Hw4: Twitter Random Walk Algorithms

Course: Massive Data Analytics

Proffesor: Dr. Iman Gholampoor

Mohammad Mehdi Zare

* **Introduction**:

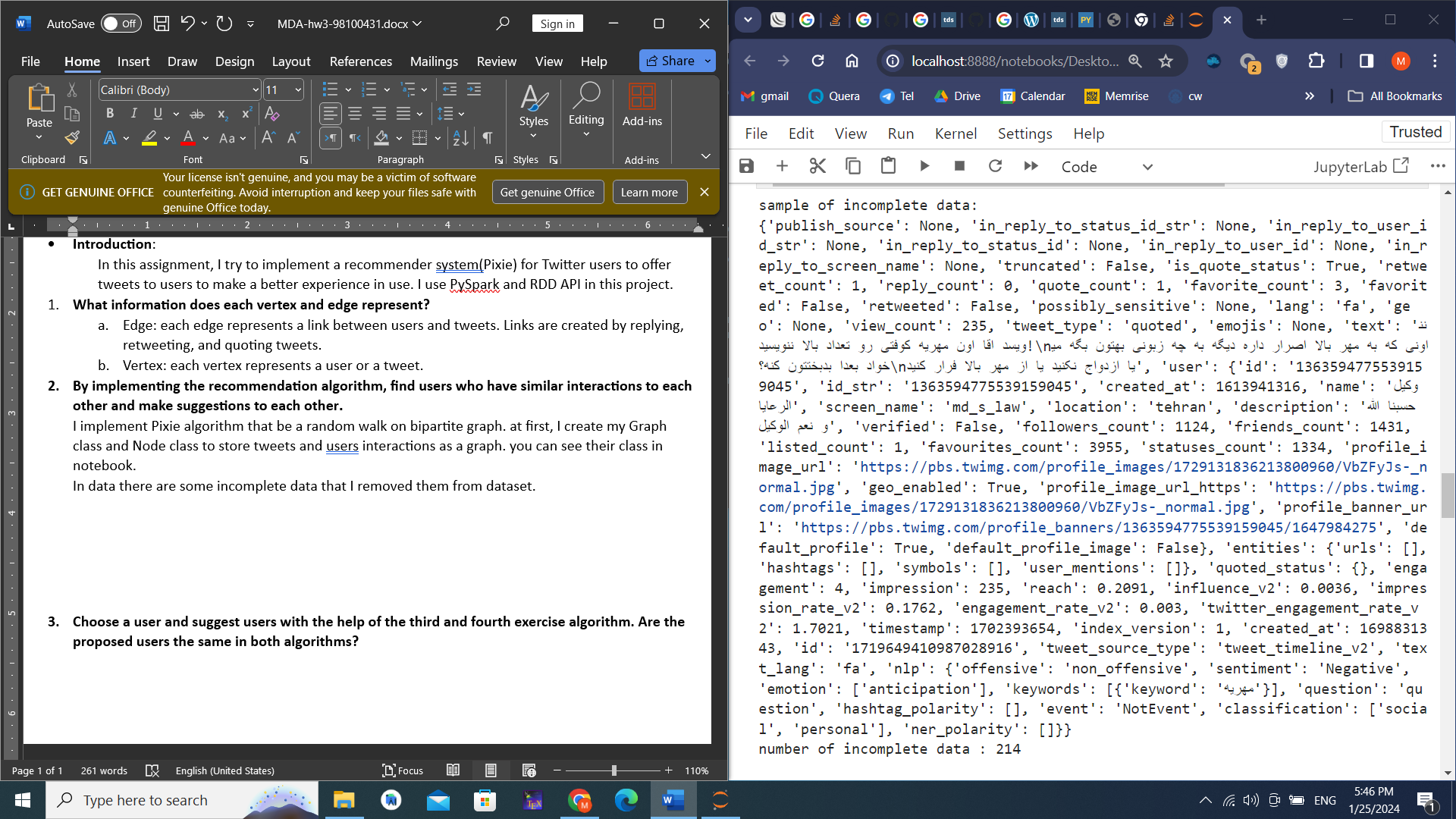
In this assignment, I try to implement a recommender system (Pixie) for Twitter users to offer tweets to users to make a better experience in use. I use PySpark and RDD API in this project.

1. **What information does each vertex and edge represent?**
   1. Edge: each edge represents a link between users and tweets. Links are created by replying, retweeting, and quoting tweets.
   2. Vertex: each vertex represents a user or a tweet.
2. **By implementing the recommendation algorithm, find users who have similar interactions to each other and make suggestions to each other.**

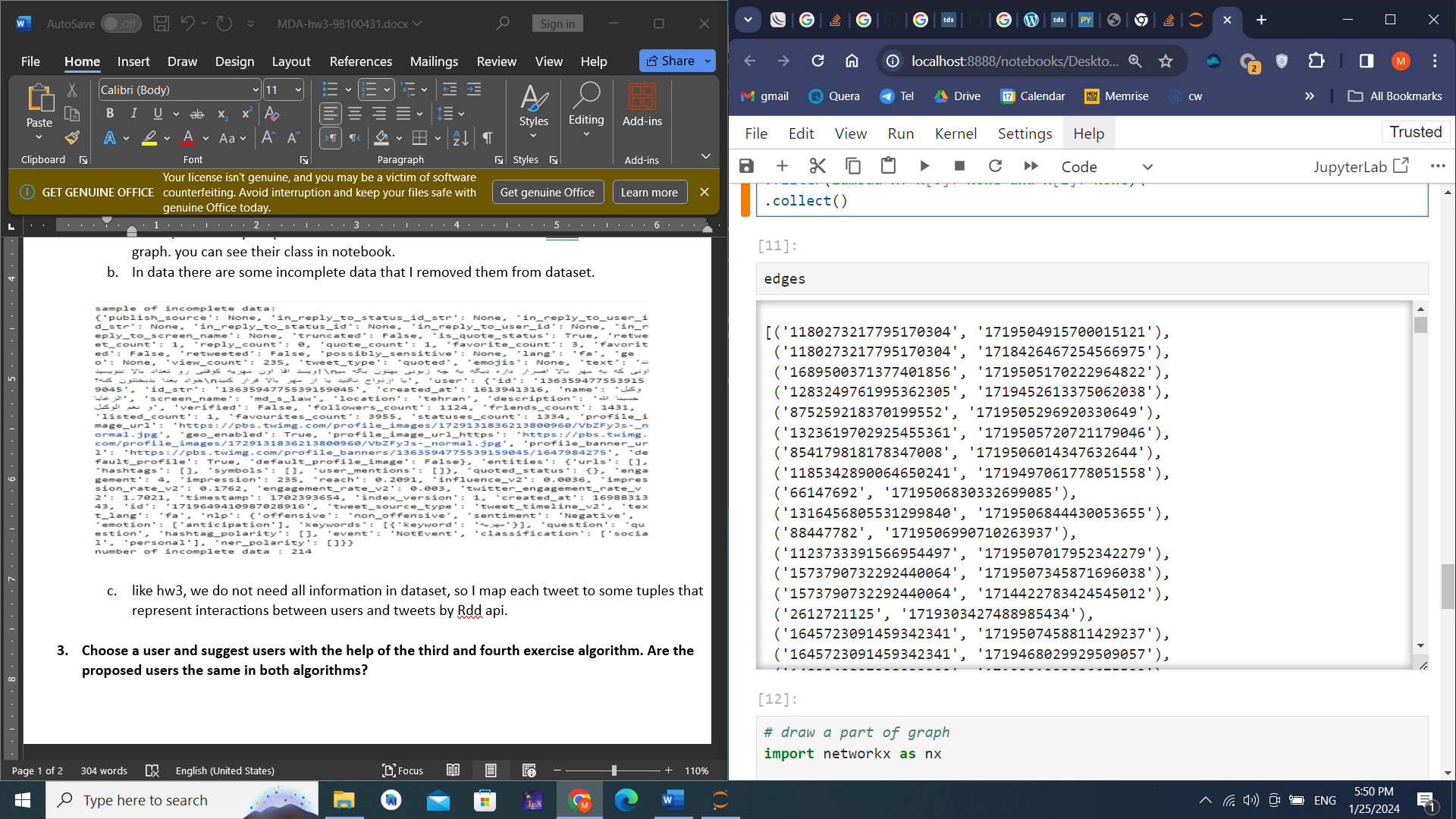
I implement the Pixie algorithm which is a random walk on a bipartite graph.

I do the below steps to clean data and implement the pixie algorithm:

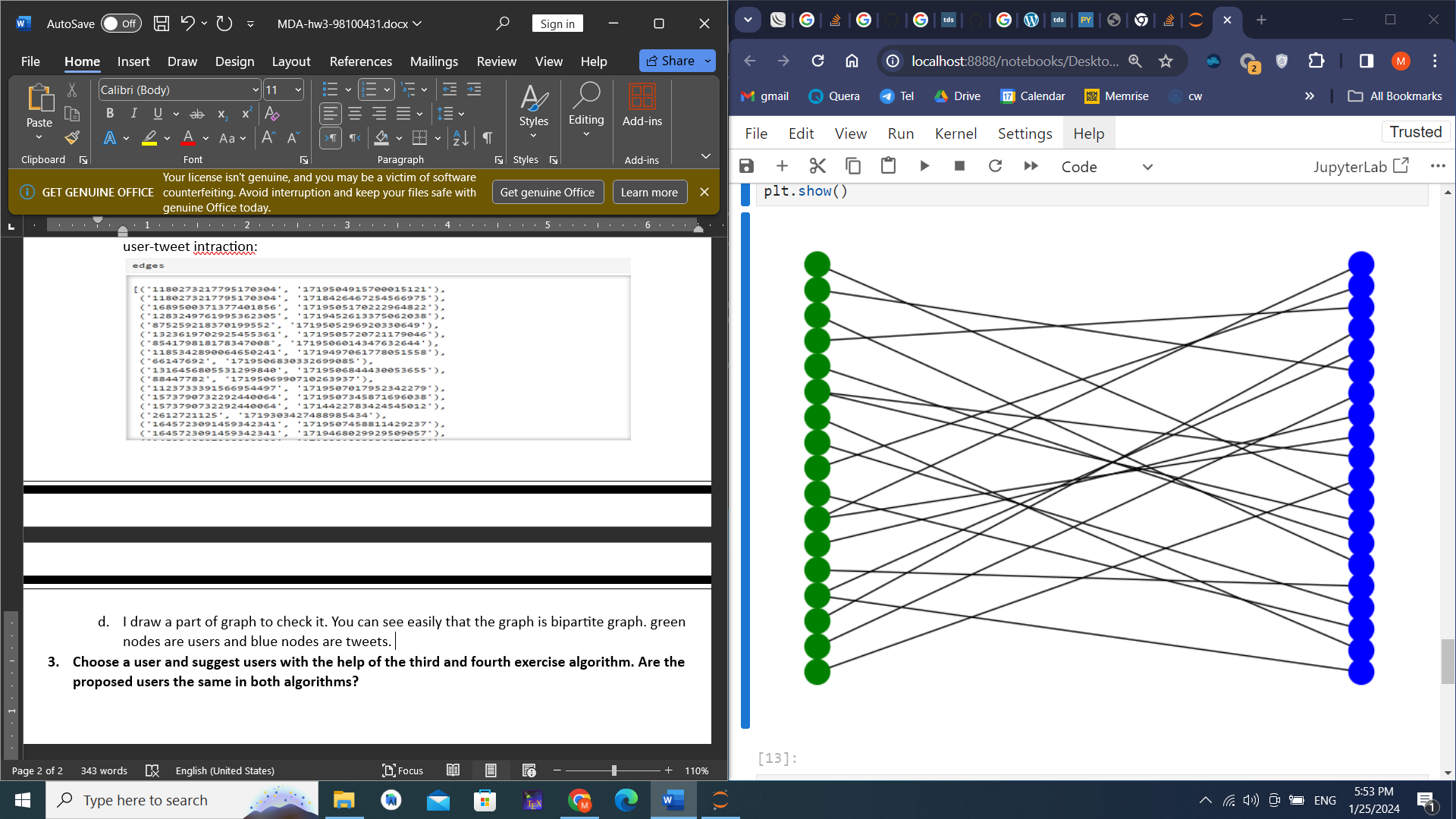
* 1. At first, I create my Graph class and Node class to store tweets and user interactions as a graph. you can see their class in the notebook.
  2. In data there are some incomplete data that I removed from the dataset.



* 1. like hw3, we do not need all the information in the dataset, so I map each tweet to some tuples that represent interactions between users and tweets by Rdd API. Here a list of edges of graph or user-tweet interaction:



* 1. I draw a part of graph to check it. You can see easily that the graph is bipartite graph. green nodes are users and blue nodes are tweets.

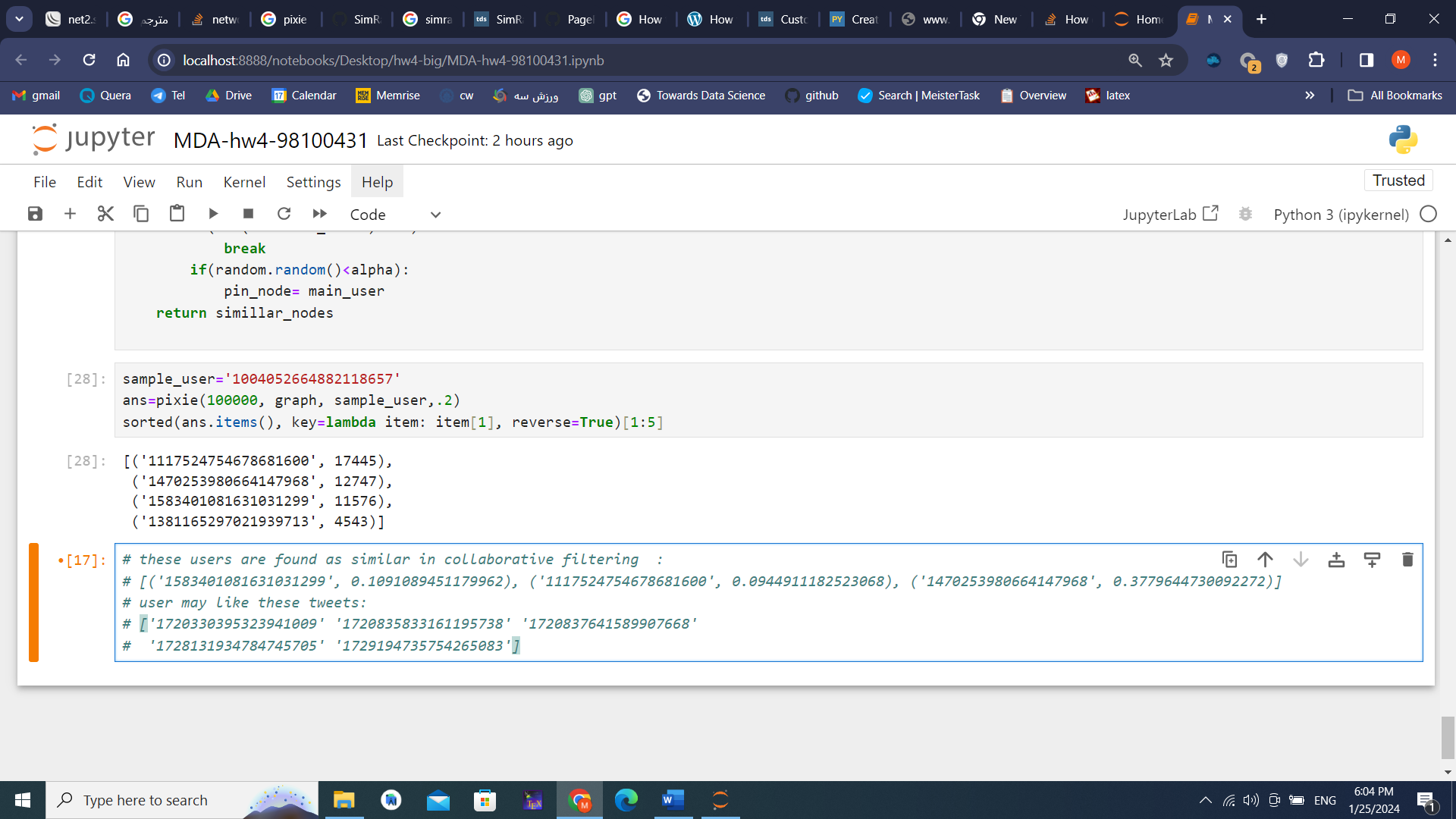


* 1. Now I define a function as pixie that gets a user ID and finds similar users. In this function, inputs are the number of iterations, the graph of interactions, the user ID, and alpha for teleport to the user node. The function returns similar users after
     1. Find 100 similar users that are visited by random walkers at least 20 times.
     2. Or, after random walker walk enough(number of iterations)

Then return similar users to sample users.

1. **Choose a user and suggest users with the help of the third and fourth exercise algorithms. Are the proposed users the same in both algorithms?**

I input the same user ID to the Pixie algorithm as I input in HW3. You can see the results in below picture:





The first three similar users are the same as the outputs of the algorithm in HW3(collaborative filtering- its outputs comment)

Therefore, it works well as a collaborative filtering algorithm in HW3.

1. **Does the algorithm need to obtain the model from the beginning for a new user who is added to this matrix? (online or offline)**

Yes. We have to create the graph, retrain the model, and do all computations again. Because adding a new user can make changes to the graph and edges. Therefore, the probability of random walk can change. So we have to walk randomly again on the graph and find similar users.

1. **What are the advantages and disadvantages of each algorithm?**
   1. **Collaborative Filtering:**
      1. **Advantages:**
         1. **User-Centric:**

Collaborative filtering relies on user behavior and preferences, making it user-centric. It recommends items based on the behavior and preferences of similar users.

* + - 1. **No Need for Item Attributes:**

It doesn't require information about the items themselves. The system learns from user interactions and doesn't rely on detailed item characteristics.

* + - 1. **Serendipity:**

Collaborative filtering can introduce users to new items or products based on the preferences of users with similar tastes, leading to serendipitous discovery.

* + 1. **Disadvantages**:
       1. **Cold Start Problem:**

It struggles with the cold start problem, where it's challenging to provide accurate recommendations for new items or users with limited interaction history.

* + - 1. **Sparsity:**

Collaborative filtering relies heavily on the availability of user-item interactions. In situations where data is sparse, it may struggle to provide accurate recommendations.

* + - 1. **Scalability:**

As the number of users and items grows, the collaborative filtering approach can become computationally expensive and challenging to scale.

* 1. **Pixie:**
     1. **Advantages:**
        1. **Personalization:**

Pixie relies on deep learning models that can capture complex patterns and relationships in user behavior, leading to more personalized recommendations.

* + - 1. **Content-Aware:**

It can take advantage of additional item features and content information, making it more effective in scenarios where item attributes are essential.

* + - 1. **Handling Cold Start:**

Deep learning models, such as Pixie, can be more robust in handling cold-start problems by learning representations from various features.

* + 1. **Disadvantages:**
       1. **Data Requirements:**

Pixie often requires large amounts of data for training, and obtaining labeled data for recommendation systems can be challenging in some cases.

* + - 1. **Interpretability**:

Deep learning models, including Pixie, can be complex and lack interpretability. Understanding why a particular recommendation was made might be challenging.

* + - 1. **Training Complexity:**

Training deep learning models can be computationally intensive, requiring powerful hardware and significant resources.

**conclusion:** The Pixie is a very fast algorithm and can run for each user in a little time, but it is offline. On the other hand, collaborative filtering is online but has more computation. Therefore Each method has its own advantages. Depending on the problem, one can use any of them or combine them.