UFUG2602 (2024 Spring) Final Project: Interactive Approximate Text Search

Deadline: 23:59 **26 May 2024 (Sun)**

Introduction:

In the final project, you will develop an interactive and flexible approximate text search tool using data structures and algorithms learned in the course, with the help of the `curses` library for a terminal-based user interface. Specifically, the outcome will enable users to *search for multiple (exact or approximate)* ***substring*** *matches (and optionally with complex logics) for each line of a text file efficiently.*

Tasks:

- **Trie-based search algorithm**
  - Implement a ***Trie*** (c.f., links in the Reference) for:
    - efficient **word insertion**
    - finding both **exact matches** and **approximate matches** (with a given error threshold)
    - Input k keywords: $\{w_1, w_2, \ldots, w_k\}$, define two matches of a keyword as:

$$m(w_i) = \begin{cases} w_i & \text{, if } len(w_i) < \texttt{APPROX\_LEN\_THRESHOLD} \\ \{w \mid ed(w, w_i) \leq \delta, w \in F\} \end{cases} \tag{1}$$

      - where

$$\delta = \begin{cases} \texttt{ERROR\_THRESHOLD} & \text{, if } \texttt{ERROR\_THRESHOLD} \geq 1 \\ len(w_i) \cdot \texttt{ERROR\_THRESHOLD} & \text{, otherwise.} \end{cases} \tag{2}$$

      - F is the the set of words contained in the input text file. Words are extracted from the text using

```
words = re.split('[^a-zA-Z]+', lines)
```

- **Flexible search**
  - User can input k keywords: $\{w_1, w_2, \ldots, w_k\}$.
  - The resulting lines are defined as those are contains at least one match from $m(w_i)$ for all i. That is, we assume the *conjunctive* semantics of the search keywords. The **error** of a line is defined as the sum of edit distances of each w_i with its matching word in the line (taking the minimum if there are multiple matches in the same line). The resulting lines should be **ordered** by the error, and then the line number.
  - For example, consider the file that contains two lines:

```
"Elliot Smith", "276 Eastern Avenue Kingsford", 22, "Geology"
"Elliott White", "3/76 Astern Ave Kensington", 23, "Sociology"
```

    If the search keywords are: `liot ology`, then both lines will match with the error 0 and ranked in the order of there index (as the index of the first line is 1, and the index of the second line is 2).

- **File Loading**
  - Load a text file where we treat each line as a record. Then index the record using the ***Trie*** data structure to facilitate the search.
- **A user-friendly text-based interface**

- Integrate the search functionality into the provided skeleton code based on the `curses` library
- Enable users to input keywords, view search results, and navigate through them.

Note: Most of the python interpreter's built-in functions and its corresponding operations are supported, such as set(), list() and other commonly used methods. And we list some lib which probably will be used during the project development. You can search all the corresponding help documents in the website https://docs.python.org/3.9/index.html.

Instructions:

- **Environment Setup**: Make sure that your Python interpreter version is higher than 3.9 (supporting `curses` and `@cache`) . And install the required libraries (e.g., `curses`) for your development environment.
- **Implementation**: Follow the provided tasks and guidelines in the code template to develop the trie-based keyword search application.
- **Testing**: **Thoroughly** test your implementation, including edge cases and varying input scenarios, to ensure correct functionality and performance.
- **Documentation**: Include relevant comments and docstrings in your code to explain complex logic, algorithms, and data structures.
- **Submission**: Submit your completed Python script in Canvas by the deadline.

Submission:

- Submit the hole project file in `.zip` format containing the complete implementation.
- Ensure all code is **well-commented**, explaining the purpose and logic of each function and significant blocks of code.
- No report is required, but your comments should be detailed enough to understand your implementation and design choices.

Note: NO LATE SUBMISSION will be accepted.

Grade Policy

- We will check the **correctness, readability and coding style** of your written code. (25%)
- And we will evaluate the correct output and functionality of the code in different test cases (**including** the three test cases given). (75%)

Testing Usages

Testing usages 1 & 2 is for the basic part, which means that you all should check your codes with them.

```
# Just a description of some hyperparameters for your testing

# Testing usage 1


APPROX_LEN_THRESHOLD = 3
ERROR_THRESHOLD = 1

test_file_name = "test_data_01.csv"

input0 = "ogy"  # Approximate matching

# Expecting Outputs:
```

```
# 4,A knowledge growth and consolidation framework for lifelong machine learning systems,"lifelong machine learning, oblivion criterion, knowledge topology and acquisition, declarative learning",Machine Learning I,2014
# 5,A Hybrid Genetic-Programming Swarm-Optimisation Approach for Examining the Nature and Stability of High Frequency Trading Strategies,"sociology, statistics, noise, testing, prediction algorithms, algorithm design and analysis, genetics",Real-time Systems and Industry,2014
# 1,Prediction of Sunspot Number Using Minimum Error Entropy Cost Based Kernel Adaptive Filters,"kernel methods,error entropy,information theoretic learning",Machine Learning Algorithms for Environmental Applications ,2015
# 2,A Data-Driven Method to Detect the Abnormal Instances in an Electricity Market,"electricity market,data mining,anomaly detection",Machine Learning in Energy Applications,2015

input1 = "logy"   # Approximate matching

# Expecting Outputs:

# 4,A knowledge growth and consolidation framework for lifelong machine learning systems,"lifelong machine learning, oblivion criterion, knowledge topology and acquisition, declarative learning",Machine Learning I,2014
# 5,A Hybrid Genetic-Programming Swarm-Optimisation Approach for Examining the Nature and Stability of High Frequency Trading Strategies,"sociology, statistics, noise, testing, prediction algorithms, algorithm design and analysis, genetics",Real-time Systems and Industry,2014

input2 = "gy"   # Exact matching

# Expecting Outputs:

# 2,A Data-Driven Method to Detect the Abnormal Instances in an Electricity Market,"electricity market,data mining,anomaly detection",Machine Learning in Energy Applications,2015
# 4,A knowledge growth and consolidation framework for lifelong machine learning systems,"lifelong machine learning, oblivion criterion, knowledge topology and acquisition, declarative learning",Machine Learning I,2014
# 5,A Hybrid Genetic-Programming Swarm-Optimisation Approach for Examining the Nature and Stability of High Frequency Trading Strategies,"sociology, statistics, noise, testing, prediction algorithms, algorithm design and analysis, genetics",Real-time Systems and Industry,2014
```

```
# Testing usage 2


APPROX_LEN_THRESHOLD = 5
ERROR_THRESHOLD = 2

test_file_name = "test_data_02.csv"

input0 = "quir"

# No Outputs

# Outputs:
#

input1 = "conv Learn" # Exact + Approximate matching

# Expecting Outputs:

# Outputs:
# 1,Learning Good Features To Track,"object tracking, convolutional neural network, feature learning",Feature Extraction and Selection,2014
```

```
# 3,A Cyclic Contrastive Divergence Learning Algorithm for High-order RBMs,"high-order rbms, cyclic
contrastive divergence learning, gradient approximation, convergence, upper bound",Neural Networks
I,2014
# 2,Human action recognition based on recognition of linear patterns in action bank features using
convolutional neural networks,"human action recognition, action bank features, deep convolutional
network",Neural Networks I,2014

input2 = "Conv net"   # Exact + Exact matching

# Expecting Outputs:

# Outputs:
# 1,Learning Good Features To Track,"object tracking, convolutional neural network, feature
learning",Feature Extraction and Selection,2014
# 2,Human action recognition based on recognition of linear patterns in action bank features using
convolutional neural networks,"human action recognition, action bank features, deep convolutional
network",Neural Networks I,2014
# 3,A Cyclic Contrastive Divergence Learning Algorithm for High-order RBMs,"high-order rbms, cyclic
contrastive divergence learning, gradient approximation, convergence, upper bound",Neural Networks
I,2014
# 4,Facial expression recognition using kinect depth sensor and convolutional neural
networks,"convolutional neural networks (cnn), facial expression recognition",Neural Networks
I,2014
# 7,Human action recognition based on MOCAP information using convolution neural
networks,"convolutional neural networks (cnn), motion capture (mocap)",Neural Network II,2014
# 11,One-shot periodic activity recognition using Convolutional Neural Networks,"human activity
recognition, convolutional neural networks (cnn)",Neural Network II,2014

input3 = "Netwo Featu"  # Appr + Appr matching

# Expecting Outputs:

# Outputs:
# 1,Learning Good Features To Track,"object tracking, convolutional neural network, feature
learning",Feature Extraction and Selection,2014
# 2,Human action recognition based on recognition of linear patterns in action bank features using
convolutional neural networks,"human action recognition, action bank features, deep convolutional
network",Neural Networks I,2014
# 9,Multi-Variable Neural Network Forecasting Using Two Stage Feature Selection,"forecasting,
feature selection, neural networks",Neural Network II,2014

input4 = "Fearu Netwo"  # Some spelling mistakes in inputs

# Expecting Outputs:

# Outputs:
# 1,Learning Good Features To Track,"object tracking, convolutional neural network, feature
learning",Feature Extraction and Selection,2014
# 2,Human action recognition based on recognition of linear patterns in action bank features using
convolutional neural networks,"human action recognition, action bank features, deep convolutional
network",Neural Networks I,2014
# 9,Multi-Variable Neural Network Forecasting Using Two Stage Feature Selection,"forecasting,
feature selection, neural networks",Neural Network II,2014
# 3,A Cyclic Contrastive Divergence Learning Algorithm for High-order RBMs,"high-order rbms, cyclic
contrastive divergence learning, gradient approximation, convergence, upper bound",Neural Networks
I,2014
# 5,Improving Performance on Problems with Few Labelled Data by Reusing Stacked Auto-
Encoders,"transfer learning, deep learning, artificial neural networks",Neural Networks I,2014
# 10,Adaptive restructuring of radial basis functions using integrate-and-fire neurons,"machine
learning, radial basis functions, neural networks, feed-forward networks",Neural Network II,2014
```

Testing usage 3 is for the **bonus** part, you can ignore it if you just want to finish the basic tasks.

```
# Testing usage 3


APPROX_LEN_THRESHOLD = 5
ERROR_THRESHOLD = 1

test_file_name = "test_data_02.csv"

input0 = "perf | multi" # OR testing 1

# Expecting Outputs:

# Outputs:
# 5,Improving Performance on Problems with Few Labelled Data by Reusing Stacked Auto-
Encoders,"transfer learning, deep learning, artificial neural networks",Neural Networks I,2014
# 9,Multi-Variable Neural Network Forecasting Using Two Stage Feature Selection,"forecasting,
feature selection, neural networks",Neural Network II,2014

input1 = "perfo | multi"  # OR testing 2

# Expecting Outputs:

# Outputs:
# 5,Improving Performance on Problems with Few Labelled Data by Reusing Stacked Auto-
Encoders,"transfer learning, deep learning, artificial neural networks",Neural Networks I,2014
# 9,Multi-Variable Neural Network Forecasting Using Two Stage Feature Selection,"forecasting,
feature selection, neural networks",Neural Network II,2014
# 11,One-shot periodic activity recognition using Convolutional Neural Networks,"human activity
recognition, convolutional neural networks (cnn)",Neural Network II,2014

input2 = "netwo (conv | activ)"

# Expecting Outputs:

# Outputs:
# 1,Learning Good Features To Track,"object tracking, convolutional neural network, feature
learning",Feature Extraction and Selection,2014
# 2,Human action recognition based on recognition of linear patterns in action bank features using
convolutional neural networks,"human action recognition, action bank features, deep convolutional
network",Neural Networks I,2014
# 3,A Cyclic Contrastive Divergence Learning Algorithm for High-order RBMs,"high-order rbms, cyclic
contrastive divergence learning, gradient approximation, convergence, upper bound",Neural Networks
I,2014
# 4,Facial expression recognition using kinect depth sensor and convolutional neural
networks,"convolutional neural networks (cnn), facial expression recognition",Neural Networks
I,2014
# 7,Human action recognition based on MOCAP information using convolution neural
networks,"convolutional neural networks (cnn), motion capture (mocap)",Neural Network II,2014
# 11,One-shot periodic activity recognition using Convolutional Neural Networks,"human activity
recognition, convolutional neural networks (cnn)",Neural Network II,2014
# 10,Adaptive restructuring of radial basis functions using integrate-and-fire neurons,"machine
learning, radial basis functions, neural networks, feed-forward networks",Neural Network II,2014
```

We provide some of the testing usages above to you in `unittest` codes, you are suggested to check all of them in your project to make sure all the outputs from your codes is correct.

Reference

- Trie Data Structure:
  - For Beginners: https://www.hackerearth.com/practice/data-structures/advanced-data-structures/trie-keyword-tree/tutorial/
  - Advanced 1: https://www.geeksforgeeks.org/introduction-to-trie-data-structure-and-algorithm-tutorials/
  - Advanced 2: https://en.wikipedia.org/wiki/Trie
- Python Dictionary (Hash Tabel):
  - https://stackoverflow.com/questions/327311/how-are-pythons-built-in-dictionaries-implemented
  - https://csrgxtu.github.io/2020/04/21/Python-Dict-a-new-implementation-by-pure-Python/
  - https://www.jessicayung.com/how-python-implements-dictionaries/
  - https://peps.python.org/pep-0412/
- Levenshtein distance:
  - For Beginners: https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0
  - Advanced 1: https://www.geeksforgeeks.org/introduction-to-levenshtein-distance/
  - Advanced 2: https://www.baeldung.com/cs/levenshtein-distance-computation
  - Advanced 3: https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance
- Python `curses` Lib:
  - You do **not** really need to touch this `---` **all you need to do is to implement the `search` function in the skeleton code.**
  - Help doc: https://docs.python.org/3/library/curses.html

Good luck, and remember to test thoroughly to ensure your interactive text search and approximate matching tool are as effective and efficient as possible!

Bonus part

Logical Combinations of Keywords

- Enhance the search functionality to handle **logical combinations** of keywords, enabling users to specify complex search queries involving multiple keywords and Boolean operators. Implement support for the following syntax:
  - `match` means `approximate match` using the logic defined in the basic part.
  - **Conjunction (`AND`):** `(keyword1 keyword2)` searches for lines matching both `keyword1` and `keyword2`.
  - **Disjunction (`OR`):** `(keyword1 | keyword2)` searches for lines matching either `keyword1` or `keyword2`.
- Compose these logical expressions using parentheses to form nested groups, allowing for arbitrarily complex combinations. For example:
  - `((a | b) c) | (d e)` searches for lines that match either `(a or b) and c` or `(d and e)`.

  Note: Using inputs with different order of the logical operators may return different results with the same threshold setting (Here you just need to satisfy simple testing cases like the given tests.).

- For evaluating different logical combinations, defining **a reasonable error metric** for different cases (especially after introducing `OR` here) is important. You should pay attention to how to design the ranking strategy here to make sure your ordered results are reasonable both cognitively and logically.(*Hints: Based on the given edit distance, you can consider the logic is just slightly different from Boolen*), and **write out your design ideas in comments**. Specifically, for conjunction-only queries, the ranking order should be the same as that defined in the basic part.
- Order the resulting lines according to their error values, then by line numbers, as in the original search functionality.

Ensure that your implementation handles these bonus features seamlessly within the existing user-friendly text-based interface, allowing users to input complex search queries and view the results in a structured and navigable manner.

Submission of Bonus part

- Implement the bonus part based **on your existing Python script submission**. (*The original supported functionalities are also keeped. *)
- **Clearly document the new added code and its logic** related to the bonus features, maintaining the same level of commenting and explanation as in the main project.

By implementating these bonus features, you will further enhance the versatility and precision of your interactive text search and approximate matching tool, empowering users to perform sophisticated searches with ease. Thoroughly test your implementation with various logical combinations and edge cases to ensure optimal functionality and performance.