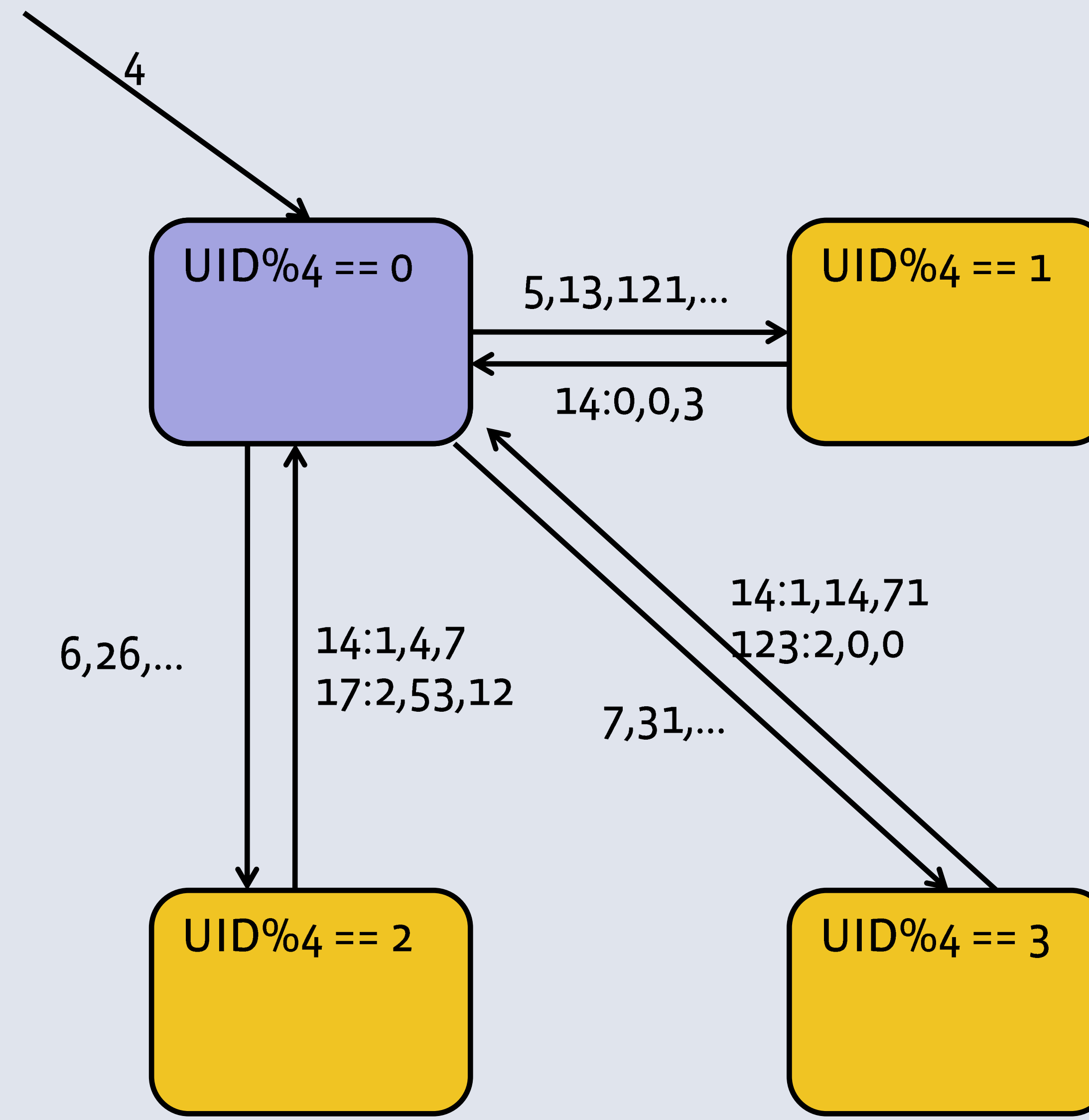
Suggestions generation

- Simple example with 4 machines
- User 4 requests PYMK
  - User 4 is friends with 5,6,7,13,26,31,121,...
- Sends requests for FoFs to all other machines (also some local)



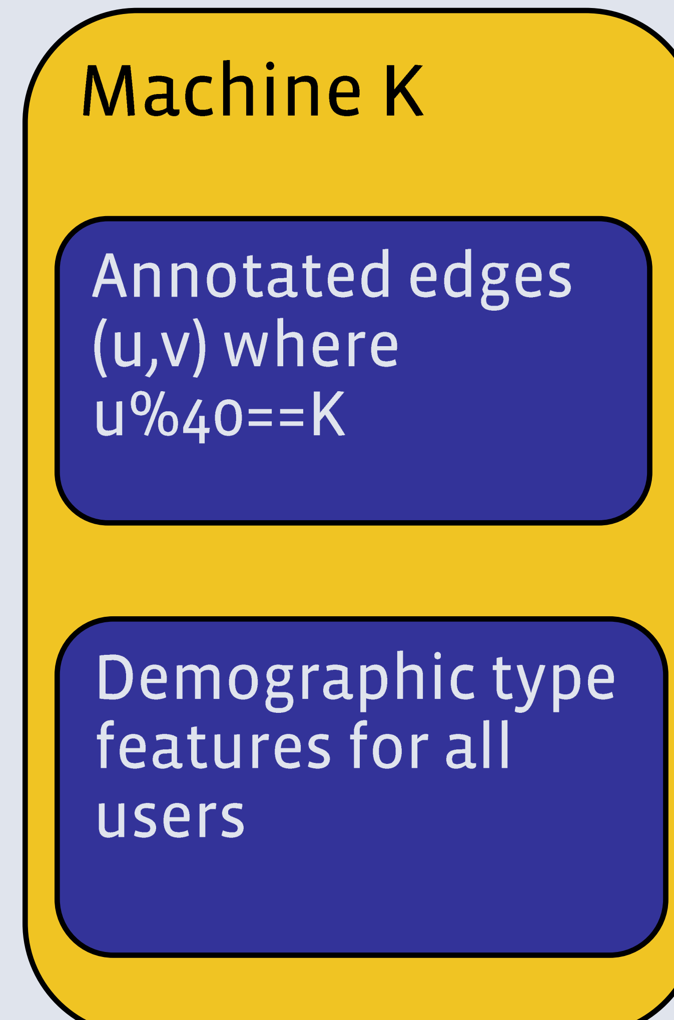
Suggestions generation

- Simple example with 4 machines
- User 4 requests PYMK
  - User 4 is friends with 5,6,7,13,26,31,121,...
- Sends requests for FoFs to all other machines (also some local)
- Feature vectors for each FoF are aggregated
  - 14:2,18,81
    17:2,53,12
    123:2,0,0
    ...



Making things fast and memory efficient

- Can’t afford to run full decision tree evaluation on all 40K FoFs for every person
  - Use heuristics to narrow the field
  - Select top 5K by time-weighted mutual friends feature
    - Use linear-time rank-N algorithm to find cutoff (no N log N sorting)
    - Run full decision tree algorithm only on them
- Don’t want to use network to get age, gender, etc. for 5K users
  - Every machine has a local in memory copy
- Select top 100 out of fully ranked 5K
  - Only these are eligible to be shown
  - To ensure diversity, temporarily blacklist any suggestion seen by a user over 4 times



Making things fast and memory efficient

System ranks 8,600,000 suggestions per second

Machine K

Annotated edges (u,v) where u%40==K

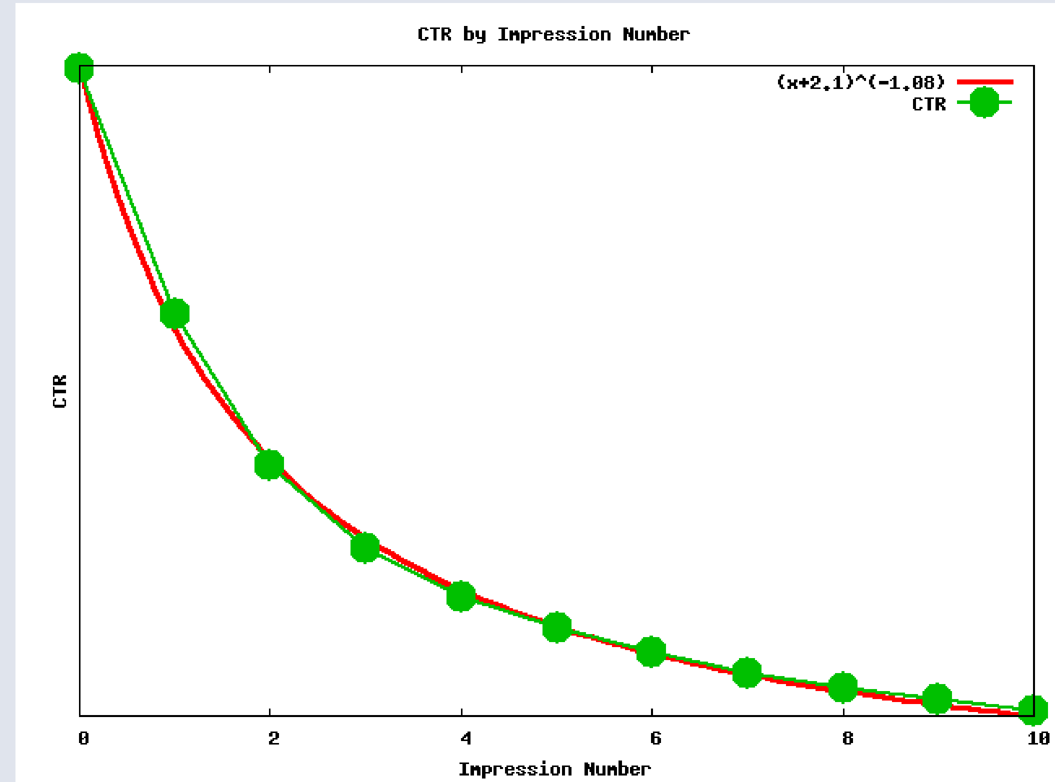
Demographic type features for all users

Agenda

1	Who to suggest?
2	Static, offline predictions
3	Dynamic, online reranking
4	Performance/Wrap-Up

Showing the best suggestion every time

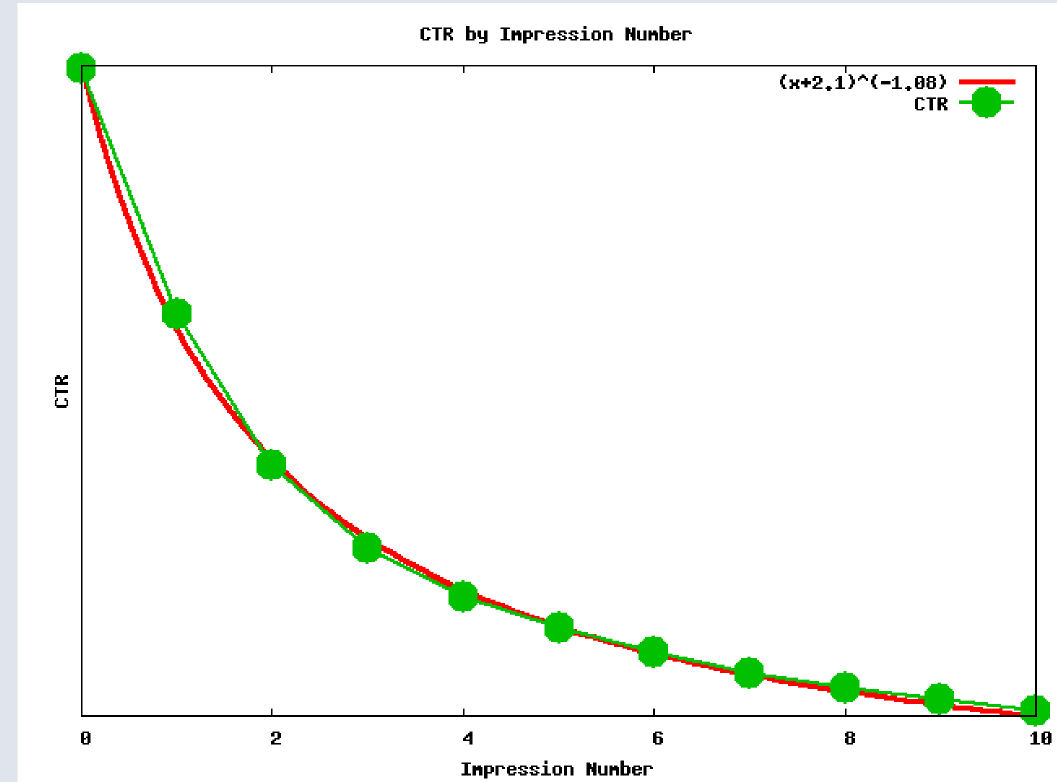
- To optimize the suggestions, we re-rank after every impression
  - Decision models can only be run once per 2 days
    - They output a score for each (u,w_i) pair
  - Can't do much too much computation for each impression, but can do a little
    - Simple features are available at each impression, for each suggestion
      - $score(u,w_i)$, number of impressions for (u,w_i), friend count(u), friend count(w_i)



Showing the best suggestion every time

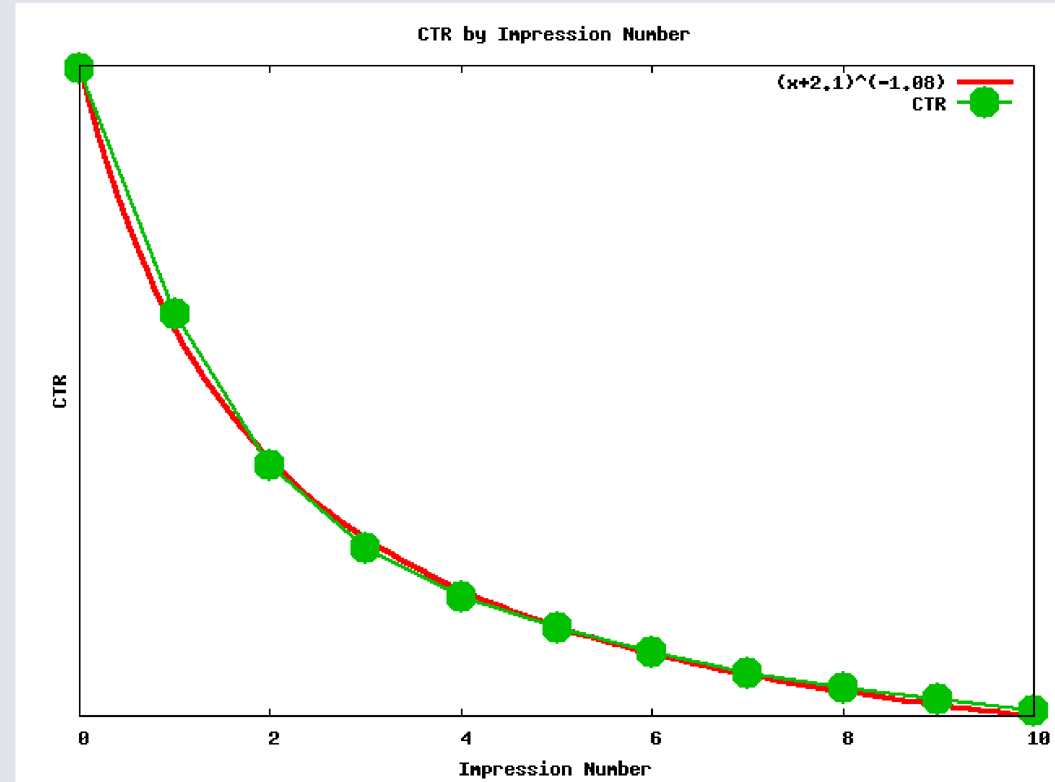
- To optimize the suggestions, we re-rank after every impression
  - Decision models can only be run once per 2 days
    - They output a score for each (u,w_i) pair
  - Can't do much too much computation for each impression, but can do a little
    - Simple features are available at each impression, for each suggestion
      - $score(u,w_i)$, number of impressions for (u,w_i), friend count(u), friend count(w_i)

Combine what is available with score to re-rank via Logistic Regression



Showing the best suggestion every time

- To optimize the suggestions, we re-rank after every impression
  - Decision models can only be run once per 2 days
    - They output a score for each (u,w_i) pair
  - Can't do much too much computation for each impression, but can do a little
    - Simple features are available at each impression, for each suggestion
      - score(u,w_i), number of impressions for (u,w_i), friend count(u), friend count(w_i)



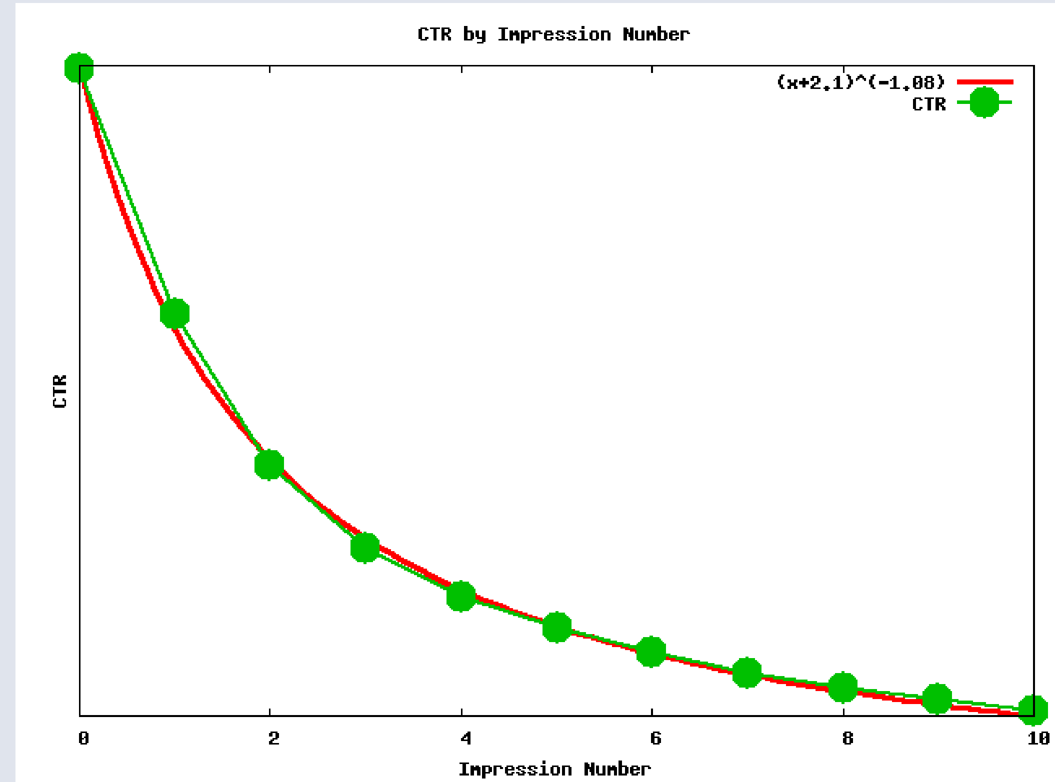
Suggestion	Impressions	CTR Prediction
Alice	0	0.048
Bob	0	0.031
Carol	0	0.027
David	0	0.025

Combine what is available with score to re-rank via Logistic Regression

Showing the best suggestion every time

- To optimize the suggestions, we re-rank after every impression
  - Decision models can only be run once per 2 days
    - They output a score for each (u,w_i) pair
  - Can't do much too much computation for each impression, but can do a little
    - Simple features are available at each impression, for each suggestion
      - $score(u,w_i)$, number of impressions for (u,w_i), friend count(u), friend count(w_i)

Combine what is available with score to re-rank via Logistic Regression

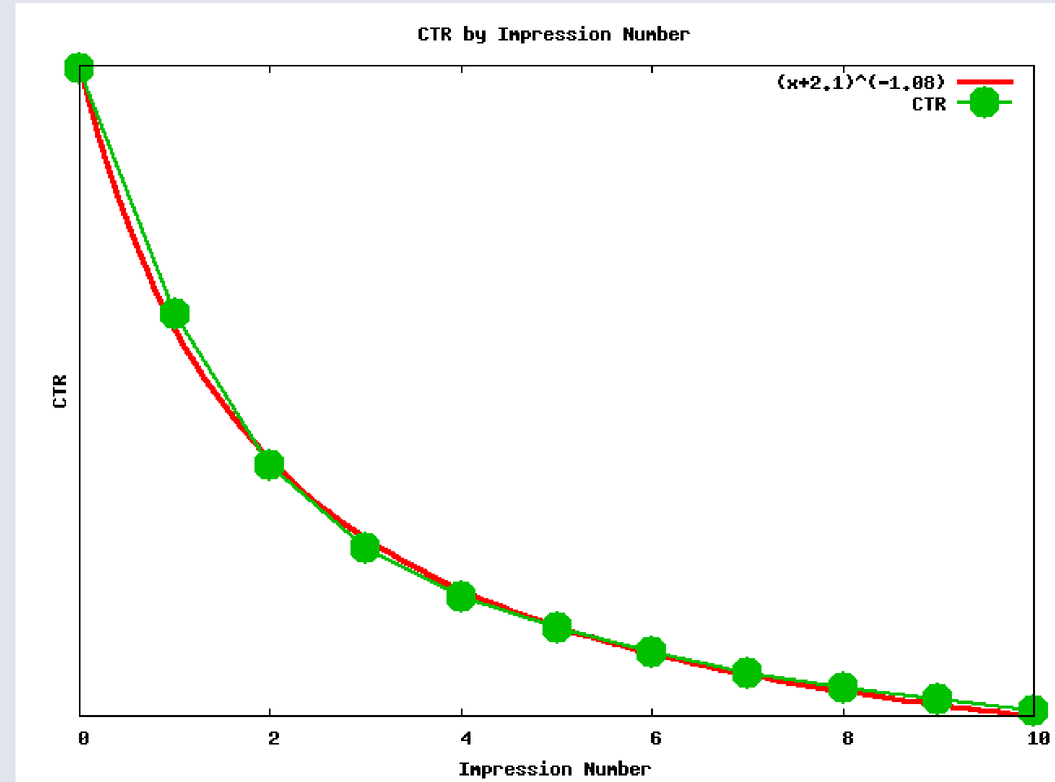


Suggestion	Impressions	CTR Prediction
Carol	0	0.027
David	0	0.025
Alice	1	0.025
Bob	1	0.016

Showing the best suggestion every time

- To optimize the suggestions, we re-rank after every impression
  - Decision models can only be run once per 2 days
    - They output a score for each (u,w_i) pair
  - Can't do much too much computation for each impression, but can do a little
    - Simple features are available at each impression, for each suggestion
      - $score(u,w_i)$, number of impressions for (u,w_i), friend $count(u)$, friend $count(w_i)$

Combine what is available with score to re-rank via Logistic Regression

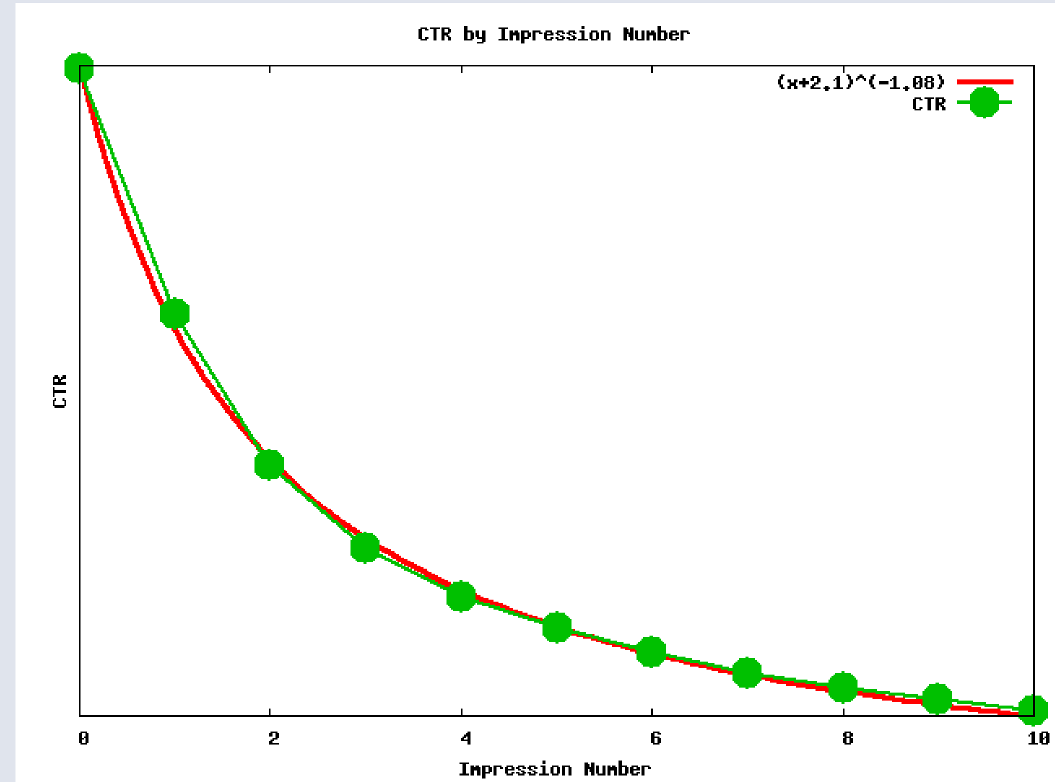


Suggestion	Impressions	CTR Prediction
Alice	1	0.025
Bob	1	0.016
Carol	1	0.014
David	1	0.012

Showing the best suggestion every time

- To optimize the suggestions, we re-rank after every impression
  - Decision models can only be run once per 2 days
    - They output a score for each (u,w_i) pair
  - Can't do much too much computation for each impression, but can do a little
    - Simple features are available at each impression, for each suggestion
      - $score(u,w_i)$, number of impressions for (u,w_i), friend count(u), friend count(w_i)

Combine what is available with score to re-rank via Logistic Regression

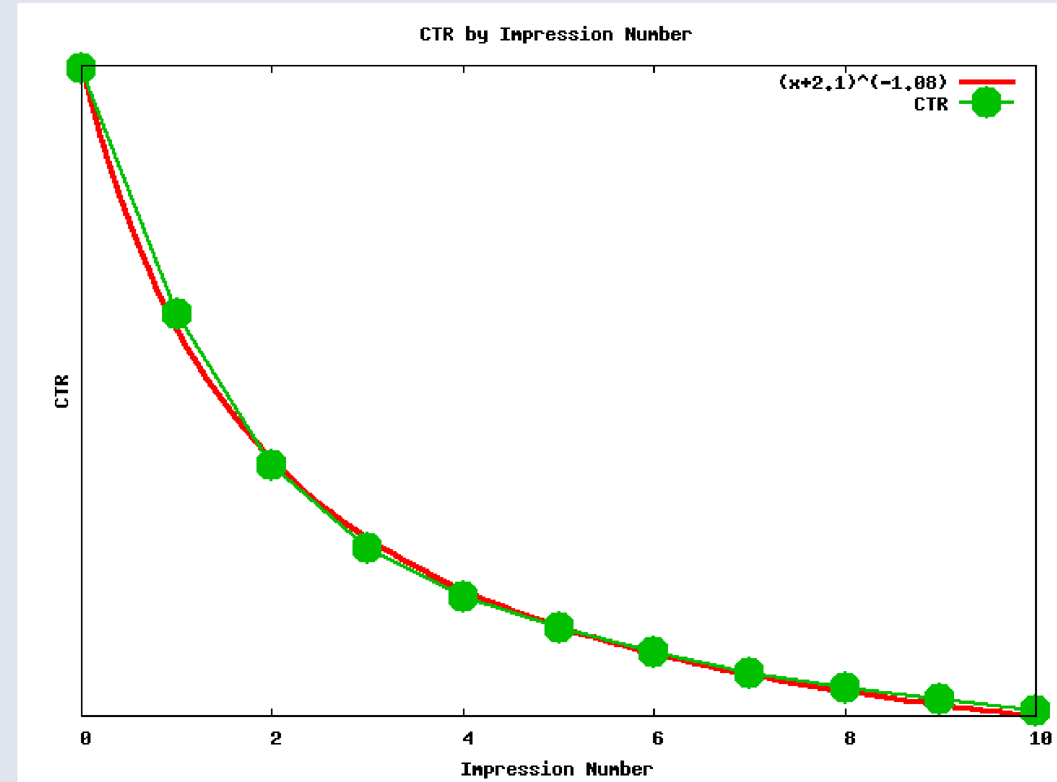


Suggestion	Impressions	CTR Prediction
Alice	2	0.016
Carol	1	0.014
David	1	0.012
Bob	2	0.010

Showing the best suggestion every time

- To optimize the suggestions, we re-rank after every impression
  - Decision models can only be run once per 2 days
    - They output a score for each (u,w_i) pair
  - Can't do much too much computation for each impression, but can do a little
    - Simple features are available at each impression, for each suggestion
      - $score(u,w_i)$, number of impressions for (u,w_i), friend count(u), friend count(w_i)

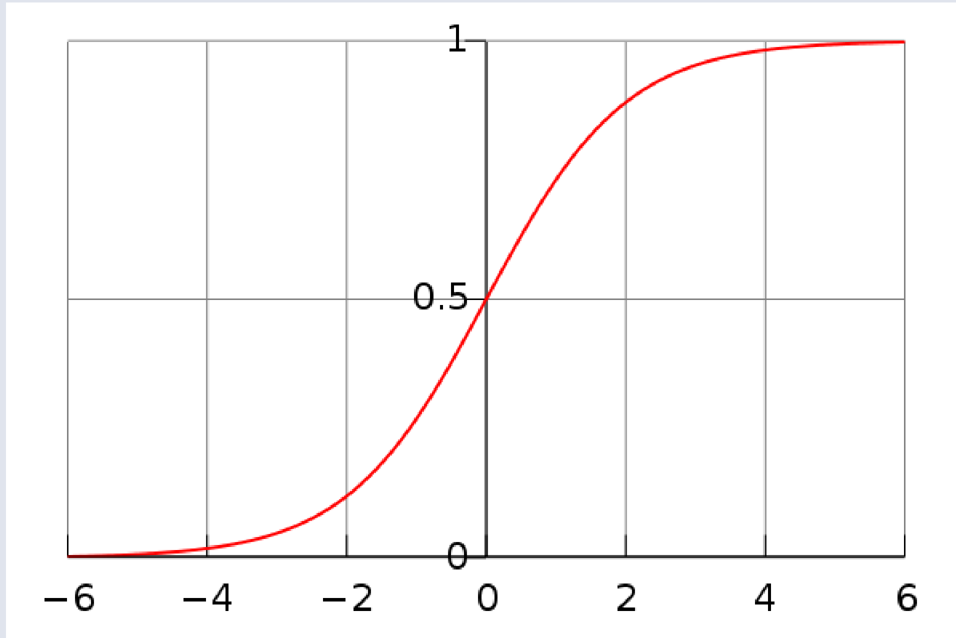
Combine what is available with score to re-rank via Logistic Regression



Suggestion	Impressions	CTR Prediction
David	1	0.012
Alice	3	0.011
Bob	2	0.010
Carol	2	0.009

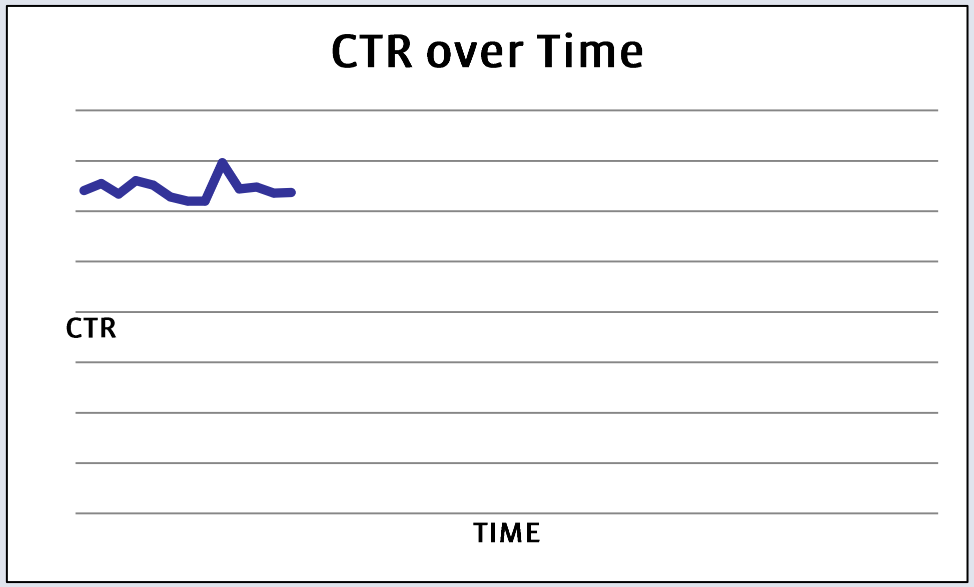
Reranking with logistic regression

- Most important features have to do with offline score and user's PYMK history
  - What score did the decision trees give?
  - How many friends has the user added through PYMK in the last week
  - How many has she rejected?
  - How many suggestions did we make?
  - How many times have we shown her each suggestion?



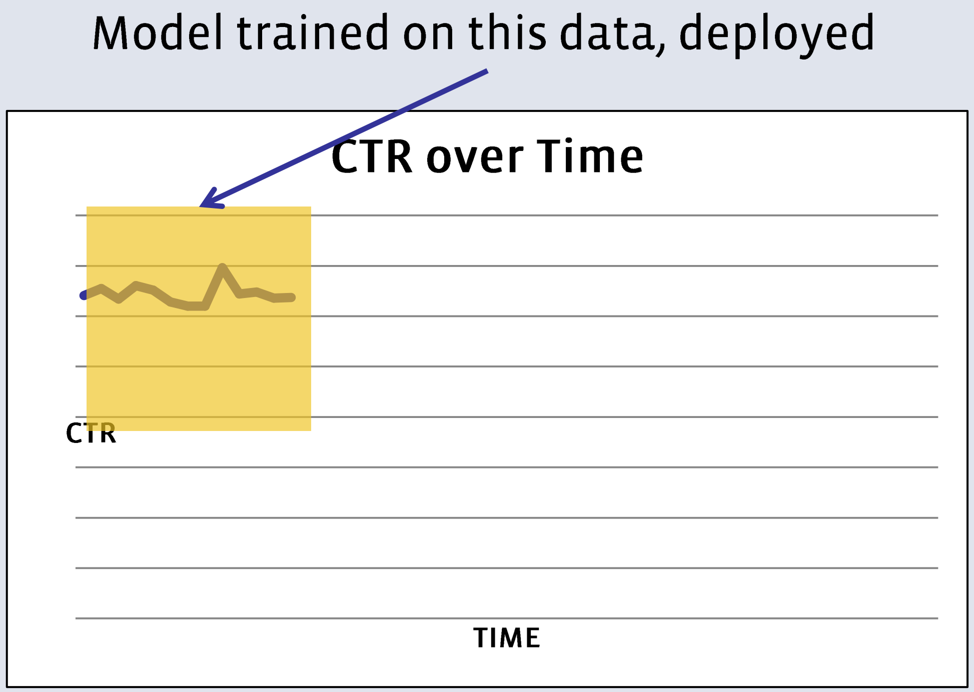
- Simple to implement, lots of software to learn coefficients
  - Using user history data to personalize gives HUGE improvements!

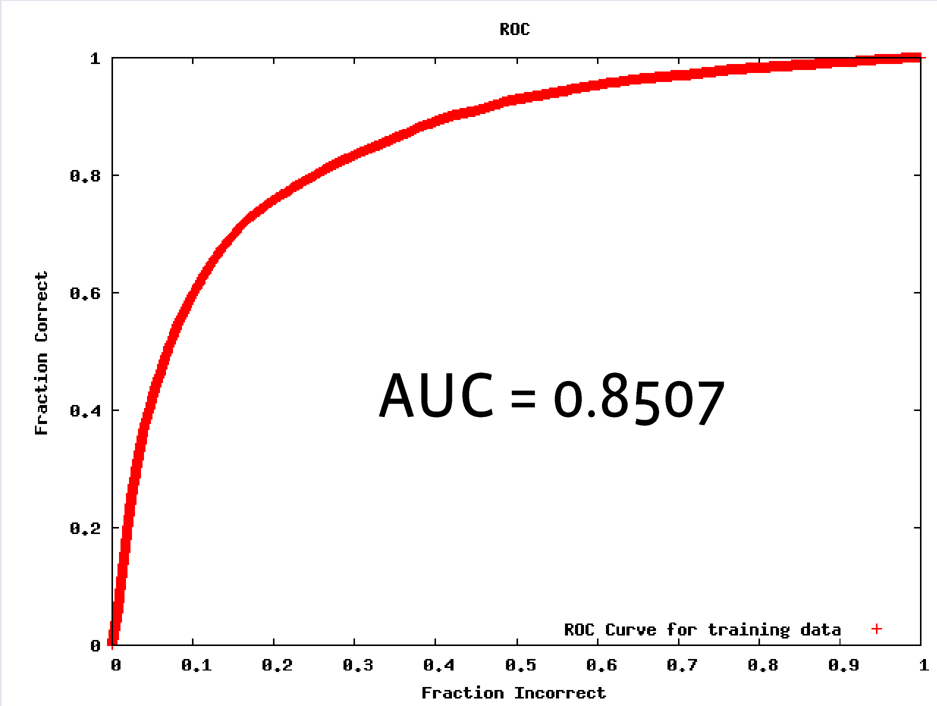
Machine Learning Challenges



- Good predictions on previous data don't always work out
  - May give high scores to suggestions not represented in previous dataset
- If training from scratch, requires a few iterations to converge
  - Moving towards more online system

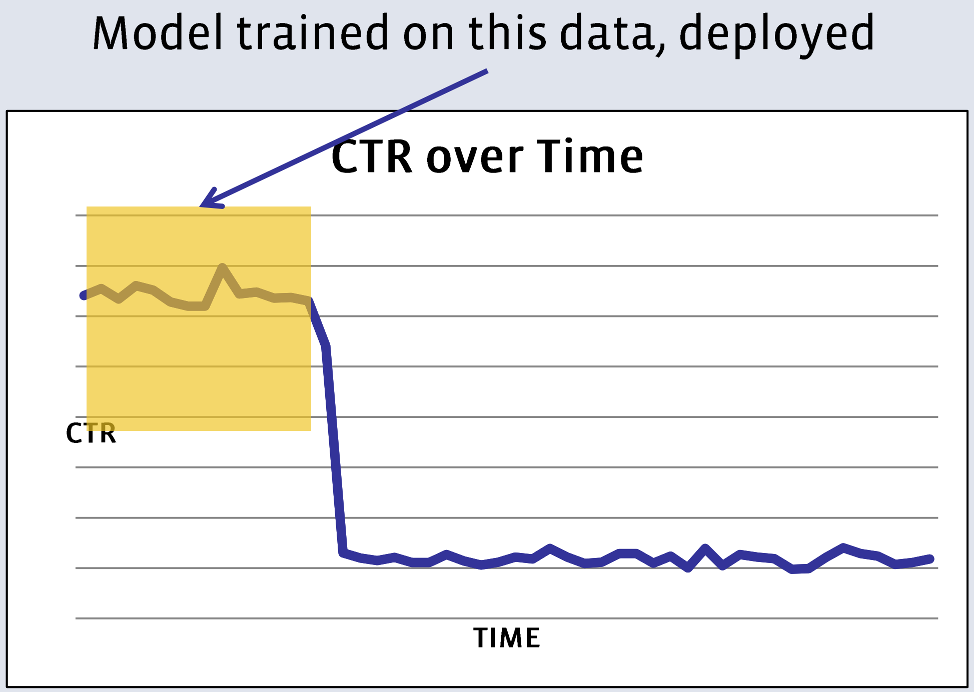
Machine Learning Challenges

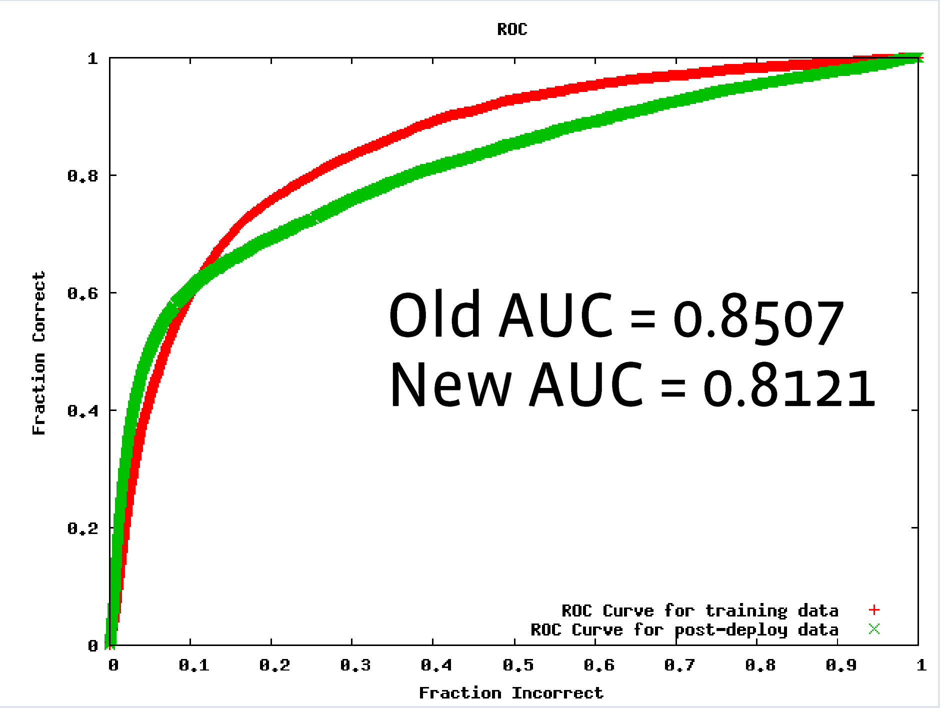




- Good predictions on previous data don't always work out
  - May give high scores to suggestions not represented in previous dataset
- If training from scratch, requires a few iterations to converge
  - Moving towards more online system

Machine Learning Challenges

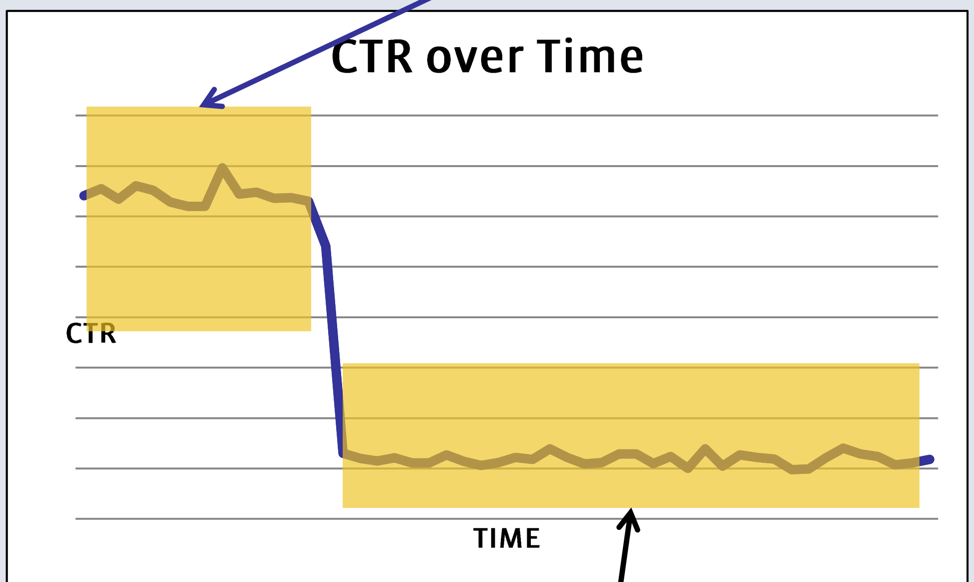




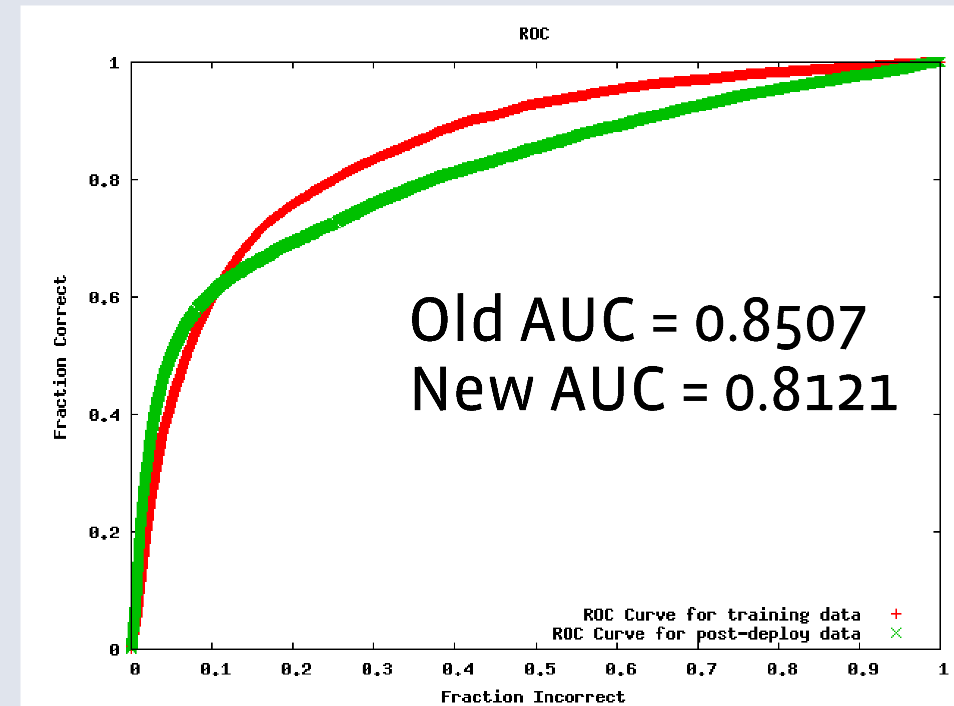
- Good predictions on previous data don't always work out
  - May give high scores to suggestions not represented in previous dataset
- If training from scratch, requires a few iterations to converge
  - Moving towards more online system

Machine Learning Challenges

Model trained on this data, deployed

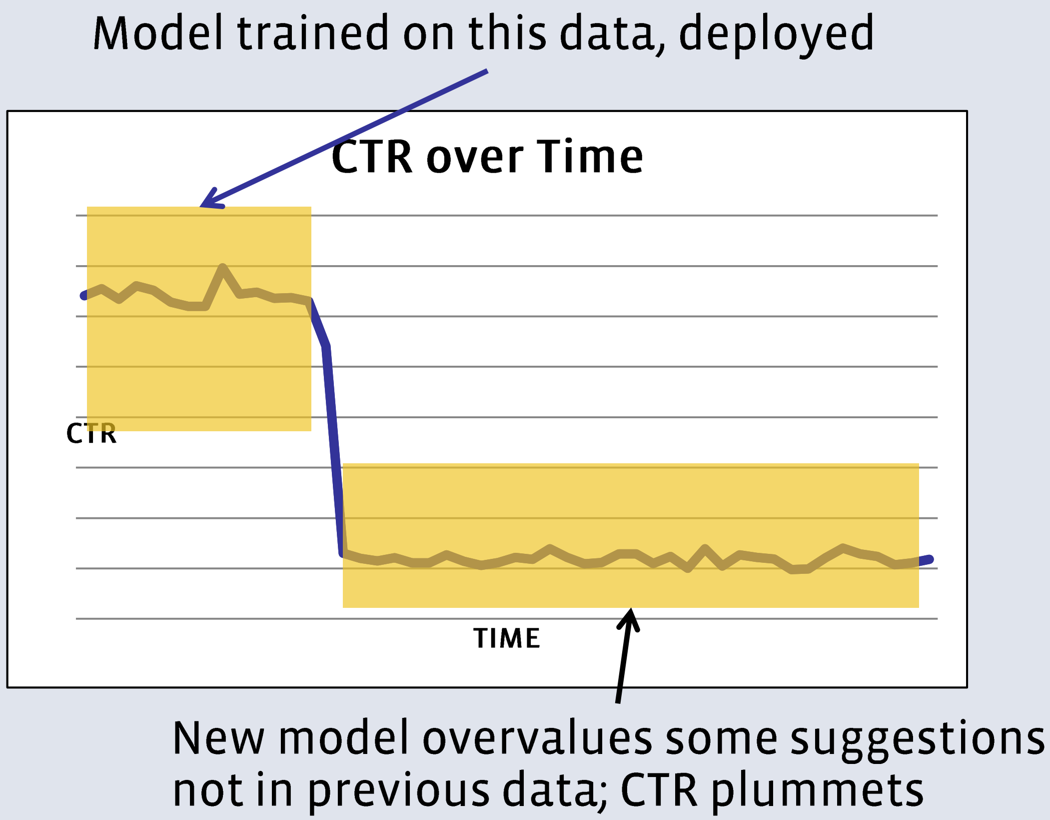


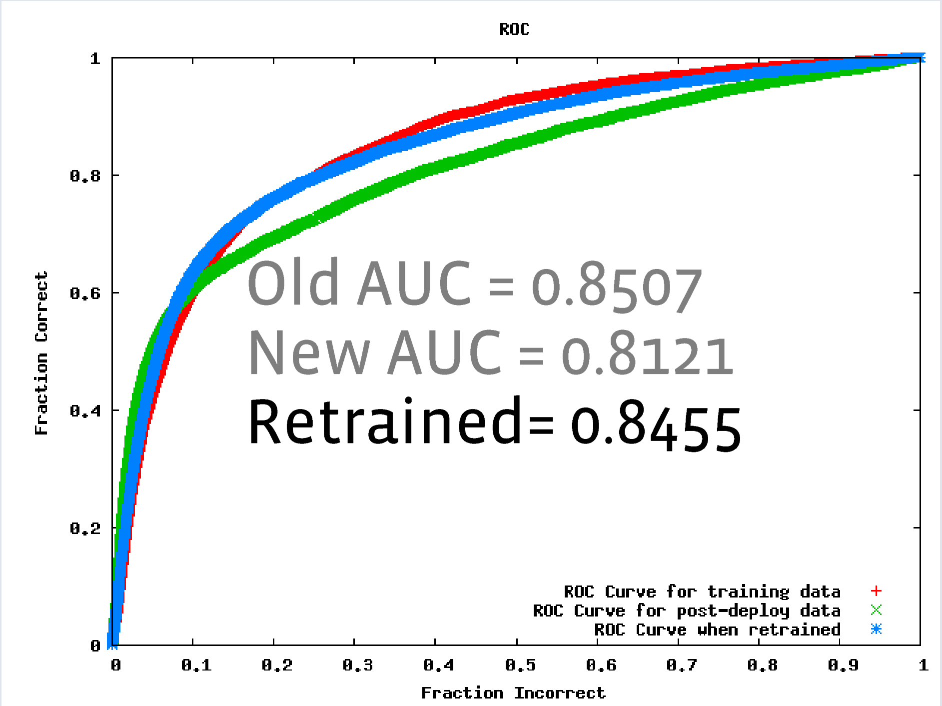
New model overvalues some suggestions
not in previous data; CTR plummets



- Good predictions on previous data don't always work out
  - May give high scores to suggestions not represented in previous dataset
- If training from scratch, requires a few iterations to converge
  - Moving towards more online system

Machine Learning Challenges





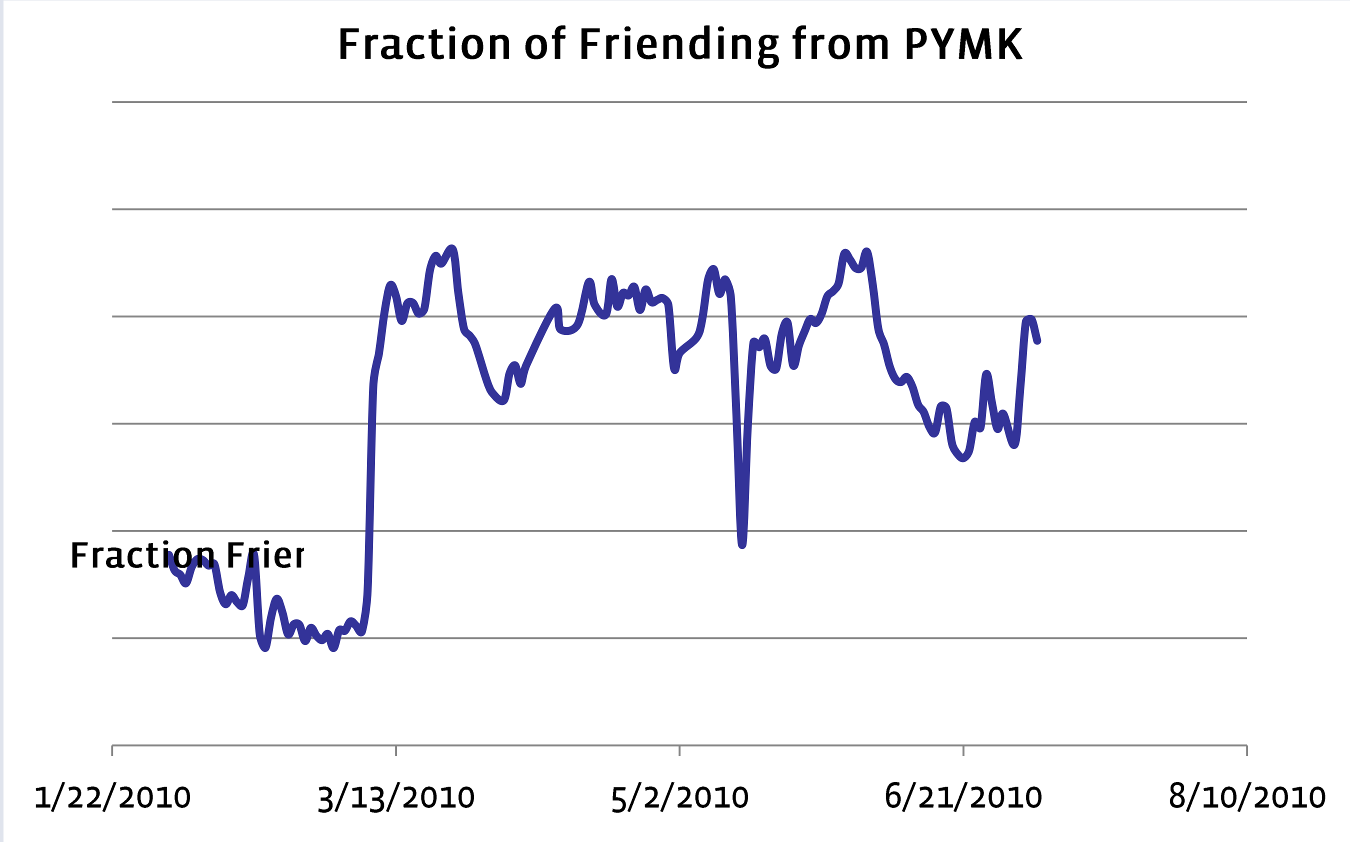
- Good predictions on previous data don't always work out
  - May give high scores to suggestions not represented in previous dataset
- If training from scratch, requires a few iterations to converge
  - Moving towards more online system

Agenda

1	Who to suggest?
2	Static, offline predictions
3	Dynamic, online reranking
4	Performance/Wrap-Up

Performance

- Two performance metrics
  - Friendships created
  - Click-through Rate
- Can always increase one at the cost of the other



- Initial launch of offline model and CTR prediction in early march
  - Recent poor performance due to memcache problems (losing all user-view history data)
  - Overall, increase in total adds up 60%
  - At the same time, CTR prediction has cut impressions have been cut by 1/3
  - Hence, CTR is up by 130%

Takeaways

- Edge annotations are useful features
  - Coefficient helps us a little, creation time more
- Huge performance wins from simple user customization
  - Learn what people use, what they ignore, show them what they like!
- Context matters
  - Use the main content of the page to inform what else to show

Questions