CSC111 Project 2 Proposal: JetSetGo

Xin Lei Lin, Evan Su, Rick Chen, Yi He (Robert) Li

March 5, 2024

Problem Description and Research Question

Nowadays there are millions of people travelling by plane everyday. By planning our flight reasonably, we can save tons of time and money, which not only make our lives more efficient but also improve our life quality. Thus, a good flight recommending system is essential for most of the people. Therefore, our group project is an efficient flight recommending system. Our project aims to provide the best flight route for every user from a student to a billionaire. We plan to use several different algorithms to enhance the flight search experience.

The final goal of our project is taking the user and its location as input, recommend similar or different places to go and based on user's current location, and its characteristics, all while recommend the shortest, the cheapest and the most-valued flight to the user, ensuring a wonderful flight experience.

Computational Plan

The data our team will use includes:

Graphs and trees represent the central way that transportation lines and airports will be represents. Each vertex will represent airports around the world, and the edges in the graph will represent the existence of a flight between two airports. This data will come from the pyflightdata API. [1]

We will also create a class object for each destination containing information about the climate and costs of living at each location (retrieved from "Global Cost of Living" [2] and "Average Temperature of Cities" [3]). Our recommend system will then utilize these pieces of data to recommend similar locations to the user. For our program to identify "similar" locations, we will employ clustering algorithms to group certain locations together, though the specific hyper parameters will require fine tuning.

The K-nearest neighbour (KNN) algorithm is a simple and widely used machine learning algorithm for classification and regression tasks [4]. It belongs to the class of instance-based or lazy learning algorithms, meaning that it does not learn an explicit model from the training data, but instead stores the entire training dataset and uses it during prediction.The basic idea behind the KNN algorithm is to classify new data points based on the class labels of their closest k neighbours in the training data. The "k" refers to the number of neighbours to consider, which is usually chosen by the user. One of the main advantages of the KNN algorithm is its simplicity and ease of implementation. However, it can be computationally expensive and memory-intensive when dealing with large datasets, as it requires storing the entire training dataset in memory. Additionally, choosing the appropriate value of k and distance metric can be challenging and requires some experimentation. KNN mainly involves two hyperparameters, K value and distance function. K value : how many neighbours participate in the KNN algorithm. k should be tuned based on the validation error. Distance function : Euclidean distance is the most used similarity function, with a few alternatives such as Manhattan distance, Hamming Distance, and Minkowski distance.

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems [5]. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector

Machine.

Hence, for the aforementioned clustering algorithms, we will be employing libraries such as Tensorflow and/or Scikit-learn, in addition to data manipulation libraries such as numpy and pandas.
On the other hand, KNNs can be implemented with Scikit-Learn, which is a machine learning library with supervised models and data manipulation tools, such as train_test_split() for our training and validation data, while the KNN module renders an easy-to-use KNN model [6].
Another method could be to implement the KNNs, and SVM from scratch, which could be a good challenge, to further our understanding of those machine learning tools.
Finally, we might use Tensorflow or PyTorch, two powerful machine learning libraries. Yet a small problem would arise if another user wants to test our models: a lot of dependencies and packages must be installed, as running models saved in Protobuf file format is quite a hassle. [7]

Additionally, as mentioned in the problem description, we will be employing certain algorithms, such as Dijkstra's, Depth-First-Search (DFS)/Breadth-First-Search (BFS), and algorithms such as Kosajaru's algorithm to find strongly connected components. As the goal of our project is to find "optimal" flights, we will use Dijkstra's shortest path algorithm [8] on a weighted, directed graph to determine the best path to match a set of criteria, for example, the cheapest route or the fastest route. The weighting for each edge will be a "score" determined by a formula integrating such criteria. We note that Dijkstra's is particularly useful in our case of directed graphs [9], since many flights are one-way and not bidirectional. Additionally, one thing we will also need to consider is the ordering and timing of flights. We can do this by filtering real-time flight data so that each flight/edge in the graph is at a time that is before any of its subsequently connected flights.

Of course, all of this assumes that there exists a flight path between two airports. To ensure this, we can use graph traversal techniques like BFS and DFS [10]. We could use DFS to determine whether there exists a series of flights connecting two airports, and BFS to determine the minimum number of flights necessary. Combined with Dijkstra's algorithm, we can create a comprehensive analysis of different paths and suggest optimal flight paths for the user.

The use of Kosajaru's algorithm will be useful, as a secondary graph structure, containing various information for strongly connected components. This will ensure faster computational search of the large database, in order to determine the existence of a flight between two cities, by commuting through hubs of central airports. [11]

To display our final results and insights to the user, we will use a graphical plotting library, notably `plotly` [12], to display an interactive world map and superimpose our suggested flight paths and points of interest onto it. This allows the user to visualize potential flight paths and hover over them for more information. In terms of gathering the user's criteria for optimal flights, we will using a simple command-line system. We might also implement a simple user system where each user is uniquely identified by their name and their flight preferences are saved into a JSON file.

References

[1] 'pyflightdata API documentation'. pyflightdata, 2018. https://pyflightdata.readthedocs.io/en/latest/pyflightdata.html.

[2] 'Global Cost of Living'. kaggle, 2022. https://www.kaggle.com/datasets/mvieira101/global-cost-of-living

[3] 'Average Temperature of Cities' kaggle, 2022. https://www.kaggle.com/datasets/swapnilbhange/average-temperature-of-cities

[4] "What is the k-nearest neighbors (KNN) algorithm?". IBM, 2024.

https://www.ibm.com/topics/knn: :text=The%20k%2Dnearest
%20neighbors%20(KNN)%20algorithm

[5] Finley, Thomas, and Thorsten Joachims. "Supervised clustering with support vector machines." Proceedings of the 22nd international conference on Machine learning. 2005.
https://dl.acm.org/doi/abs/10.1145/1102351.1102379
casatoken=dQhUVlYlS3YAAAAA:2HiXOF90ou5rq89-
SN_ujcsGk9PQKsiIm8hzGw_6EUSAynAGLjOukpo6H6iW6Hwy7FnDUz3xkJ4z

[6] “Learn.” Scikit, scikit-learn.org/stable/. Accessed 2 Mar. 2024.

[7] “Tensorflow.” TensorFlow, www.tensorflow.org/. Accessed 2 Mar. 2024.

[8] ‘Dijkstra’s Algorithm’. Wikipedia, 23 Feb. 2024. Wikipedia, https://en.wikipedia.org/w/index.php?title=Dijkstra%27s_algorithm&oldid=1209859384.

[9] ‘Directed Graph’. Wikipedia, 26 Nov. 2023. Wikipedia, https://en.wikipedia.org/w/index.php?title=Directed_graph&oldid=1186925478.

[10] ‘Graph Traversal’. Wikipedia, 2 Feb. 2024. Wikipedia, https://en.wikipedia.org/w/index.php?title=Graph_traversaloldid=1202449254.

[11] GfG. “Strongly Connected Components.” GeeksforGeeks, GeeksforGeeks, 17 Jan. 2024, www.geeksforgeeks.org/strongly-connected-components/.

[12] Lines. https://plotly.com/python/lines-on-mapbox/. Accessed 2 Mar. 2024.