

Recommender Systems



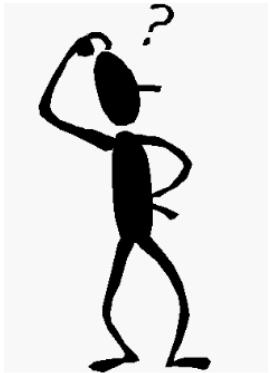
TATRAS
DATA | ALGORITHMS | INSIGHT

Dr. Sarabjot Singh

Co-founder & Chief Data Scientist

ssanand@tatrasdata.com

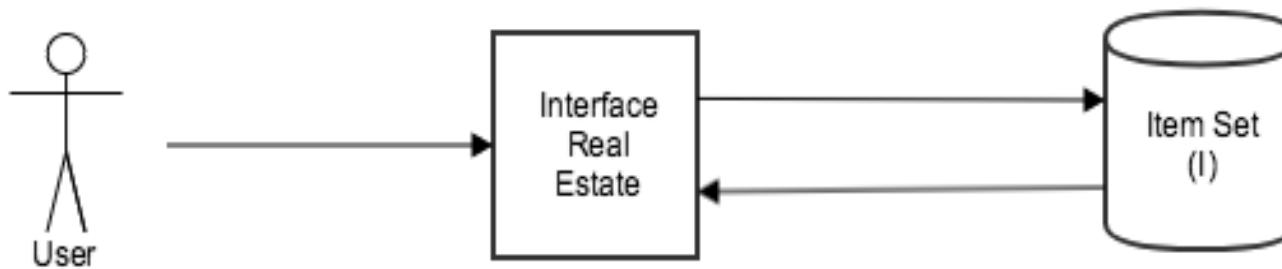
Too many Items, too little time



Why search when you can filter?

WHAT PROBLEM DO RECOMMENDER SYSTEMS SOLVE?

- Users lost in a sea of 'items', I
- Real estate of the 'window' to Items is limited

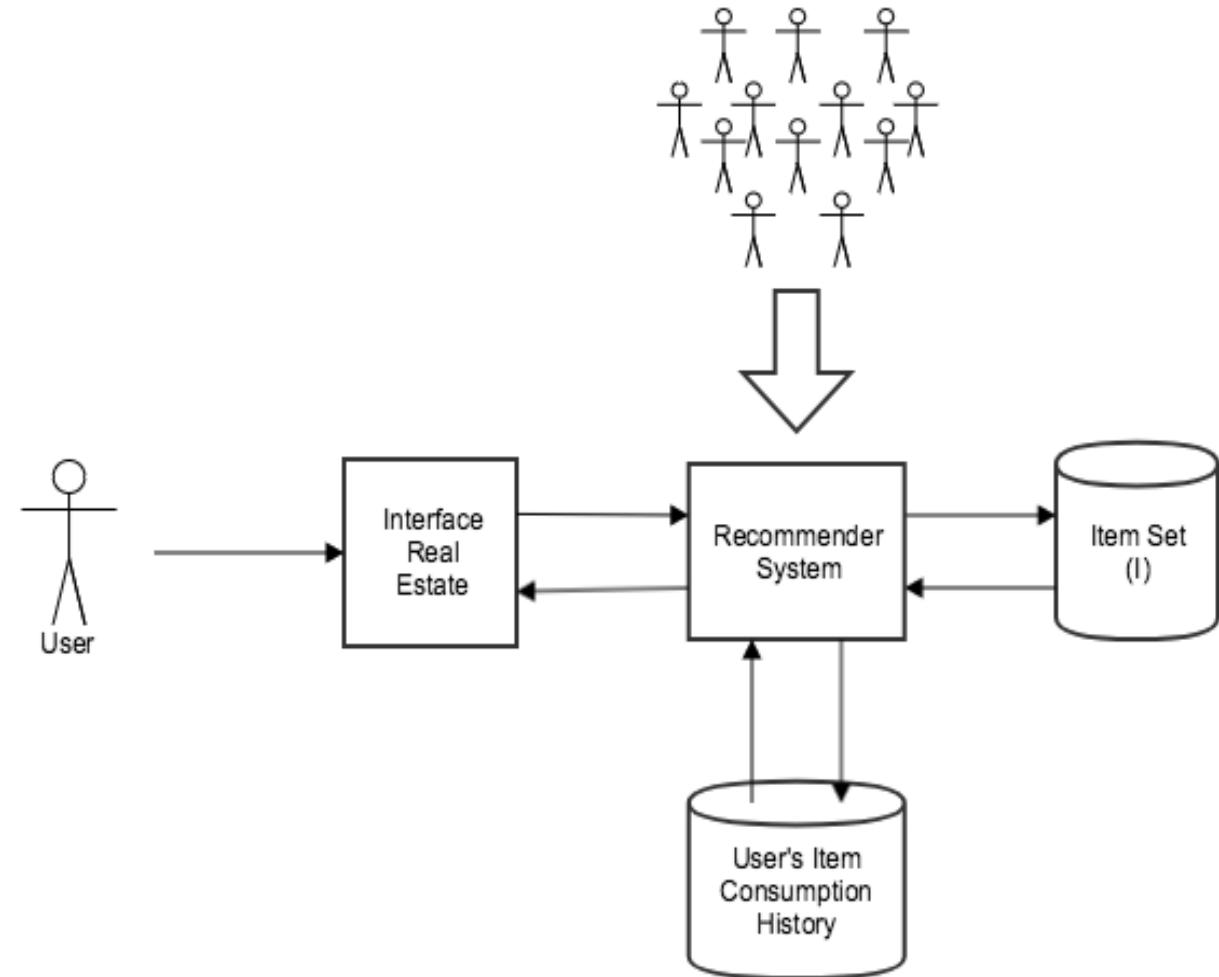


- Information Retrieval Vs. Information Filtering
 - Explicit search terms Vs. Implicit Consumption Traits
 - The user gets what (s)he wants without needing to 'ask' for it



TYPES OF RECOMMENDER SYSTEMS

- **Content based Filtering**
 - Track what the user consumes
 - Recommend Items similar to those consumed in the past
 - Need item descriptions
 - Narrow Vision/ Low diversity
- **Collaborative Filtering**
 - Track all user consumption
 - No need for item descriptions
 - Recommend items consumed by users who have a similar consumption pattern to the user of interest
 - More serendipity
 - First automated recommender based on collaborative filtering GroupLens (1994)
- **Hybrid Approaches**
 - Combines both approaches to address shortcomings



RECOMMENDATION AS RATING PREDICTION

- Recommendation can be viewed as a rating prediction problem
 - Given a set of triples

$$\{(u_j, i_k, r_{jk}) : u_j \in U, i_k \in I, r_{jk} \in \{1, 2, \dots, r\}\}$$

- approximate a function

$$f: (u_j, i_k) \rightarrow [1, r]$$

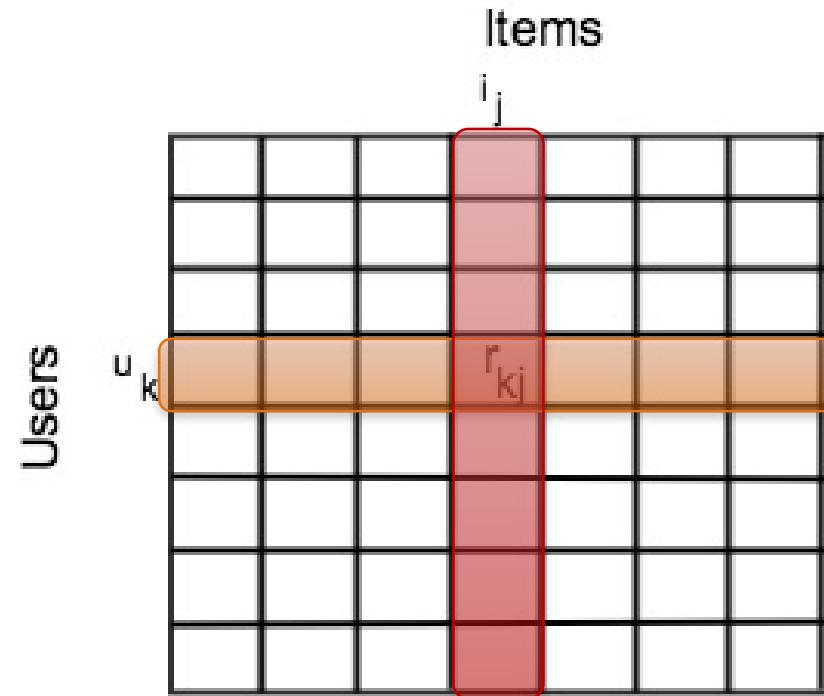
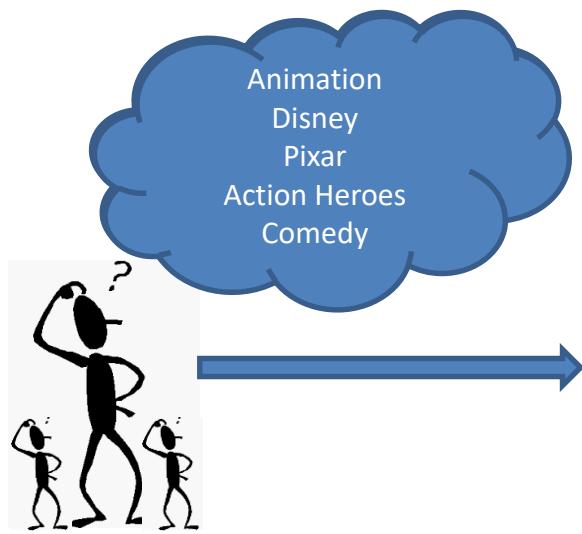
- The accuracy of the function is computed using metrics such as

$$mse = \frac{1}{n} \sum_{j,k} (r_{jk} - f(u_j, i_k))^2 \quad \text{and} \quad mae = \frac{1}{n} \sum_{j,k} |r_{jk} - f(u_j, i_k)|$$

where n is the number of triples in the test set



THE DATA: USER ITEM RATING MATRIX (R)



$$f: U \times I \times C \rightarrow [1, r]$$

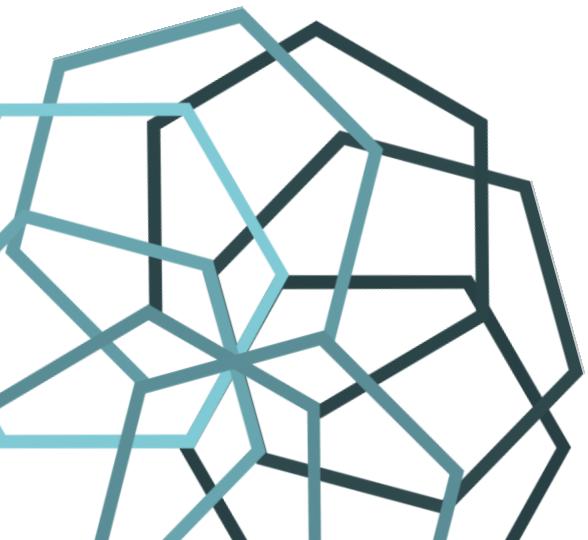


Users	Titanic	Batman	Inception	SuperMar	Spiderma	matrix
Michele	2.5		3	3.5	2.5	3
SATYA	3	3.5		5	3	3.5
PRANAV	2.5	3		3.5		4
SURESH		3.5	3	4	2.5	
TOM		4	2	3	2	3
LEO	3	4		5	3.5	3
CHAN		4.5		4	1	

USER-BASED COLLABORATIVE FILTERING

- The basic premise is that similar users like similar items
 - Find the neighbourhood consisting of k users judged to be most similar for the active user, u_a
 - Recommend items to user, u_a , that have been liked by users in their neighbourhood, not already consumed by u_a
 - The result: “People who liked *those* also liked *this*”
- Calculating User Similarity
 - Cosine Similarity

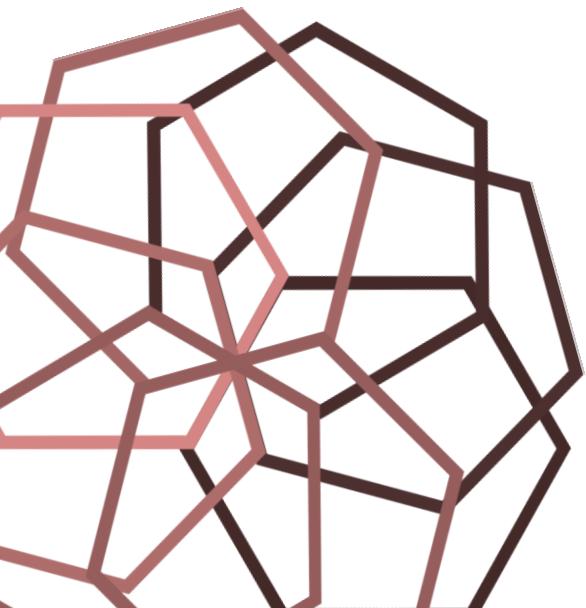
$$s_{ab} = \cos\theta = \frac{\nu_a \cdot \nu_b}{|\nu_a| |\nu_b|}$$



EXAMPLE

	i_1	i_2	i_3	i_4	i_5	i_6	$ u $	$\text{sim}(u_4, u_j)$	$\text{Avg}(u_j)$
u_1	2	4	3				5.39	0.517	3
u_2		3	4	3	1		5.92	0.735	2.75
u_3	1		3	4			5.10	0.955	2.67
u_4	?	?	4	?	4	1	5.74		3
u_5		3		2		4	5.39	0.129	3

What do you think the missing ratings are going to be?



PREDICTING THE RATINGS

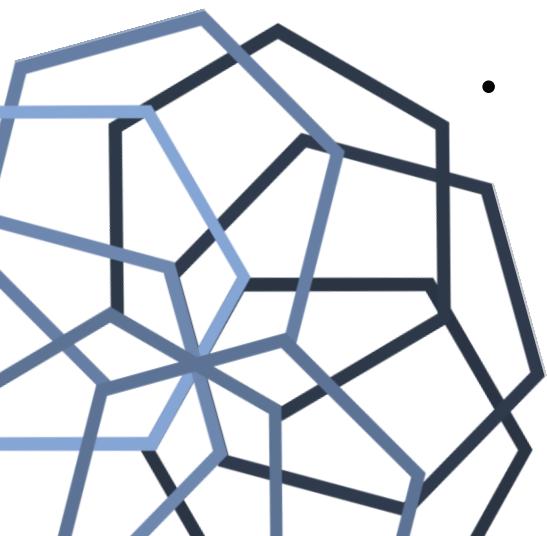
- Rating predicted for each item not previously consumed by the active user

$$\widehat{r}_{aj} = \frac{\sum_{u_k \in N(a)} s_{ak} r_{kj}}{\sum_{u_k \in N(a)} s_{ak}}$$

- 
- i1 = (0.955*1)/0.955 = 1
 - i2 = (0.517*2+0.129*3)/0.646 = 1.421/0.646 = 2.2
 - i4 = (0.735*4+0.517*3-0.129*2)/1.381 = 3.43

COLD START, NEW ITEM LATENCY & SPARSITY

- Cold Start
 - Items not rated by any user, cannot be recommended
 - Users who have not consumed/rated can't be recommended to
 - Fall back to Most Popular Items (Impersonalized)
- The time between an item being introduced to the item catalogue and the first time it gets recommended is called Item Latency
 - How can this be reduced?
- Sparsity of Ratings often leads to inaccurate estimation of similarity as well as ratings
- Should all items be treated equal?
 - Everyone loves Spiderman – does he add value to similarity measurement



ALTERNATIVE SIMILARITY MEASURES

- The adjusted cosine similarity
 - Used in user based collaborative filtering to adjust for differences in user rating scales

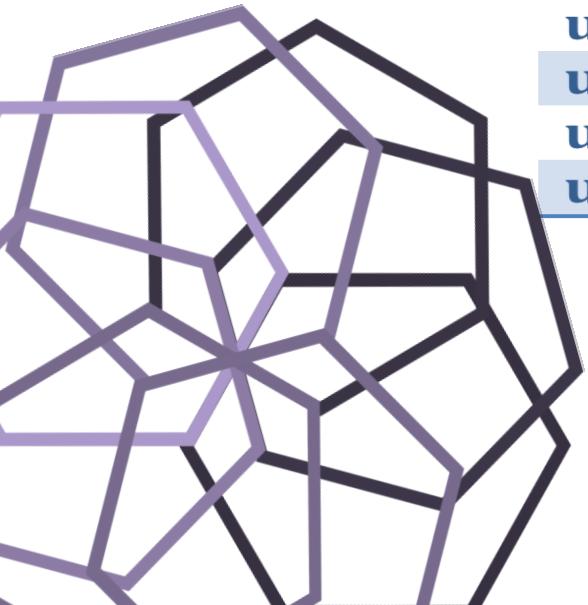
$$s_{ab} = \frac{\sum_{i_j \in R(a) \cap R(b)} (r_{aj} - \mu_a)(r_{bj} - \mu_b)}{\sqrt{\sum_{i_j \in R(a)} (r_{aj} - \mu_a)^2} \sqrt{\sum_{i_j \in R(b)} (r_{bj} - \mu_b)^2}}$$



PREDICTING THE RATINGS

- Adjusting for users with different rating scales

$$\widehat{r}_{aj} = \mu_j + \frac{\sum_{u_k \in N(a)} s_{ak} (r_{kj} - \mu_j)}{\sum_{u_k \in N(a)} s_{ak}}$$



	i ₁	i ₂	i ₃	i ₄	i ₅	i ₆	u-avg(u)	sim(u ₄ , u _j)	Avg(u _j)
u ₁		2	4	3			1.41	0.288	3
u ₂			3	4	3	1	2.18	0.749	2.75
u ₃	1		3		4		2.16	0.314	2.67
u ₄	?	?	4	?	4	1	2.45		3
u ₅		3		2		4	1.41	-0.577	3

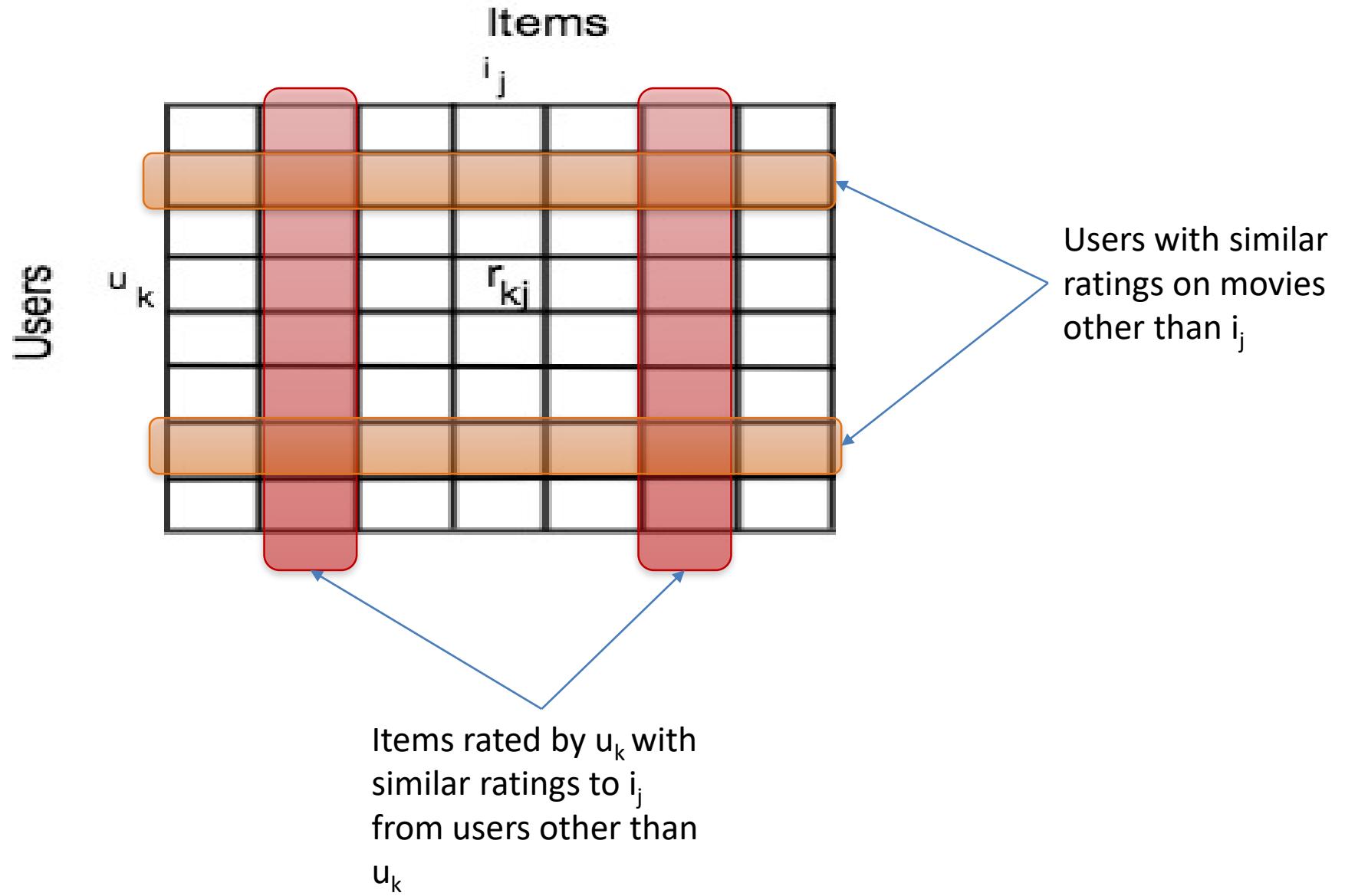
- i₁ = 3 + (0.314*(1-2.67))/0.314 = 1.33
- i₂ = 3 + (0.288*(2-3))/0.288 = 2
- i₄ = 3 + (0.749*(4-2.75)+0.288(3-3))/1.037 = 3.9

IMPACT OF ADJUSTED COSINE

	i₁	i₂	i₃	i₄	i₅	i₆	 u 	sim(u₄, u_j)	Avg(u_j)
u₁		2	4	3			5.39	0.517	3
u₂			3	4	3	1	5.92	0.735	2.75
u₃	1		3		4		5.10	0.955	2.67
u₄	1	2.2	4	3.43	4	1	5.74		3
u₅		3		2		4	5.39	0.129	3

	i₁	i₂	i₃	i₄	i₅	i₆	 u-avg(u) 	sim(u₄, u_j)	Avg(u_j)
u₁		2	4	3			1.41	0.288	3
u₂			3	4	3	1	2.18	0.749	2.75
u₃	1		3		4		2.16	0.314	2.67
u₄	1.33	2	4	3.9	4	1	2.45		3
u₅		3		2		4	1.41	-0.577	3

ITEM BASED COLLABORATIVE FILTERING



ITEM-BASED COLLABORATIVE FILTERING

- Instead of finding the neighbourhood of a user, find the neighbourhood of the target item from items rated by the active user
- Similarity of items computed using Pearson Correlation

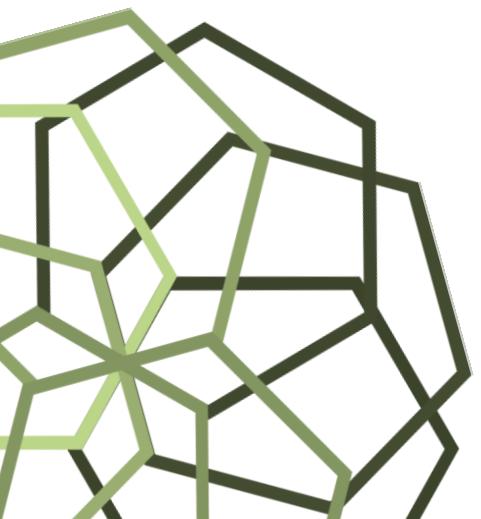
$$\rho_{ij} = \frac{\sum_{u \in U(i) \cap U(j)} (r_{ui} - \mu_i)(r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U(i) \cap U(j)} (r_{ui} - \mu_i)^2} \sqrt{\sum_{u \in U(i) \cap U(j)} (r_{uj} - \mu_j)^2}}$$

- As the number of common users rating the items may be small, the similarity score is often “dampened”

$$s_{ij} = \frac{n_{ij}}{\lambda_2 + n_{ij}} \rho_{ij}$$

- Now ratings for items for the active user are calculated using

$$\widehat{r}_{ak} = \frac{\sum_{i_j \in N(k)} s_{kj} r_{aj}}{\sum_{i_j \in N(k)} s_{kj}}$$



EXAMPLE

Given the ratings matrix

	i₁	i₂	i₃	i₄	i₅	i₆
u₁	4	2	4	3		2
u₂	1		3	4	3	1
u₃	2		3		4	1
u₄			4		4	1
u₅		3		2		4

Using Pearson's correlation between item vectors we can calculate the pairwise similarity of items

	i₁	i₂	i₃	i₄	i₅	i₆
i₁	1	0	0.945	-1	1	0.94
i₂	0	1	0	-1	0	1
i₃	0.94	0	1	-1	0.5	0.816
i₄	-1	-1	-1	1	0	-0.98
i₅	1	0	0.5	0	1	0.5
i₆	0.94	1	0.817	-0.98	0.5	1

Assuming we are looking for movies to recommend to u₄:

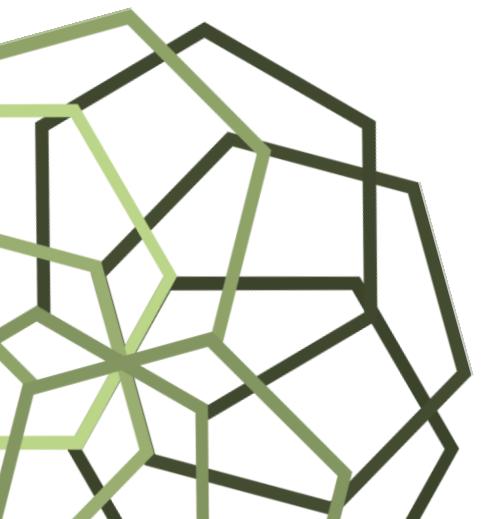
- i₁ = (0.945*4+1*4+0.94*1)/2.885 = 3.02
- i₂ = (1*1)/1 = 1
- i₄ = ?

Neighbourhood based approaches

- How dependent is the predicted rating on the similarity of the neighbours?

$$\widehat{r}_{ak} = \frac{\sum_{i_j \in N(k)} s_{kj} r_{aj}}{\sum_{i_j \in N(k)} s_{kj}}$$

- Weights add up to one, so it's the relative similarity of neighbours that counts



Rating Prediction using Matrix Factorization

	i1	i2	i3	i4	i5	i6
Tim		2	4	3		
Tom			3	4	3	1
Peter	1		3		4	
Paul			4		4	1
Kathy		3		2		4



↓

The diagram illustrates the decomposition of a user rating matrix into two latent factor matrices. A large blue bracket groups the top row of the user rating matrix (labeled i1) with the first two columns of the user latent factor matrix (labeled h1 and h2). A large blue bracket groups the top row of the item latent factor matrix (labeled i1) with the first two columns of the item latent factor matrix (labeled h1 and h2). A blue L-shaped arrow points from the user rating matrix towards the user latent factor matrix, indicating the projection of user ratings onto the latent space defined by user latent factors. Another blue L-shaped arrow points from the item latent factor matrix towards the item latent factor matrix, indicating the projection of items onto the latent space defined by item latent factors.

	h1	h2
Tim		
Tom		
Peter		
Paul		
Kathy		

	h1	h2
i1		
i2		
i3		
i4		
i5		
i6		

Rating Prediction using Matrix Factorization

	i1	i2	i3	i4	i5	i6
Tim		2	4	3		
Tom			3	4	3	1
Peter	1		3		4	
Paul			4		4	1
Kathy		3		2		4

→

	h1	h2
Tim	1.38	1.22
Tom	0.51	1.7
Peter	1.39	0.78
Paul	0.80	1.88
Kathy	2.41	0.46

←

	h1	h2
i1	0.69	0.04
i2	1.13	0.43
i3	1.46	1.43
i4	0.41	2.10
i5	2.09	1.2
i6	1.66	-0.06

	i1	i2	i3	i4	i5	i6
Tim	1.01	2.10	3.78	3.14	4.37	2.21
Tom	0.42	1.32	3.18	3.78	3.11	0.73
Peter	0.99	1.92	3.16	2.22	3.85	2.26
Paul	0.64	1.74	3.88	4.29	3.95	1.21
Kathy	1.69	2.93	4.2	1.98	5.61	3.97

MATRIX FACTORIZATION

- The ratings are a reflection of the user's taste that is driven by certain domain dependant latent factors e.g. plot, actor, cinematography etc.
- Assuming k latent factors, we can approximate the rating matrix

$$R \approx PQ$$

where P and Q are $n \times k$ and $m \times k$ matrices respectively

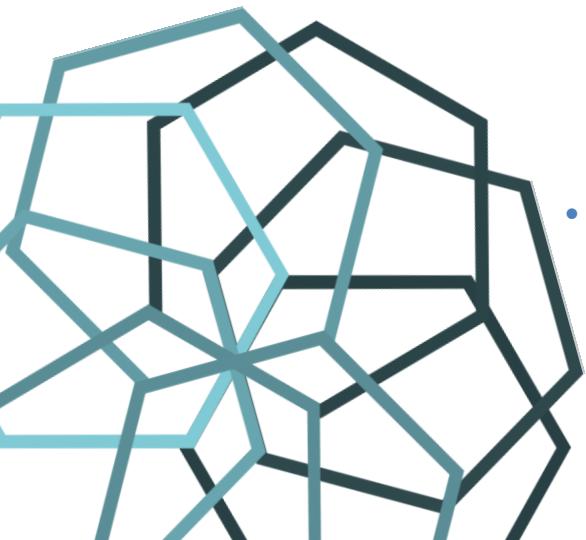
- P is a representation of users based on their 'k' interests
- Q is a representation of movies based on its ability to fulfil the user's preferences
- Typically use an iterative optimization algorithm to learn P and Q such that, they minimize

$$\sum_{ij} (r_{ij} - \hat{r}_{ij})^2 + \frac{\beta}{2} (\|P\|^2 + \|Q\|^2)$$

where, $\beta = 0.02$ and $\hat{r}_{ij} = p_i \cdot q_j^T$

- Multiplying P and Q to obtain an approximation of R provides ratings to items, predicted for a user who had not previously consumed (or at least not rated) the item

Number of Parameters?



```
def MF(X,num_dims,step_size,epochs,thres,lam_da):
    import scipy
    P = scipy.sparse.rand(X.shape[0],num_dims,1,format='csr')
    P=scipy.sparse.csr_matrix(P/scipy.sparse.csr_matrix.sum(P, axis=1))
    Q = scipy.sparse.rand(num_dims, X.shape[1],1,format='csr')
    Q=scipy.sparse.csr_matrix(Q/scipy.sparse.csr_matrix.sum(Q, axis=0))

    prev_error = 0
    for iterat in range(epochs):
        errors = X - make_sparse(P.dot(Q),X.indices,X.indptr)
        mse=np.sum(errors.multiply(errors))/len(X.indices)
        if abs(mse-prev_error) > thres:
            P_new = P+step_size*2*(errors.dot(Q.T)-lam_da*P)
            Q_new = Q+step_size*2*(P.T.dot(errors)-lam_da*Q)
            P = P_new
            Q = Q_new
            prev_error = mse
        else:
            break
    if iterat%1 == 0:
        print(iterat,mse)
    print(P.dot(Q).todense())
    return P,Q
```

Movielens

- Number of Users: 610
- Number of movies: 9724
- Number of Ratings: 100,836

```

0 12.658056670291035
1 12.3679403784389
2 12.054535441993796
3 11.71142099647553
4 11.333958784038623
5 10.919755000620956
.....
.....
98 1.1990709981876604
99 1.1908870413582395

```

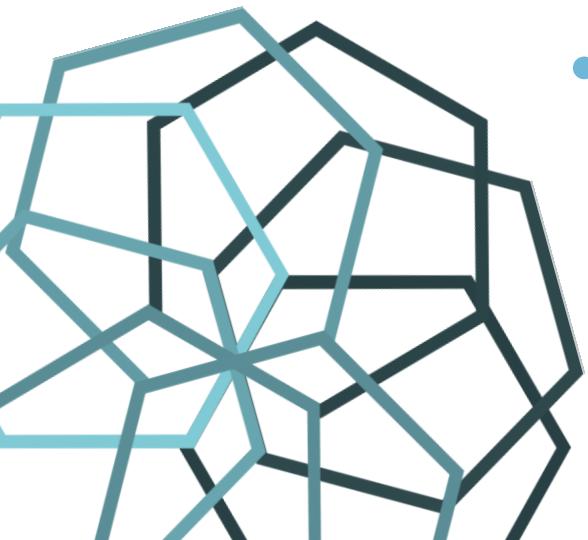
movielens	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585
userId																			
1	4.0	0.0	4.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

movielens	title	genres
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	Jumanji (1995)	Adventure Children Fantasy
3	Grumpier Old Men (1995)	Comedy Romance
4	Waiting to Exhale (1995)	Comedy Drama Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action Crime Thriller
7	Sabrina (1995)	Comedy Romance

[[5.07329863 4.32746854 4.29565133 ... 1.19216286 1.22835445 1.73026716]
[2.45394037 2.09411183 2.08228058 ... 0.56524101 0.57710553 0.83368864]
[1.40258223 1.19868363 1.20233245 ... 0.34040145 0.33131866 0.46761349]
...
[3.98635237 3.44580781 3.41018843 ... 0.94605886 0.94480902 1.3559413]
[2.3257932 1.99572822 1.98339592 ... 0.55158046 0.54639789 0.78277657]
[5.13201075 4.37643347 4.35846042 ... 1.20405442 1.20816191 1.74238842]]

Stochastic Gradient Descent

- Stochastic Gradient Descent
 - Prediction Error for each user, item pair
$$e_{ui} = r_{ui} - q_i^T p_u$$
 - Update P and Q
$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda \cdot p_u)$$
 - Loop through all user, item pairs
- Alternating Least Squares (ALS)
 - q_i and p_u are not known, hence convex cost function
 - Alternately fix q_i and p_u so as to make the cost function quadratic and hence making the cost function convex



Incorporating Biases

- Decomposing the rating

$$r_{ui} = \mu + r_i + r_u + \hat{r}_{ui}$$

Average rating Item bias User bias Interaction rating
How does the user deviate from the average?

Everyone loves Star Wars ($r_i=0.8$)

Derek is a harsh rater ($r_u= -0.4$)

Average rating = 3.4

Derek's interaction with Star Wars contributed 0.5

$$r_{ui} = 3.4 + 0.8 - 0.4 + 0.5 = 4.3$$

How do we learn the biases?



Learning the biases

- Decouple the computation of b_i and b_u

$$b_i = \frac{\sum_{u \in R(i)} (r_{ui} - \mu)}{\lambda_1 + |R(i)|}$$

$$b_u = \frac{\sum_{i \in R(u)} (r_{ui} - \mu - b_i)}{\lambda_2 + |R(u)|}$$

- λ_1 and λ_2 are regularization parameters that shrink the biases towards zero when the number of ratings for an item or ratings by a user is small

- Alternatively, compute b_i and b_u symmetrically by minimizing the cost function

$$b^* = \operatorname{argmin}_{(b_i, b_u)} \sum_{(u, i) \in \mathcal{K}} (r_{ui} - \mu - b_i - b_u)^2 + \lambda \left(\sum_u b_u^2 + \sum_i b_i^2 \right)$$

Number of Parameters?



Adding in the user item interaction term

- The predicted rating is $\widehat{r}_{ui} = \mu + b_i + b_u + p_u q_i^T$
 - Assuming k dimensions used in decomposing the rating matrix, the number of unknown parameters in this model are $(n+m)(k+1)$
- The Cost function is defined as

$$b^* = \underset{(b_i, b_u)}{\operatorname{argmin}} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \mu - b_i - b_u - p_u q_i^T)^2 + \lambda \left(\sum_u b_u^2 + \sum_i b_i^2 + \|P\|^2 + \|Q\|^2 \right)$$

- Update Rules

$$\begin{aligned} e_{ui} &= r_{ui} - \widehat{r}_{ui} \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda \cdot b_i) \\ b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda \cdot b_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda \cdot q_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda \cdot p_u) \end{aligned}$$



λ is the learning rate (step size) and γ is the regularization constant

Reducing the number of Parameters

- Typically the number of users is much larger than the number of items e.g. Netflix Prize data contained 480,189 users and 17,770 movies with a total of 100,480,507 ratings
- Patarek proposed modelling the user as a linear combination of item vectors

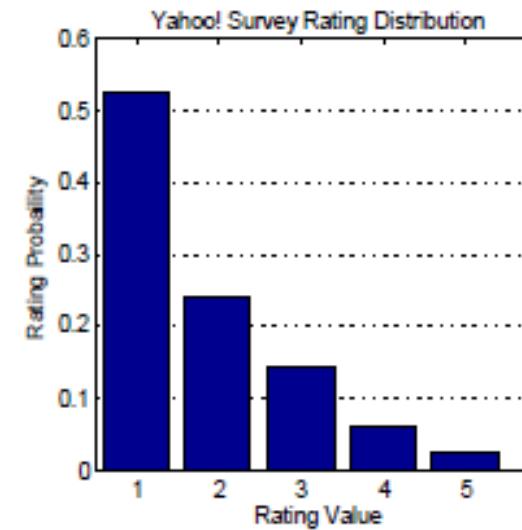
$$p_u = \frac{1}{\sqrt{1 + |I(u)|}} \sum_{i_j \in I(u)} y_j$$

- Number of parameters for the matrix factorization part is now $O(mk)$ instead of $O(n+m)k$

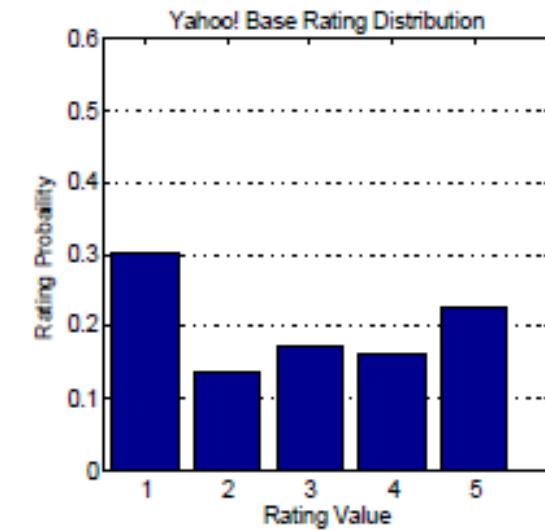


Not Missing At Random: The Bias in Rating Data

- Missing at Random means that the rating of an item has no bearing on the probability that it is going to be missing
- The complete Rating matrix is hidden and some process has hidden a large proportion of the ratings
- If the missing ratings are not missing at random it is important to understand the process that caused the rating to be missing



(a) Yahoo! Random



(b) Yahoo! User

RECOMMENDATION AS SUBSET SELECTION

- While accurately predicting ratings is one way to achieve improved recommendation
- It is more important to get the ranking of items correct
 - There is only limited real estate to push recommendations to
 - Being accurate in rating a large number of items that are not interesting to the user does not impact perceived value of the recommender
- An alternative problem statement
 - Given a set of n items, and a user u_a who has consumed m ($<< n$) items select N such that the ratings of these items by a user will be high
- Also referred to as Top-N recommendation



EVALUATION METRICS FOR TOP-N

- Withhold one rating at a time and predict it
 - If the test item exists in the recommendation list $L(u)$, it is referred to as a hit and the Top-k hit rate is defined as
$$HR = \frac{\#hits}{|L(u)|}$$
 - All hits are equal (no credit given for the rank achieved by the item), Average Reciprocal Hit Rank is defined as
$$ARHR = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|T(u)|} \sum_{i_j \in T(u)} \frac{1}{rank(i, L(u))}$$
- Alternatively, a set of items per user are withheld from training and precision and recall are computed given the recommendations, $L(u)$ of size k , for each user

$$precision@k = \frac{1}{|U|} \sum_{u \in U} \frac{|L(u) \cap T(u)|}{|L(u)|}$$

Note: $k = |L(u)|$

$$recall@k = \frac{1}{|U|} \sum_{u \in U} \frac{|L(u) \cap T(u)|}{|T(u)|}$$



Adapting Matrix Factorization to TOP-N Recommendation

- Cost function adapted to account for the fact that most missing values are uninteresting
 - Missing rating are imputed (parameter) and set to r_m
 - A weight (parameter) is assigned to each rating to account for sparsity and the fact that most ratings are missing

$$w_{ui} = \begin{cases} 1 & \text{if observed rating} \\ w_m & \text{if missing rating} \end{cases}$$

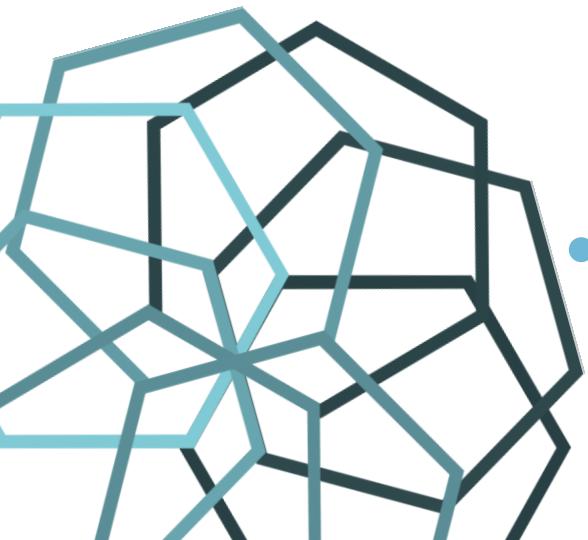
- Cost function defined over all items (not just the observed)

$$\sum_{u \in U} \sum_{i \in I} w_{ui} \left\{ (r_{ui} - (r_m + p_u q_i^T))^2 + \lambda \left(\sum_d p_{ud}^2 + q_{id}^2 \right) \right\}$$

- Hyper-Parameters, r_m , w_{ui} , and λ are set using cross validation to maximize recall@k (Top-k HR)

Restricted Boltzmann Machine

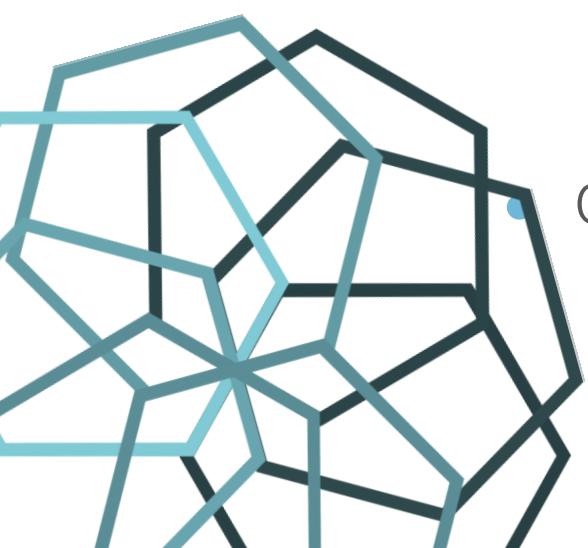
- Restricted Boltzmann Machine is a stochastic neural network
 - One layer of visible units
 - Ratings for movies are fed through the visible layer
 - $|I| \times r$ units
 - Netflix winning team used a binary input (rated/not rated) rather than the actual rating
 - One layer of hidden units
 - Parameter representing the number of hidden factors
 - A bias (always on)
 - Each visible unit is connected to all hidden units (undirected edge – symmetrically weighted connections)



Restricted Boltzmann Machine

- Due to the sparse rating matrix, each user is modelled using a different RBM
 - Same number of hidden units
 - Visible nodes is equal to the number of movies rated by the user
 - One training example per RBM, V , a $r \times m$ binary matrix (here $m = |I(u)|$)
 - Weights and biases are tied together (shared across user RBMs)
- Conditional multinomial distribution models each movie rating

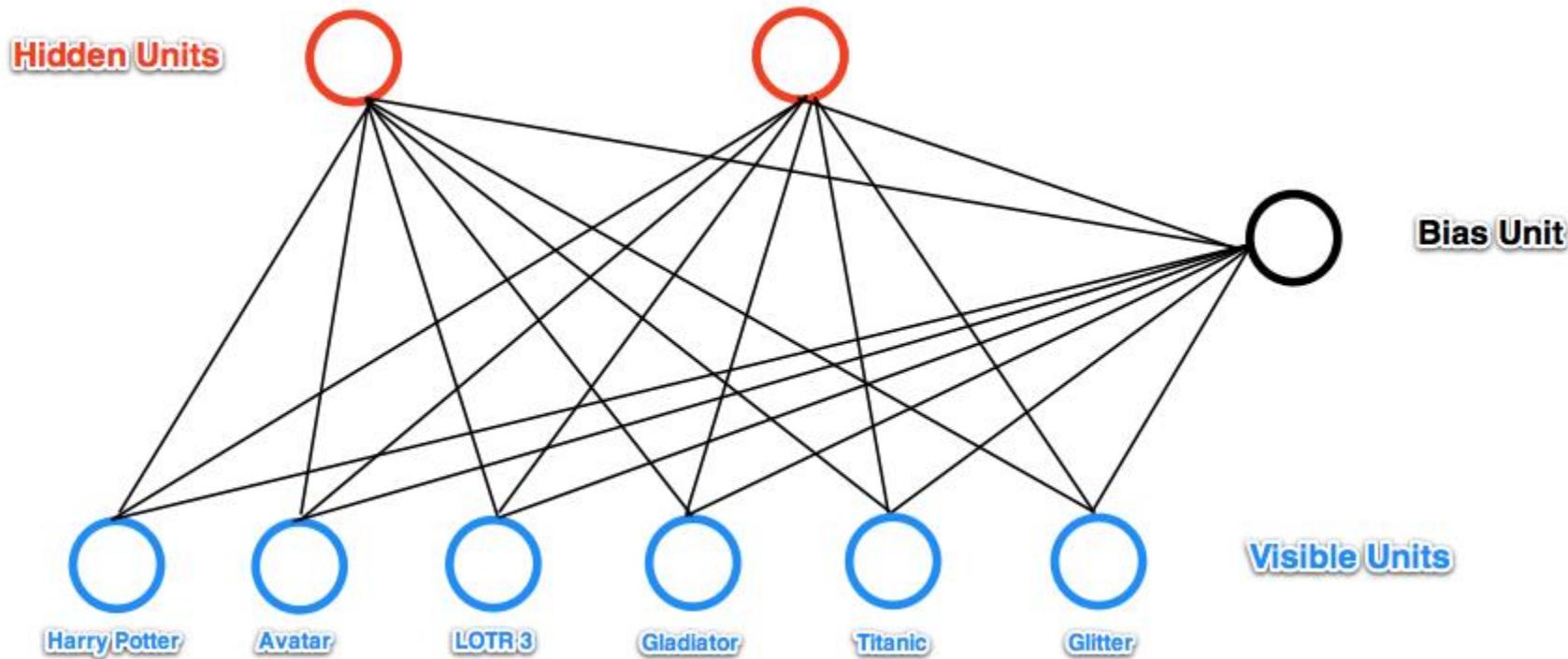
$$p(v_i^k = 1 | h) = \frac{\exp(b_i^k + \sum_{j=1}^K W_{ij}^k h_j)}{\sum_{l=1}^F \exp(b_i^l + \sum_{j=1}^F W_{ij}^l h_j)}$$



Conditional Bernoulli distribution models a hidden unit

$$p(h_j = 1 | V) = \sigma\left(b_j + \sum_{i=1}^m \sum_{k=1}^K v_i^k W_{ij}^k\right)$$

Restricted Boltzmann Machine



$$p(V) = \sum_h \frac{\exp(-E(V, h))}{\sum_{V', h'} \exp(-E(V', h'))} \quad \text{where} \quad E(V, h) = - \sum_{i=1}^m \sum_{j=1}^F \sum_{k=1}^K W_{ij}^k h_j v_i^k - \sum_{i=1}^m \sum_{k=1}^K v_i^k b_i^k - \sum_{j=1}^F h_j b_j$$

Gradient Ascent on $\log p(V)$ to learn W

Contrastive Divergence

- Take a training example, Set the states of the visible units to these
- Next, update the states of the hidden units using the logistic activation rule described above:
 - for the j th hidden unit, compute its activation energy $a_j = b_j + \sum_i w_{ij}v_i$
 - set h_j to 1 with probability $\sigma(a_j)$ and to 0 with probability $1-\sigma(a_j)$
 - Then for each edge e_{ij} , compute $\text{Positive}(e_{ij})=v_i \cdot h_j$ (i.e., for each pair of units, measure whether they're both on)
- Now **reconstruct** the visible units in a similar manner:
 - for each visible unit, compute its activation energy $a_i = b_i + \sum_j w_{ij}h_j$
 - set v_i to 1 with probability $\sigma(a_i)$ and to 0 with probability $1-\sigma(a_i)$
 - Then update the hidden units again and compute $\text{Negative}(e_{ij})=v_i \cdot h_j$ for each edge.
- Update the weight of each edge e_{ij} by $w_{ij} = w_{ij} + \lambda(\text{Positive}(e_{ij}) - \text{Negative}(e_{ij}))$
- Repeat over all training examples.

CONTENT BASED FILTERING

- Recommend items that are similar to items previously consumed by the user
- Similarity measurement can get complex
 - Simple approach is to “vectorize” the content

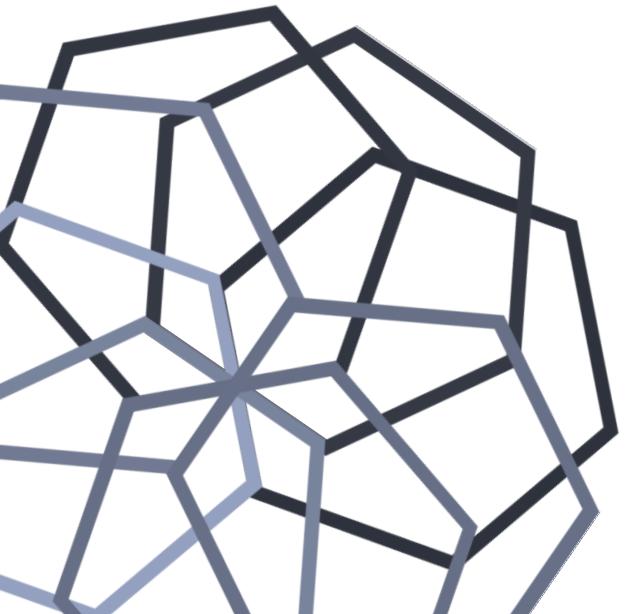
बीजेपी विधायक फिर बोर्लीं, गरबा पंडालों में नहीं आए मुसलमान

पहले भी रहा विवादित बयानों से विधायक का नाता

पिछले साल नवरात्र में दुर्गा पूजा पंडालों में मुस्लिम युवाओं के प्रवेश को बैन करने का बयान देकर सुर्खियों में रही मध्यप्रदेश के इंदौर की बीजेपी विधायक अपने विवादास्पद बयान से फिर सुर्खियों में है।

बीजेपी विधायक ने मुसलमानों के खिलाफ एक बार फिर बवाल खड़ा कर देने वाला बड़ा बयान दिया है। विधायक ने बकरीद को लेकर विवादित टिप्पणी कर डाली।

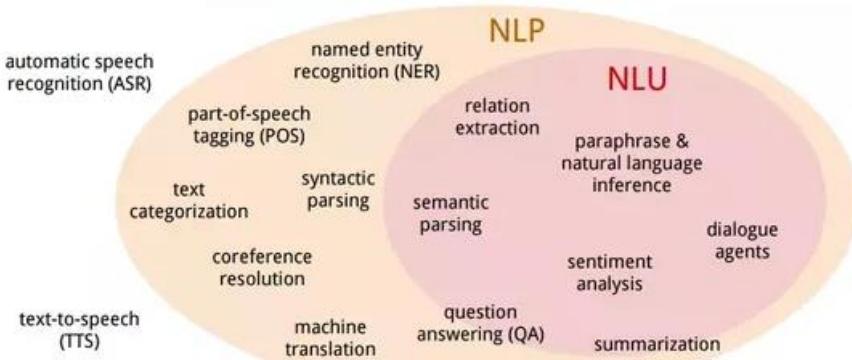
ठाकुर ने कहा कि मुस्लिम्स को अगर कुर्बानी ही देनी है तो सांकेतिक कुर्बानी दें। जानवरों की कुर्बानी देने का उन्हें अधिकार नहीं है। वहीं उन्होंने अपने पुराने बयान को दोहराते हुए फिर कहा कि दुर्गा पूजा में मुसलमानों के जाने पर रोक लगाई जानी चाहिए।



Why is the article of interest to the consumer?

- Is he a BJP (or Usha Thakur) supporter?
- Is he from Indore?
- Is he a Muslim or Hindu?
- Is he concerned about the rising levels of communal rhetoric in the country?

.....



- Documents can be hyperlinked
- Links typically are not random



Documents

- have different lengths
- contain sequence of words selected from a vocabulary
- Authored by people with different writing styles and vocabularies



Need to convert documents to vectors

- Ideal Representation for allowing retrieval of relevant documents for a user query
- How can I model documents so that I can generate new ones that have similar statistical properties to those that humans write
- Symbols are unwieldy
- How can I find numeric (continuous) representations for symbols so I can manipulate them and create accurate models

Tfidf, LSA

PLSA LDA Maximum Likelihood
Hidden Markov Models
Conditional Random Fields

Word Embedding,
Recursive Neural Nets

Documents as vectors

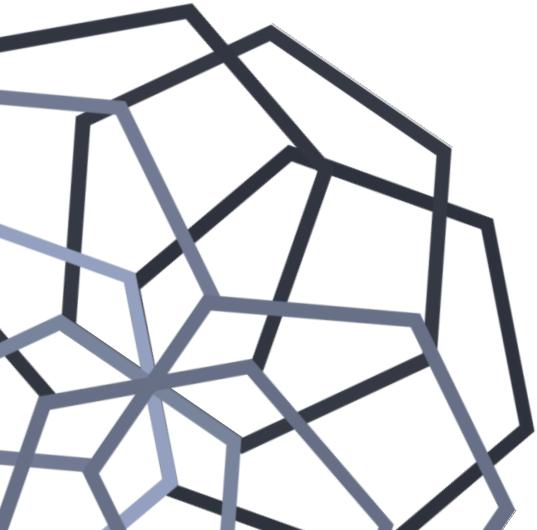
- The Vector Space Model
 - Each document is represented as a vector of weights
 - Weight represents how much the term, t_k , contributes to the semantics of the document
 - Each term (typically a word or unigram) in a predefined vocabulary, V , is a dimension of this space
 - Also referred to as Bag of Words
 - Ignore ordering of words
 - “the poor man ate the food” and “the man ate the poor food” have the same bag of words representation
- Typical processing of text
 - Tokenize (Words) and remove punctuation
 - stem words
 - Use heuristics to collapse multiple words to the same root e.g. *organize*, *organizes*, and *organizing* get mapped to *organiz*
 - remove stop words
 - Stop words are words that do not typically contribute to the semantics of the document e.g. a, an, the, and, but...

REPRESENTING ARTICLES AS VECTORS

- Preprocess the article
 - Tokenize
 - Lemmatize/Stem
 - Remove Stop Words
- Derive weights for each token
 - Represent importance of token in describing the document

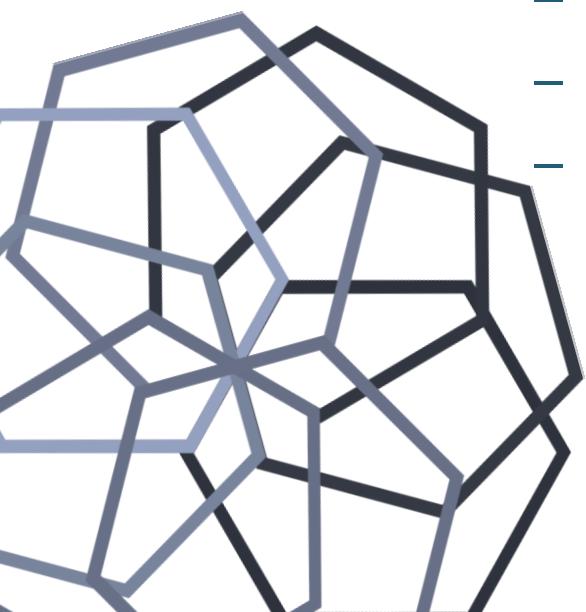
$$tfidf(w, d) = \frac{n_d(w)}{|d|} \ln \frac{N}{df(w)}$$

- Each document is now represented as an n-dimensional vector of weights
- Also known as the Bag of Words representation as we loose token sequencing information
- The above is a “Syntactic Representation” of a document



EXAMPLE

- Suppose the following eight texts are in our corpus
 - d1:Human Machine interface for ABC computer applications
 - d2:A survey of user opinion of computer system response time
 - d3:The EPS user interface management system
 - d4:System and human system engineering testing of EPS
 - d5:Relation of user perceived response time to error measurement
 - d6:The generation of random, binary, ordered trees
 - d7:The intersection graph of paths in trees
 - d8:Graph minors IV: Width of trees and well-quazi-ordering



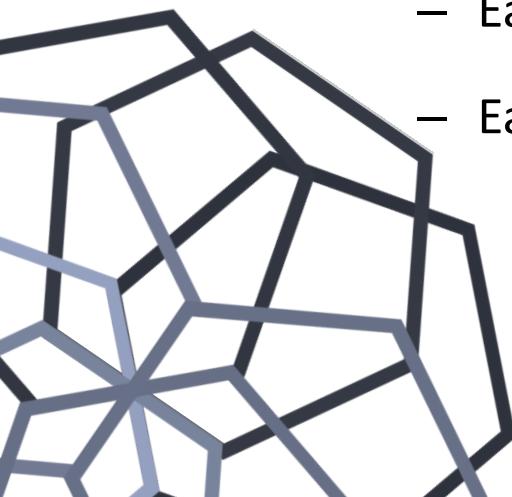
EXAMPLE: BAG OF WORDS

	d1	d2	d3	d4	d5	d6	d7	d8	df
Human	1	0	0	1	0	0	0	0	2
Interface	1	0	1	0	0	0	0	0	2
Computer	1	1	0	0	0	0	0	0	2
User	0	1	1	0	1	0	0	0	3
System	0	1	1	2	0	0	0	0	3
response	0	1	0	0	1	0	0	0	2
Time	0	1	0	0	1	0	0	0	2
EPS	0	0	1	1	0	0	0	0	2
Survey	0	1	0	0	0	0	0	0	1
Trees	0	0	0	0	0	1	1	1	3
Graph	0	0	0	0	0	0	1	1	2
minors	0	0	0	0	0	0	0	1	1

$$tfidf(\text{human}, d1) = \frac{1}{6} \ln\left(\frac{8}{2}\right) = 0.23$$

Feature Extraction: Latent Semantic Analysis

- Assumes some latent semantic space obscured by randomness of word choice e.g. human vs user
 - Lexical level (“what was said”) Vs Semantic level (“what was meant”)
 - Method for removing “noise” in the “semantic signal”
 - Estimating the hidden concept space (associates syntactically different but semantically equivalent terms/ documents)
- Given an $m \times n$ matrix X consisting of the word frequency counts for m words in n documents
- A singular value decomposition of X results in three matrices U , S and V^T
 - The singular values in S provide a basis for reducing the dimensionality of U and V by choosing the r largest values
 - Each row, u_i , in U is a representation of a word in r -dimensional space
 - Columns (right singular vectors) are the Eigen vectors of XX^T
 - Each row, v_i , in V is a representation of a document in r -dimensional space
 - Columns (left singular vectors) are the Eigen vectors of X^TX



SVD Example

U

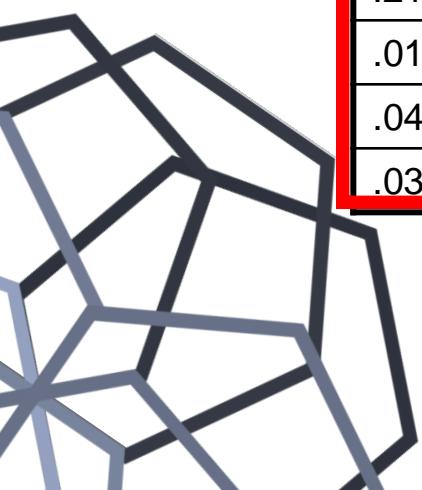
.22	-.11	.29	-.41	-.11	-.34	.52	-.06	-.41
.2	-.07	.14	-.55	.28	.5	-.07	-.01	-.11
.24	.04	-.16	-.59	-.11	-.25	-.3	.06	.49
.4	.06	-.34	.1	.33	.38	0	0	.01
.64	-.17	.36	.33	-.16	-.21	-.17	.03	.27
.27	.11	-.43	.07	.08	-.17	.28	-.02	-.05
.27	.11	-.43	.07	.08	-.17	.28	-.02	-.05
.30	-.14	.33	.19	.11	.27	.03	-.02	-.17
.21	.27	-.18	-.03	-.54	.08	-.47	-.04	-.58
.01	.49	.23	.03	.59	-.39	-.29	.25	-.23
.04	.62	.22	0	-.07	.11	.16	-.68	.23
.03	.45	.14	-.01	-.3	.28	.34	.68	.18

S

3.34									
	2.54								
		2.35							
			1.64						
				1.5					
					1.31				
						.85			
							.56		
								.36	

V^T

.2	.61	.46	.54	.28	0	.01	.02	.08
-.06	.17	-.13	-.23	.11	.19	.44	.62	.53
.11	-.5	.21	.57	-.51	.1	.19	.25	.08
-.95	-.03	.04	.27	.15	.02	.02	.01	-.03
.05	-.21	.38	-.21	.33	.39	.35	.15	-.6
-.08	-.26	.72	-.37	.03	-.3	-.21	0	.36
.18	-.43	-.24	.26	.67	-.34	-.15	.25	.04
-.01	.05	.01	-.02	-.06	.45	-.76	.45	-.07
-.06	.24	.02	-.08	-.26	-.62	.02	.52	-.45



LSA Example (Reconstruction with 2 dimensions)

	1	2	3	4	5	6	7	8	9
Human	1	0	0	1	0	0	0	0	0
Interface	1	0	1	0	0	0	0	0	0
Computer	1	1	0	0	0	0	0	0	0
User	0	1	1	0	1	0	0	0	0
System	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
Time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
Survey	0	1	0	0	0	0	0	0	1
Trees	0	0	0	0	0	1	1	1	0
Graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

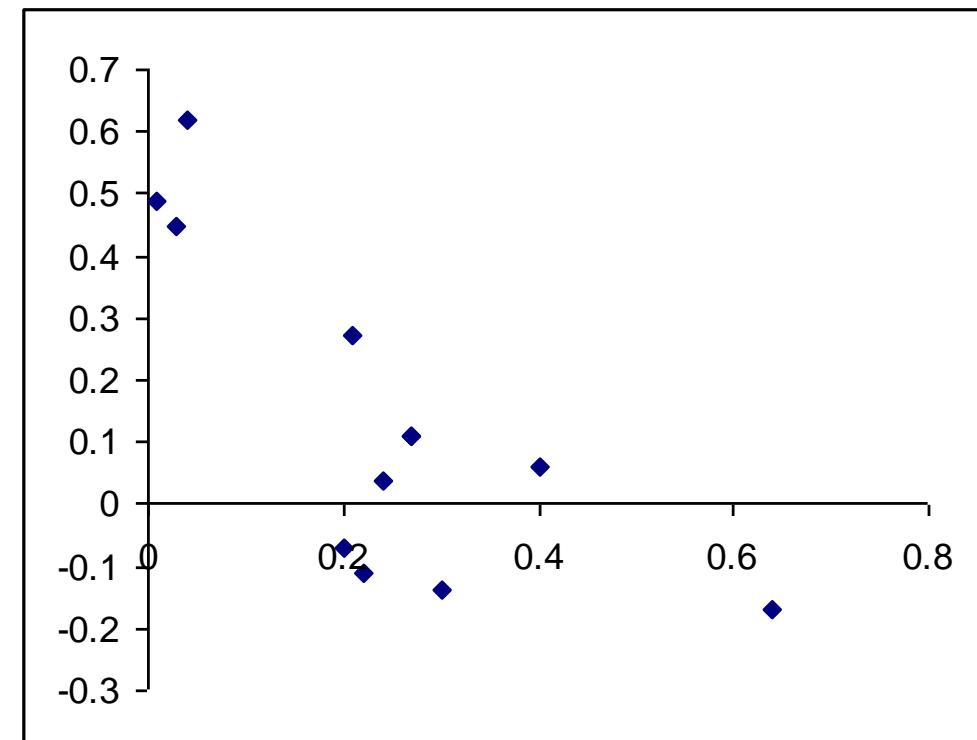
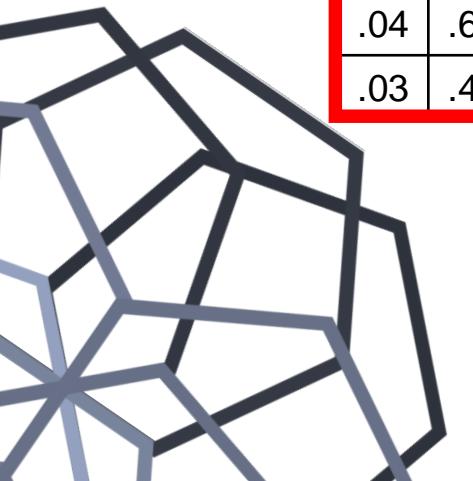
	1	2	3	4	5	6	7	8	9
Human	.16	.4	.38	.47	.18	-.05	-.12	-.16	-.09
Interface	.14	.37	.33	.4	.16	-.03	-.07	-.1	-.04
Computer	.15	.51	.36	.41	.24	.02	.06	.09	.12
User	.26	.84	.61	.7	.39	.03	.08	.12	.19
System	.45	1.23	1.05	1.27	.56	-.07	-.15	-.21	-.05
response	.16	.58	.38	.42	.28	.06	.13	.19	.22
Time	.16	.58	.38	.42	.28	.06	.13	.19	.22
EPS	.22	.55	.51	.63	.24	-.07	-.14	-.2	.11
Survey	.1	.53	.23	.21	.27	.14	.31	.44	.42
Trees	-.06	.23	-.14	-.27	.14	.24	.55	.77	.66
Graph	-.06	.34	-.15	-.3	.2	.31	.69	.98	.85
minors	-.04	.25	-.1	-.21	.15	.22	.5	.71	.62

The matrix resulting from the reduced number of dimensions is the optimal approximation (with respect to least squares error) of the original matrix X

Word Similarity

U

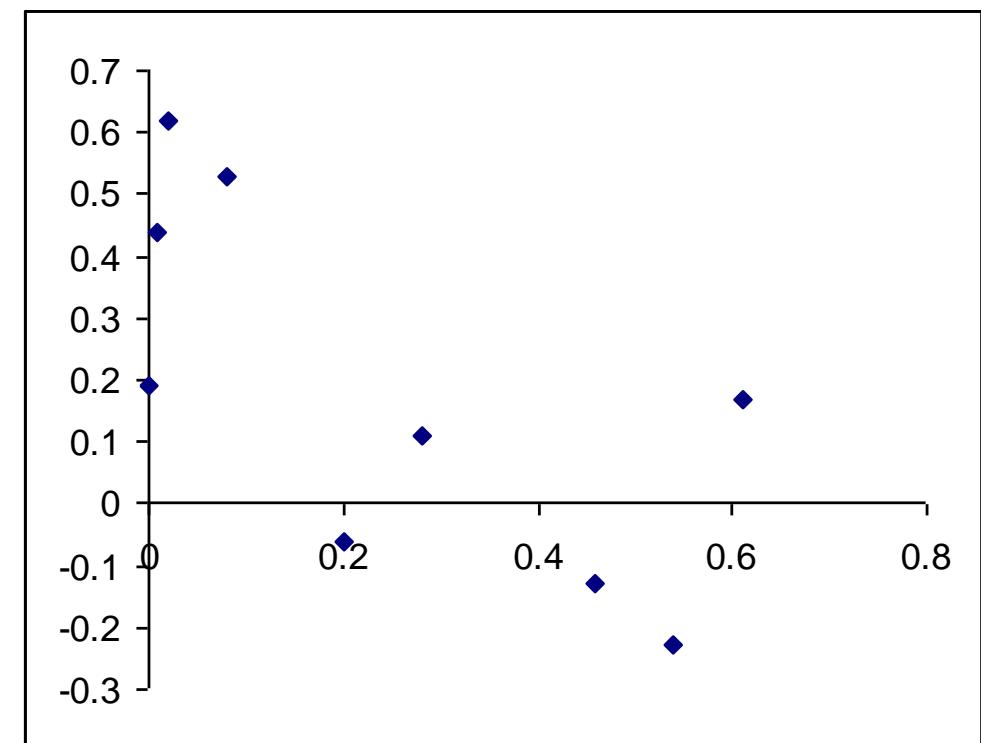
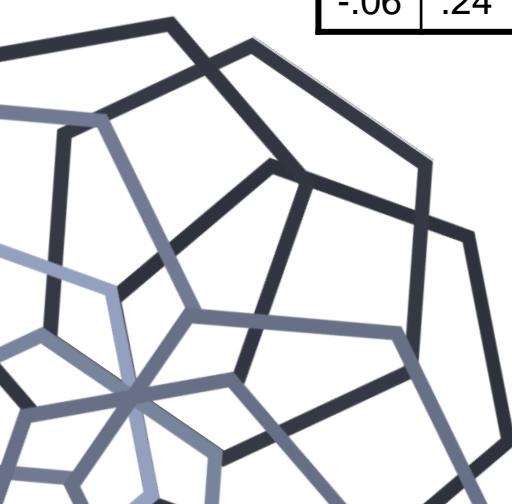
.22	-.11	.29	-.41	-.11	-.34	.52	-.06	-.41
.2	-.07	.14	-.55	.28	.5	-.07	-.01	-.11
.24	.04	-.16	-.59	-.11	-.25	-.3	.06	.49
.4	.06	-.34	.1	.33	.38	0	0	.01
.64	-.17	.36	.33	-.16	-.21	-.17	.03	.27
.27	.11	-.43	.07	.08	-.17	.28	-.02	-.05
.27	.11	-.43	.07	.08	-.17	.28	-.02	-.05
.30	-.14	.33	.19	.11	.27	.03	-.02	-.17
.21	.27	-.18	-.03	-.54	.08	-.47	-.04	-.58
.01	.49	.23	.03	.59	-.39	-.29	.25	-.23
.04	.62	.22	0	-.07	.11	.16	-.68	.23
.03	.45	.14	-.01	-.3	.28	.34	.68	.18



Document Similarity

V^T

.2	.61	.46	.54	.28	0	.01	.02	.08
-.06	.17	-.13	-.23	.11	.19	.44	.62	.53
.11	-.5	.21	.57	-.51	.1	.19	.25	.08
-.95	-.03	.04	.27	.15	.02	.02	.01	-.03
.05	-.21	.38	-.21	.33	.39	.35	.15	-.6
-.08	-.26	.72	-.37	.03	-.3	-.21	0	.36
.18	-.43	-.24	.26	.67	-.34	-.15	.25	.04
-.01	.05	.01	-.02	-.06	.45	-.76	.45	-.07
-.06	.24	.02	-.08	-.26	-.62	.02	.52	-.45

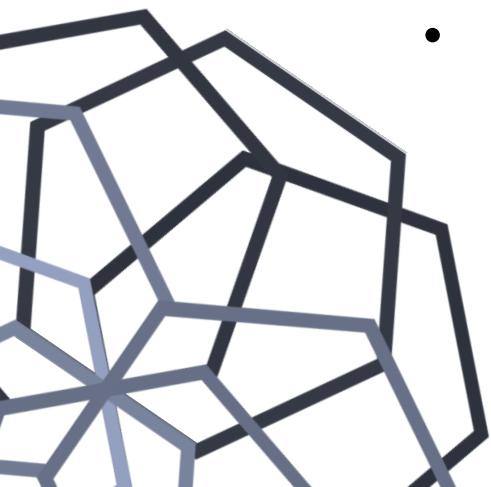


Probabilistic Models of Documents

- Generative Model
 - Assume that the corpus has been generated from a mixture model parameterized by θ
 - A mixture model is a probability distribution defined by a linear combination of individual probability distributions (known as components) C_j
 - Each C_j is parameterized by $\theta_j \subseteq \theta$
 - Generally assume a one-to-one correspondence between components and classes of documents

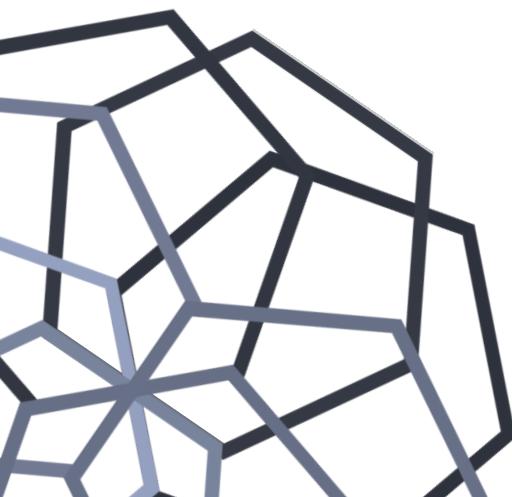
$$p(d_i|\theta) = \sum_{j=1}^k p(d_i|C_j)p(C_j)$$

- Two models for computing $P(d_i|C_j;\theta)$ (assuming conditional independence of words given a class)
 - Multi-variate Bernoulli Model
 - Multinomial Distribution Model



Bernoulli Distribution

- Bernoulli Trail is an experiment whose outcome is
 - Random
 - Binary (success or failure)
- A Bernoulli process consists of repeatedly performing independent and identical (i.i.d) Bernoulli trails
- Bernoulli Distribution is a Discrete Probability Distribution
 - Takes a value of 1 (success) with probability θ and a value 0 (failure) with probability $1-\theta$
 - Probability mass function defined as



Multivariate Bernoulli Document Model

- Documents in D are described by vocabulary, V
- Each document, d_i , is defined by a binary vector of dimensionality $|V|$
 - Dimension t of the vector d_i , d_{it} , is set to 0 if the word, $w_t \in V$, does not appear in the document else it is set to 1
 - $P(d_i | c_j; \theta)$ can then be calculated as

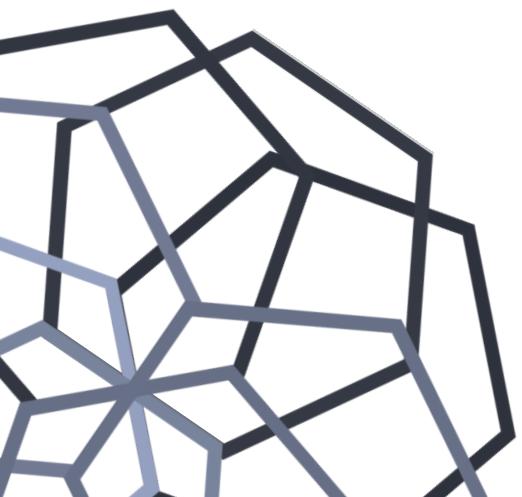
$$p(d_i | C_j) = \prod_{t=1}^{|V|} p(w_t | C_j; \theta)^{d_{it}} (1 - p(w_t | C_j; \theta))^{(1-d_{it})}$$

- $P(w_t | c_j; \theta)$ is estimated from the data as

$$p(w_t | C_j; \theta) = \frac{1 + \sum_{i=1}^{|D|} d_{it} p(C_j | d_i)}{2 + \sum_{i=1}^{|D|} p(C_j | d_i)}$$

- Prior Probabilities, $P(c_j; \theta)$ are estimated as

$$p(C_j | \theta) = \frac{\sum_{i=1}^{|D|} p(C_j | d_i)}{|D|}$$

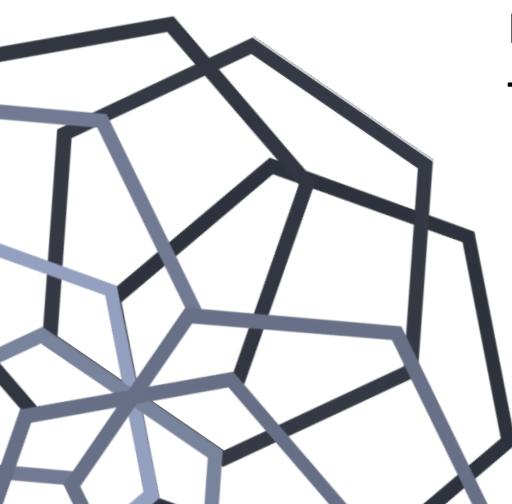


Multinomial Distribution

- Generalization of the binomial distribution
 - Probability distribution of the number of successes in ‘n’ i.i.d Bernoulli trials

$$p(r; n, p) = \frac{n!}{r!(n-r)!} p^r (1-p)^{1-r}$$

- Each trial results in one of k outcomes with probabilities p_1, p_2, \dots, p_k
- Given n trials the probability distribution followed by the random variable $X=(X_1, X_2, \dots, X_k)$, where X_i is the number of times that the i-th outcome is observed is defined as


$$p(x_1, x_2, \dots, x_n; n, p_1, p_2, \dots, p_n) = n! \prod_{i=1}^k \frac{p_i^{x_i}}{x_i!}$$

Multinomial Document Model

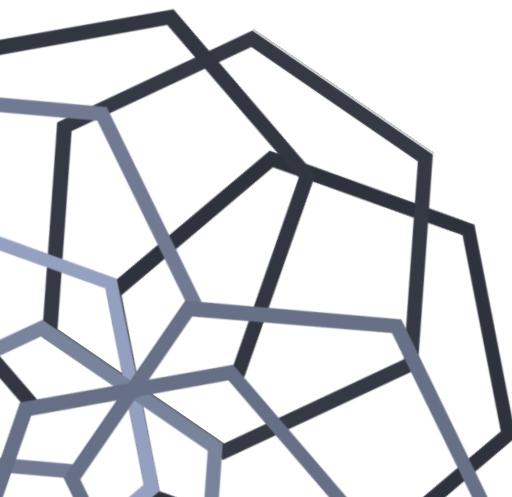
- Captures word frequency
- A document, d_i , is represented as a sequence of word events
 - $P(d_i|c_j; \theta)$ can then be calculated as

$$p(d_i|c_j; \theta) = p(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{p(w_t|c_t; \theta)^{N_{it}}}{N_{it}!}$$

- Note, the equation assumes that document length is independent of the class, though this is not necessary
- $P(w_t|c_j; \theta)$ is estimated from the data as

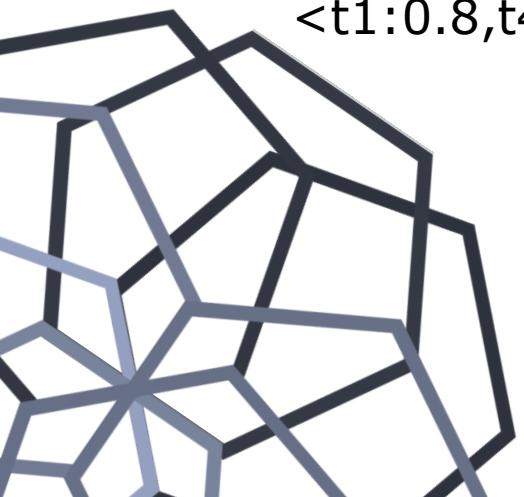
$$p(w_t|C_j; \theta) = \frac{1 + \sum_{i=1}^{|D|} N_{it} p(C_j|d_i)}{|V| + \sum_{t=1}^{|V|} \sum_{i=1}^{|D|} N_{it} p(C_j|d_i)}$$

- Laplace Smoothing (a special case of Lidsone smoothing, $\lambda=1$) is applied in this estimation



TOPIC MODELS

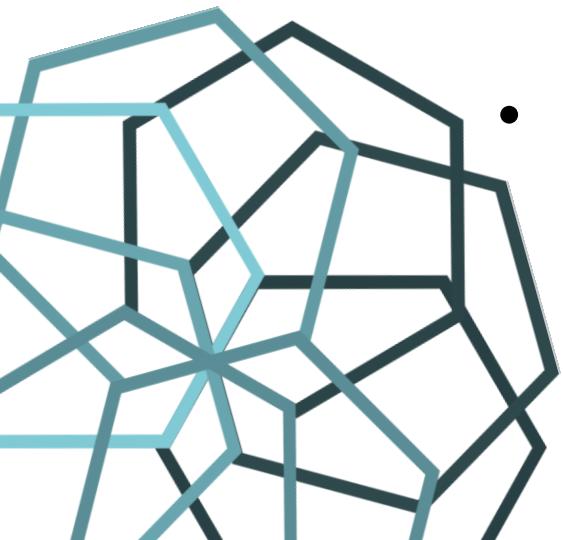
- A language has a vocabulary
- Words from the vocabulary are chosen to form a document, d , of length $|d|$
 - Words are chosen by the author. Different authors would write the same article differently.
- The words in a document reflect a mix of *unobserved* topics
- Each topic defines a probability distribution over the words in the vocabulary
 - The word “cricket” is more likely to appear in a document about sports than in a document about cooking
 - D1: <bat: 0.3,drive:0.1,umpire:0.05,tendulkar:0.2,lawyer:0.4> and D2: <kohli:0.3,australia:0.2,warner:0.3,accused:0.2,sue:0.1> may have a similar Topic representation with a high Topic weight for topic 1 below, for example <t1:0.8,t4:0.2>



Topic 1		Topic 2		Topic 3	
term	weight	term	weight	term	weight
game	0.014	space	0.021	drive	0.021
team	0.011	nasa	0.006	card	0.015
hockey	0.009	earth	0.006	system	0.013
play	0.008	henry	0.005	scsi	0.012
games	0.007	launch	0.004	hard	0.011

Latent Dirichlet Allocation

- Probabilistic model of a corpus that
 - Assigns high probability to members of the corpus; and
 - Assigns high probability to other “similar” documents
- Uses one parameter more than the mixture of unigrams
 - α is k -dimensional as opposed to the $k-1$ prior probabilities of the classes
- Documents are represented as mixtures over latent topics



The generative process

- Length of the document: How many words?
- What is the document about?
 - Topic Distribution (t_1, t_2, \dots, t_k) such that

$$\sum_{t=1}^k t_k = 1$$

Length: 14 words

T1: 0.6 T2: 0.3 T7: 0.1

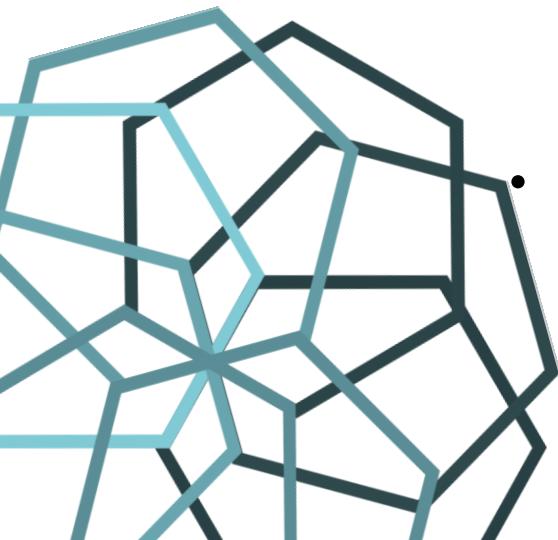
	T1		T2		T7	
kohli	0.016	law	0.018	violent	0.02	
cover	0.011	lawyer	0.016	scuffle	0.019	
batsman	0.011	interpret	0.015	assault	0.018	
pavilion	0.010	testimony	0.015	anger	0.017	

- What are the words?
 - Drawn from, $P(V|t)$, where V is the vocabulary
 - a multinomial distribution of words for each topic

- Note

- No consideration to Grammar
- Positions of words in the document irrelevant

kohli court over
wicket pitch tea
bowler lawyer
anger pavilion
teammates case
assault testimony



A tangential question:

How many Martians will buy milk?

- What is the probability of finding a transaction that contains milk, p ?
 - p itself is a random variable
 - So what is the distribution of p ?
 - What is our prior belief in its value?

- Beta Distribution

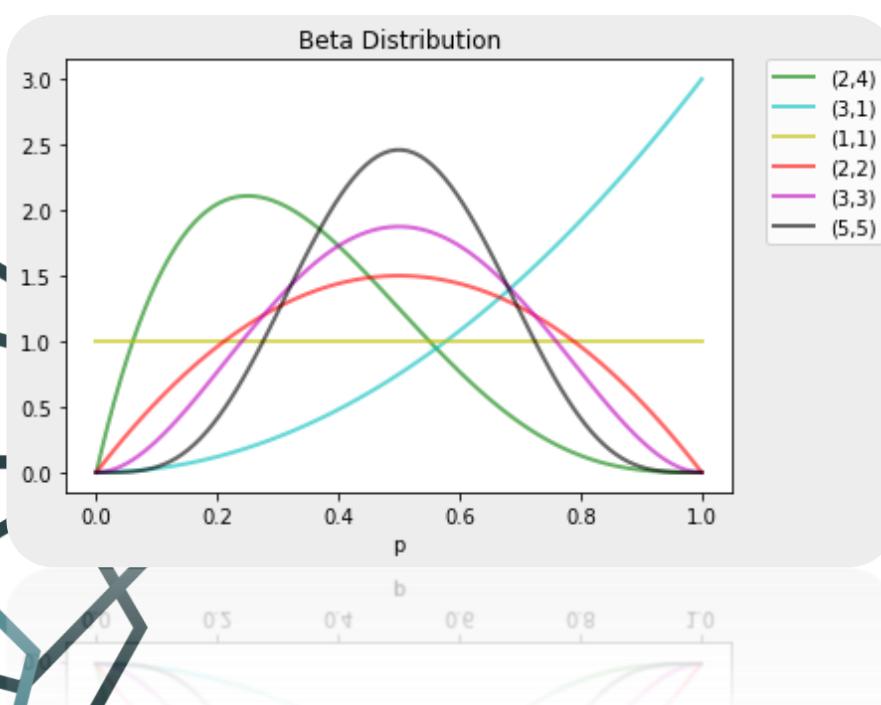
- Two shape parameters α, β

$$f(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

- Note that $x \in [0,1]$
- Intuitively, α and β are the ‘imaginary’ counts of the two outcomes that lead to our prior belief in x

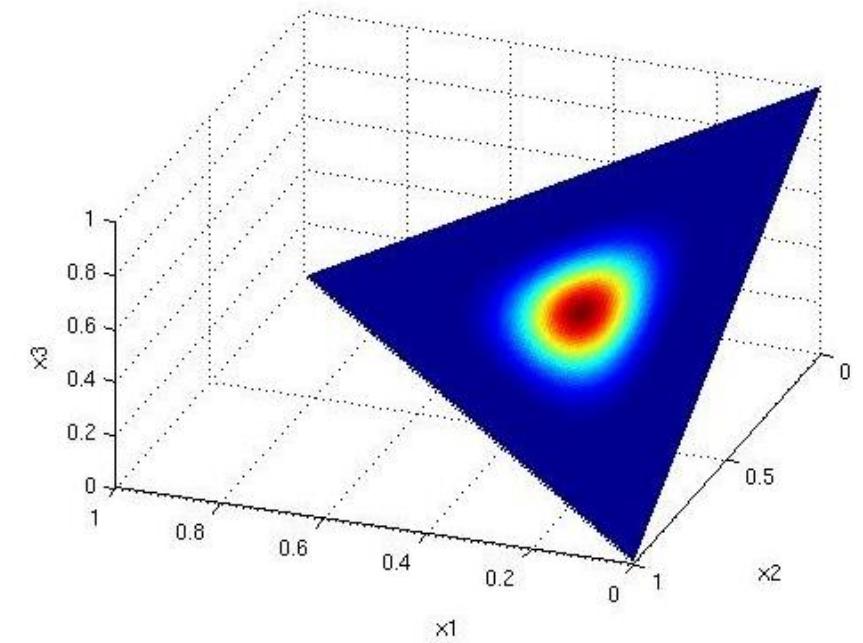
Given some Data, D containing a milk purchases our of $n..$

$$p(\text{milk}|D) = \frac{a + \alpha}{n + \alpha + \beta}$$



Generating the topic vector for a document

- To generate a documents we need a way to generate k-tuples
 - Each element represents the proportion of words generated from a particular topic
- The Dirichlet Distribution
 - Generalization of the beta distribution
 - Parameterised by k-vector, α



LDA: Generative Model

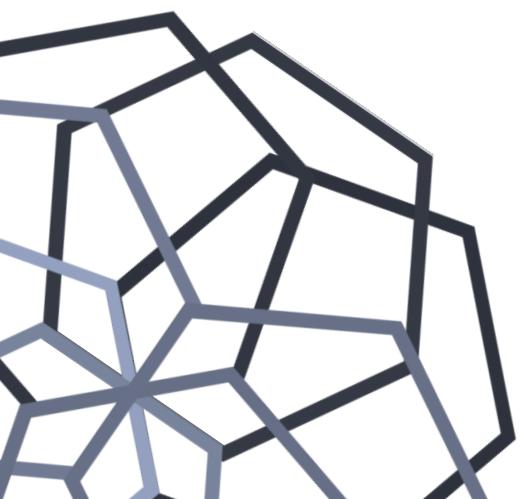
- Choose N (length of the document) using a Poisson Distribution

$$f(N; \lambda) = \frac{\lambda^N e^{-\lambda}}{N!}$$

- Choose θ using a k dimensional Dirichlet Distribution (α)
 - k is the number of topics
 - Γ is the gamma function

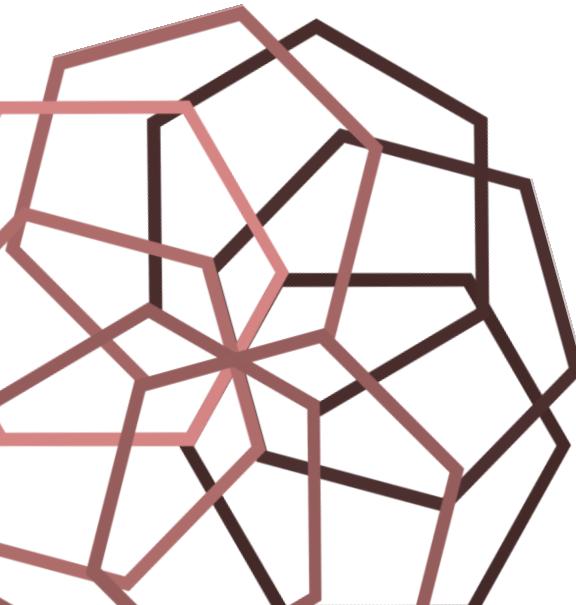
$$p(\theta|\alpha) = \frac{\Gamma \sum_{i=1}^k \alpha_i}{\prod_{i=1}^k \Gamma(\alpha_i)} \theta_1^{\alpha_1-1} \theta_2^{\alpha_2-1} \dots \theta_k^{\alpha_k-1}$$

- Choose each of the N words, w_n , in the document by
 - Choosing a topic z_n using the multinomial distribution define by the parameters θ
 - Choose a word w_n from the multinomial distribution $p(w_n|z_n, \beta)$
 - β is a $k \times |V|$ matrix where the i -th row consists of the probability values for selecting each word given z_i



LATENT VARIABLES, TOPIC MODEL AND SEMANTIC REPRESENTATIONS

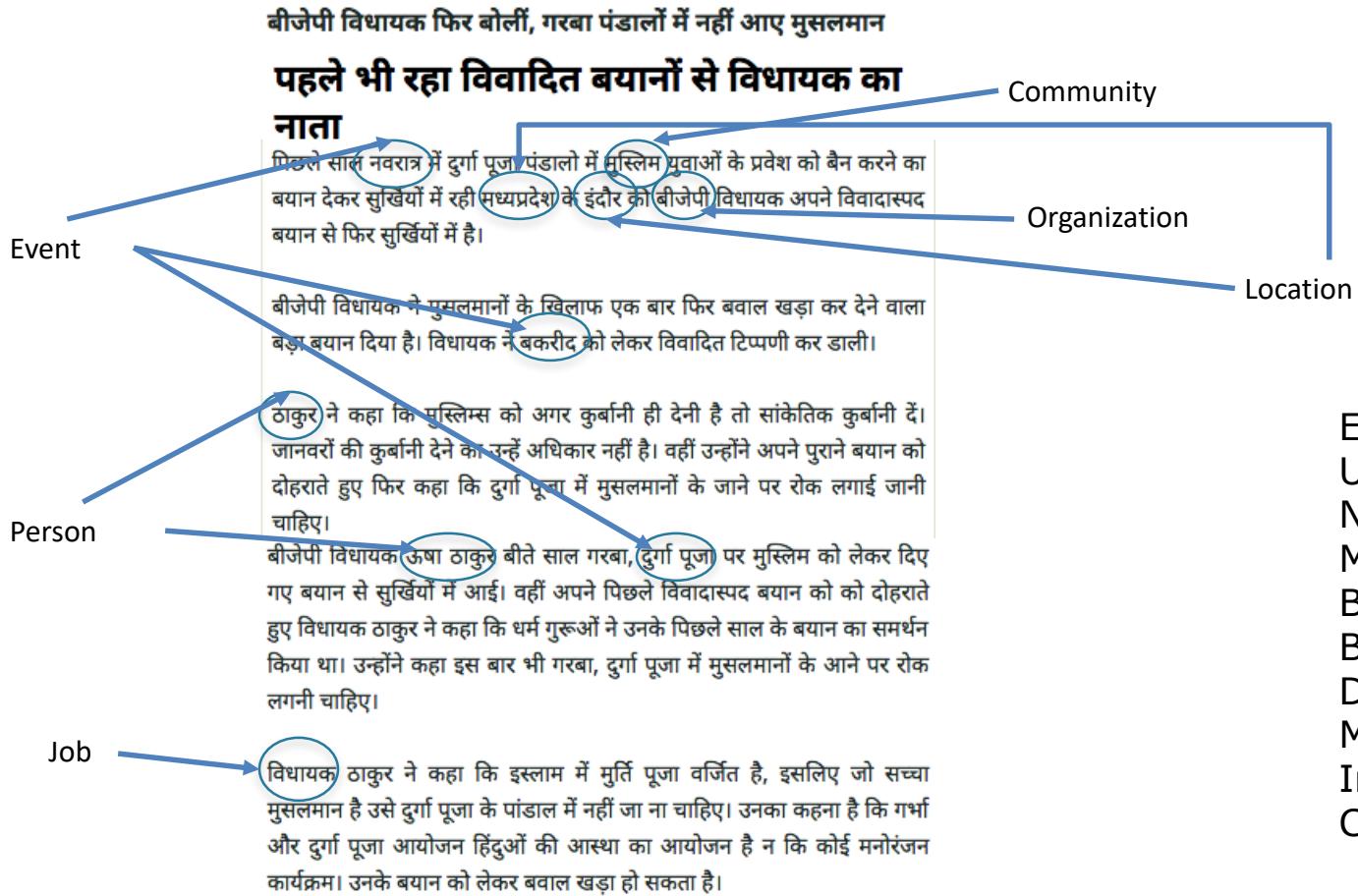
- A language has a vocabulary
- Words from the vocabulary are chosen to form a document, d , of length $|d|$
 - Words are chosen by the author. Different authors would write the same article differently.
- The words in a document reflect a mix of *unobserved (latent)* topics
- Each topic defines a probability distribution over the words in the vocabulary



	Government	Unions	Agricultural	Pricing	Cricket	Weather		
सरकार	0.012	सरकार	0.01		टीम	0.022	मौसम	0.011
कर्मचारियों	0.009	गेहूं	0.009		मैच	0.011	आँधी	0.009
प्रदर्शन	0.008	रुपये	0.008		क्रिकेट	0.01	गर्मी	0.009
मांग	0.008	कोर्ट	0.007		खेल	0.008	बारिश	0.009
अध्यक्ष	0.008	किसानों	0.007		विकेट	0.007	तूफान	0.009
धरना	0.007	केंद्र	0.007		खिलाफ	0.005	तेज	0.008
आंदोलन	0.06	खरीद	0.006		मुकाबले	0.004	रात	0.007

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
D0	0	0	0	0.19	0.16	0.36	0	0	0.28	0	0
D1	0.07	0.19	0.34	0.38	0	0	0	0	0	0	0
D2	0.21	0.1	0	0	0	0.68	0	0	0	0	0

CONTENT BASED FILTERING



Extracted Entities & Tags:

Usha Thakur
Navratri
Muslim
BJP
Bakr Id
Durga Puja
Madhya Pradesh
Indore
Controversial Communal Rhetoric

CONTENT SIMILARITY

- Given a news story (d_i) with
 - tfidf vector t_i
 - Topic vector T_i
 - Feature Vector f_i
- Similarity between two news stories can be computed as

$$sim(d_1, d_2) = \alpha sim(t_1, t_2) + \beta sim(T_1, T_2) + (1 - \alpha - \beta) sim(f_1, f_2)$$

Example:

D1: The situation in Sirsa is tense as the army was called in to maintain law and order.

$t_1:<\{\text{army}:0.69\}, \{\text{called}:0.48\}, \{\text{law}:0.72\}, \{\text{maintain}:0.51\}, \{\text{order}:0.64\}, \{\text{situation}:0.55\}, \{\text{Sirsa}:0.77\}, \{\text{tense}:0.64\}>$
 $T_1:<\text{Law and Order}: 0.8, \text{Haryana}:0.2>$
 $f_1:<\text{location:sirsa}:1>$

D2: The curfew imposed by the armed forces stopped the violent clashes in Boston.

$t_2:<\{\text{armed}:0.69\}, \{\text{clash}:0.6\}, \{\text{curfew}:0.77\}, \{\text{force}:0.62\}, \{\text{impose}:0.69\}, \{\text{stop}:0.49\}, \{\text{violent}:0.66\}, \{\text{Boston}:0.77\}>$
 $T_2:<\text{Law and Order}: 0.8, \text{USA}:0.2>$
 $f_2:<\text{location:Boston}:1>$

CONTENT SIMILARITY

- Standard approach to computing similarity between vectors is cosine similarity

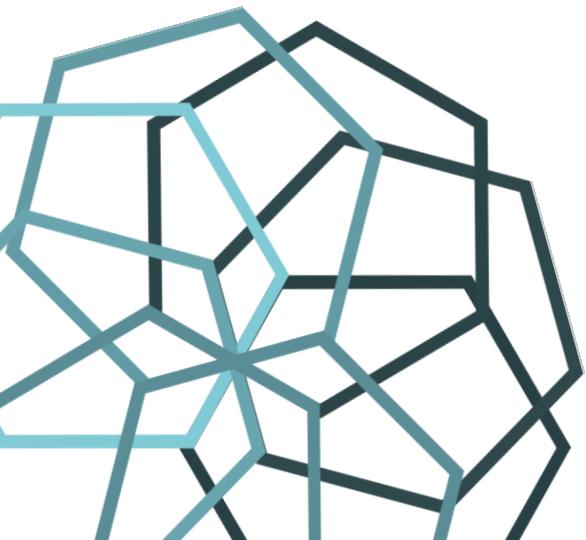
$$\cos\theta = \frac{\mathbf{v}_a \cdot \mathbf{v}_b}{|\mathbf{v}_a| |\mathbf{v}_b|}$$

- $\text{sim}(\mathbf{t}_1, \mathbf{t}_2) = 0$
- $\text{sim}(\mathbf{T}_1, \mathbf{T}_2) = \frac{0.8 \times 0.8}{\sqrt{0.64 + 0.04}\sqrt{0.64 + 0.04}} = 0.35$
- $\text{sim}(\mathbf{f}_1, \mathbf{f}_2) = 0$
- If $a = 0.3$ and $b = 0.3$ then $\text{sim}(\mathbf{D}_1, \mathbf{D}_2) = 0.105$



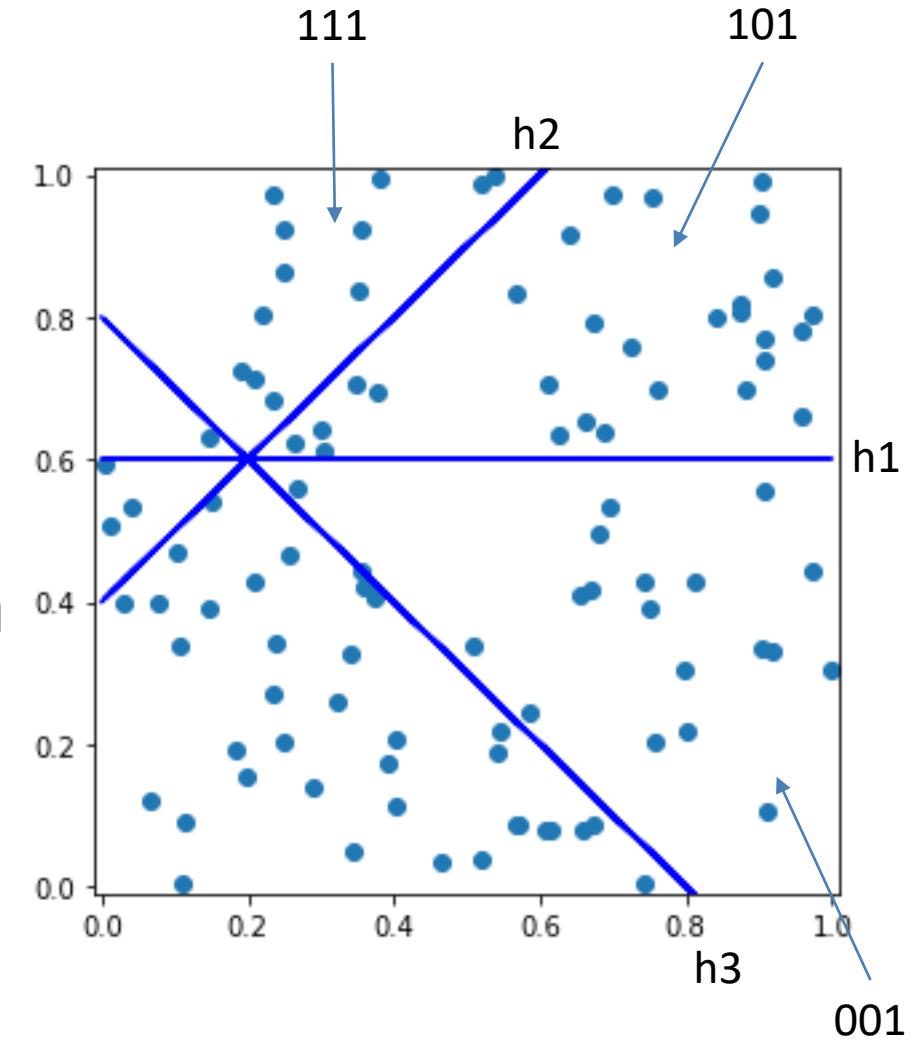
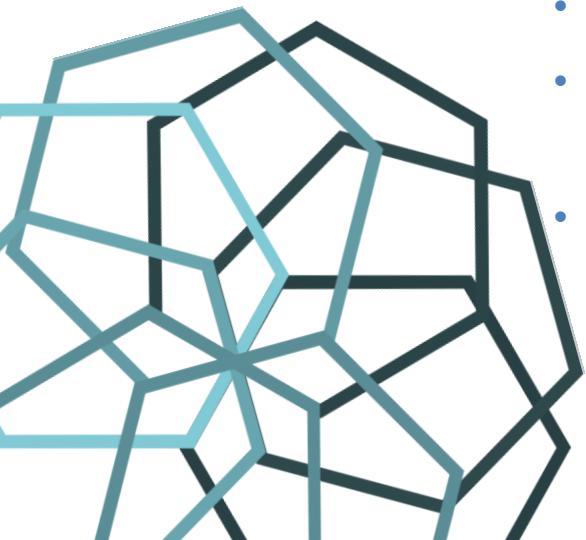
Locality Sensitive Hashing

- Finding similar documents is required in a number of applications
 - Content based filtering
 - Nearest Neighbour based classification (authorship attribution)
 - Plagiarism detection
- Finding similar documents is $O(n)$
- Solution
 - Hash documents into buckets based on their similarity
 - To find the nearest neighbours of the new document
 - Assign a new document to a bucket
 - Search only those documents that are in the same bucket as new documents
 - Quality of hashing defined by
 - Number of false positives (number of non-neighbours that end up in the same bucket as the new document)
 - Number of false negatives (number of neighbours that end up in a different bucket)



Hashing documents

- Creating a hash function (of k bits)
 - Generate Random hyperplanes that cut the instance space into two
 - Assign a bit 1 to one side of each hyperplane and 0 to the other
 - Note that some “neighbours” are in a different bucket so will not be retrieved
- Solution
 - Creating multiple hash functions
 - For the target document, find all buckets it belongs to and take the union of these buckets, $B=UB_i$
 - Search of neighbours in the union of all buckets, B



How likely are we to miss neighbours?

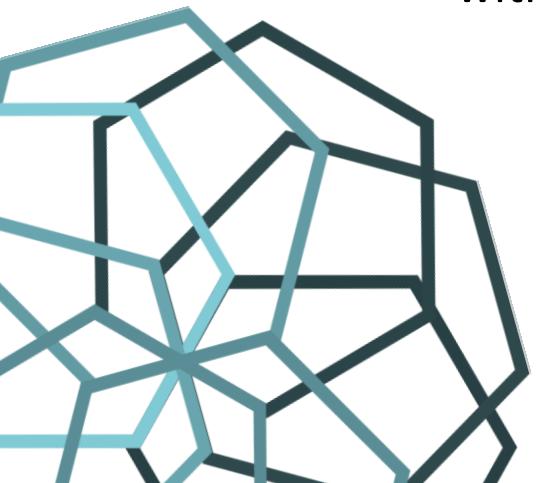
- Computational Cost
 - Assuming a d-dimensional vector, $O(kd)$
 - Nearest neighbour search in a bucket
 - Average number of documents in each bucket is $n/2^k$
 - Hence $O(dn/2^k)$
 - Assuming L buckets
 - $O(kd + dn/2^k)L = O(\log N)$ if $k \approx \log N$
- Probability to a false Negative (Neighbour that does not get hashed within the same bucket as the target document)

$$\prod_{i=1}^L p(h_i(a) \neq h_i(b)) = p(h_i(a) \neq h_i(b))^L$$

$$= (1 - p^K)^L = (1 - \left(1 - \frac{2\theta}{\pi}\right)^K)^L$$

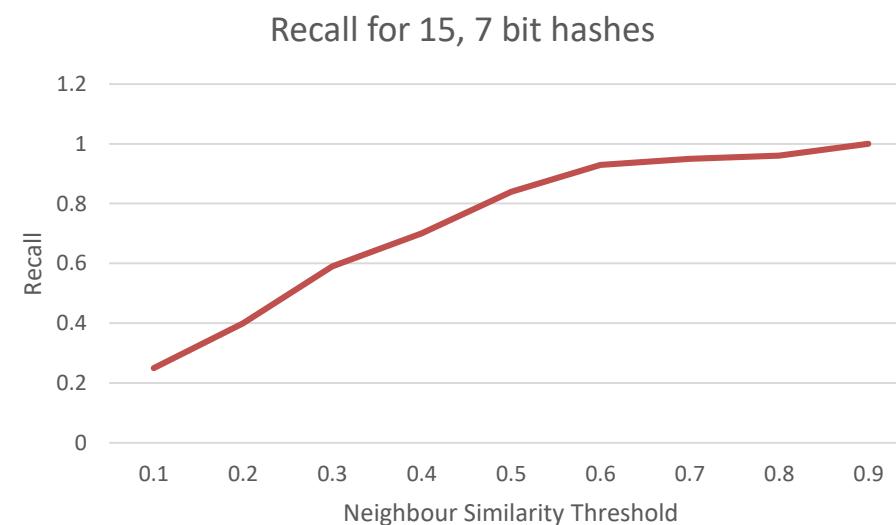
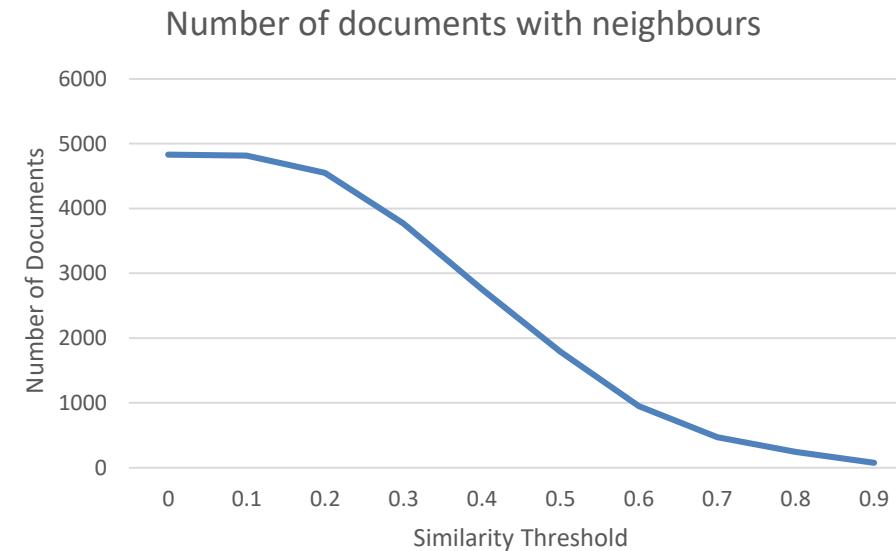
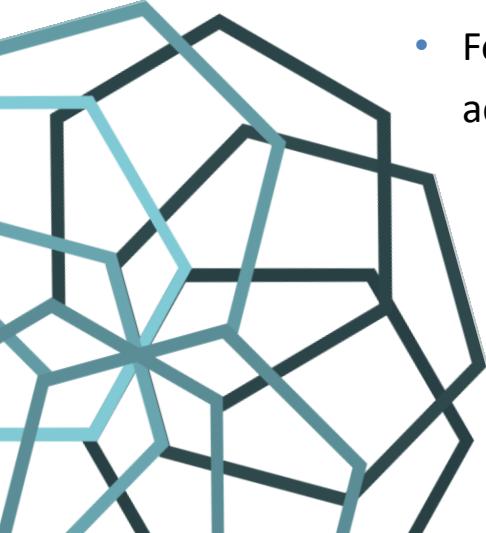
Probability that a bit is the same

$$\theta = \arccos(a^T b)$$



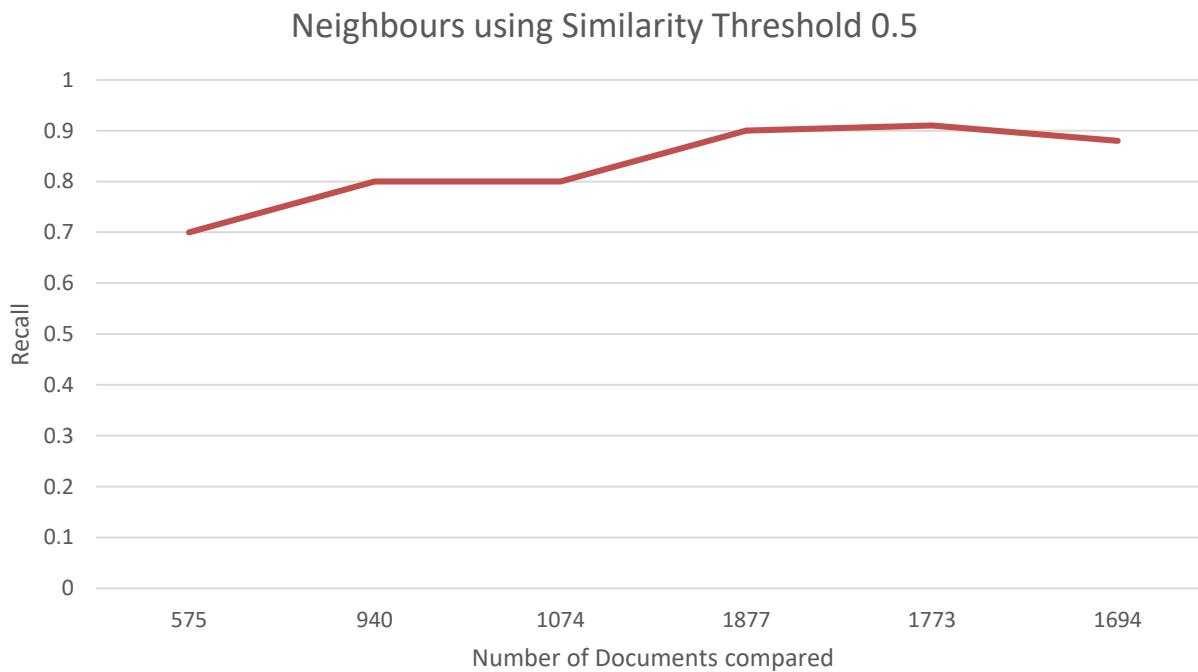
Evaluation

- Data consists of 4830 news articles
- Evaluation Metric: Recall based on neighbour similarity threshold
- 15 hash functions, each consisting of 7 planes used to partition the space of documents (128 buckets)
 - Average number of documents compared: 580 (app. 12%)
 - For thresholds ≥ 0.6 , recall > 0.9 can be achieved

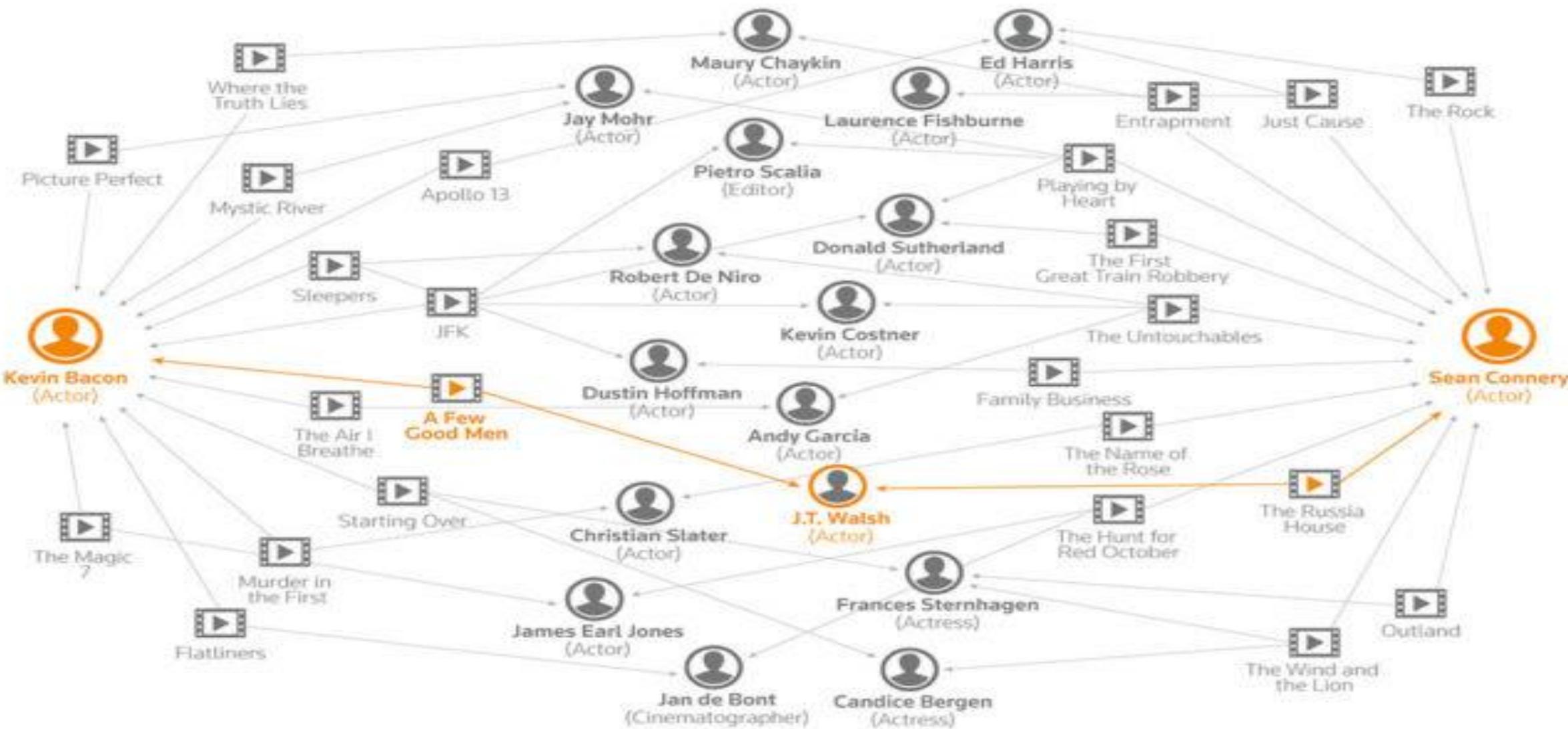


Evaluation

- Recall of 0.9 can be achieved with lower thresholds of neighbour similarity using fewer buckets and hash functions
 - Results in a larger number of documents to be compared (app. 30% of documents to achieve 0.9 recall when neighbours are defined as those with similarity > 0.5)



REPRESENTING CONTENT AS A GRAPH

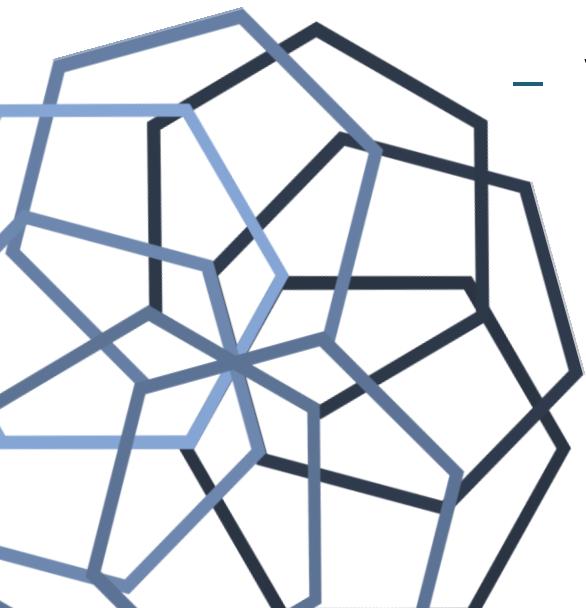


CREATING A USER PROFILE

- Represent the user in the same vector space as items
- Rocchio's algorithm
 - Borrowed from information retrieval literature on relevance feedback

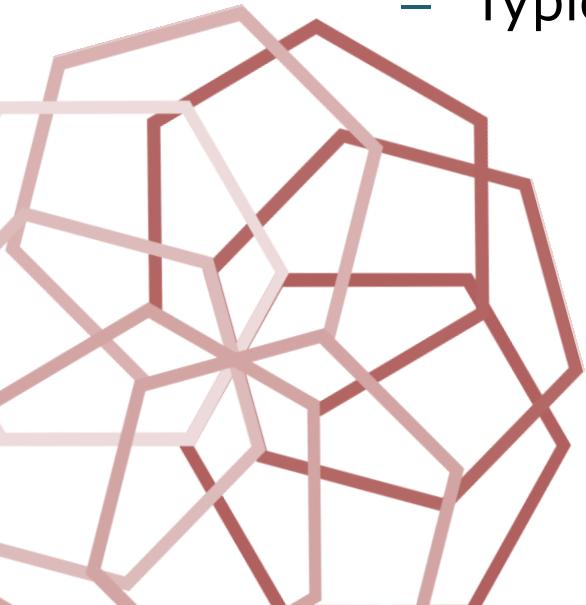
$$u_a = \sum_{i_j \in I_a} r_{uj} i_j$$

- Variants assign more weight recent articles

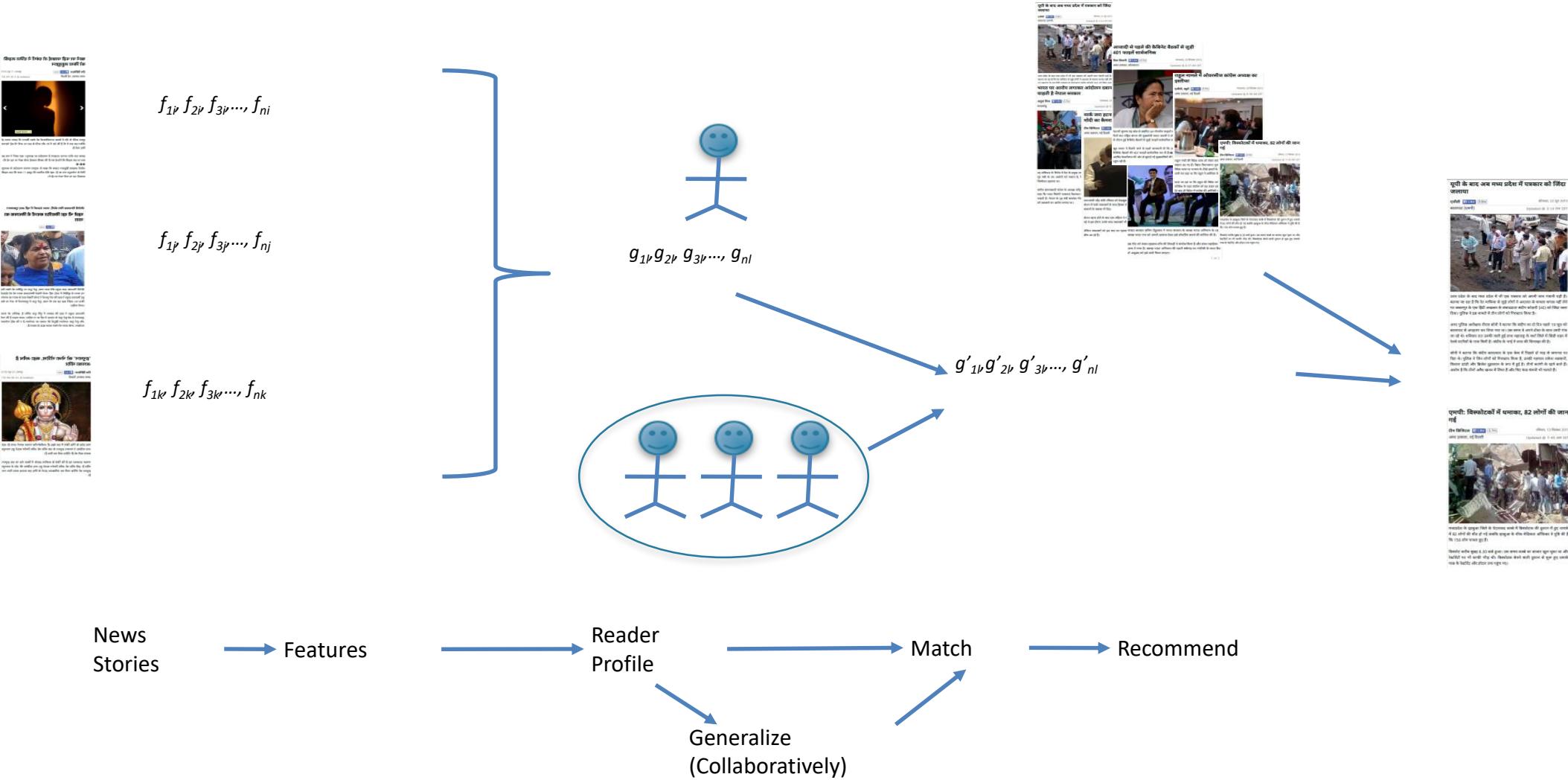


CONTENT BASED FILTERING

- Pros
 - New Items can be recommended immediately to the existing customer base
 - No data needed other than item descriptions
 - Does not depend on how “run of the mill” a user is
- Cons
 - Typically results in narrow focus (No “widening the horizons” of the user)



BACK TO NEWS RECOMMENDATION

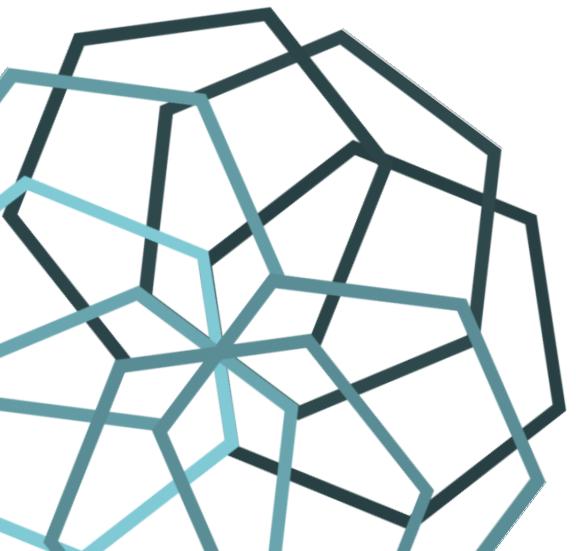


Recommending: A continuum

- When should we begin personalization?
 - Content Based: Too specific too soon
 - Collaborative: Poor Neighbourhood Identification

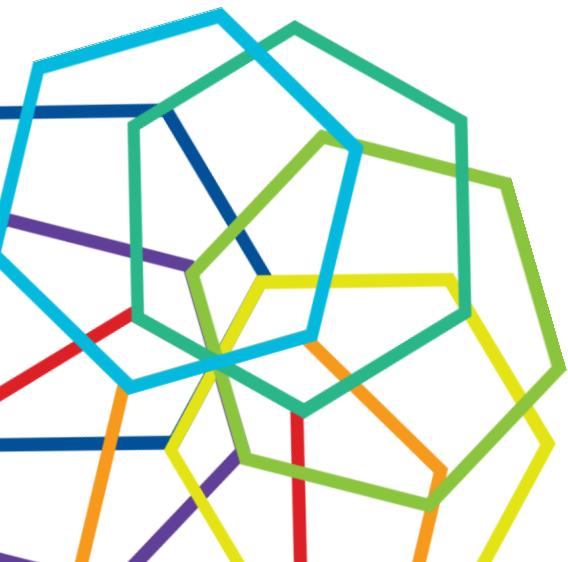


- Note these are individual weights, specific to the user
 - w2 is based on the strength of the signal from the target user
 - w1 is based on the signal strength from the neighbourhood and how the user has responded to recommendations from his neighbourhood



Graph based recommender systems

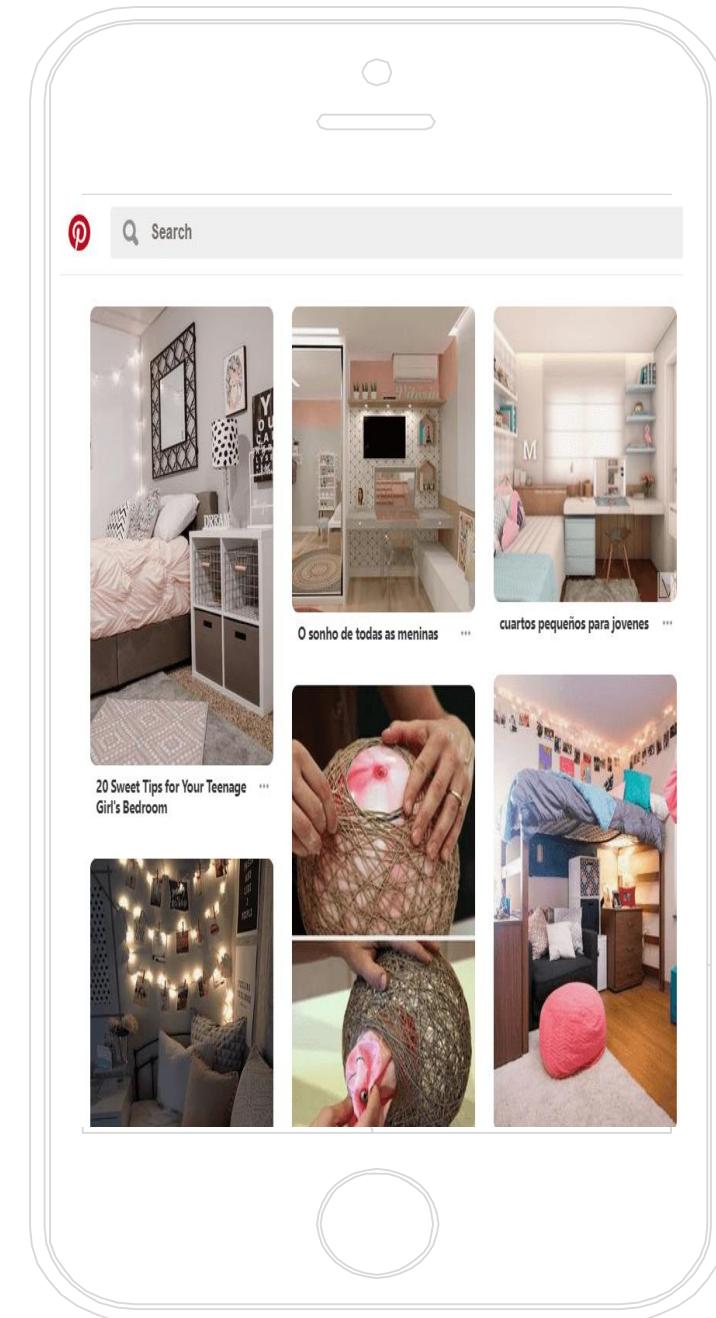
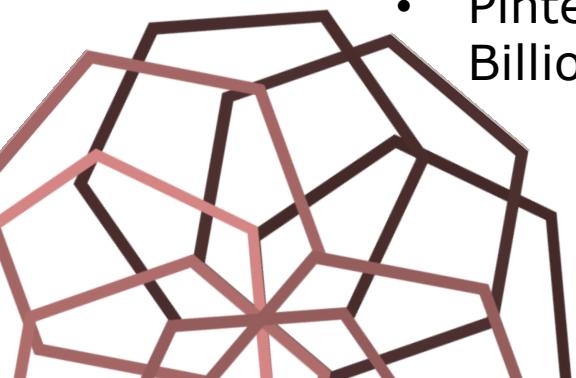
- Who can use them?
 - Pinterest or any similar platform based out of the concept set of items related to set of classes.
 - Graph based recommendation systems serve as skeleton for this purpose
- What sort of graphs are we looking at?
 - A network in which every node is connected to every other node is Fully connected graph.
 - If some set of nodes in a graph can be divided into two sets such that nodes in one set are connected to the other set alone. These graphs are Bipartite graphs



ABOUT PINTEREST

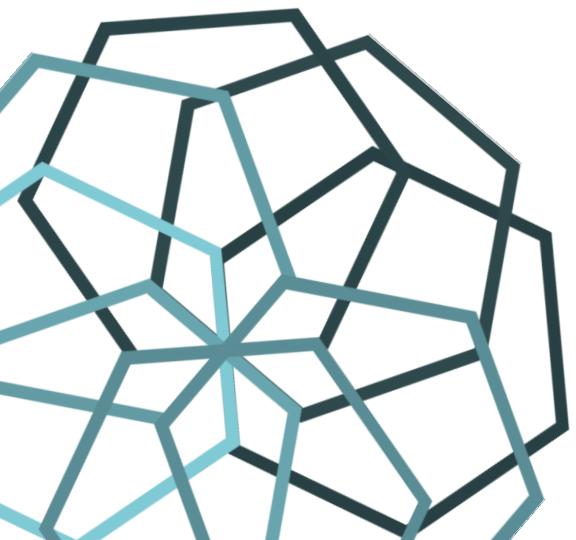


- Pinterest is a social bookmarking site
- It is a social network that allows users to share, curate, discover new interests by “pinning” images to their user wall
- It allows users to explore, discover and curate and share content (images) based on their lifestyle and interests
- It also provides links to sites where consumers can buy products or learn more about them
- Pinterest is currently valued at \$12.3 Billion



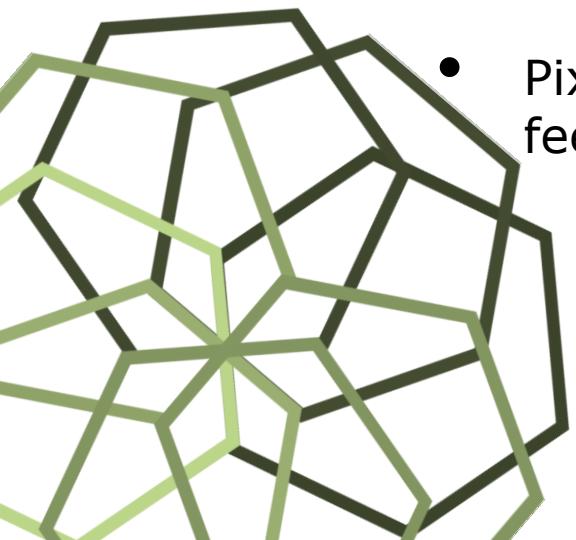
PIXIE, PINTEREST'S RECOMMENDATION SYSTEM

With a total user base of 250+ million, Pinterest has been constantly scaling the number of pins saved and matching them to people with overlapping interests, serving more than 10 billion recommendations every day.



WHAT IS PIXIE ?

- Pixie is a flexible, graph-based system for making personalized recommendations in real-time
- The goal is to create a system that could provide relevant and narrow recommendations at sub-second response rates as Pinners scroll through the home feed
- As Pinterest's main recommender system, Pixie is applied across all product surfaces
- Pixie powers recommendations across Pinterest in Related Pins, Home feed and Explore, and accounts for about half of all Pins saved.



HOW DOES IT WORK ?

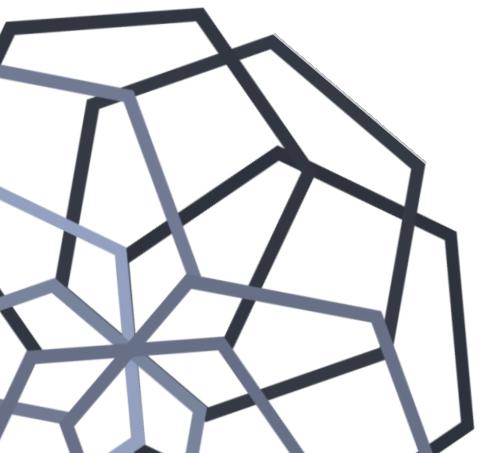
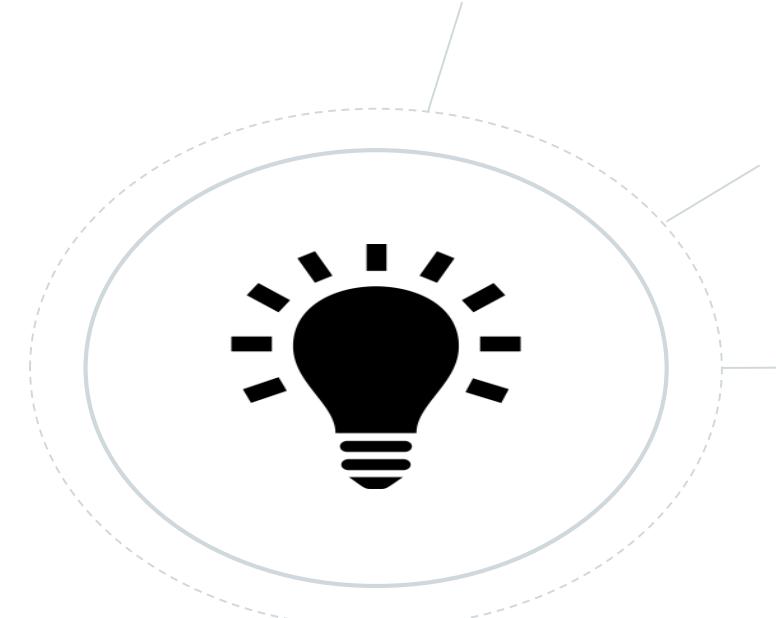
- Starting from a bipartite graph where each edge shows that a person saved a Pin to a board.
- This graph captures a huge amount of rich data from users (Pinner) with more than 100 billion edges and several billion nodes.
- Developed advanced machine-learning systems like **Pinnability** that predict how relevant an idea is to a Pinner, the first challenge is figuring out which of the more than 100 billion Pins to even consider since we can't possibly score them all at once.
- Pixie solves the candidate generation problem by starting graph traversal from a set of nodes that are already known to be currently relevant to the Pinner. Then, it only examines the portion of the graph nearest to these nodes by using a biased random walk algorithm to estimate the Personalized PageRank. The walk starts from multiple Pins and finds recommendations at the intersection of all of them.



THE CHALLENGE

- The data set of pinterest is unique as it's created from how people describe and organize Pins and boards, and it results in countless Pins that have been added hundreds of thousands of times.
- From this dataset, we know two valuable things: how those Pins are organized based on the context people add as they save and the Pinner's interests.
- With more than 175 billion Pins in the system, they are dealing with a huge bipartite graph.

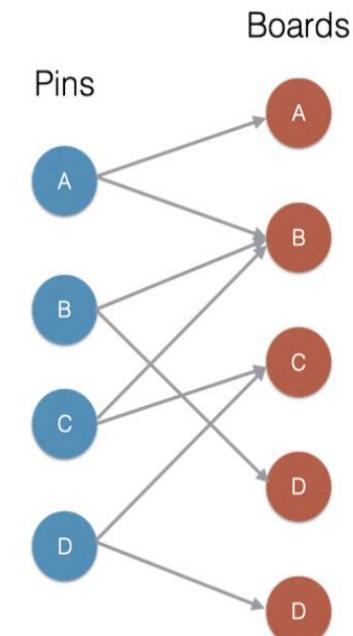
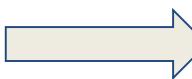
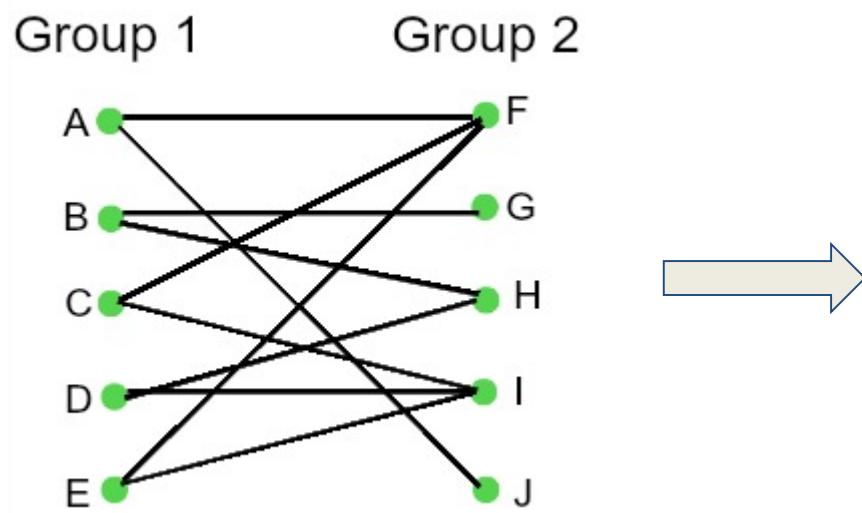
The challenge of the recommendation problem was figuring out how to narrow down the best Pin for the best person at the best time.



THE SOLUTION

- This is where the graph-based recommender system comes in: we know a set of nodes that are already interesting to a Pinner, so we start graph traversal from there.
- Pixie then finds the Pins most relevant to the user by applying a random walk algorithm for 100,000 steps. At each step, it selects a random neighbor and visits the node, incrementing node visit counts as it visits more random neighbors. We also have a probability Alpha, set at 0.5, to restart at node Q so our walks do not stray too far.
- The nodes that have been visited 14 and 16 times are the ones that are most closely related to the query node.
- Once the random walks are complete, we know the nodes which have been visited most frequently are the ones most closely related to the query node.
- Pixie continuously repeats this process in real-time as the data grows, so the users are always able to keep narrowing down their searches and find the exact ideas they're looking for to pursue their goal.
- Instead of just one starting point, Pixie also operates with multiple starting nodes where different weights are assigned based on the different actions a user can take on a Pin, whether it's zooming in, saving the Pin, or something else.

Bipartite graphs



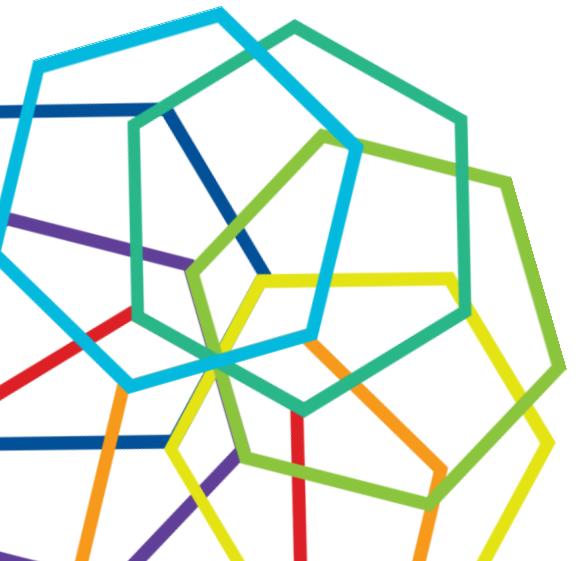
Traversals and the theory

- How do these graphs are generated?
 - Users upload pins and they pin them onto boards.
 - Over a period of time, curation by humans generates this graph and this will help us in associations.
- How are recommendations generated?
 - Traversals by Random walk
- Can we do any better?
 - Biased random walks based on probabilistic selection



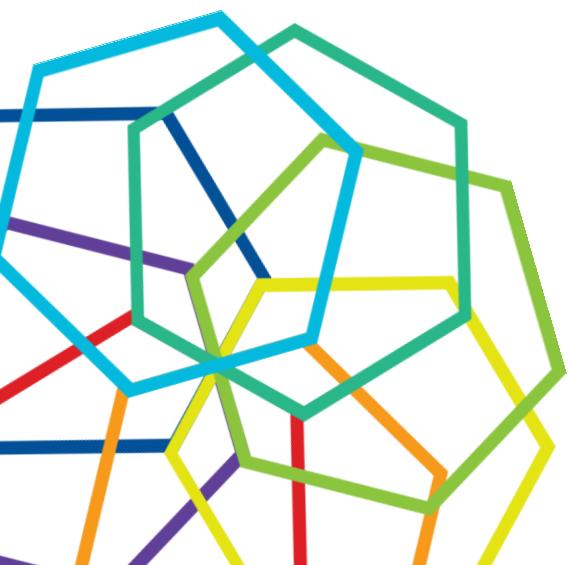
Traversals and its theory

- Bias in graphs
 - Traversing to more related boards
 - Connections between board to pin is based upon relatedness and simple relatedness metric like
$$p(\text{board}(k)/\text{pin}(j)) = n(\text{pin}(j) \rightarrow \text{board}(k))/n(\text{pin}(j))$$
- By above, the most likely board you will traverse to from a pin is the one that users had most associated the pin with
- The same goes for the selection of pins from a board.
$$p(\text{pin}(j) | \text{board}(k)) = n(\text{board}(k) \rightarrow \text{pin}(j))/n(\text{board}(k))$$



Biased Random walk

- Each step we take from each node is sampled from a discrete distribution
- Generating recommendations with multiple parallel walks
- How do you deal with multiple repeated recommendations over all paths?
 - Objects are scored based on the number of times they repeat across a set of paths
 - If two different source paths produce same object, this item should be rated higher



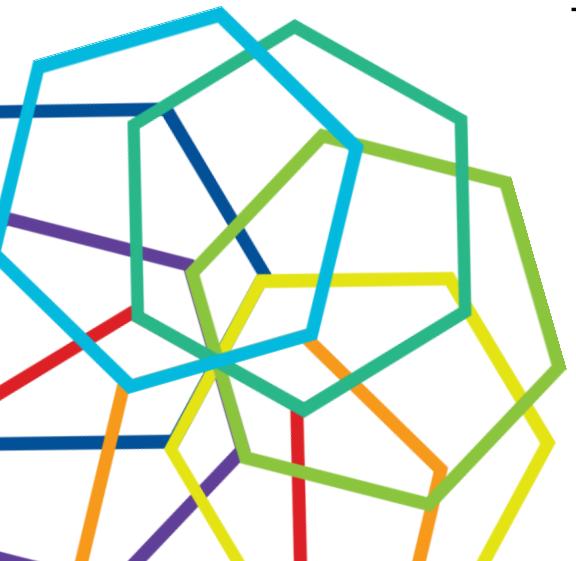
Biased Random walk - Scoring

- $score(pin(k)) = (\sum \sqrt{visitCount(walk(i))}))^{**2}$
- Let say we are having three parallel walks and a node has been visited one time in every single path. By the formula above, we give a score of 9 to this pin.
- Similarly a pin that is visited two times in one path and single time on second path will have a score of 5.82



Traversals and its theory

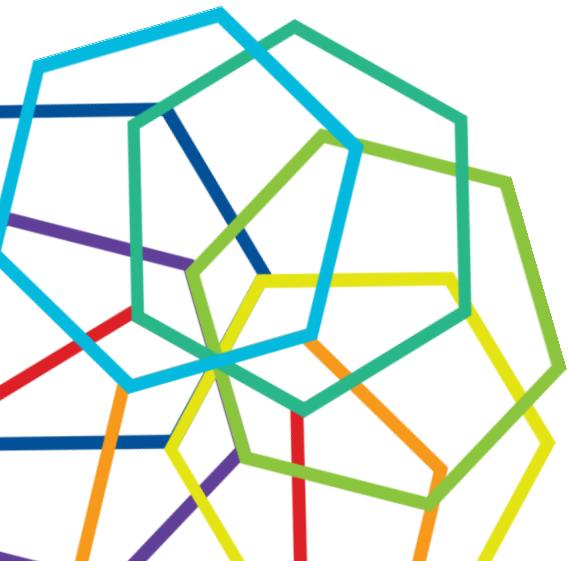
- A walk based on previous metrics will produce a fair set of recommendations
- How can we improve them?
 - What about freshness of an object in the pool?
 - How can we generate items related to user?
 - Can we include ideas from collaborative filtering into this?



Pin freshness and what is it

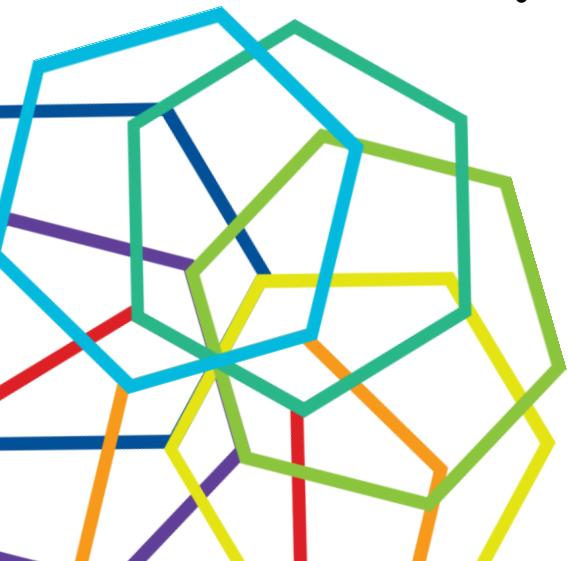
- It is a scalar value indicating how old the pin is
- An example scaling giving recent pins more weight can be like

$$freshness(pin) = C * e^{(-\lambda * n_{daysDiff})}$$



Profiles and its role

- To generate user related content, profile generation is key
- Since a user is defined by what he sees, likes, repins and searches for, we primarily focus on calculating profiles for pins themselves
- How can we do pin profiling?
 - Title and tags given by the user, if any
 - Content of the pin
 - Object recognition on Images to auto generate tags



Generating profiles

- From text content
 - Using vectorization models to generate low dimensional representation like Topic Models, Word2Vec, GloVe etc.
- Since a board is a set of pins associated with them. A board profile can be a
 - Average of all profiles of pins pinned on it
 - Weighted average of pins with weights being the number of times each pin got pinned to a board

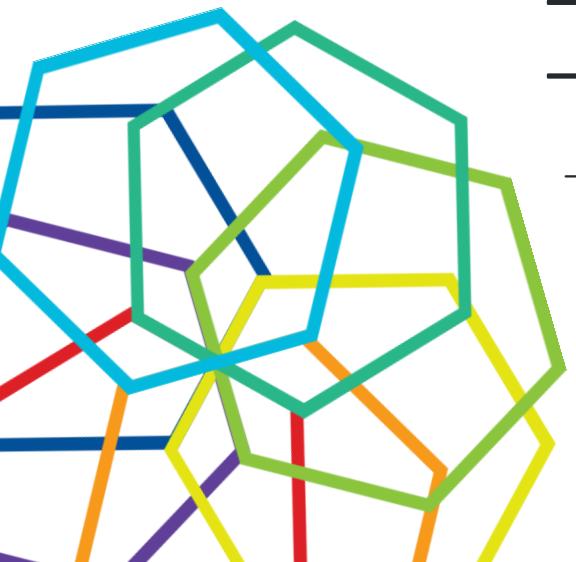


User profiles as a driver

- We use user profiles to drive recommendations
- How do we do that?
 - We select source node profile and use user profile to transform the board profiles to user bias

```
pin_board_probability_array *  
(self.user_profile.dot(self.board_profiles.T))
```
 - We use the normalised form of this to select the best board.
 - We use a similar transformation to jump from a board to pin

```
_board_pin_probability_array *  
(self.user_profile.dot(self.pin_profiles.T)) * self.pin_freshness
```



Pinterest's Pixie - Algorithm

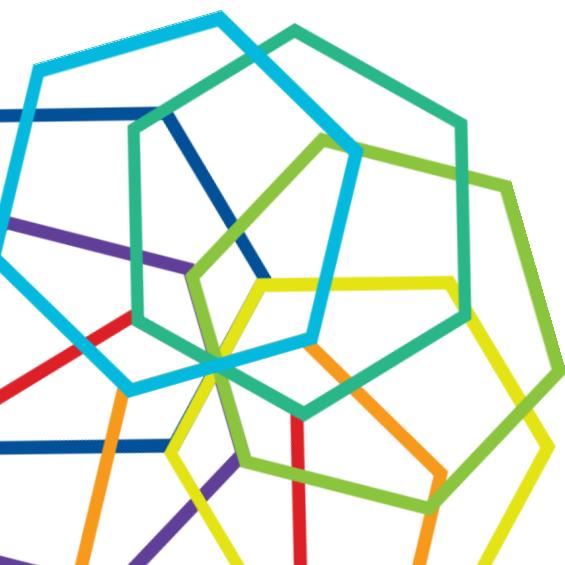
Algorithm 2 Pixie Random Walk algorithm with early stopping.

```
PIXIERANDOMWALK( $q$ : Query pin,  $E$ : Set of edges,  $U$ : User  
personalization features,  $\alpha$ : Real,  $N$ : Int,  $n_p$ : Int,  $n_v$ : Int)  
1: totSteps = 0,  $V = \vec{0}$   
2: nHighVisited = 0  
3: repeat  
4:   currPin =  $q$   
5:   currSteps = SampleWalkLength( $\alpha$ )  
6:   for  $i = [1 : currSteps]$  do  
7:     currBoard =  $E(currPin)[\text{PersonalizedNeighbor}(E,U)]$   
8:     currPin =  $E(currBoard)[\text{PersonalizedNeighbor}(E,U)]$   
9:      $V[currPin]++$   
10:    if  $V[currPin] == n_v$  then  
11:      nHighVisited++  
12:    totSteps += currSteps  
13: until totSteps  $\geq N$  or nHighVisited  $> n_p$   
14: return  $V$ 
```

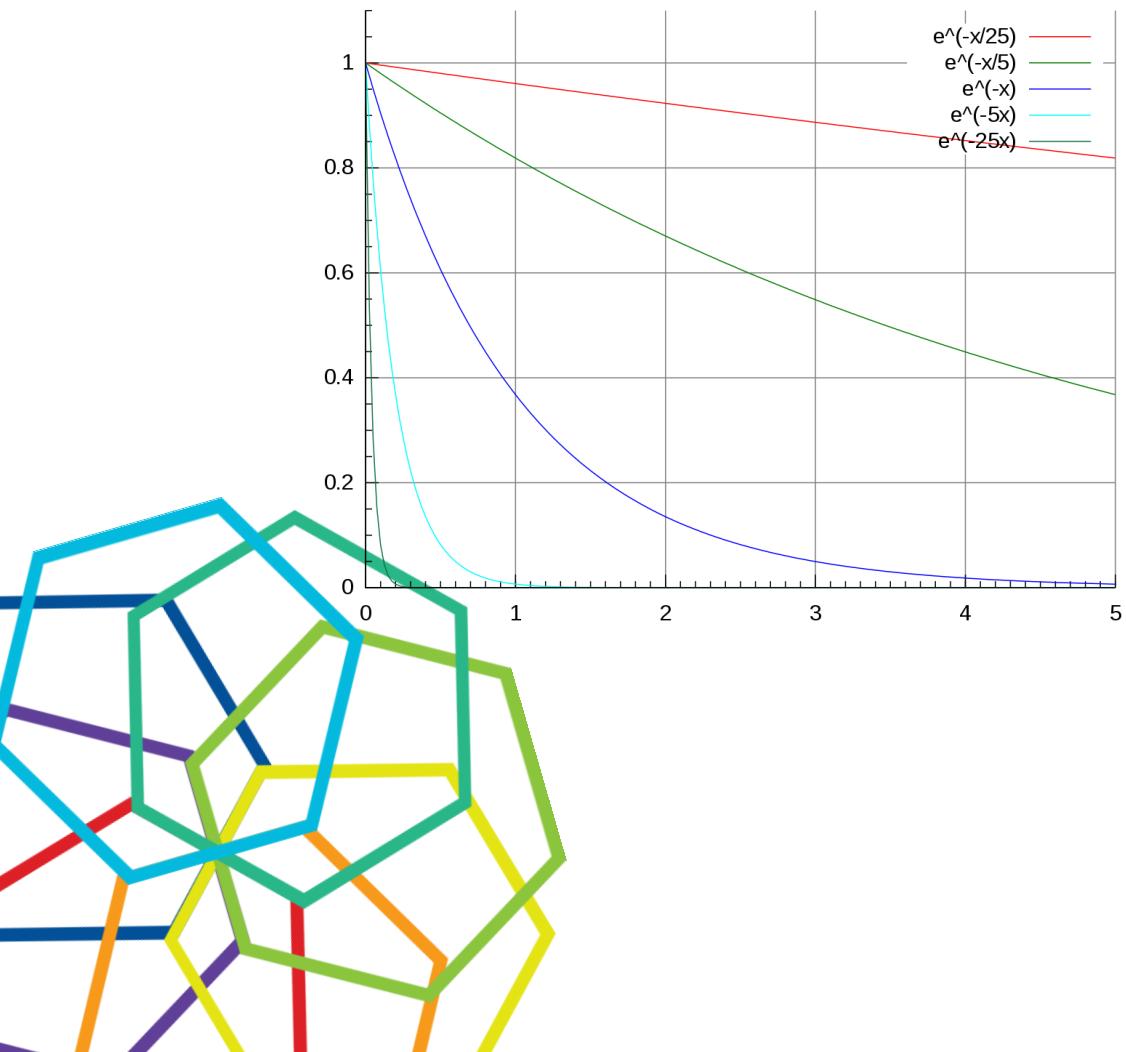
Algorithm 3 Pixie recommendations for multiple pins.

```
PIXIERANDOMWALKMULTIPLE( $Q$ : Query pins,  $W$ : Set of weights for  
query pins,  $E$ : Set of edges,  $U$ : User personalization features,  $\alpha$ :  
Real,  $N$ : Int)
```

```
1: for all  $q \in Q$  do  
2:    $N_q = \text{Eq. 2}$   
3:    $V_q = \text{PIXIERANDOMWALK}(q, E, U, \alpha, N_q)$   
4: for all  $p \in G$  do  
5:    $V[p] = \left( \sum_{q \in Q} \sqrt{V_q[p]} \right)^2$   
6: return  $V$ 
```



Multiple user profiles for better recommendations



- Most users tend to have a permanent and temporary tastes and it is quite hard to catch both of them in one profile.
- We use temporary and permanent profiles to generate recommendations.
- Temporary and permanent profiles have a different decay rate when calculating profiles.

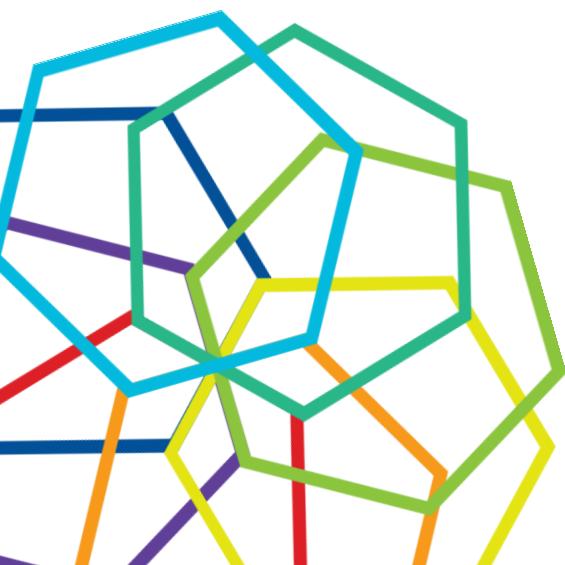
Scaling recommender systems

- How scalable is the system?
 - Pinterest uses 120GB of RAM to load their board-pin graphs.
 - Dealing with Sparse matrices for traversals.
 - If the connectivity is weak in the network, use sparse matrices to generate them.
 - It can use [DOK](#), [CSC](#) or [CSR](#) (Compressed Sparse Column/Row matrices).
 - A low memory footprint arrays can be made with above data structures



Scaling recommender systems

- Where will they be lacking?
 - Slicing rows and columns
 - Each matrix form comes with pros and cons.
 - While CSR matrix has faster row splicing, column splicing is incredibly slow.
 - While CSC matrix has faster column splicing, row splicing is incredibly slow.
 - Since we need to switch between boards to pins and vice versa, we use both matrices in switching to find appropriate rows and columns.



Faster recommendations + cache ...

- While arrays can only do so much, pre-caching to serve recommendations can tremendously increase user experience on the platform
- Pre-calculating the recommendations for active users can act as load distributor for your system.



EVALUATION OF THE RECOMMENDER SYSTEM

- No time stamp associated with ratings
 - Hide some ratings from the user item matrix
 - Build model on remaining data
 - Predict ratings for hidden the hidden ratings
 - Compute accuracy metrics
- With time stamp



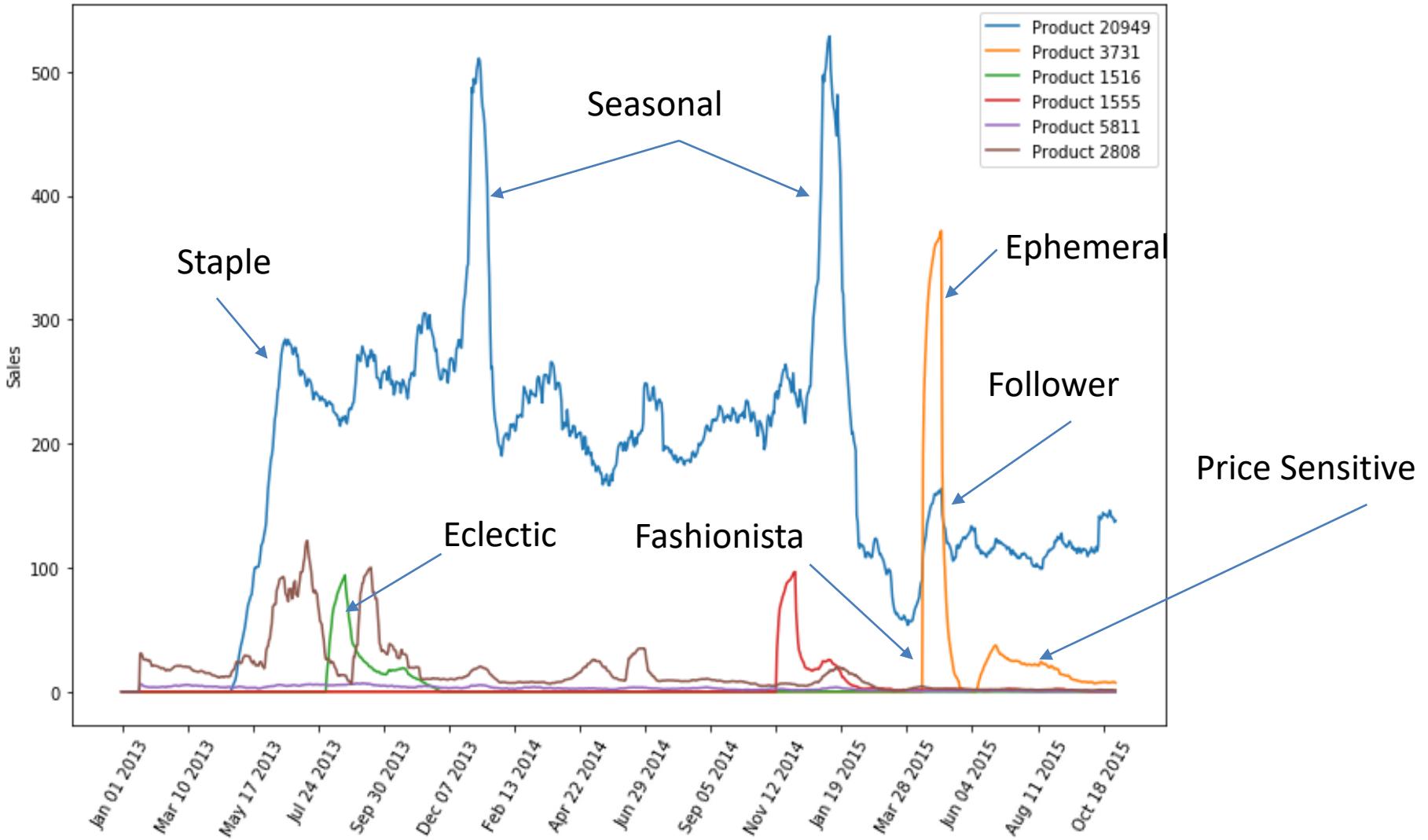
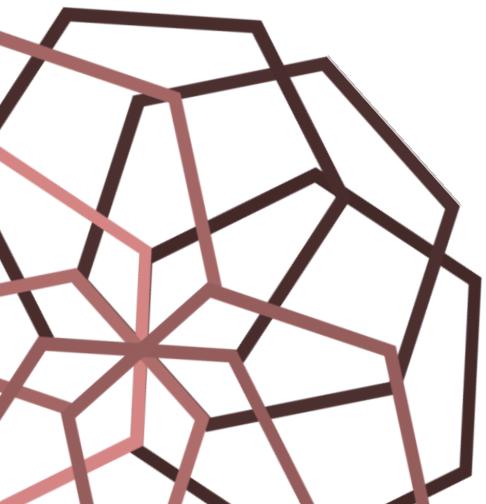
FEATURE EXTRACTION FASHION RECOMMENDATION

- Customer Registration data
 - Preferences: Colours, Brands...
 - Qualitative measures: "what looks good on me?" or fashion confidence
 - Physical characteristics
 - Budgetary Guidelines
 - Demographics
- Purchasing behavior
 - What product did they buy and when?
- Product Features
 - Price
 - Colour
 - Brand
- Item View data
 - What did they view and when?

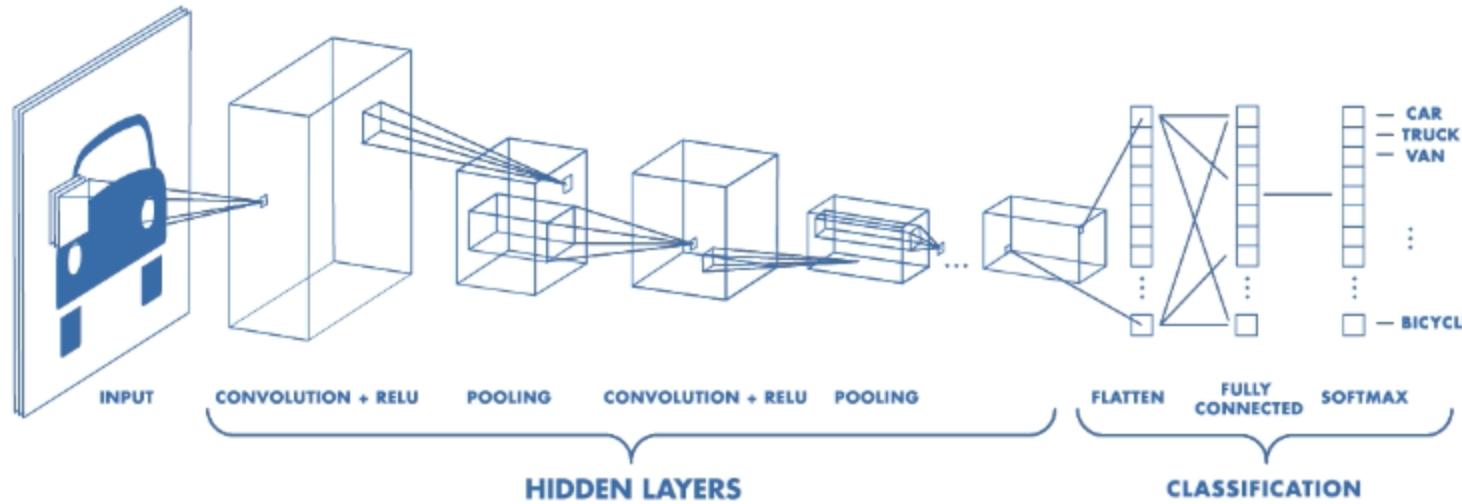


FEATURE ENGINEERING

Are all items equally important?

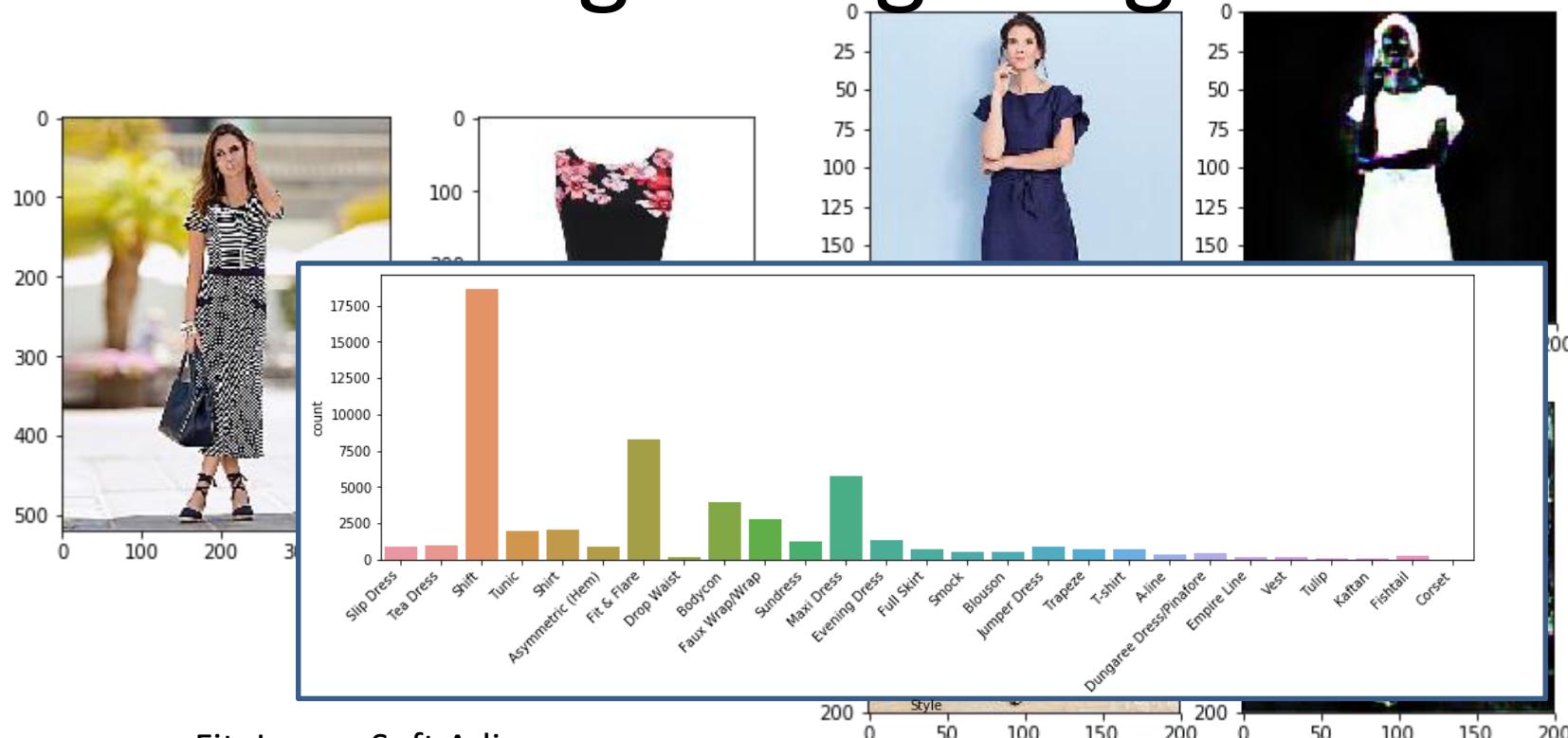


Convolutional Nets and Transfer Learning



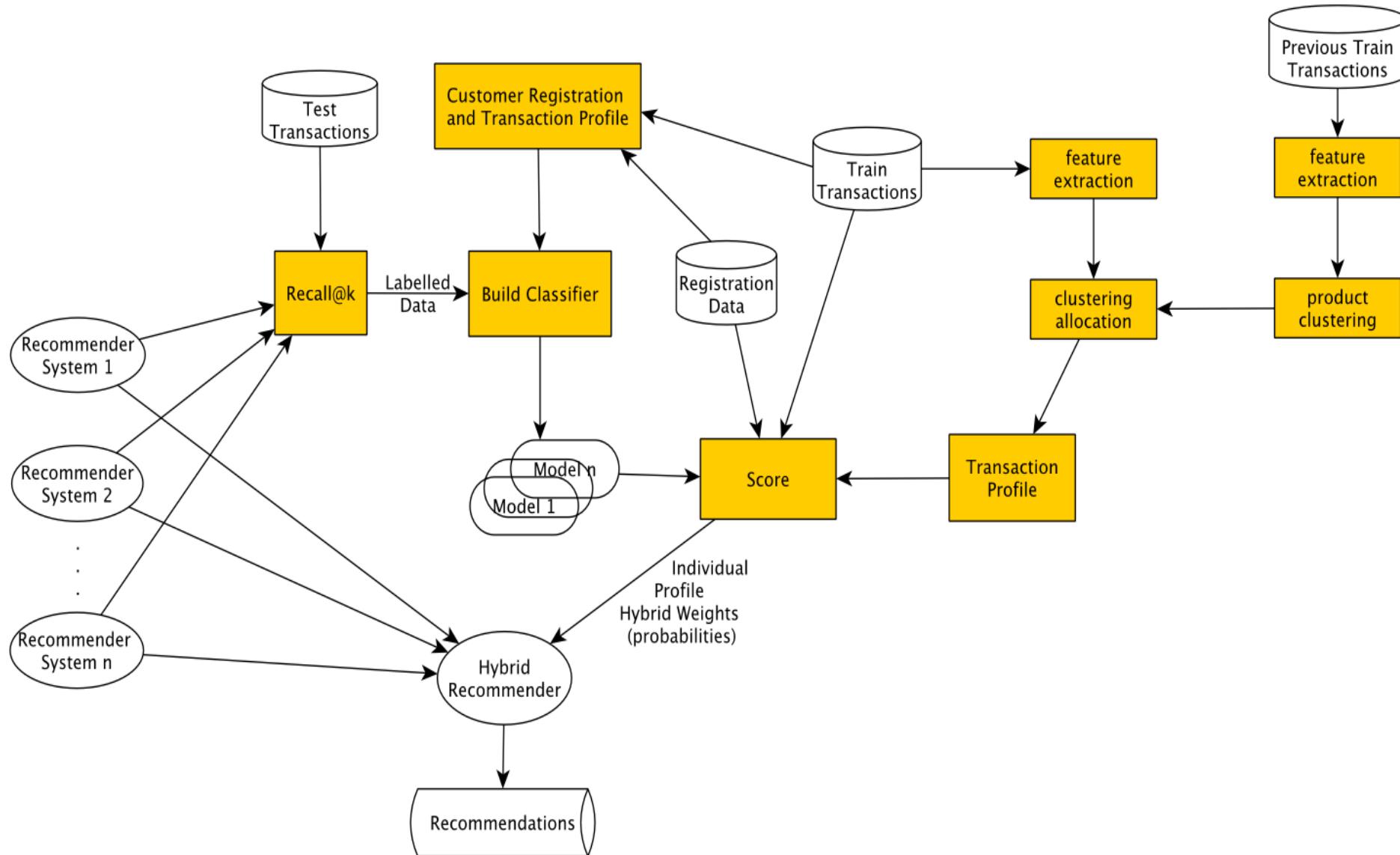
Source: <https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>

Learning to Tag Images



- Fit: Loose, Soft A-line,.....
- Style: Slip Dress, Fit and Flare,...
- Neckline: V-neck, Crew, Scoop, Cowl draped neck
- Sleeve Style: Spaghetti straps, Balloon, Angel/Waterfall...
- Sleeve Length: Sleeveless, Short Sleeves, $\frac{3}{4}$ sleeves,...
- Dress Length: Mid Thigh, On the knee, Just above the knee
- (Predominant) Color: Black, Navy, Primrose Yellow, Diva Blue, ...
- Pattern: Plain, Florals, Marl, Paisley....

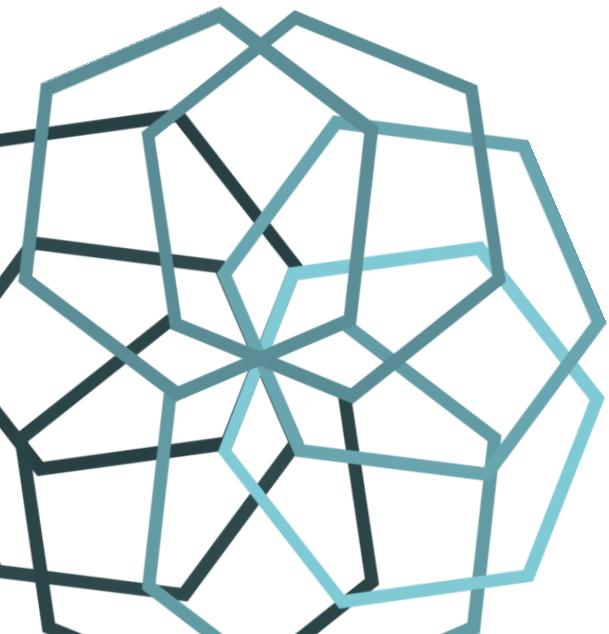
PERSONALIZED RECOMMENDATION

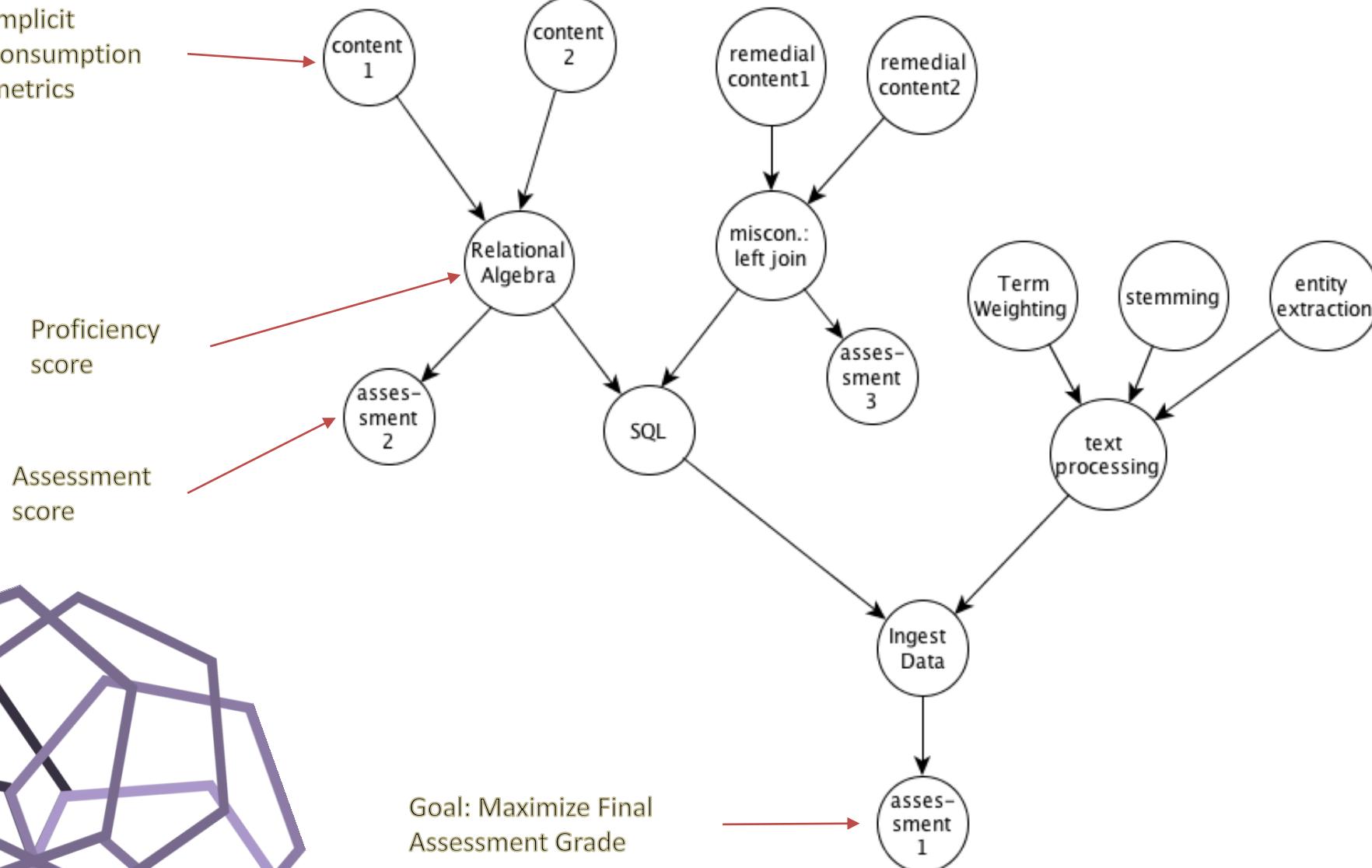


MODELLING ITEM DEPENDENCIES

ADAPTIVE LEARNING

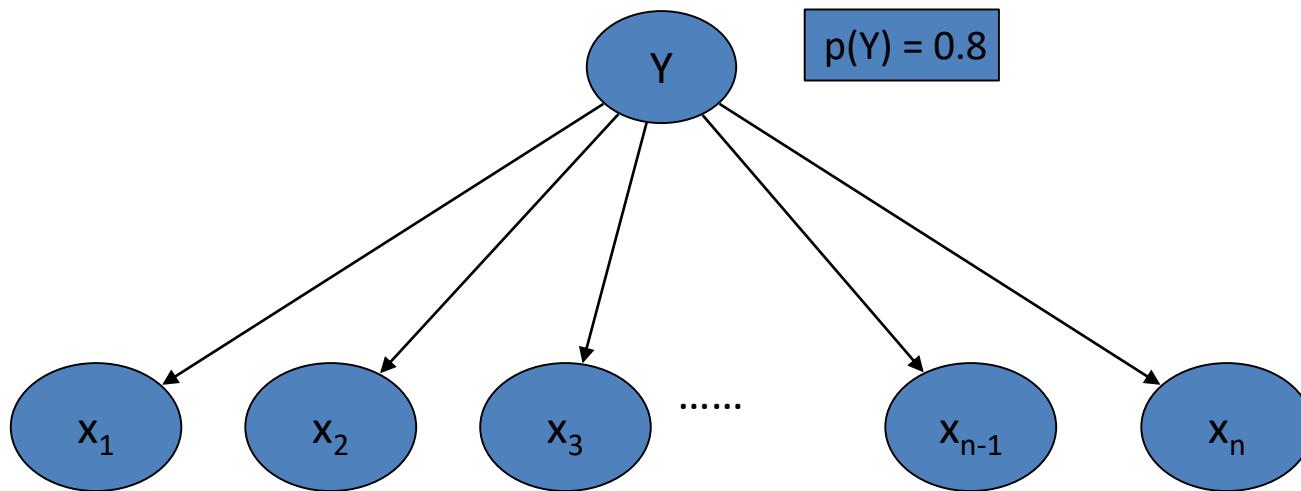
- Objective
 - To provide individualized pathways through content on a course that builds proficiency in a particular skill
 - Improved learning
 - Speed of learning/certification
 - Given the current state of the student, what content should he consume next?
- Challenge
 - Content to be consumed may require prerequisite knowledge
 - Dependencies between items
 - Dependencies between skill proficiency and items





THE BBN FOR NAÏVE BAYES

Y	$\neg Y$
x_1	0.7 0.1
$\neg x_1$	0.3 0.9



Number of Parameters: $2n+1$ instead of $2^{n+1}-1$

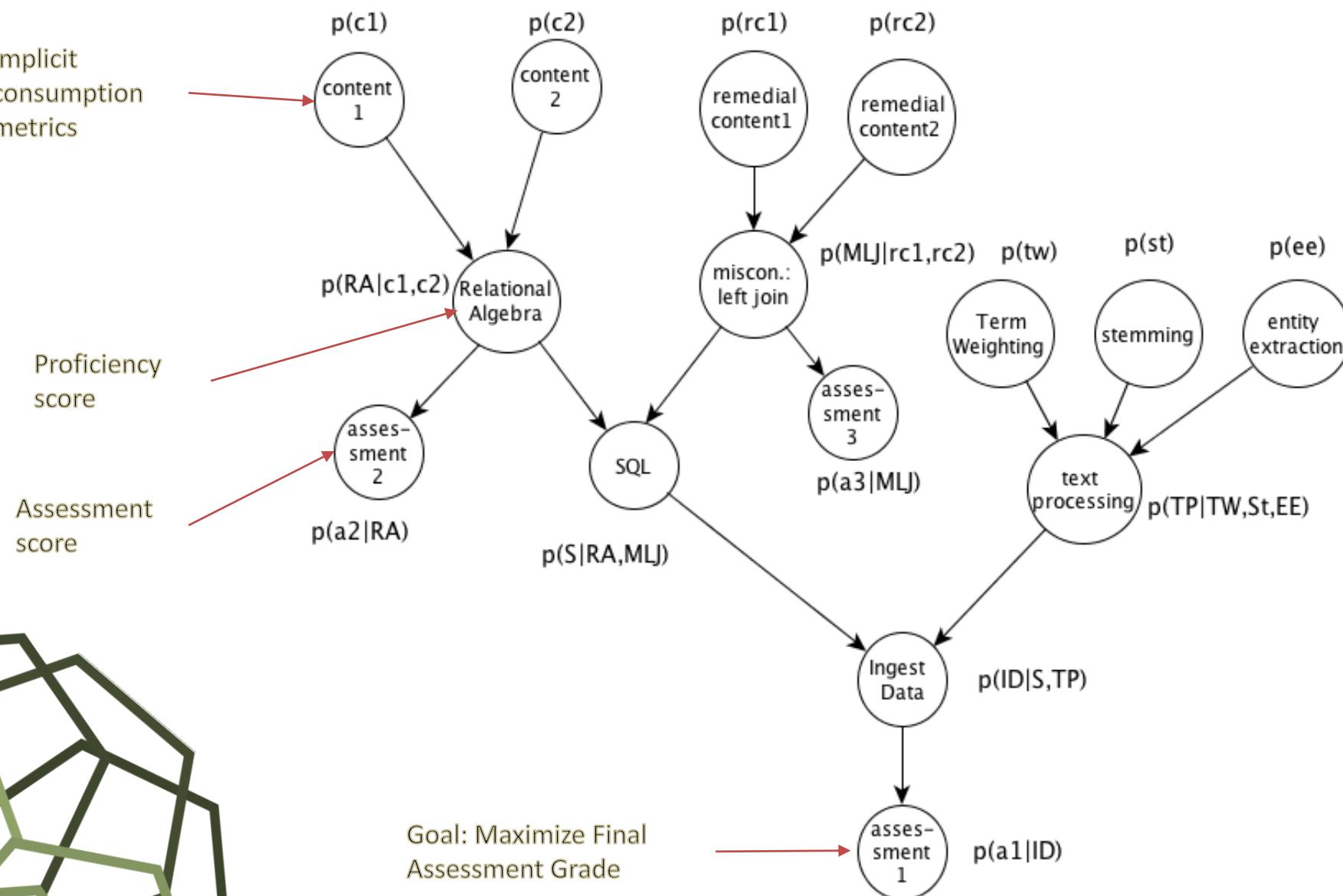
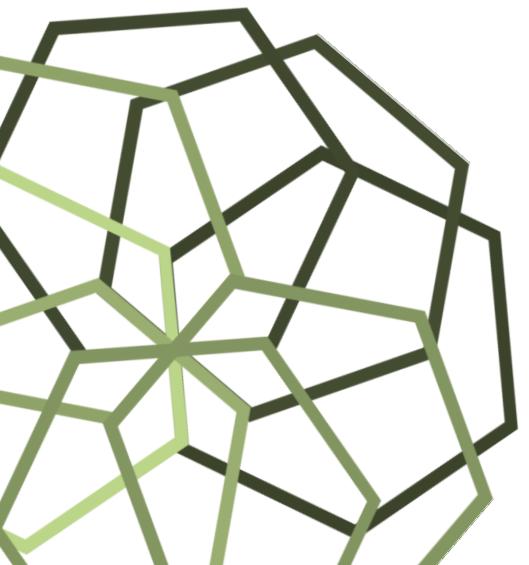
BAYESIAN BELIEF NETWORKS

- Given
 - a problem domain described using a set of n random variables $V = \{X_1, X_2, \dots, X_n\}$
 - a joint probability distribution P defined on V
 - a DAG $G = \langle V, E \rangle$
- A Bayesian Network
 - Is a directed acyclic graph
 - Each variable corresponds to a node in the graph
 - Parents of a node X_i are a subset of the variables with a direct influence on the node
 - Represents a full joint probability distribution over these variables by defining
 - Local conditional probability distributions $P(X_i | \text{Parents}(X_i))$
 - Called the conditional probability table
 - Assertions of conditional independence
- By the Chain Rule

$$P(X_1 X_2 \dots X_n) = P(X_n | X_1 X_2 \dots X_{n-1}) P(X_{n-1} | X_1 X_2 \dots X_{n-2}) \dots P(X_2 | X_1) P(X_1)$$

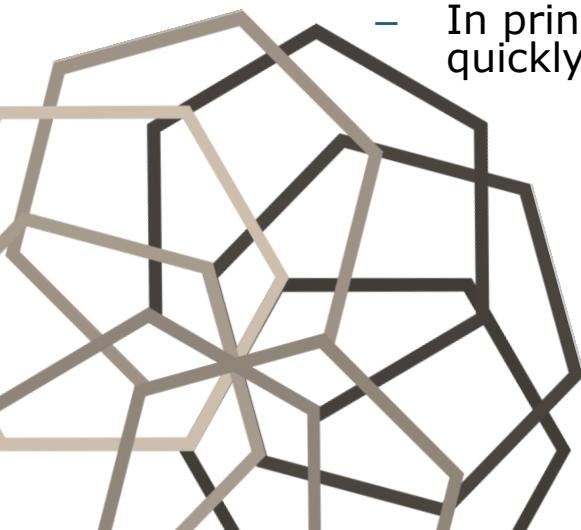
- Given a BBN

$$p(X_1, X_2, \dots, X_n) = \prod_{i=1}^n p(X_i | PA_{X_i})$$



CONVERSATIONAL RECOMMENDATION

- Collaborative Filtering and content based filtering fail for users that have not consumed any item
- Conversational Recommenders aim to ask “searching questions” that can help recommend items in the absence of any prior knowledge
- **Pros**
 - More Human Like
 - Explicit preference elicitation
 - Very useful in complex product sales
- **Cons**
 - Takes time/multiple steps to get recommendations
 - In principle, the recommender should be able to learn over time as to which questions allow it to quickly reach a good recommendation in the current context



CONVERSATIONAL RECOMMENDATION



I would like to eat at a restaurant that has:

Cuisine Price
Style Atmosphere Occasion

I would like to eat at a restaurant just like:

PREFERENCE ELICITATION

 *Entree Results*

The Los Angeles restaurant you chose is:

Chinois On Main	
2709 Main St. (bet. Rose Ave. & Ocean Park Blvd.), Santa Monica, 310-392-9025	
Pacific New Wave	\$30-\$50
Extraordinary Decor, Extraordinary Service, Near-perfect Food, Hip Place To Be, On the Beach, Great for People Watching, Parties and Occasions, Weekend Brunch, Weekend Lunch, Fabulous Wine Lists	

We recommend:

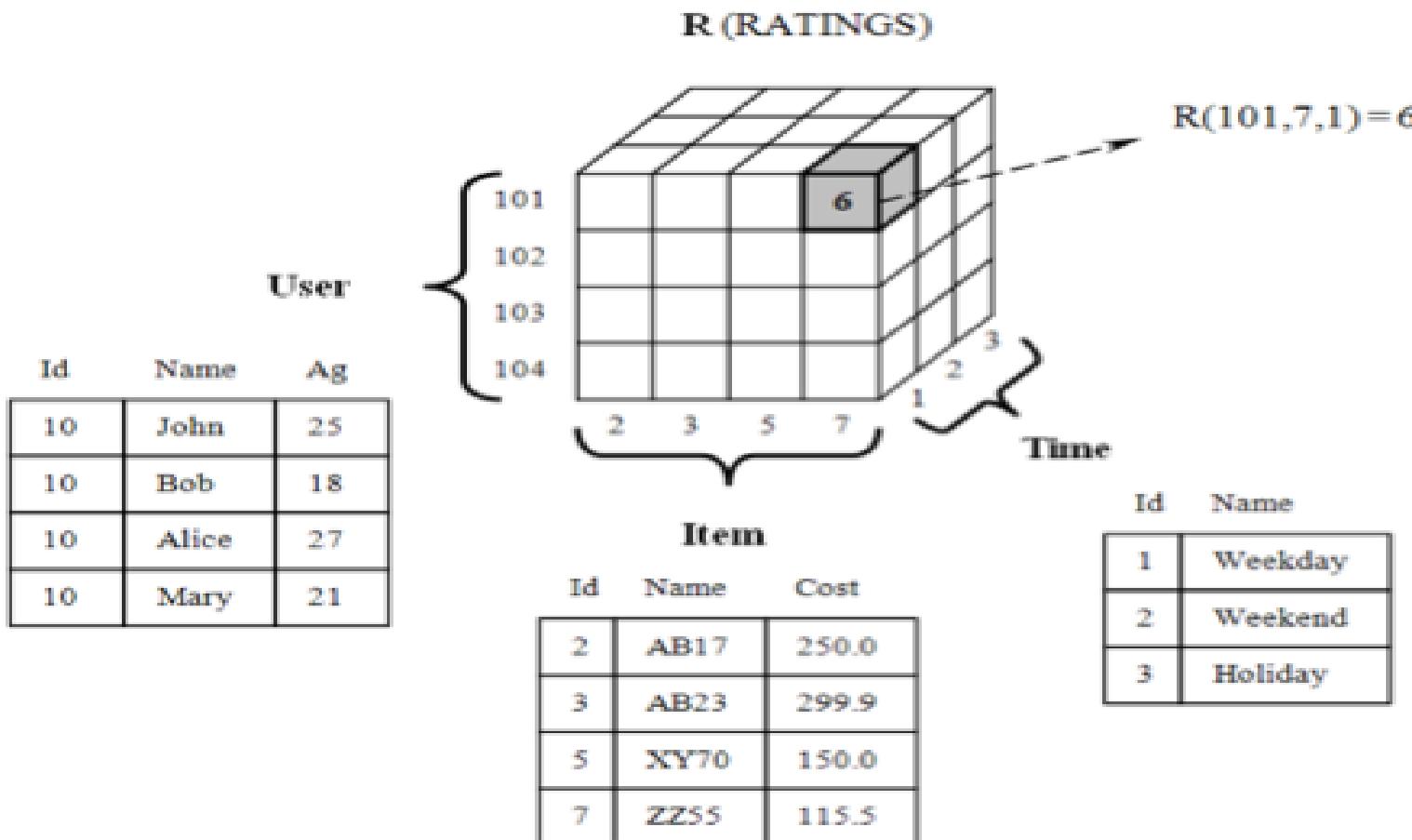
Yoshi's Cafe	
3257 N. Halsted St. (Belmont Ave.), Chicago, 312-248-6160	
Asian, Japanese, French (New)	\$30-\$50
Extraordinary Decor, Extraordinary Service, Near-perfect Food, Need To Dress, Prix Fixe Menus, Quiet for Conversation, Very Busy - Reservations a Must, Romantic, Good Out of Town Business, Fabulous Wine Lists, Game, Parking/Valet	

less \$\$ nice cuisine
Less \$\$

traditional creative lively quiet

CONTEXTUAL RECOMMENDATION

- Context for user within which (s)he consumed the item



Group Recommendation

- Recommenders are typically built to cater to the preference of a single user
- How can we recommend items to groups
 - For example
 - a group of friends that want to watch a movie together
 - a meal for a family
 - music to be played in a car
- Effect of others
 - Emotional Contagion: Others enjoying impact on our experience
 - Conformity
 - Normative influence
 - Informational influence

Approaches to Group Recommendation

$$f: U \times I \times \varphi(U) \rightarrow [1, r]$$

- Profile Aggregation
 - Is averaging ratings a good idea?
 - Drawbacks
 - Individual preferences are lost
 - The aggregated profile is for a ‘user’ that is not in the group
- Recommendation Aggregation
 - Create a ranked list of recommendation for each user in the group
 - Aggregate the lists using a voting mechanism
 - Arrows’ theorem: there is no fair voting system

Arrow's Theorem

- No rank-order voting system can be designed that satisfies these three fairness criteria:
 - If every voter prefers alternative X over alternative Y, then the group prefers X over Y
 - If every voter's preference between X and Y remains unchanged when Z is added to the slate, then the group's preference between X and Y will also remain unchanged
 - There is no dictator: no single voter possesses the power to always determine the group's preference.

Group Recommendation

- Kemeny-Optimal aggregation [Dwork et al., 2001]
 - Given a distance function between two ranked lists (Kendall tau distance)
 - Given some input ranked lists to aggregate
 - Compute the ranked list (permutation) that minimize the average distance to the input lists

Kemeny Optimal Aggregation

- Kemeny optimal aggregation is expensive to compute (NP hard – even with 4 input lists)
- There are other methods that have been proved to approximate the Kemeny-optimal solution
 - Borda count – no more than 5 times the Kemeny distance
 - Spearman footrule distance – no more than 2 times the Kemeny distance [Coppersmith et al., 2006]
 - Average – average the predicted ratings and sort
 - Least misery - sort by the min of the predicted ratings

Table 1 Example of individual ratings for ten items (A to J)

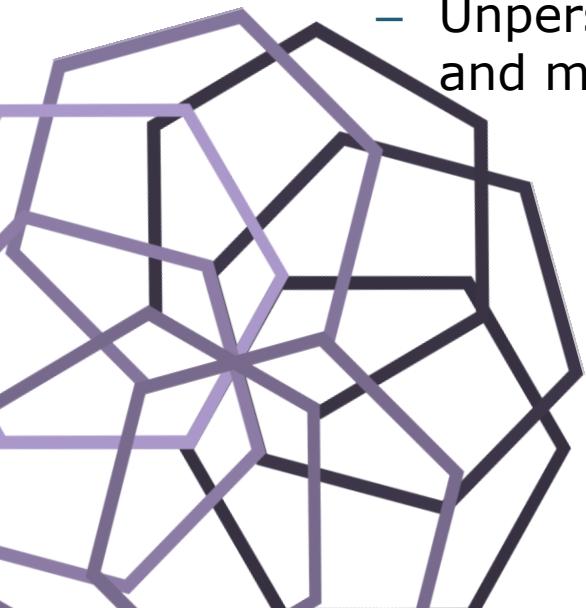
	A	B	C	D	E	F	G	H	I	J
Peter	10	4	3	6	10	9	6	8	10	8
Jane	1	9	8	9	7	9	6	9	3	8
Mary	10	5	2	7	9	8	5	6	7	6

Table 2 Example of the Least Misery Strategy

	A	B	C	D	E	F	G	H	I	J
Peter	10	4	3	6	10	9	6	8	10	8
Jane	1	9	8	9	7	9	6	9	3	8
Mary	10	5	2	7	9	8	5	6	7	6
Group Rating	1	4	2	6	7	8	5	6	3	6

RECOMMENDATION DIVERSITY

- One of the goals of recommenders is to provide idiosyncratic items
 - The more diverse the recommendations, the more opportunities there are to recommend such items
 - Balance accuracy with diversity
 - Individual Diversity: Measured by average dissimilarity between pairs of recommended items
 - Aggregate Diversity: Diversity of recommendations across users
 - Unpersonalized recommendations for example, “Top Searches” perform poorly and more long tail products get recommended



CONCLUDING NOTE

- Focus on rating accuracy may not be the best approach
- Diversity of Recommendations key for gaining “signal”
- Innovative Feature Extraction can improve quality of recommendation
- Item dependencies need to be understood and modelled
- Contextual Recommendation
 - Beyond location/Time of Day
- Personalization of Lists of Items or Groups of Users
 - Recommending a menu for a dinner
 - Add to the mix – Constraints: “Ensure a balanced diet for the dinner”
 - Recommending a play list of a journey
- Conversational Recommenders
 - Tweaks and Critiques



Bibliography

- RESNICK, P., IACOVOU, N., SUCHAK, M., BERGSTROM, P., AND RIEDL, J. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of CSCW*.
- KONSTAN, J., MILLER, B., MALTZ, D., HERLOCKER, J., GORDON, L., AND RIEDL, J. 1997. GroupLens: Applying collaborative filtering to Usenet news. *Commun. ACM* 40, 3, 77–87.
- D.W. Oard and J. Kim, “Implicit Feedback for Recommender Systems”, *Proc. 5th DELOS Workshop on Filtering and Collaborative Filtering*, pp. 31–36, 1998.
- A. Paterek, “Improving Regularized Singular Value Decomposition for Collaborative Filtering”, *Proc. KDD Cup and Workshop*, 2007.
- R. Salakhutdinov, A. Mnih and G. Hinton, “Restricted Boltzmann Machines for Collaborative Filtering”, *Proc. 24th Annual International Conference on Machine Learning*, pp. 791–798, 2007
- Geffory Hinton, “A Practical Guide to Training Restricted Boltzmann Machines”, <https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf>
- Ben Marlin and Richard Zemel. "Collaborative prediction and ranking with non-random missing data." *Recsys-2009: ACM Conference on Recommender Systems*
- Yehuda Koren and Robert Bell and Chris Volinsky. MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS. *IEEE Computer*, 2009
- Koren, Yehuda. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'08)
- Mukund Deshpande and George Karypis. Item-Based Top-N Recommendation Algorithms, *ACM Transactions on Information Systems*, Vol. 22, No. 1, January 2004, Pages 143–177
- Eksombatchai et al., Pixie: a system for recommending 3+ billion items to 200+ million users in real-time WWW'18
- Y. Hu, Y. Koren, C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets

Thanks!

Any questions?

You can find me at:

ssanand@tatasdata.com