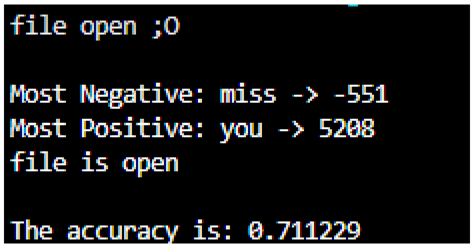
Southern Methodist University
Luke Voinov
CS 5393–002
In Collaboration with Craig Truitt

Project #5



```
file open ;O

Most Negative: miss -> -551
Most Positive: you -> 5208
file is open

The accuracy is: 0.711229
```

DSString.cpp + DSString.h

The DSString.cpp file defines the implementation of the DSString class, a custom string class with various functionalities. Here's a summary of its contents:

1. **Helper Functions**:
   - cstr_length: Calculates the length of a C-string.
   - cstr_copy: Copies a C-string to another location.
2. **Constructors and Destructor**:
   - **Default Constructor**: Initializes an empty string.
   - **C-string Constructor**: Initializes the string with a given C-string.
   - **Copy Constructor**: Creates a copy of another DSString.
   - **Character Constructor**: Creates a string with n copies of a character.
   - **Assignment Operator**: Copies another DSString.
   - **Destructor**: Cleans up allocated memory.
3. **Compatibility with std::string**:
   - **Constructor from std::string**: Initializes the string with a std::string.
   - **Assignment Operator from std::string**: Copies a std::string.
   - **Conversion Operator to std::string**: Converts DSString to std::string.
4. **Member Functions**:
   - clear: Clears the string.
   - empty: Checks if the string is empty.
   - find: Finds a substring, C-string, or character.
   - append: Appends another DSString, C-string, or multiple copies of a character.
   - erase: Erases a portion of the string.
   - replace: Replaces a portion of the string with another DSString or C-string.
   - insert: Inserts another DSString, C-string, or multiple copies of a character at a specified position.
   - erase (range-based): Erases a range of characters based on a predicate.
   - begin and end: Returns pointers to the beginning and end of the string.
   - length: Returns the length of the string.
   - operator[]: Accesses a character at a specified index.

    - operator+: Concatenates two DSString objects or a character to the DSString.
    - operator==, operator!=, operator<: Compares two DSString objects.
    - operator+=: Appends a C-string or DSString to the current DSString.
    - substring: Returns a substring of the DSString.
    - toLower: Converts the string to lowercase.
    - c_str: Returns a C-string representation of the DSString.
5. **Stream Operators**:
    - operator<<: Outputs the string to an output stream.
    - operator>>: Inputs a string from an input stream.
    - getline: Reads a line from an input stream with or without a delimiter.

sentimentMain.cpp

The sentimentMain.cpp file implements a sentiment analysis program using a Trie data structure. Here's a detailed summary of its contents:

Includes and Namespace

- **Includes**:
    - <iostream>: For input and output operations.
    - "trie.h": Header file for the Trie data structure.
    - <fstream>: For file handling operations.
- **Namespace**: Uses the std namespace.

Function Declarations

- vector<int> predictSentiment(TrieNode tr, TrieNode* root): Predicts sentiments for the test dataset.
- double findAccuracy(vector<int> pS): Calculates the accuracy of the sentiment predictions.

main Function

- **Trie Initialization**:
    - Creates a TrieNode object tr.
    - Creates a root TrieNode pointer root.
- **Variables**:
    - findCol: Tracks the current column in the CSV file.
    - sentiment: Stores the sentiment value.
    - str: Temporary string for reading lines.
    - words: Vector to store parsed words from a sentence.
    - predictedSentiments: Vector to store predicted sentiments.

    - accuracy: Stores the calculated accuracy.
- **File Handling**:
    - Opens the training dataset file train_dataset_20k_modified.csv.
    - Checks if the file is open; if not, prints an error message and exits.
    - Reads the file line by line, parsing the sentiment and tweet text.
    - Inserts words from the tweet into the Trie with their associated sentiment.
    - Closes the training file.
- **Sentiment Prediction and Accuracy Calculation**:
    - Calls tr.mostChargedWords() to identify the most sentiment-charged words.
    - Calls predictSentiment(tr, root) to predict sentiments for the test dataset.
    - Calls findAccuracy(predictedSentiments) to calculate the accuracy of the predictions.
    - Prints the accuracy.

predictSentiment Function

- **Variables**:
    - pSent: Predicted sentiment.
    - findCol: Tracks the current column in the CSV file.
    - str: Temporary string for reading lines.
    - words: Vector to store parsed words from a sentence.
    - posAndNeg: Vector to store positive and negative sentiment counts.
    - allPredictions: Vector to store all predicted sentiments.
    - posPredict, negPredict, total: Variables for sentiment calculation.
- **File Handling**:
    - Opens the test dataset file test_dataset_10k.csv.
    - Checks if the file is open; if not, prints an error message and returns.
    - Reads the file line by line, parsing the tweet text.
    - Analyzes each word in the tweet using the Trie to determine sentiment.
    - Calculates the total sentiment score and determines the predicted sentiment.
    - Stores the predicted sentiment in allPredictions.
    - Returns the vector of all predicted sentiments.

findAccuracy Function

- **Variables**:
    - str: Temporary string for reading lines.
    - total: Counts the number of correct predictions.
    - average: Stores the calculated accuracy.
- **File Handling**:
    - Opens the test dataset file with actual sentiments test_dataset_sentiment_10k.csv.
    - Reads the file line by line, comparing predicted sentiments with actual sentiments.
    - Increments total for each correct prediction.
    - Calculates the accuracy as the ratio of correct predictions to total predictions.
    - Returns the accuracy.

Summary

The program reads a training dataset to build a Trie with words and their associated sentiments. It then predicts sentiments for a test dataset by analyzing the words in each tweet and calculating a sentiment score. Finally, it compares the predicted sentiments with actual sentiments to determine the accuracy of the predictions. The program uses file handling, string parsing, and the Trie data structure to perform sentiment analysis.

Trie.h

The `trie.h` file defines a `TrieNode` structure and its associated functions for managing a Trie data structure, which is used for storing and analyzing words with sentiment scores. Here's a detailed summary:

Includes and Namespace

- **Includes**:
  - `<iostream>`: For input and output operations.
  - `<vector>`: For using the `vector` container.
  - `<unordered_map>`: For using the `unordered_map` container.
  - `<fstream>`: For file handling operations.
  - `"DSString.h"`: Custom string class header file.
  - `<cmath>`: For mathematical functions.
- **Namespace**: Uses the `std` namespace.
- **Pragma Once**: Ensures the file is included only once during compilation.

`TrieNode` Structure

- **Members**:
  - `unordered_map<char, TrieNode*> leaf`: Maps characters to child `TrieNode` pointers.
  - `int sentiment`: Sentiment score of the word.
  - `int maxPos`: Maximum positive sentiment score.
  - `int maxNeg`: Maximum negative sentiment score.
  - `DSString posWord`: Word with the maximum positive sentiment.
  - `DSString negWord`: Word with the maximum negative sentiment.
  - `bool endOfWord`: Flag indicating the end of a word.
- **Constructor**: Initializes `endOfWord` to `false`.

Member Functions

- `insertLetter`:
    - Inserts a word into the Trie and updates its sentiment score.
    - Updates the maximum positive and negative sentiment words.
- `searchWord`:
    - Searches for a word in the Trie.
    - Returns `true` if the word is found and is marked as the end of a word.
- `parseSentence`:
    - Parses a sentence into individual words.
    - Removes non-alphanumeric characters from words.
- `parseWord`:
    - Parses a word by removing non-alphanumeric characters and converting to lowercase.
- `mostChargedWords`:
    - Prints the words with the most positive and negative sentiment scores.
- `analyzeWord`:
    - Analyzes a word and updates the sentiment scores in a provided vector.
- `normalize`:
    - Normalizes a value using the base-10 logarithm.

Summary

The `TrieNode` structure and its functions are designed to build and manage a Trie for storing words with associated sentiment scores. The Trie can insert words, search for words, parse sentences into words, and analyze words to update sentiment scores. It also identifies and prints the words with the highest positive and negative sentiment scores. The `normalize` function is used to normalize sentiment values using logarithms.

Bibliography

CoPilot

https://www.geeksforgeeks.org/trie-insert-and-search/