Homework 1

COSC 6590: Special Topics – Application Development with Large Language Models (Dr. Chandra N Sekharan)

Sainadth Pagadala
A04359181

Code Files - LLM_Learning/Homework1 at main · sainadth/LLM_Learning

findURLS.ipynb - Question1

mapAisle.ipynb - Question2

In the file cnbc.html you will see a lot of places where you will find URLs of the type http:// If you want to extract only the URLs (anything starting with http including https), how would you think about solving the problem. Think of a good algorithm that is efficient for counting the URLs. Come ready to discuss this next week. Is your approach efficient? If so, argue how so?

My approach to solve this problem includes the prowess of regular expressions in string matching.

Algorithm:

1. Load the html file content
2. Regex pattern matching (2 regex for general URLs and encoded URLs)
    a. Regular URLs

```
r'https?://[^\s\)\"\'\}\]\)]+'
```

h – matches character
t – matches character
t – matches character
p – matches character
s – matches character
? - either 0 or 1 occurrence of preceding token ‘s’
: - matches character
/ - matches character
/ - matches character

[^] - negation set (matches any character not in the list which could be at the end of URLs such as “, ‘, space (\s),},],) etc., which are common delimiters in URLs)

+ - either 0 or 1 occurrence of preceding token i.e., negation set

b. Encoded URLs

```
r'https?:\\[^\s\)\"\'\}\]\)]+'
```

h – matches character
t – matches character
t – matches character
p – matches character
s – matches character
? - either 0 or 1 occurrence of preceding token ‘s’
: - matches character
\ - matches character (\u002F ~ /)
\ - matches character (\u002F ~ /)
[^] - negation set (matches any character not in the list which could be at the end of URLs such as “, ‘, space (\s),},],) etc., which are common delimiters in URLs)

+ - either 0 or 1 occurrence of preceding token i.e., negation set

3. Save the URLs to output files

Efficiency of my algorithm

1. Linear Time Complexity O(N)

   Regex does single parse to search and match the expression, i.e., the regex scans the input string from start to finish once. Regex engines are optimized for pattern matching so they minimize backtracking and redundant checks. Unlike DOM parsers which includes additional processing such as interpreting HTML tags, tree building etc., are not required.

2. Simple and Maintainable

   Regular expression is self-explanatory of what we are trying to achieve clearly. With the well-defined standardized documentation one can easily understand the regex and can easily make changes to the expression. Very few lines of code can achieve complex matching.

3. Memory Usage

Unlike DOM parsing which requires storing a tree structure of the given HTML in RAM, regex only stores the html contents there by present no overhead in memory usage. This approach is very efficient for large files.

Let us say you are setting up a grocery store online and you are required to virtually set up items for shopping online, so shoppers can search for them using names of the aisle categories. For instance, eggs may be mapped to the aisle breakfast food or food. The below file contains two columns: grocery items and aisle category. How would you map each grocery item to an aisle name? If an item does not have an appropriate category to map, then you can use a generic "other" aisle category. Think of an algorithm to solve this problem and write the steps of the algorithm clearly. Is your approach efficient? If so, argue how so?

My approach to solve this problem is to use dictionaries to categorize a grocery item. If there is no appropriate aisle, then we categorize it as "other".

Algorithm:

1. Load the excel data
2. Data Preprocessing
    a. Text normalization – convert all the grocery item names and aisle category names to lowercase and strip off any leading/trailing spaces. This would help us to match items case inventively.
3. Build Dictionary
    a. For all the mapped grocery items, we create a dictionary with grocery item as the key and the corresponding aisle category as value. We chose the first non-empty aisle category to create the dictionary.
    b. We skip all the unmapped grocery items for now.
    { grocery_item_1 : aisle_category_1, grocery_item_2 : aisle_category_2, ...}
4. Mapping Grocery Items
    a. For unmapped grocery items we check if we have an entry in the dictionary and assign it to each item.
    b. If there is no corresponding Aisle Category we assign “other”.

Efficiency of my algorithm

1. Linear Time Complexity O(N)
    Building the mapping dictionary is linear O(N) - N is no of rows in the excel

All dictionary lookups are O(1)
Can rapidly handle thousands of items.

2. Space Complexity O(K)
   Requires space for K unique mapped items
3. Scalable
   New items/categories can be easily added to the dictionary and large datasets can be easily handled since the dictionary lookups take constant time O(1)

This is the efficient approach as both Time and space complexity are at most linear.