Final Project

NTU Data Mining 2014
TA: Jhao-Yin Li
Lecturer: Prof. Ming-Syan Chen

Topics

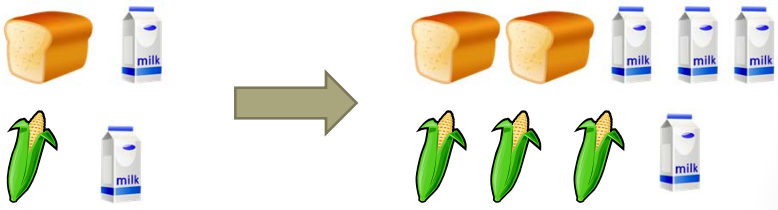
- TOPIC1: Distributed Mining
  - Project 1: Distributed Pattern Mining in Hadoop
- TOPIC2: Link Prediction
  - Project 2: Co-authorship Prediction
- TOPIC3: Social Influence
  - Project 3: Adoption Prediction

Important Information

- 2-3 people/1 Team (shown in abstract)
- Sum up the last digits of your student IDs
  - 3N+1 -> P1
  - 3N+2 -> P2
  - 3N -> P3
- Average scores for each project will be adjusted to approximately the same
- Abstract due: 2014/11/23 22:00 (Sun.)
- Presentation: 2014/12/31 or 2015/1/7
- Report due: 2015/1/22 22:00 (Thu.)

Project 1:
Distributed Pattern Mining in Hadoop

- Goal: Design a pattern mining algorithm which supports multiple occurrences of an item in the same transaction and implement your proposed algorithm in Hadoop

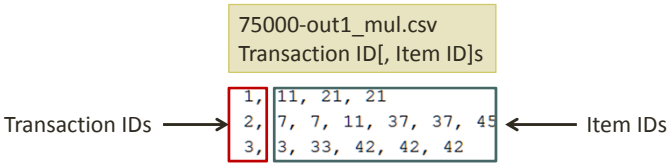


Project 1 Tips

- You should use Hadoop to implement this project and output the correct answers efficiently
- Your code should preserve the flexibility to set any minimum support threshold for the users
- You may extend Apriori or FP-growth or design your own pattern mining algorithm to support multiple occurrences of an item
  - p.s. You may refer to the existing implementation/concept of pattern mining in Hadoop for your own implementation, e.g.
    - Apriori on Hadoop: http://sourceforge.net/p/apriorimapred/wiki/Home/

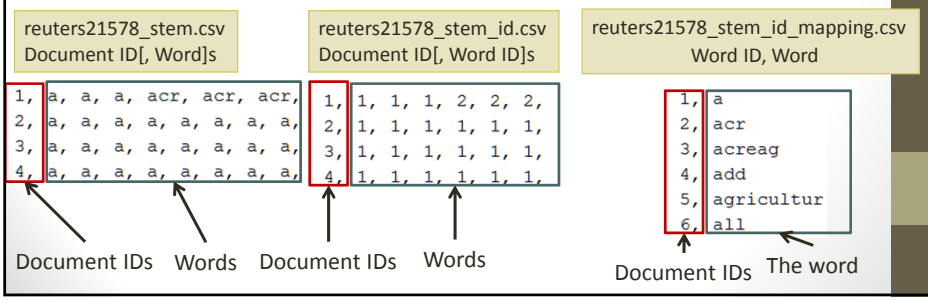
Project 1 Data Set #1

- **Extended Bakery dataset**
  - pastry items and coffee drinks
  - 75,000 receipts (no item information)
  - 75000-out1.csv: original file
  - 75000-out1_mul.csv: multiple occurrences of an item in the same transaction (synthetic, 0~2 duplicated, i.e. 1~3 occurrences of an item)



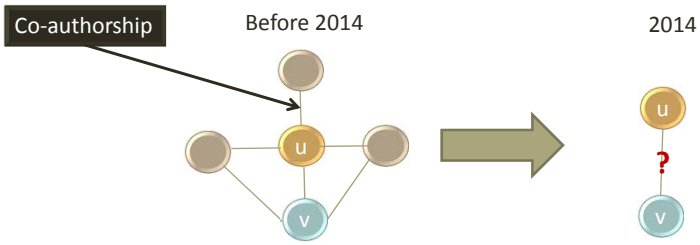
Project 1 Data Set #2

- **Reuters21578**
  - 19042 preprocessed Reuter news documents
  - reuters21578.csv: lowercasing
  - reuters21578_stem.csv: lowercasing + stemming
  - “”_id.csv: “Word“ -> “Word ID”
  - “”_id_mapping.csv: the mapping of “Word ID“ and “Word”



Project 2: Co-authorship Prediction

- Goal: Given two author u and v and history data on collaborations between authors, predict whether u and v will collaborate with each other again (Co-authorship: two authors collaborate with each other in a published paper)



Project 2 Data Set

- A small collaboration network (CN1)
  - authors: 175899
  - collaborations: ~0.8 million
- A large collaboration network (CN2)
  - authors: 484990
  - collaborations: ~3.7 million
- Do tests by yourselves
  - Time slices
    - E.g. Train: 2008~2009, Test: 2010

CN1.txt
author1 author2 year conference paper

```
0 1 2008 0 0
2 3 2008 1 1
3 4 2008 1 1
3 2 2008 1 1
5 6 2008 2 2
```

e.g. Authors 2, 3 & 4 has a Paper 1 on a conference/journal 1 in 2008.

Project 2 Validation

- Dataset: CN3.txt
  - 410827 authors and ~2.9 million collaborations between 2008 and 2012
- Input: CN3_query.txt
  - 20000 author pairs which appear in "CN3.txt"
  - 10000 pairs have collaborations in 2013.
- Please predict whether author1 will collaborate with author2 in 2013. (1: yes; 0: no)
- Output: CN3_answer.txt
- Baseline: All "1" or All "0"
  - Accuracy: 0.5

CN3_query.txt
author1 author2

```
100026 100029
100083 100084
100102 100103
10012 9318
100123 91604
100149 161462
100164 100180
```

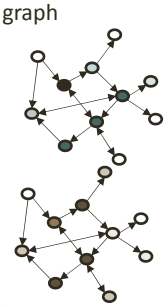
CN3_answer.txt
(just an example)

```
0
1
0
1
0
1
0
```

Project 3: Adoption Prediction

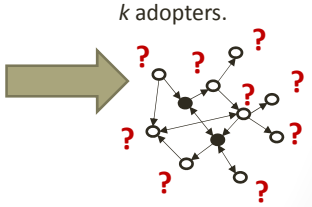
- Goal: Given a social network, a set of initial adopters of an idea *A* in this social network, and the adoptions on other ideas (except *A*), predict who will adopt the idea *A* in the following

The adoptions of some ideas (include time, degree, nodes) & social graph



+ The initial adopters of an idea *A*

Predict the subsequent *k* adopters.



Project 3 Training Data

- A social graph: "graph.txt"
  - data format:
    - node_id its_neighbor_1 its_neibhbor_2....
    - e.g. for node_id = 4, there are 6 neighbors
      - 4 180 188 190 194 197 199
  - The graph is undirected
- A list of adoptions: "training.txt"
  - data format:
    - node_id idea_id time degree
  - Each line of the list records that a *node* adopts a certain *idea* in a specific *time* and the *degree* of this adoption is known
  - The value of *degree* is ranging from 0 to 1
    - 1 is strongly positive, while 0 is strongly negative
  - The list contains 1000 ideas

graph.txt
node_id neighbor_1 … neighbor_k

```
3 271 294 309 324
4 180 188 190 194 197 199
5 104 194 808 890 1064 1070
6 12 46 97 142 291 428 438 4
7 412 431 528 532 533 539 54
8 45 69 162 401 656 661 665
```

tranining.txt
node_id idea_id time degree

```
16450 3 2007/01/01 0.8
18815 4 2007/01/01 0.6
5971 5 2007/01/01 0.7
5971 6 2007/01/01 0.7
5971 7 2007/01/01 0.7
5971 8 2007/01/01 0.4
14317 9 2007/01/01 1
```

Project 3 Testing Data (1/3)

- Given a set of initial adopters of an idea, please find the subsequent 100 adopters of the idea
  - You can report less than 100 nodes (no more than 100 nodes)
  - The order of node IDs does not matter.
- Please evaluate your answer on testing data by F1-score, i.e. F-Measure

	actual class (expectation)	
predicted class (observation)	**tp** (true positive) Correct result	**fp** (false positive) Unexpected result
	fn (false negative) Missing result	**tn** (true negative) Correct absence of result

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

F-measure is the harmonic mean of precision and recall:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

reference: http://en.wikipedia.org/wiki/F1_scores

Project 3 Testing Data (2/3)

- Do tests on the following 3 datasets and report their F1-scores in your presentation and your final report. Each set contains 10 ideas. Each line in the file belongs to the same idea. The order of the nodes in each line follows the temporal order. Thus, the first node in a line is the first one who adopts the corresponding idea
  - Use the first 10% initial adopters as the question ("test_data_q1.txt")
    - The following 90% are given for you to calculate F1-score ("test_data_a1.txt")
  - Use the first 20% initial adopters as the question ("test_data_q2.txt")
    - The following 80% are given for you to calculate F1-score ("test_data_a2.txt")
  - Use the first 30% initial adopters as the question ("test_data_q3.txt")
    - The following 70% are given for you to calculate F1-score ("test_data_a3.txt")
- You may split the training data in order to do more tests
- ps. We only reveal who are the first X%. We do not reveal the information of *time* and *degree*

Project 3 Testing Data (3/3)

e.g. test_q*i*.txt
node1_id node2_id ... (in temporal order)

```
18815 9068 2516 14186 2844 17678 20889 4865 3805 17640 17639 8510 17647
27753 14955 21439 12145 7252 20520 16639 16206 10658 26429 8436 5770 22672 15
18815 9415 21454 15057 14317 14955 3838 17640 12961 17647 8172 17678 25340 25
4865 17678 17640 9415 13847 2858 24775 9068 21965 2787 21477 25667 15777 6071
29003 23579 1185 7398 7860 27066 10219 7173 2177 3752 10538 24261 16460 21089
20108 3403 3805 17640 8352 7494 14955 25030 4385 12734 9081 17678 15053
```

Each line in the file represents the first X% adopters of an idea.
Thus, in the "test_a*i*.txt" represents the last (100-X)% adopters of an idea.
The format of "test_a*i*.txt" is the same as the format of "test_q*i*.txt"".

Project 3 Validation

- The problem is the same as the one mentioned in the testing.
- The setting is similar. There are 3 validation datasets:
  - "valid_data_q1.txt" (the first 10% adopters)
  - "valid_data_q2.txt" (the first 20% adopters)
  - "valid_data_q3.txt" (the first 30% adopters)
- Please output your answers into a file named "teamID_*i*.txt" for each validation dataset
  - e.g. If you belong to Team 3 and the answers are for the first validation dataset, the file name should be "3_1.txt"
  - Remind:
    - You can report less than 100 node IDs (no more than 100 nodes)
    - The order of node IDs does not matter
- TA will evaluate your answers by F1-score finally

Project 3 Baseline

- A simple base line:
  - Suppose that *n* initial adopters are given and we want to find *k* subsequent adopters
  - Let *C* contain all neighbors of *n* initial adopters
  - Let *C'* contain all neighbors of nodes in *C*
  - Rank the nodes in *C'* by the following score function and extract top *k* nodes as the answer
    - For a node *u*, its score function is *s(u)* and *s(u)="the number of initial adopters who is u's neighbor"*
- Your results should beat the baseline ☺

Hand In

- All projects
  - Presentation slides (Before your presentation)
  - Report and the followings
- Project 1
  - Your Java source codes
  - Precompiled Jar of your Java source codes for Hadoop
  - ReadMe
  - Environment & Usage
- Project 2
  - Your source codes (no restrictions on programming languages)
  - ReadMe
  - Environment & Usage
  - Your answers for the validation dataset
- Project 3
  - Your source codes (no restrictions on programming languages)
  - ReadMe
  - Environment & Usage
  - Your answers for three validation datasets