Workshop Week 11 - COMP20008 2021

1. Suppose you are conducting data linkage between two databases, one with m records and the other with n records (assume $m < n$). Under a basic approach, $m \times n$ record comparisons will be needed.

   - Assume there are no duplicates. What is the maximum number of record matches? What is the corresponding number of non-matching comparisons required in this circumstance? **Answer: maximum number of record matches is m, number of non-matches is m* (n-1)**

   Now suppose a blocking method is employed, where each record is assigned to exactly one block. Assume this method results in b number of blocks.

   - What is the smallest possible number of comparisons? **0 (all blocks contain only records from one of the two databases)**
   - What is the smallest possible number of comparisons that will return all matches? **(m)** What is the value of b? **(anything between m+ 1 and n)**
   - What is the largest possible number of comparisons? **($m \times n$, when $b = 1$)**
   - If records are evenly allocated to b blocks, how many comparisons will be needed? $(m/b) \times (n/b) \times b = m \times n/b$
   - What is the advantage with a large b? What is the advantage with a small b? **(fast, inaccurate vs slow, accurate)**
   - In practice, a record is assigned to more than one block and records are not evenly allocated to blocks. How would this affect your analysis of large b vs small b?
     - **Dominant block size is the bottleneck for efficiency. So, even if b is large, the blocking can still lead to inefficient record linkage**
     - **Small b tends to correspond to inefficiency; but it does not automatically guarantee accuracy. If matches are all off blocks, accuracy can still be low.**
   - What is the advantage of records being assigned to multiple blocks? **To reduce the chance of matched pairs not allocated to a common block. Matched pairs may not have the same parts of the records in common, sometimes, a blocking key may miss a pair but captured by another blocking key.**

2. Explain how the data duplicate detection problem relates to the data linkage problem. **Data duplicate detection refers to finding records in the same dataset that belong to the same entity. For example in a medical dataset if a patient changed their name, we would like the history to link back to the same user. Thus in structured databases (or csv's) we could have two records from the**

same patient with different names although having identical data otherwise,

What are the major differences?**(When we match in the same dataset it is more likely that we have to do fuzzy (approximate) matching - as in the name change example. There could also be privacy issues if the matching is not done properly. How do we do matching without snooping at data?).**

3. One may evaluate the output of a data linkage system according to how many records are linked correctly and how many records are linked incorrectly.

   - What are the reasons two records could be linked incorrectly? **(They do not correspond to the same entity. E.g. two persons with accidentally the same name, but not the same date of birth)**
   - Suppose a false positive (FP) is two records that are linked by the system, which a human believes should not have been linked. Suppose a false negative (FN) is when two records are not linked by the system, which a human believes should have been linked. A true positive (TP) is two records linked by the system which a human believes should have been linked and a true negative (TN) is two records not linked by the system which a human believes should not have been linked.
     - What are the relative sizes of the categories TP, TN, FP, FN? In practice, how might one calculate these sizes?
       Construct the confusion matrix

	Actual:Link=true	**Actual:Link=false**
Prediction:link=true	**Small (TP)**	**Small (FP)**
Prediction: link=false	**Small (FN)**	**Very large (TN)**

     - It is desirable to minimise both FP and FN, but it may be difficult to minimise both simultaneously. Give an example application where minimising FP is more important than minimising FN. Give an example application where minimising FN is more important than minimising FP.

       More important to minimise FP: e.g. Centrelink - might want to avoid sending out debt recovery letters to people who don't actually owe a debt
       Minimise FN: e.g. Medical screening tests - Often used to flag people who may have a particular disease for further tests. Don't want a FN to mean that people who actually do have the disease don't receive further tests and treatment. Also national security, don't want a FN to mean that a potential terrorist plot is not followed up on.
       In practice there's often a trade-off involving the resources available for further investigations. We are willing to have a number of false positives to ensure we're avoiding false negatives, but don't want so many false positives that we can't manually screen and investigate them all.

4. Consider the following dataset:

User	Iron Man	Superman	Batman	Spiderman	Ant-Man	Wonder Woman
Anne	3		3	3.5	2.5	3
Bob	4	3.5	2.5	4	3	3
Chris	3	3	3			4
Dave		3.5	2	4	2.5	
Eve		3		3	2	5
Frank	2.5	4		5	3.5	5
Gary		4.5	3	4	2	

- Use the Item-based recommender systems approach discussed in lectures to predict

  Frank's rating for Batman. First comput averages for each movie:

  $$\{IronMan : 3.13, Superman : 3.58, Batman : 2.7,$$
  $$Spiderman : 3.92, Antman : 2.58, WonderWoman : 4\}$$

  Use these as imputed values for all missing entries.
  Then calculate the similarity scores with Batman for each movie using $sim(i_i, i_j) = \frac{1}{1+d(i_i,i_j)}$ where $d(i_i, i_j) = \sqrt{\sum_{k=1}^{m}(r_{ki} - r_{kj})^2}$:

  $$\{IronMan : 0.34, Superman : 0.27,$$
  $$Spiderman : 0.21, Antman : 0.36, WonderWoman : 0.20\}$$

  Finally, use the weighted average of the three most similar values to predict the final result: $(0.36 \times 3.5 + 0.34 \times 2.5 + 0.27 \times 4)/(0.36 + 0.34 + 0.27) = 3.29$
- Use the User-based recommender systems approach to predict Frank's rating for Batman First comput averages for each person:

  $$\{Anne : 3, Bob : 3.33, Chris : 3.25, Dave : 3, Eve : 3.25, Frank : 4, Gary : 3.38\}$$

  Use these as imputed values for all missing entries.
  Then calculate the similarity scores with Frank for each person using $sim(i_i, i_j) = \frac{1}{1+d(i_i,i_j)}$ where $d(i_i, i_j) = \sqrt{\sum_{k=1}^{m}(r_{ki} - r_{kj})^2}$:

  $$\{Anne : 0.245, Bob : 0.240, Chris : 0.284, Dave : 0.236, Eve : 0.257, Gary : 0.262\}$$

  Finally, use the weighted average of the three most similar values to predict the final result: $(0.284 \times 3 + 0.262 \times 3 + 0.245 \times 3)/(0.284 + 0.262 + 0.245) = 3$
- Identify the advantages and disadvantages of each approach **The item based approach works by considering the ratings of similar movies, while the user based approach works by considering the opinions of similar users. Item based measures tend to perform better in many practical cases. Generally, users are likely to be added more frequently than movies meaning the offline computation needs to be updated more often for a user-based approach. It's also very difficult to make recommendations to new users with a user-based approach**

5. Recommender systems are challenged by the "cold start" problem - how to make recommendations to new users, about whom little is known, and how to make recommendations about new items. Suggest three strategies that might be used to address this.
For new items:

- **Find similar items in current dataset (based on description, author, title, category, etc), take the mean (or other summarization) of these neighbours as the initialization of the new item**
- **Pay someone to rate the new item**

For new users:

- **Use similar users' data to initilize the new user (possibly, based on gender, age, region, etc)**
- **Ask user questions to obtain more data in the first place, e.g. providing a collection of items and asking user to chose the ones they like**
- **Use the most popular items as the initial suggestion system wide (overall, no enough data to provide good recommendation)**
- **Use content-based recomendation at first, after getting enough data, then move to collaborative filtering method such as matrix factorization**

6. Recommender systems are sometimes criticised for over-recommending popular items to users and under-recommending rarer items. Why do you think this happens? How might it be addressed? **A Cycle:**
popular items been recommended → users are more likely to give high rates to these item→ the popularity of these items increase
Recommender system(RS) has no understanding of the items themselves, over and under recommending naturally occur as the system is performing its job

Approaches to address this:

- **Adding an extra level to manipulate the results from RS, to ensure the diversity (cover a broad range of items)**
- **Giving weight to the timeliness of a item, to prevent users get the same recommendation all the time**