AvlTree Class High-Level Pseudo Code

Class Description:

- Purpose: Implements an AVL (Adelson-Velsky and Landis) tree that maintains a balanced state through rotations.
- Key Properties:
    - root: The root node of the AVL tree.
    - uniqueTokens: Tracks the number of unique keys in the tree.

Constructor:

- Initializes the tree with an empty root and zero unique tokens.

Destructor:

- Calls makeEmpty to delete all nodes and clean up resources.

Copy Constructor and Assignment Operator:

- Handles deep copying of the tree to ensure independent trees with their own memory.

Public Methods:

1. insert(x, documentID, frequency):
    - Inserts a new key or updates an existing key with a given document and frequency.
    - Ensures the tree remains balanced after insertion.
2. contains(x):
    - Checks if a key exists in the tree.
3. findNode(x):
    - Returns the node containing the key if it exists.
4. getWordMapAtKey(key):
    - Retrieves the map of documents and frequencies associated with a key.
5. isEmpty():
    - Returns true if the tree is empty.
6. prettyPrintTree():
    - Prints a visual representation of the tree structure.
7. makeEmpty():
    - Clears the tree, deleting all nodes.
8. writeToTextFile(filename):

    - Writes the contents of the tree to a text file in a specific format.
9. readFromTextFile(filename):
    - Reads tree data from a file and reconstructs the tree.
10. getSize():
    - Returns the number of unique keys in the tree.

Private Methods:

1. writeHelper(node, outFile):
    - Recursively writes tree data to a file.
2. balance(t):
    - Balances the tree at node t if the height difference between child nodes exceeds allowed limits.
3. rotateWithLeftChild(k2):
    - Performs a single right rotation around the given node.
4. rotateWithRightChild(k1):
    - Performs a single left rotation around the given node.
5. doubleWithLeftChild(k3):
    - Performs a double rotation (right-left) around the given node.
6. doubleWithRightChild(k1):
    - Performs a double rotation (left-right) around the given node.
7. height(t):
    - Returns the height of the node t.
8. max(lhs, rhs):
    - Returns the larger of two values.

Debug Methods:

1. check_balance(node):
    - Checks if the tree is balanced at every node, used for debugging.

DocumentParser Class High-Level Pseudo Code

Class Variables:

- PersonTree: AVL tree for storing person entities.
- OrganizationTree: AVL tree for storing organization entities.
- WordsTree: AVL tree for storing words.
- filesIndexed: Counter for the number of processed files.
- stopWords: Set to store words that should be ignored during parsing.

Constructor:

- Initializes the AVL trees for persons, organizations, and words.

loadStopWords(filePath):

- Open the file at filePath.
- Read words from the file and add them to the stopWords set.

stemWord(word):

- Apply stemming algorithm to word.
- Return the stemmed word.

toLower(text):

- Convert all characters in text to lowercase.
- Return the modified string.

containsStopWords(word):

- Check if word exists in the stopWords set.
- Return true if found, otherwise false.

tokenizer(textLine):

- Split textLine into words based on spaces.
- Return a list of words.

removePunctuation(word):

- Remove punctuation and non-alphabet characters from word.
- Return the cleaned word.

runDocument(documentName):

- Open the document specified by documentName.
- Load stop words if not already loaded.
- Parse the JSON content of the document.
- Tokenize, clean, and stem the text of the document.
- For each token, if it's not a stop word, add it to the WordsTree.
- Extract person and organization entities and add them to respective trees.
- Close the file.

pushToTreePerson(token, docName, frequency):

- Insert the token into the PersonTree with the given docName and frequency.

pushToTreeOrg(token, docName, frequency):

- Insert the token into the OrganizationTree with the given docName and frequency.

pushToTreeWord(token, docName, frequency):

- Insert the token into the WordsTree with the given docName and frequency.

printDocument(filename):

- Open and parse the JSON document filename.
- Display the title and publication date of the article.

printDocumentText(filename):

- Open and parse the JSON document filename.
- Display the text content of the document.

toFile(personFile, orgFile, wordFile):

- Write the contents of PersonTree, OrganizationTree, and WordsTree to their respective files.

getFilesIndexed():

- Return the number of files that have been indexed.

QueryProcessor Class High-Level Pseudo Code

Class Variables:

- PersonTree, OrganizationTree, WordsTree: AVL trees for storing entities and words.
- vectorOfMaps: Holds results of search terms related to valid keywords.
- vectorOfBadMaps: Holds results of search terms prefixed with '-' to be excluded.
- documentFrequencyPairs: A vector of pairs to hold documents and their associated frequencies.
- searchIndex: Tracks the starting index for outputting search results in sessions.

Constructor:

- Initializes the AVL trees for persons, organizations, and words from provided references.

runQueryProcessor(search):

- Clears previous search data.
- Processes a new query.
- Sorts results by frequency.
- Outputs the top documents based on relevance.

processQuery(search):

- Splits the search string into keywords.
- For each keyword, retrieves matching documents from the appropriate AVL tree.
- Intersects results from multiple keywords to find common documents.
- Excludes documents related to negative search terms (from bad maps).
- Returns the final map of documents and frequencies.

sortDocumentsByFrequency(documentFrequencyMap):

- Converts map entries to a list of pairs.
- Sorts the list based on document frequencies in descending order.

outputDocuments(numDocuments):

- Outputs a specified number of top documents based on the previous query.
- Handles cases where fewer documents are available than requested.
- Updates the search index to manage session-based output.

SeperateString(search):

- Tokenizes the input search string.
- Identifies and processes different prefixes ("ORG:", "PERSON:", "-") to categorize search terms.
- Stores results in either vectorOfMaps for regular searches or vectorOfBadMaps for negative searches.

intersectMaps(map1, map2):

- Finds common documents between two maps.
- Sums their frequencies if they appear in both maps.
- Returns a new map with these intersected results.

excludeMaps(map, badMap):

- Removes documents found in badMap from map.
- Returns a new map excluding these documents.

removePunctuationExcept(word):

- Removes all punctuation from a word except colons and dashes, used for specific query syntax.
- Returns the cleaned word.

getTreesfromFile(personFile, orgFile, wordFile):

- Loads serialized data from files into the respective AVL trees.

getDocumentName(index):

- Returns the document name from documentFrequencyPairs at the specified index.
- Handles cases where the index is out of the valid range.

Main Function Overview

Purpose: Entry point of the application that directs the flow based on command line arguments.

Helper Functions

1. indexDirectory:
    - Input: References to DocumentParser and AVL trees.
    - Behavior: Prompts user for directory path, iterates through files, uses DocumentParser to process each file, and calculates and displays performance statistics.
    - Output: Indexing performance statistics.
2. performQuery:
    - Input: QueryProcessor and DocumentParser.
    - Behavior: Allows continuous querying with options to show more results, start a new query, print document details, or exit to main menu.
    - Output: Query results and performance statistics.
3. startUI:
    - Behavior: Provides a looped user interface for indexing directories, performing queries, writing indexes to files, and reading indexes from files.
    - Output: User-directed operations and system feedback.

main(argc, argv):

- Check Arguments: Ensures correct usage based on the number of arguments.
- Command Handling:
    - "index": Manages bulk indexing of documents from a specified directory.
    - "query": Handles a single query from the command line.
    - "ui": Initiates a user-interactive mode for various operations.