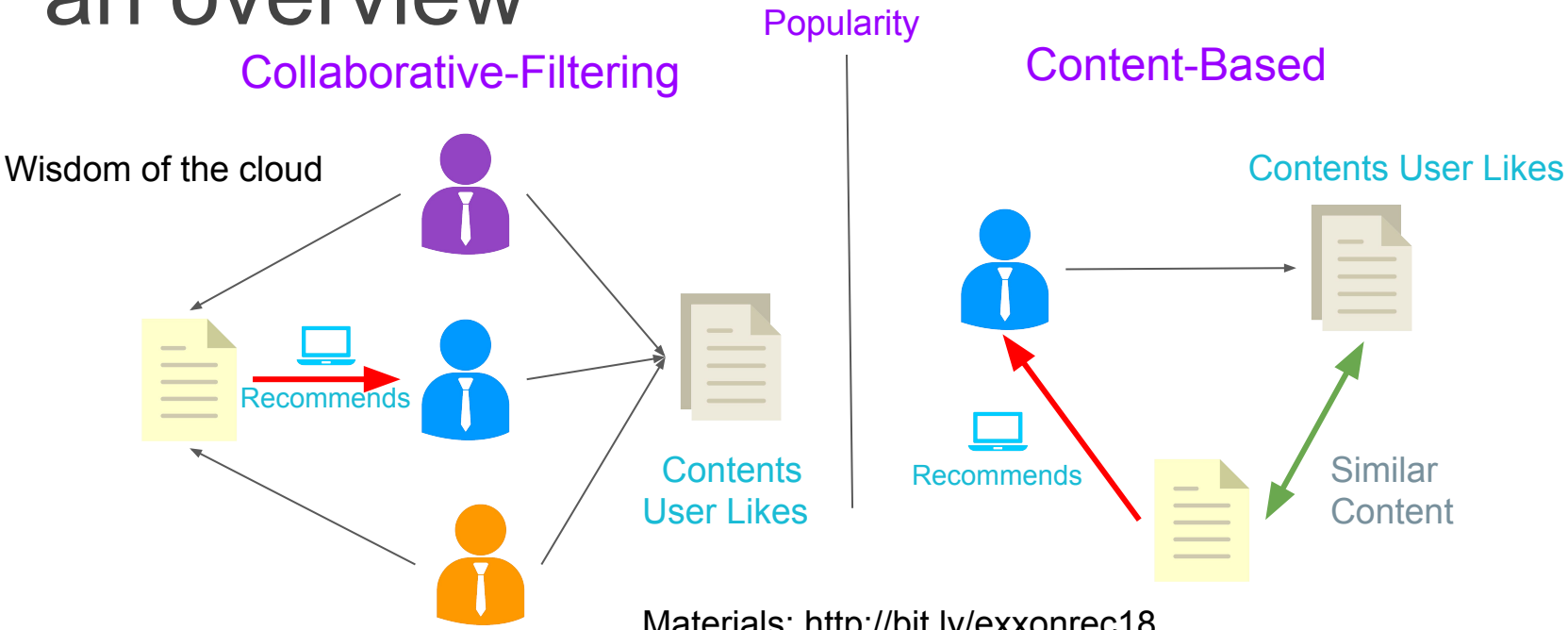
Recommendation System: an overview



Materials: http://bit.ly/exxonrec18

Agenda

Overview

Content-based recommendation system

Collaborative filtering systems

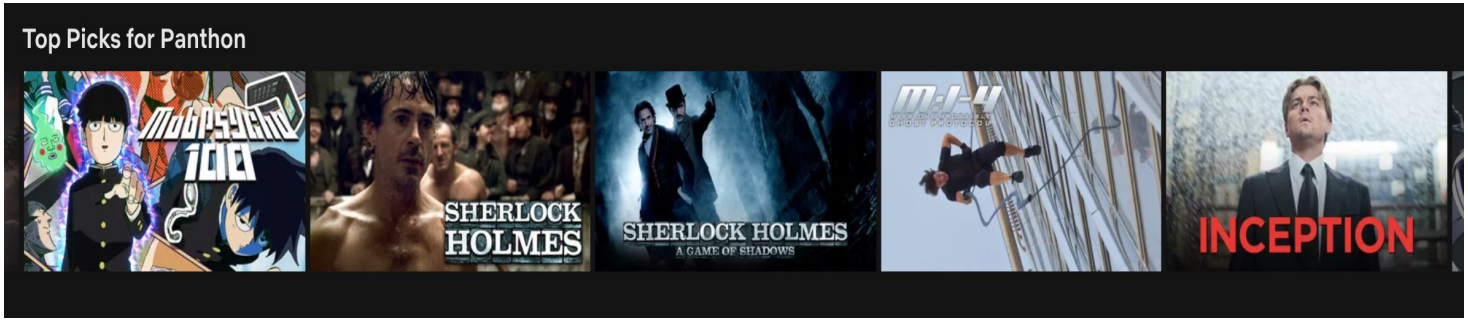
Using unstructured information

Metrics

Lab

What is the Goal of a Recommendation System?

- Predict the user's preference toward an item, and recommends it!
- Example: Netflix, Amazon



What do We Know About Our Users?

Explicit Data

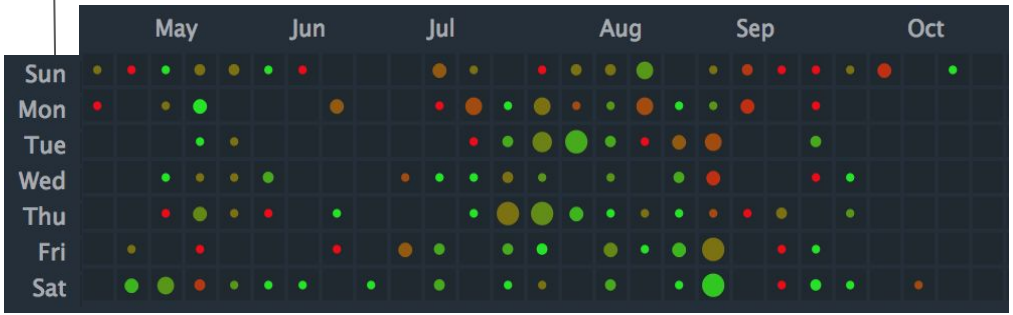
- What the Users told us
- Example: Item feedbacks

Sapporo Ichiban Chow Mein, 3.6-Ounce Packages...
Sapporo

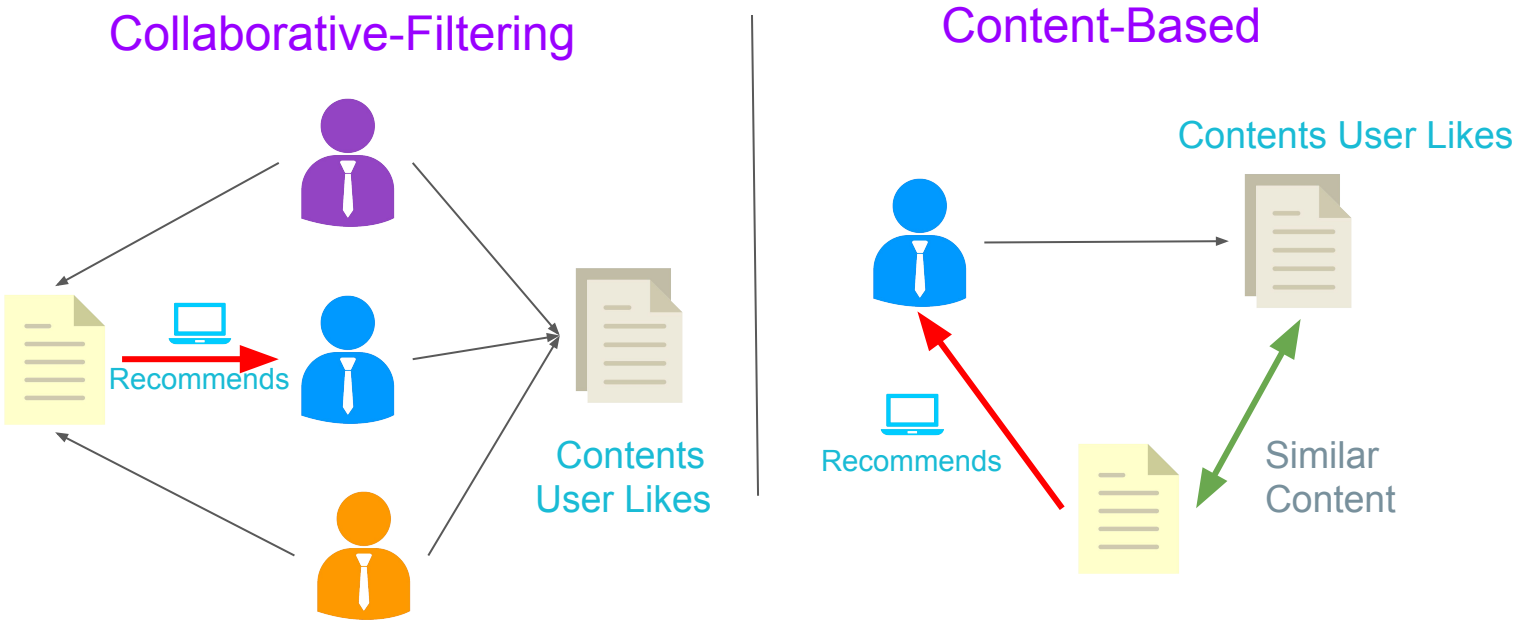
I love it

Implicit Data

- What we infer from the user's activity
- Example: Time spent browsing on a page, number of times browsing a page



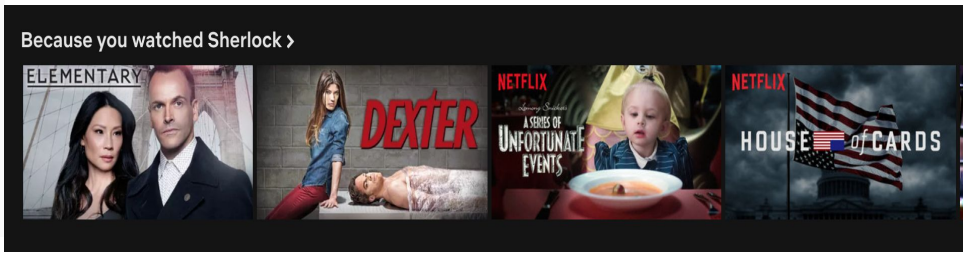
Our Options:
Collaborative vs Content-based



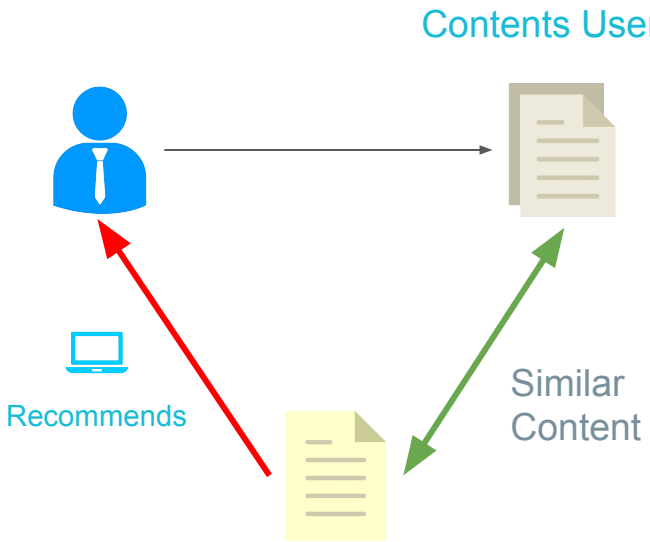
Icons made by Freepik from www.flaticon.com

Content-based Recommendation System

- Find an item similar to what the user likes
- Example: Netflix, Amazon



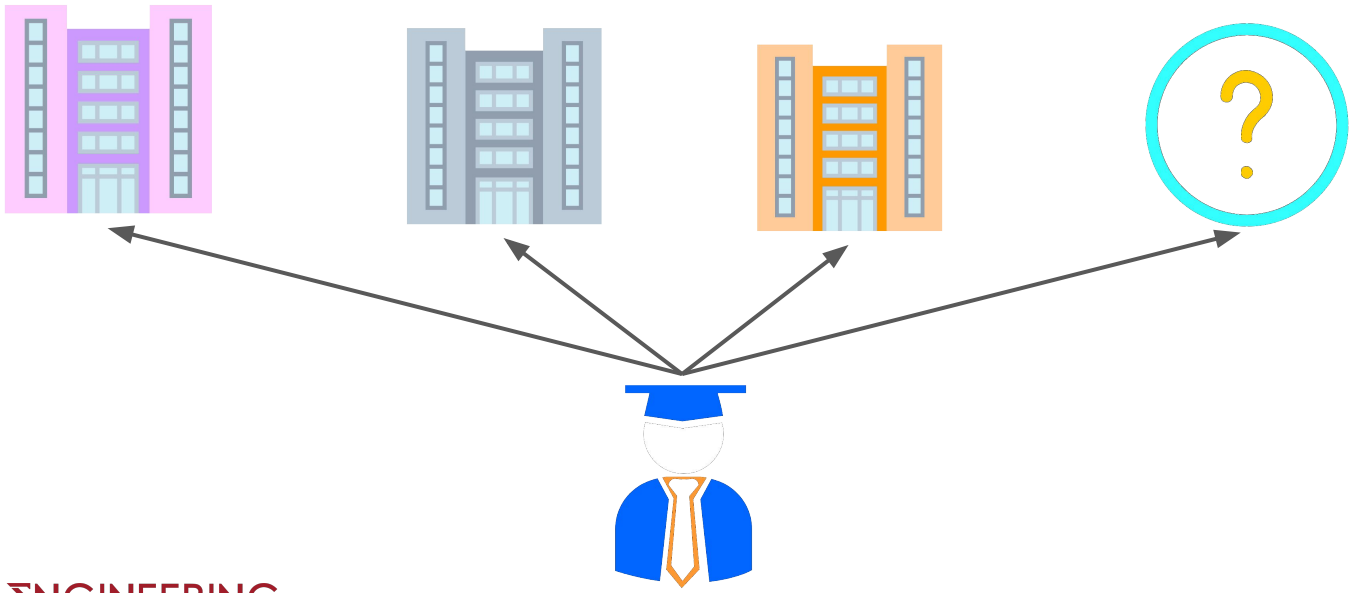
What We Need To Do:



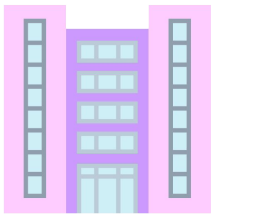
1. Find out what the user likes.
   - Which part of data do we want to use?
   - Rating? Browsing time? Browse count?
2. Construct a User Profile:
   - Which kinds of items does he like?
   - What is the similarity between the items the user likes
3. Match a new item to the User Profile
   - How much will the user like it?
4. Make Prediction!

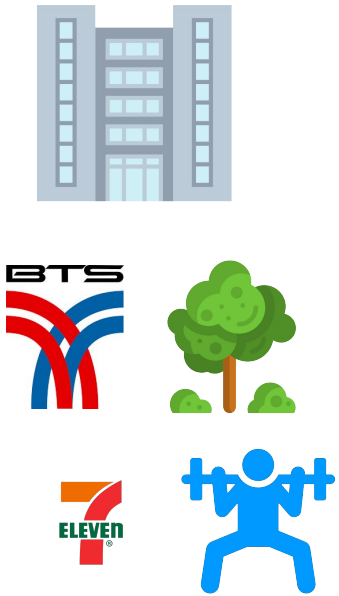
Eyeballing: What Do the Condos Have in Common?

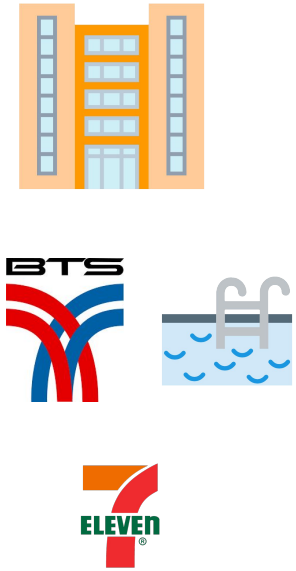
An Intuitive Example: Home browsing



Eyeballing: What Do the Condos Have in Common?



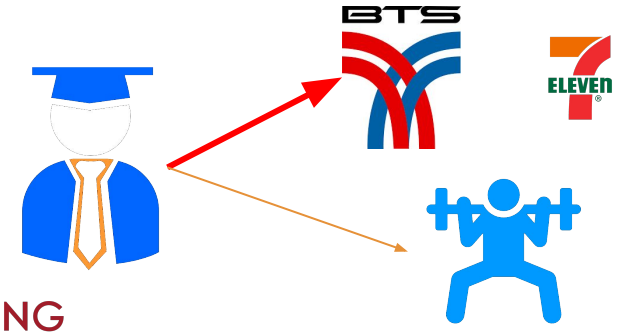




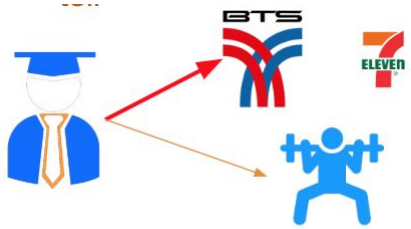
CHULA ENGINEERING
Foundation toward Innovation

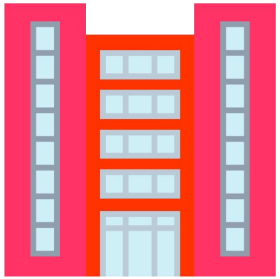
Thus, our customer's profile is:

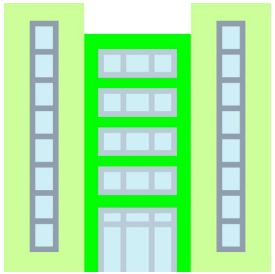
- Need to be close to the train system
- Need to be close to a convenience store
- Maybe prefer places with fitness center as well, we have too little data to tell



Thus, between

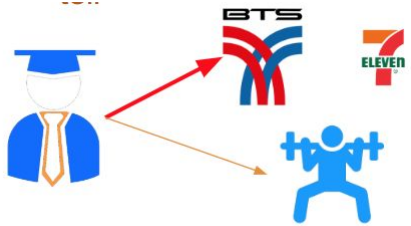


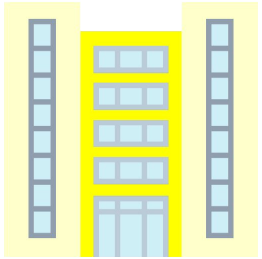


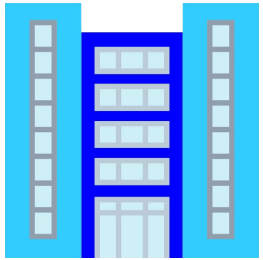


Which should we recommend?

What about?







The answer is not as obvious..

Thus we need a systematic way to tell how well a place fits out model

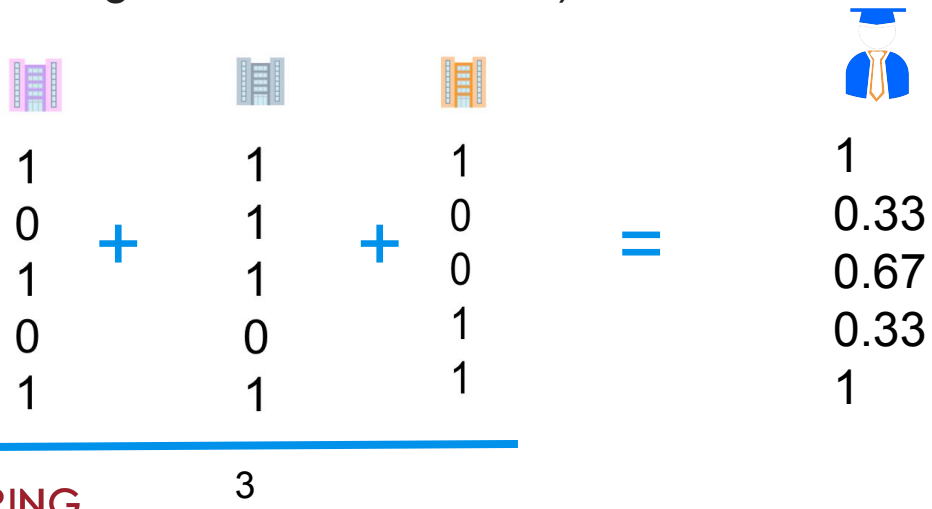
Item Profile

1 = have
0 = don't have

	1	1	1	1	0
	0	1	0	1	0
	1	1	0	1	0
	0	0	1	0	1
	1	1	1	0	1

Simple Way: we know that the customer have browsed the first three home…

So we can create a user profile by a simple average of the item profile the user browse (Or we can also do weighted average or any more complexed algorithm if we want to!)



Now we can compare which is more similar!

BTS	1	1	0
	0.33	1	0
	0.67	1	0
	0.33	0	1
7 ELEVEN	1	0	1

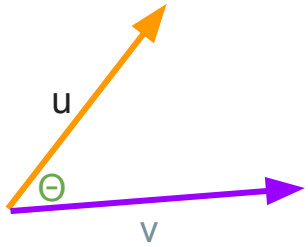
It's still not obvious… but at least we have numbers to work with!

Cosine Similarity

- Our profiles could be viewed as vectors
- We can use the Cosine Similarity to normalize and compare how similar the vectors are

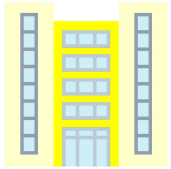
Recall that $u \cdot v = |u||v|cos\theta$

$$cos\theta = \frac{u \cdot v}{|u||v|}$$

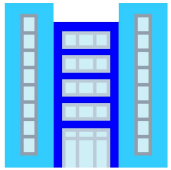


The value of cos Θ is called the Cosine Similarity, with value closer to one Indicating that the vectors point roughly in the same direction.

Example: Cosine Similarity



$$cos\theta = \frac{1 \cdot 1 + 0.33 \cdot 1 + 0.67 \cdot 1}{|1.633||\sqrt{3}|} = 0.707$$



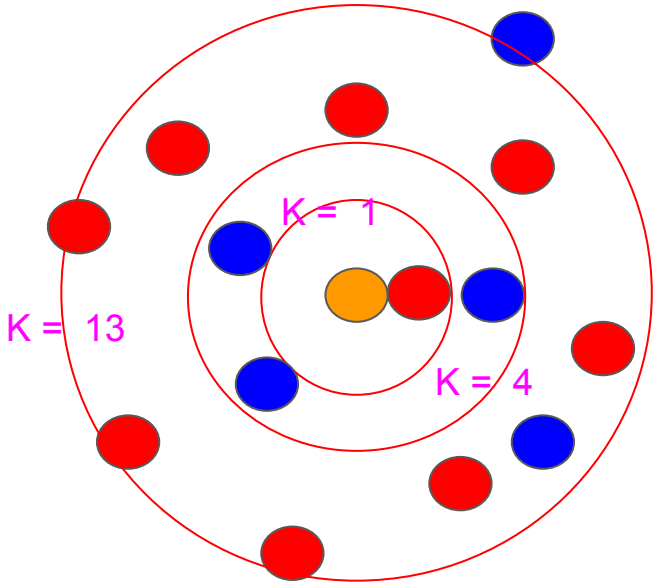
$$cos\theta = \frac{0.33 \cdot 1 + 1 \cdot 1}{|1.633||\sqrt{3}|} = 0.470$$

The yellow Condo is more similar to what our user has browsed!

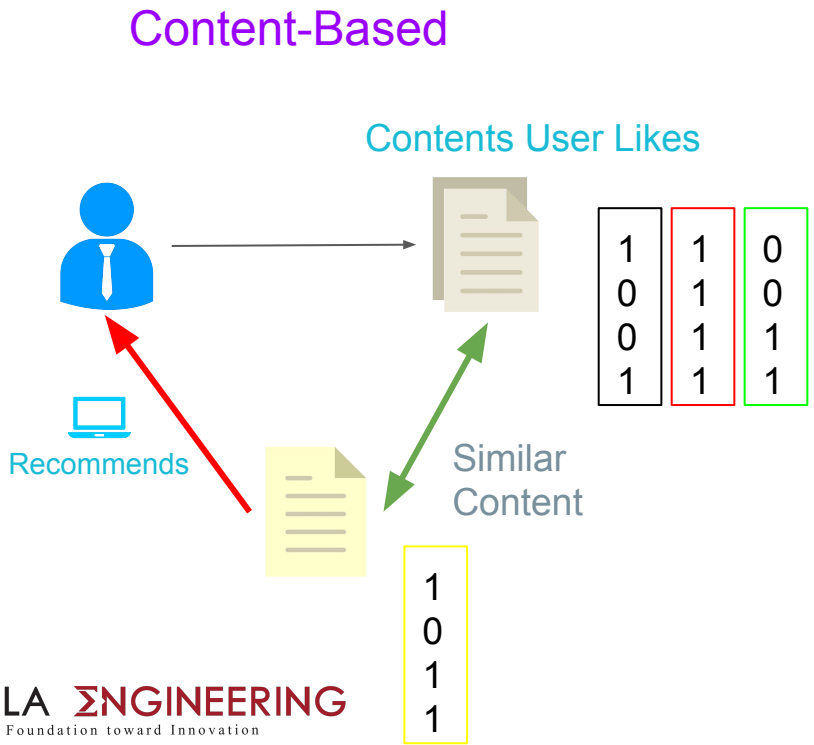
Other distance/similarity functions: Euclidean, Jaccard, Earth Mover, etc.

K-Nearest Neighbor

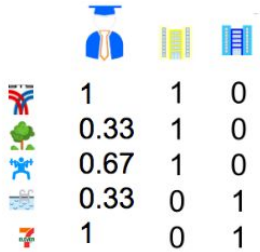
- Classify something based on k items that are most similar to it.

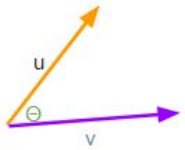


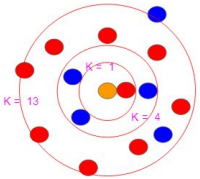
Let’s review: Content-based



- Represent items as features
- Create a user profile based on the items the user interacts with
- Find items based on the user profile







Context information

There's many information available

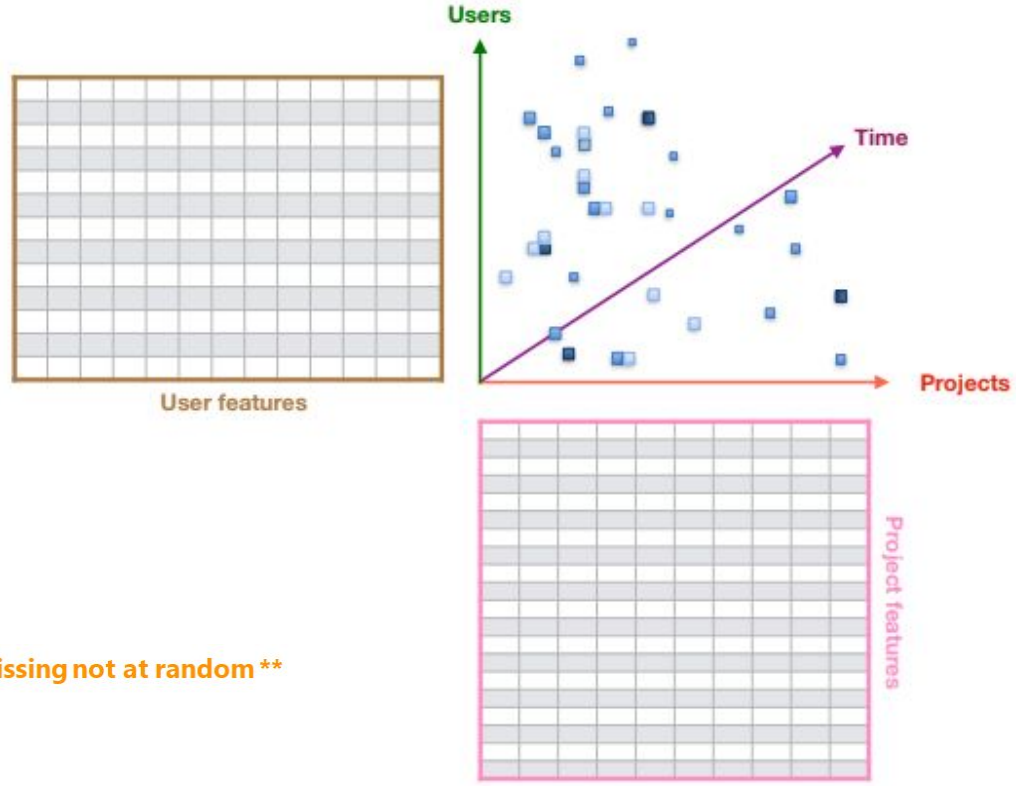
Ratings/views

Time

Missing ratings

Side information

- User information
- Item information



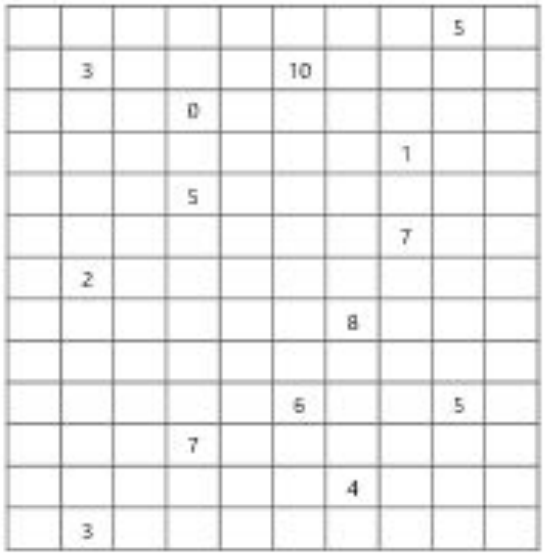
** Missing not at random **

Matrix view (ignoring time)

Conventional collaborative filtering tries to fill in the blanks of the view/rating matrices

- 0/1 matrix
- Rating matrix
- Implicit info matrix

How to fill?

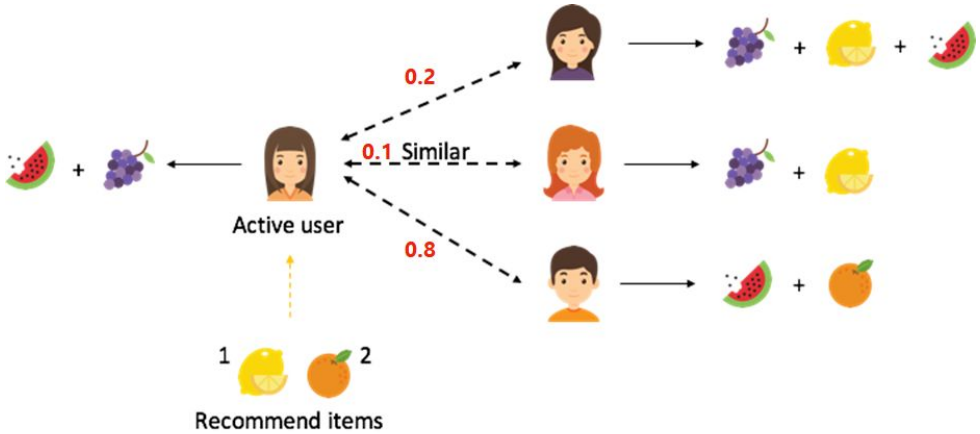


Collaborative filtering methods

- Neighbourhood
- Regression
- Matrix factorization
- Ensemble / Hybrid

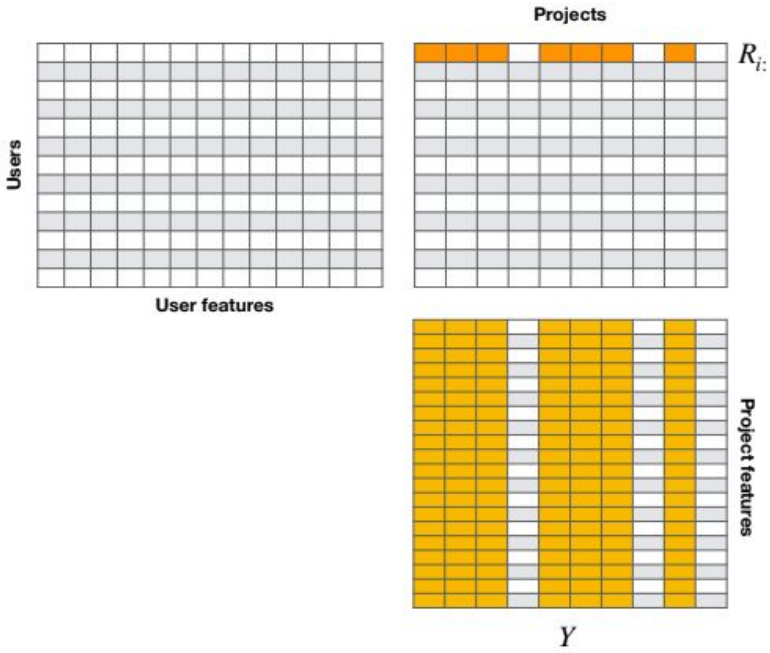
Neighborhood models

- User-user vs Item-item approach.
- Similarity methods.
- Neighbours aren’t independent.



Regression model

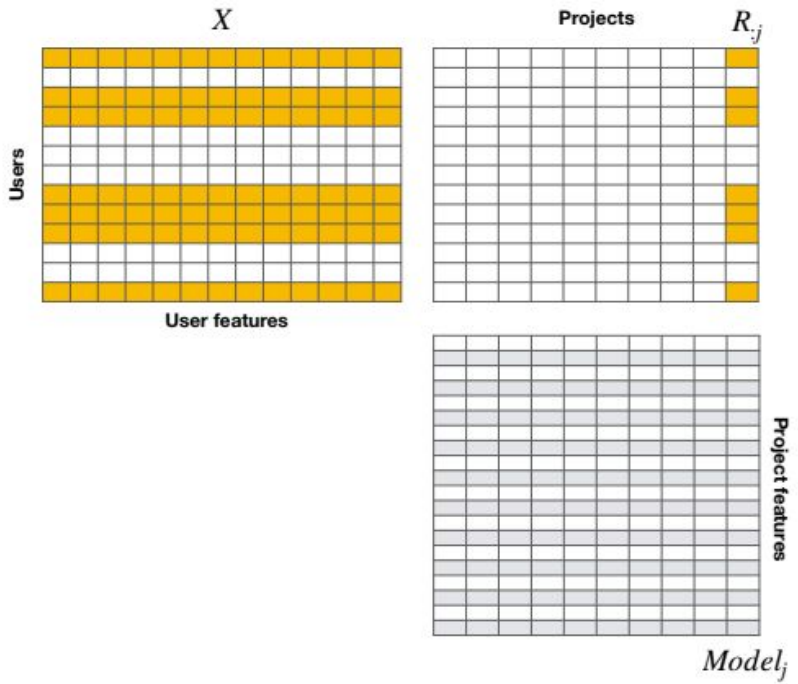
1. User-centric
    a. One model per user
    b. Usually ignore missing values (create imbalanced classes)



Regression model

1. User-centric
   a. One model per user
   b. Usually ignore missing values (create imbalanced classes)
2. Item-centric
   a. One model per item
   b. Same as user-centric

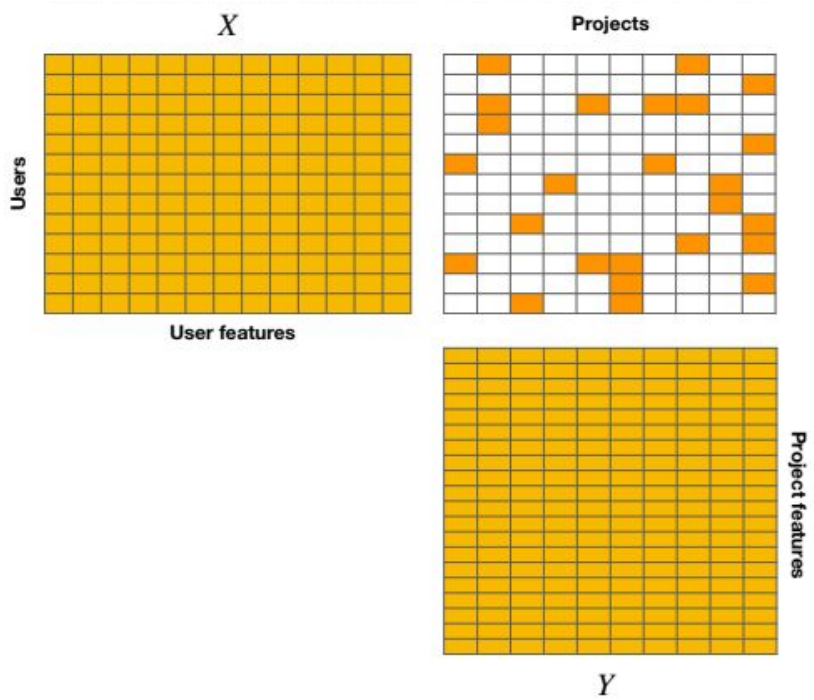
Problems?



Regression model

1. User-centric
   a. One model per user
   b. Usually ignore missing values (create imbalanced classes)
2. Item-centric
   a. One model per item
   b. Same as user-centric
3. User-item

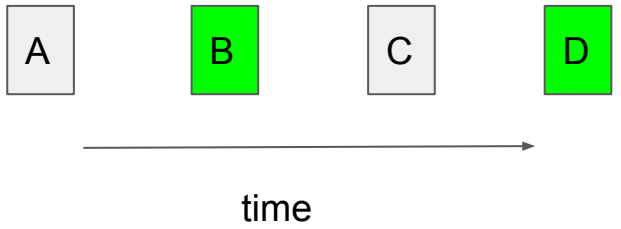
Problems?



Autoregressive model

Modeling time information (sequence)

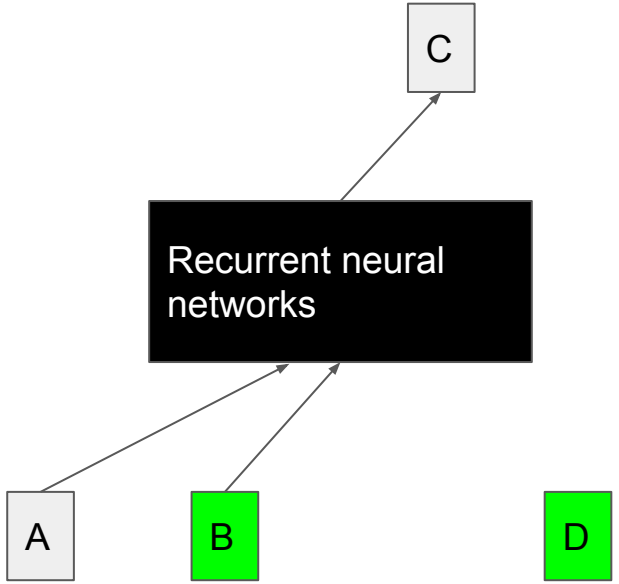
Recurrent Neural Networks



Autoregressive model

Modeling time information (sequence)

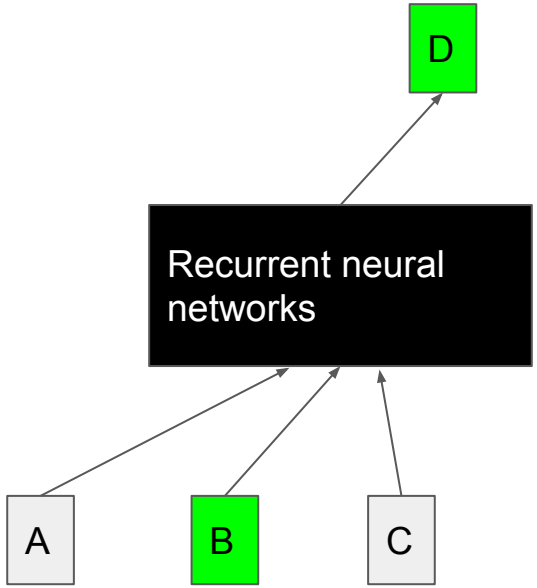
Recurrent Neural Networks



Autoregressive model

Modeling time information (sequence)

Recurrent Neural Networks



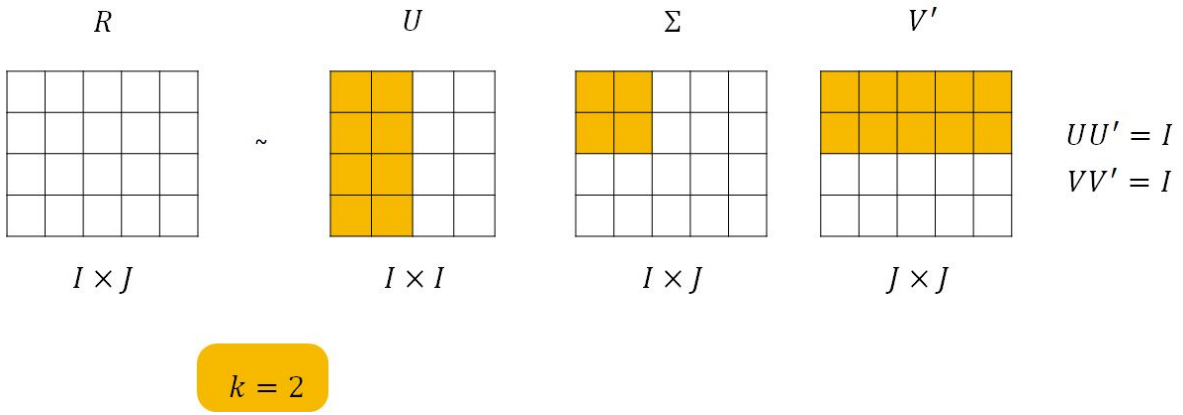
Problems?

Deep Neural Networks for YouTube Recommendations

Latent Cross: Making Use of Context in Recurrent Recommender Systems

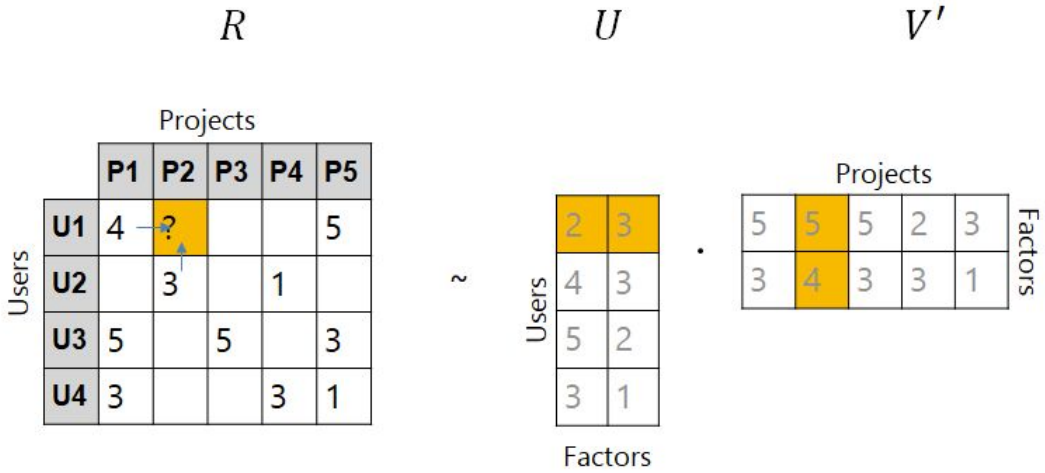
Matrix factorization

Singular Value Decomposition (SVD)



Matrix factorization

Matrix factorization



Objective function: $argmin_{U,V} \sum_{i,j} (R_{ij} - \sum_{k} U_{ik} V_{jk})^2 + Regularisation$

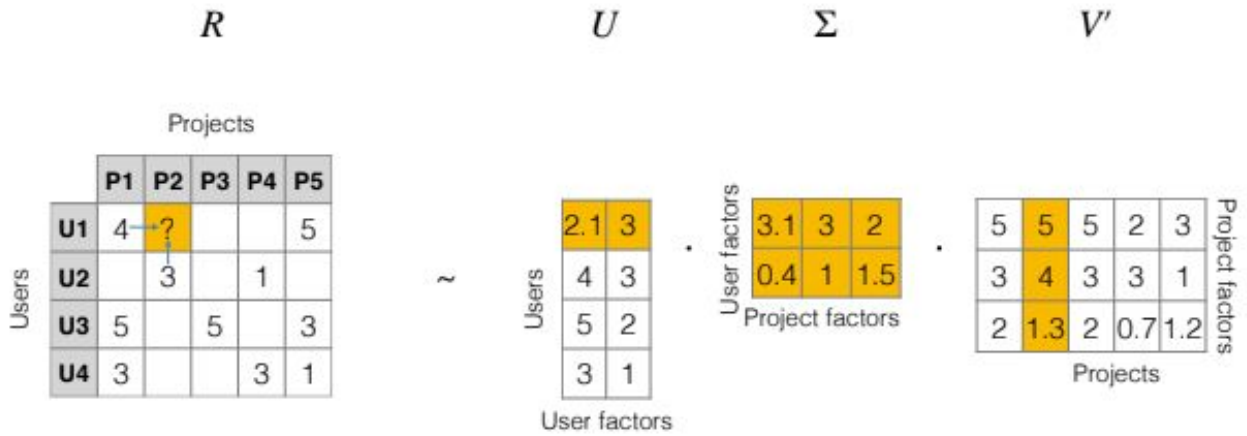
Python Packages:
-svds
-nimfa
-Implicit (https://github.com/benfred/implicit)
-pyMF (https://github.com/cthurau/pymf)

Variants

- Tri-factorisation
- Kernelised Bayesian MF (KBMF):
  - Gönen, Mehmet, Suleiman Khan, and Samuel Kaski. "Kernelized Bayesian matrix factorization." *International Conference on Machine Learning*. 2013.
  - Ammad-Ud-Din, Muhammad, et al. "Integrative and personalized QSAR analysis in cancer by kernelized Bayesian matrix factorization." *Journal of chemical information and modeling* 54.8 (2014): 2347-2359.
- NN meets MF:
  - Dong, Xin, et al. "A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems." *AAAI*. 2017.

Tri-factorization

Users and project factors do not need to lie in the same space



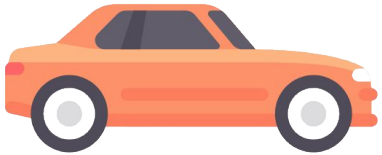
Ensemble methods

- No single model that use all information
  - Some model works well on certain kind of users
- Weighted combination (linear) or non-linear combination (classifier on top of classifiers)
- Can also train models on errors from previous models

What do we know about the items we want to recommend?

Some items come with easily recognizable attributes

Example: Cars



- Color
- Model
- Seats/Size
- Price
- Eco-friendly?

Others Don't

Example:

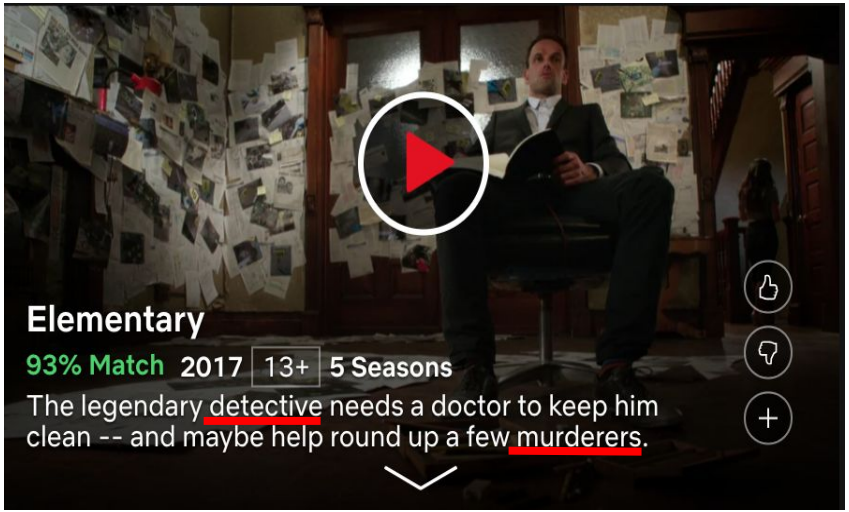
- Wikipedia Pages
- Online Articles

We have both!

Item Description

We can sometimes find related features from the description! (or sometimes name of the item)

Detective+Murderers => likely to be Crime shows

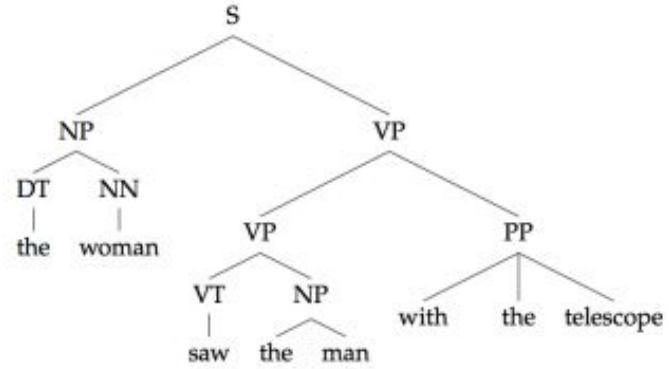


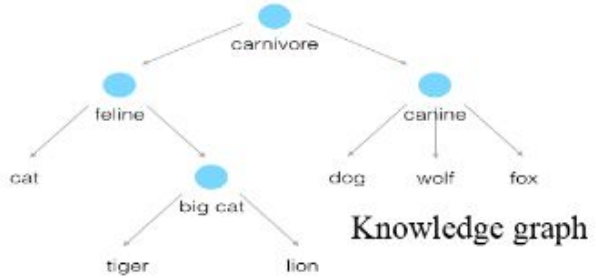
Representing text

Representing text is the most basic task of NLP

Word level

Sentence/Document level



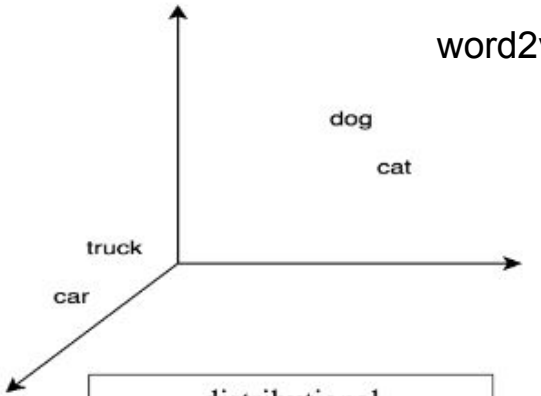


Knowledge graph

เสือ = [0 1 0 0 0 ... 0 0]

One-hot model

symbolic



distributional

TF-IDF:
Term Frequency
Inverse Document Frequency

Term Frequency (TF)

= How often does the word pop up in the description?

Inverse Document Frequency (IDF)

= How rare is it for a document to contain that word?

If a term 1) appears a lot in one document and 2) does not appear a lot in other document, then we assume it is **significant**.

TF-IDF calculation

- Term Frequency (TF) – per each document

$$TF(w) = \frac{\text{Frequency of word } w \text{ in a document}}{\text{Total number of words in the document}}$$

- Inverse Document Frequency (IDF) – per corpus (all documents)

$$IDF(w) = \log_e\left(\frac{\text{Total number of documents}}{\text{Number of documents that contain word } w}\right)$$

- TF-IDF = TF * IDF

TF-IDF

Word frequency table	ออเจ้า	มะระ	ไออ้อน	กิน
บุพเพสันนิวาส	1,000,000			5001
ไออ้อนเชฟ		3	5000	9999
ไออ้อนแมน			10000	3781
หมึกแดง		100		1090

If a term 1) appears a lot in one document and 2) does not appear a lot in other document, then we assume it is **significant**.

TF-IDF

Word frequency table	ออเจ้า	มะระ	ไออ้อน	กิน
บุพเพสันนิวาส	0.01 * 1.4			0.0005 * 0
ไออ้อนเชฟ		0.02 * 0.7	0.1* 0.7	0.08 * 0
ไออ้อนแมน			0.1* 0.7	0.04 * 0
หมึกแดง		0.06 * 0.7		0.08 * 0

$$TF(w) = \frac{\text{Frequency of word } w \text{ in a document}}{\text{Total number of words in the document}}$$

$$IDF(w) = \log_e\left(\frac{\text{Total number of documents}}{\text{Number of documents that contain word } w}\right)$$

TF-IDF

- We can use TF-IDF to transform item descriptions or title as a feature

- However, consider if there are 100,000 words that we use as features. Would this make our recommender system slow? (Sparse and high dimensional features)

- If we know หรู and มีระดับ often is included in luxury condos descriptions, maybe we can group descriptions by topics?

Topic modeling

- A description can contain several topic

คอนโดหรูสไตล์อังกฤษ แห่งแรกในเขาใหญ่ ที่ติด ถ.ธนะรัชต์ มากที่สุด 1 ห้องนอน 1 ห้องน้ำ 1 ห้องนั่งเล่น พร้อมห้องครัวแยกเป็นสัดส่วน

- Convert the proportion of topics into a set of numbers

[0, 0.2, 0, 0.45, 0, 0.35]

Eastern style topic, Western style topic, Facility topic, Layout topic, View topic, Location topic

Supervised topics modeling

If we want to create the representation this way, we need to specify the types of topics and the topics the words in the sentence belongs to.

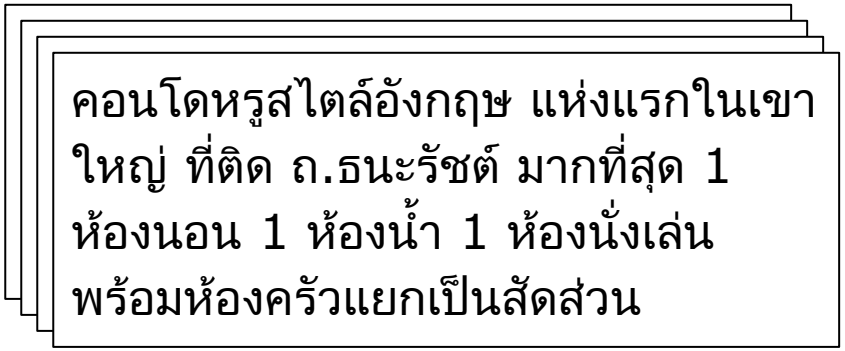
คอนโดหรูสไตล์อังกฤษ แห่งแรกในเขาใหญ่ ที่ติด ถ.ธนะรัชต์ มาก ที่สุด 1 ห้องนอน 1 ห้องน้ำ 1 ห้องนั่งเล่น พร้อมห้องครัวแยกเป็นสัดส่วน

[0, 0.2, 0, 0.45, 0, 0.35]

Eastern style topic, Western style topic, Facility topic, Layout topic, View topic, Location topic

Unsupervised topic modeling

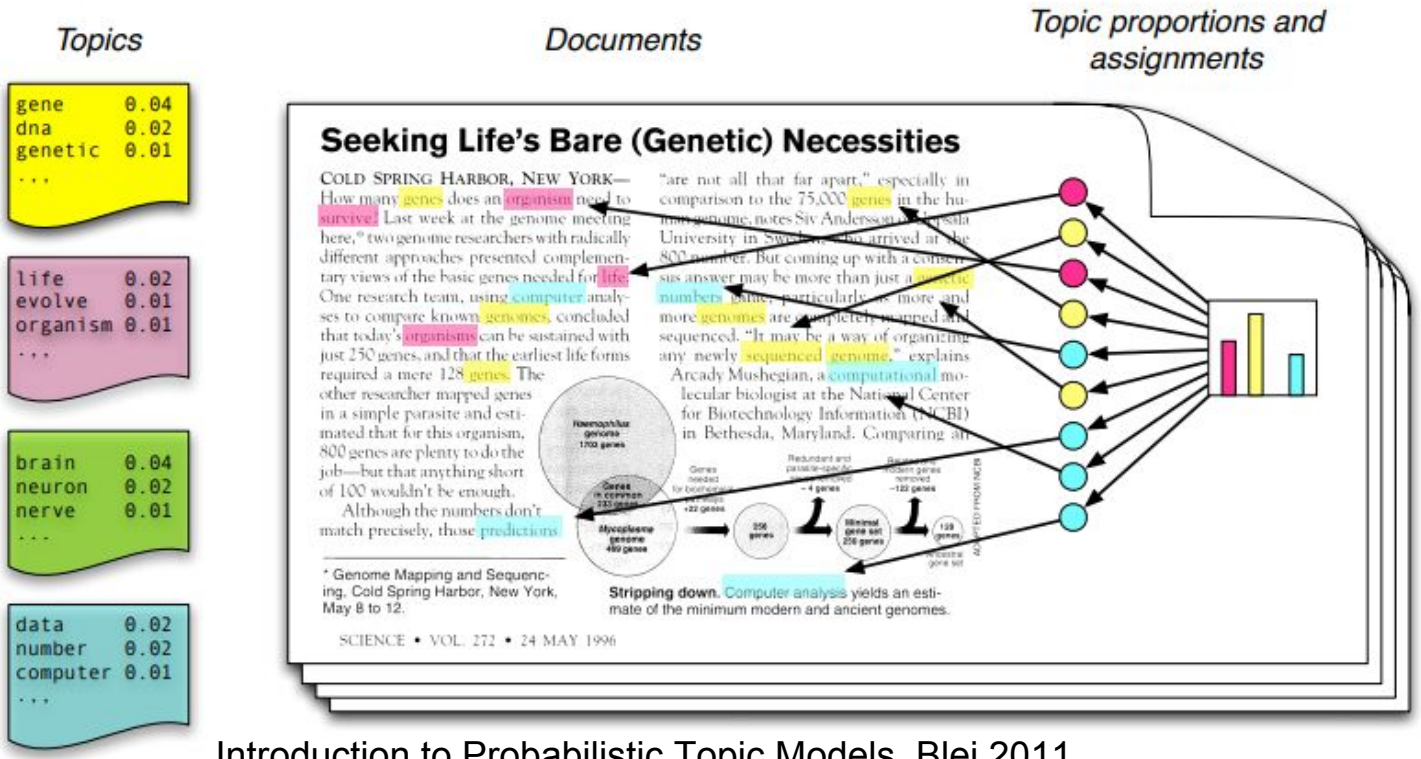
Can we let the machine learn the topics and the topic assignment by themselves?



Just give it a bunch of descriptions

Key observation: A document/description usually have sparse and coherent topics proportions

Latent Dirichlet Allocation (LDA)

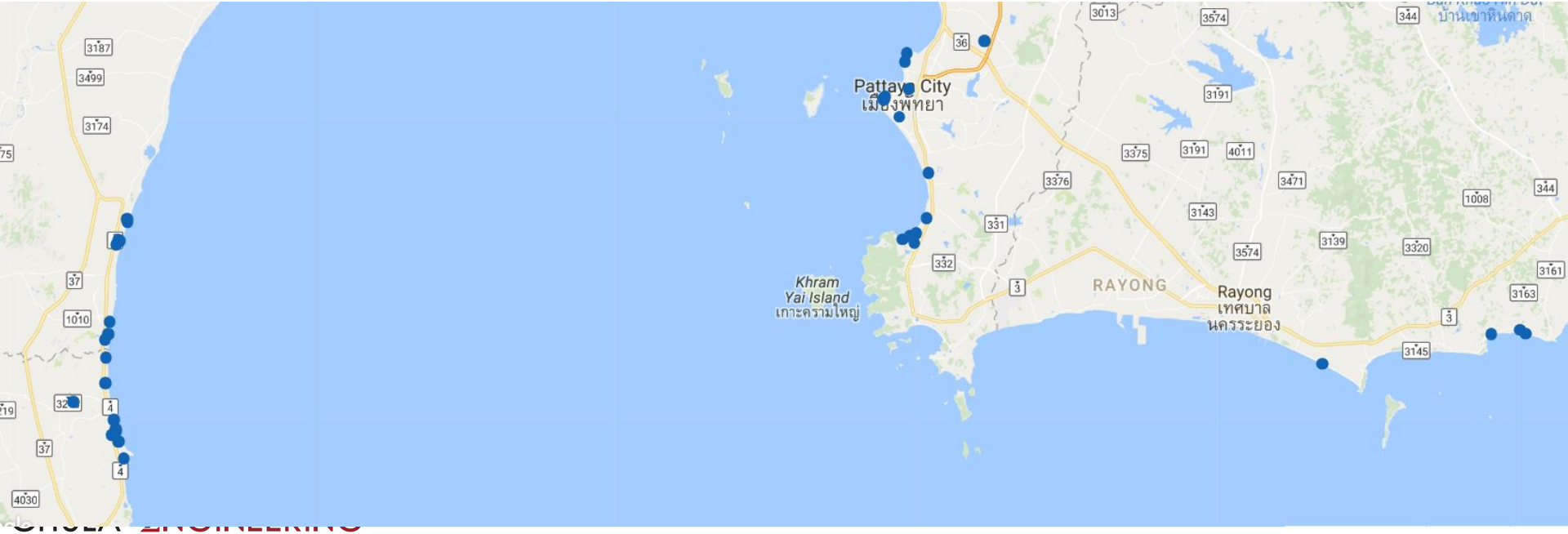


Automatically Discovers topics (Need to specify the number of topics)

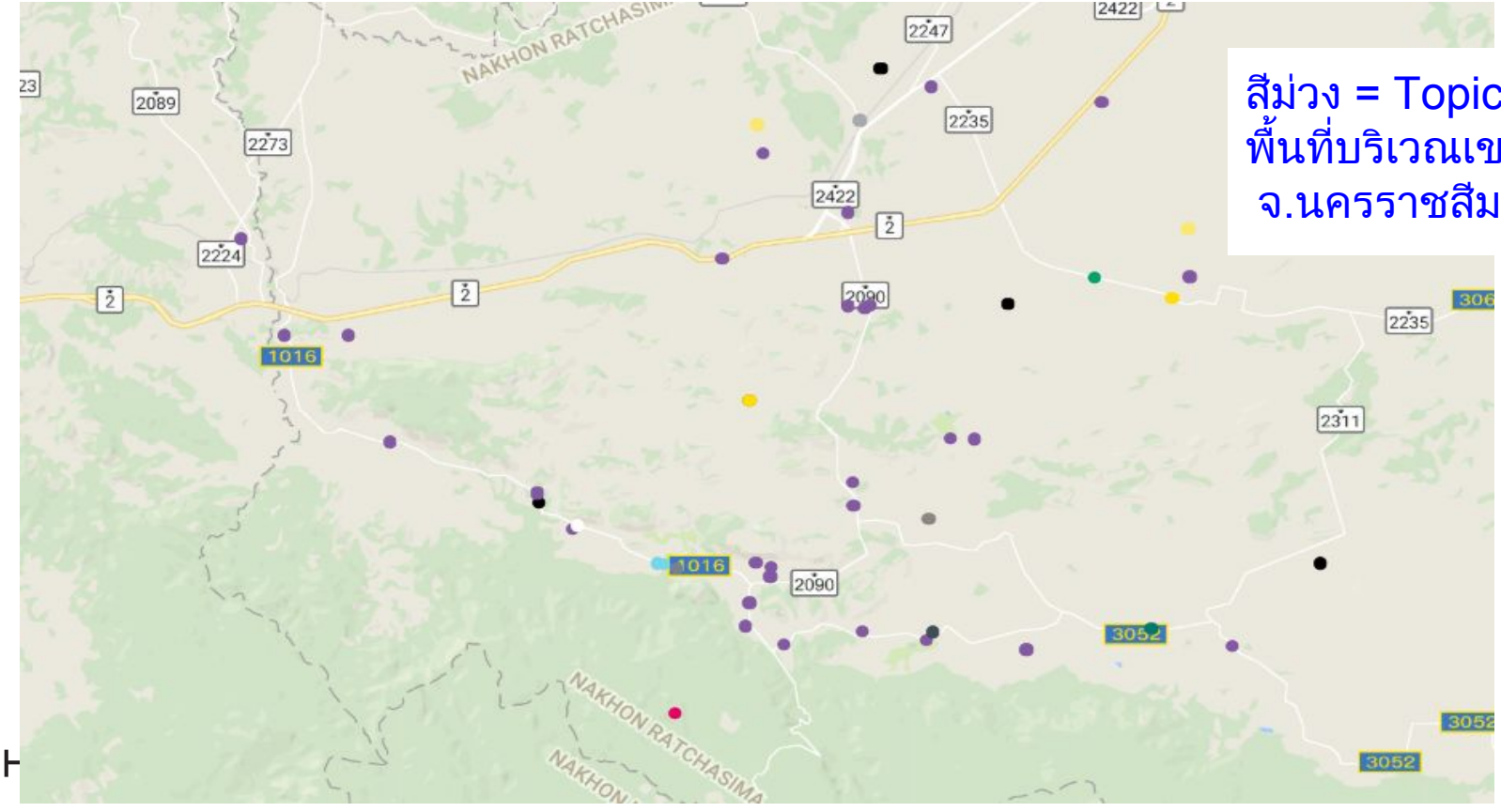
Introduction to Probabilistic Topic Models, Blei 2011
http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

LDA Examples

```
Topic 28
0.068*"วิว" + 0.058*"ทะเล" + 0.038*"คอนโด" + 0.029*"หัว" + 0.027*"คอนโดมิเนียม" + 0.025*"มองเห็น" + 0.023*"ทัศนียภาพ" + 0.
022*"ชายหาด"
```



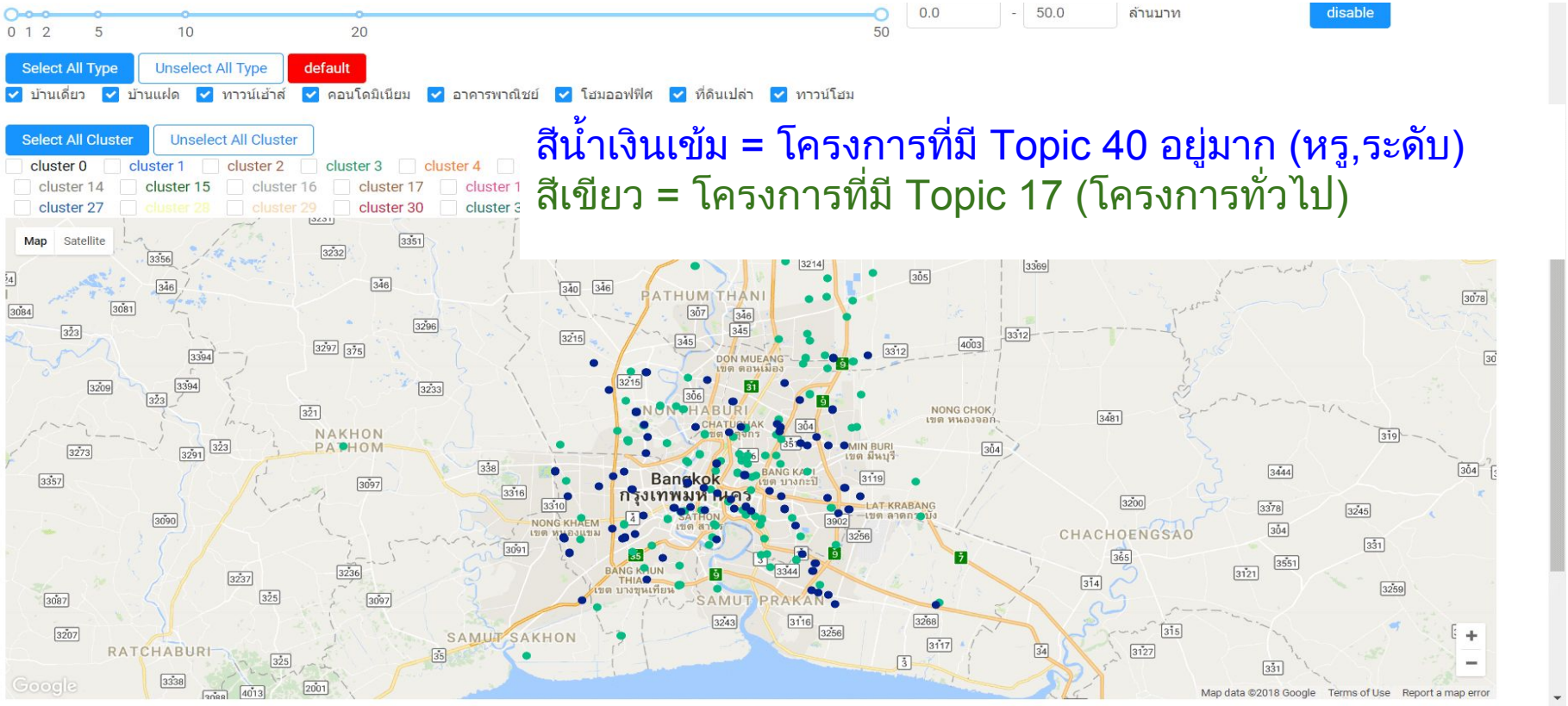
Topic 9
0.071*"ธรรมชาติ" + 0.031*"บรรยากาศ" + 0.028*"ร่มรื่น" + 0.027*"บ้าน" + 0.025*"ท่ามกลาง" + 0.025*"สวน" + 0.025*"สัมผัส" + 0.021*"พื้นที่"



สีม่วง = Topic 9
พื้นที่บริเวณเขาใหญ่
จ.นครราชสีมา

CH

Topic 40
0.115*"ระดับ" + 0.066*"เหนือ" + 0.046*"หรู" + 0.031*"ทำเล" + 0.026*"ชีวิต" + 0.026*"ใช้ชีวิต" + 0.016*"สไตล์" + 0.016*"สะท้อน"
Topic 17
0.077*"พื้นที่" + 0.060*"ออกแบบ" + 0.045*"โล่ง" + 0.039*"โปร่ง" + 0.038*"ใช้สอย" + 0.020*"ประโยชน์" + 0.018*"ห้อง" + 0.017*"อาคาร"



สีน้ำเงินเข้ม = โครงการที่มี Topic 40 อยู่มาก (หรู,ระดับ)
สีเขียว = โครงการที่มี Topic 17 (โครงการทั่วไป)

Other ways to represent a document

Words to vectors - word2vec

Document to vectors - Averaging word2vec, doc2vec and variants

We do not claim LDA is better than these but it’s cool to talk about in the workshop :)

LDA is interpretable though

Evaluation metrics

Recommendation system commonly used metrics

Best metric is via A/B testing: click through rate, conversion rate, time spent on content, etc.

Mean Square Error (MSE)

Mean Average Precision (MAP@K)

normalized Discounted Cumulative Gain (nDCG)

Evaluation: Precision and Recall

- **Precision** is the **correctness** of our recommendation

  Precision = Number of Correct Recommendation / Number Recommended

- **Recall** is the ability of our recommendation to obtain the correct recommendation

  Recall = Number of correct answer that get recommended / number of correct answers

Precision and Recall Example

We recommend [H,F,T,A,E],

the correct answer is [A,B,C,D]

- We got 1 item(A) correct out of 5 recommended items, so our precision is

  ⅕ = 0.2

- We got 1 item(A) out of 4 correct answers, so our recall is ¼ = 0.25

Evaluation: Average Precision

- Average Precision = the Sum of (precision at i) x (the change in recall)
- Example: We recommend [E,K,A,P,O,L], while the correct answer is [A,B,C,D,E]
- We get two items right at position 1 and 3
- Precision@1 = 1/1 = 1 Precision@3 = ⅔ = 0.67
- Note: Change in recall is always equal to 1/(number of correct items) = ⅕
- Average Precision = sum of precision over correct items/number of correct items = (1+0.67) = 1.67/5 = 0.334

- What if Instead of [E,K,A,P,O,L] we recommend [E,A,K,P,O,L]
- AP = (1/1+2/2)/5 = 0.4 **AP rewards correct rankings!**

Evaluation: Mean Average Precision at K (MAP@K)

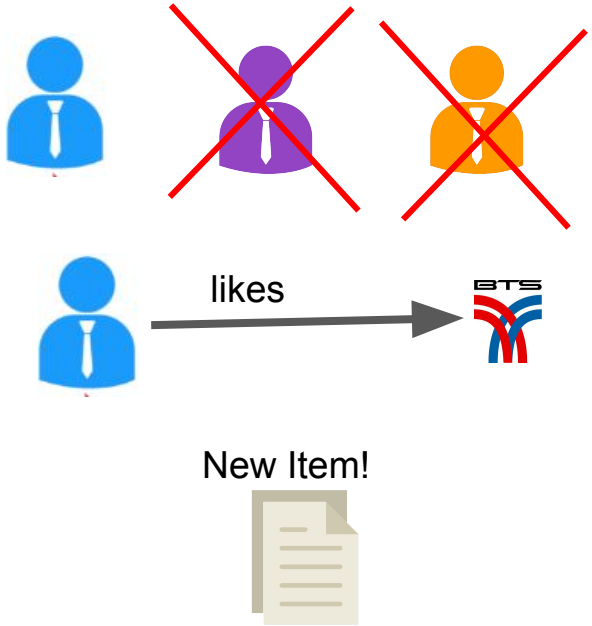
- Mean Average Precision is simply the mean of AP for multiple users

- In MAP@K, we only care about the first **K** recommendations
  - You are not penalized for incorrect answers in the K recommendations. Send K answers!
  - Change in recall = 1/min(K,number correct answer)

- MAP value both **correctness** of recommendation and **correct rankings.**

Why should we use Content-based over collaborative filtering?

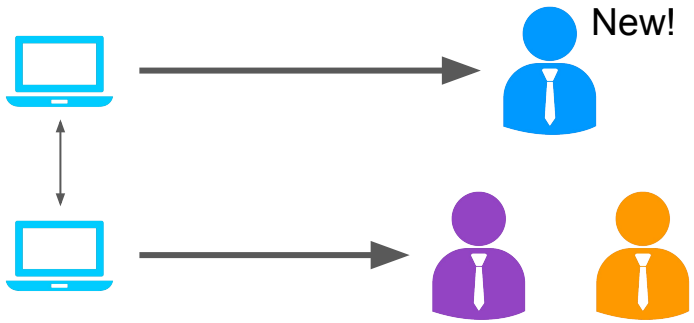
- We don’t need to know about any other users to make a prediction
- Can explain why an item is recommended (What attribute in the item that the user likes)
- Works well on new items with no reviews from other users



Cold Start Problem

- Some recommenders (content-based and collaborative filtering) build user profiles based on past histories
  - We do not have any history for a first time user!
  - Or too little history

- Potential Solution: Hybrid Recommender
  - Some recommenders do not rely on user history(non-personalized recommender etc.)



Cold Start Problem - Item side

- Collaborative filtering methods cannot give suggestions for new item that appears
- Potential Solution: suggestion based on similar items (item features)

Lab overview

1 Data exploration and preprocessing

2 NLP (word segmentation + LDA)

3 Content-based recommendation

4 MF-based recommendation

Hackathon link: http://bit.ly/homehack18rec