Assignment 2

CS-541: Artificial Intelligence

Spring 2022

In this assignment you will implement search algorithms and apply them to two real-world datasets.

In order to get full credit, you should complete both the problems without using any python scientific packages including `sklearn`, `scipy` etc. You can only use `pandas` or `numpy` for reading the data or for matrix operations. Additionally, you would need to install the `geopy` and `sacrebleu` packages.

1 NYC Taxi Data

Details The task is to perform Uniform Cost Search and A* search. The dataset contains a list of the taxi trips made in NYC in the January 2015. You will only use the following columns:

- pickup_longitude
- pickup_latitude
- dropoff_longitude
- dropoff_latitude
- trip_distance

Part 1. [5 pts] Represent the data as a graph. For the sake of simplicity, you can assume that edges are only between locations which had a valid taxi trip between them. You can either do it as an adjacency matrix or as an adjacency list. Each node in the graph will be represented by the lat/long values. Output the graph as two csv files.

1. *nodes.csv*: Containing a list of lat/long and the corresponding node ID. The file should have the following columns: nodeid, lat, long.
2. *edges.csv*: Containing tuple of node IDs which have an edges between them. The file should have the following columns: nodeid1, nodeid2

Part 2. [10 pts] Implement the Uniform Cost search where you can use the trip distances as the edge costs. The program should input two node ids from the user and output the path as well as the cost between them. You should also report the time taken to run the search algorithm.

Part 2. [15 pts] Implement A^* search using a heuristic. One idea of a heuristic value is to use straight line distance between 2 points. This can be computed using the `geopy` package. The program should input two node ids from the user and output the path as well as the cost between them. You should also report the time taken to run the search algorithm.

Note:

1. You can install geopy using the command `pip install geopy`
2. All the distance and speed values are in miles

2 Neural Machine Translation

Details The task is to implement beam search for Neural Machine Translation (NMT). This NMT model is already trained on the French to English translation task and a sample file is provided to run for few examples. The data files have already been pre-processed including tokenization and normalization.
The following files already been provided:

- *models/encoder* & *models/decoder*: trained model files
- *data-bin/fra.data* & *data-bin/eng.data*: vocabulary files created using the training data
- *data/test.fra* & *data/test.eng*: normalized and tokenized test files
- *data_utils.py* & *rnn.py*: supporting files for data processing and model initialization
- *beam_search.py*: Main file to translate input sentences

Part 1. [2.5 pts] Setup all the files and data in Google Colab and run the file *beam_search.py* on the *test.fra* file (without any changes). Save the output in a new file named *test_beam_1.out*. Use sacrebleu pacakage to compute the BLEU Score[1] performance of the model. You should use the following command and report the BLEU scores.

```
sacrebleu data/test.eng < test_beam_1.out
```

Part 2. [15 pts] Implement the beam search algorithm. You can update the `beam_search` function in the *beam_search.py* file. The function should be generic to use any beam size.

Part 3. **[10 pts]** Apply beam search on the *valid.fra* file and get output files for beam size $K = 1, 2, \cdots, 20$. Use `sacrebleu` command to compute the BLEU scores for each of the output files. Plot the validation data BLEU scores for corresponding beam sizes.

Part 4. **[2.5 pts]** Apply the beam search on the *test.fra* file using the K which gives the highest BLEU score in part 3. Use `sacrebleu` command to compute the BLEU score for the test translations.

Note:

1. You should select GPU in Google colab from Runtime > Change Runtime Type > Hardware Accelerator
2. You can install sacrebleu using the command `pip install sacrebleu`
3. You should not change the order of the files. The correct test translation (*test.eng*) should always be before the output translations.
   `sacrebleu correct_data < output_data`

Submission

This is an individual assignment. Each person should submit as a single zip file named with assignment number and the username (e.g. *HW2_akhan4.zip*). The zip file should contain the required code file and a readme file. The readme should include the following:

- one line descriptions of the code file
- Time taken to run UCS and A* searches (for Q1)
- Test data BLEU score without beam search implementation (for Q2)
- BLEU Score plot on validation data (for Q2)
- Test data BLEU score on the optimal beam size (for Q2)

Remember that after general discussions with others, you are required to work out the problems by yourself. All submitted work must be your own, though you can get help with others, so long as you cite the help. Please refer to the Stevens Honor System for clarifications.

References

[1] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.