CS5001 final project, Matching ads and viewers

—— Mianyun Ni

1. **Design Process**

   In this project, we realize the matching between same numbers of viewers and ads with their preference rank.
   Each viewer/ads information is stored in .csv file and read with python reader in main function.
   To simplify data:
   Each viewer has four attributes: name(string), isyoung(int, I or 0), isman(int, I or 0), iscatowner(int, I or 0)
   Each ads has four attributes: name(string), isforyoung(int, I or 0), isforman(int, I or 0), hascat(int, I or 0)
   Use rank_viewer and rank_ads function to generate ads_preference and viewer_preference.
   These two algorithms base on the number's of matching attributes rate
   Eg. If one viewer is man, and one ads is for man, then their matching attributes rate will plus one.
   Use dictionary to store each viewer/ads mapping to matching attributes rate. Then sort by value to generate one queue of ads_preference and viewer_preference.
   Finally, output result.

2. **Final result:**

```
Lily    fruit graden
Tom     NBA game
Lisa    Cats and girls
Jack    dior
```

   Lily matches 'fruit graden', Tom matches 'NBA game', Lisa matches 'Cats and girls', Jack matched 'dior'

3. **Pros and cons of this project**

   Pros:

1.This project is extended. User can type in unlimited data in .csv to realize complex matching

3.The result is clear, name was show up directly

Cons:

1. This project is not supported viewers and ads matching with different numbers.
2. This project only supports four attributes.

Other than name, rest attribute is simplified as Boolean.

Eg. We only care about is young or not, didn't distinguish them as a specific number of age.

3. Ranking algorithm only cares the attributes exactly same condition, that's not accurate in real situation.

4. Discussion how to extend this project

1. Same as section3 Cons, this project can be extended by optimize ranking algorithm with more complex input data.
2. Also, if input viewers and ads have different numbers, (usually, ads numbers should be larger than ads numbers), matching algorithm can also support.
3. Matching algorithm can be optimized is time complexity. In this scenario, it is O(n^2). In the most unlucky situation, after we matched the first one viewer and ads, the following viewer is more suitable for the first ads, which means we need to linear scan whole back and replace the previous matching pairs
   Using Hungarian algorithm to optimize (O(n))

5. Acknowledgements

Hungarian algorithm - Wikipedia Hungarian algorithm

3.11.0 Documentation (python.org) python doc

(354) Hungarian Algorithm - YouTube youtube video