C-string Tokenizer Documentation

Overview:

In this project, we use the C++ Standard Template Library (STL) to tokenize tweet text into individual words. Tokenization is a fundamental step in our sentiment analysis process, as it breaks down the raw text into analyzable units (tokens/words) that can be matched against our positive and negative word dictionaries.

STL Tokenization Approach:

Our implementation uses std::stringstream from the STL, which provides a stream-based approach to tokenization. This is more flexible and safer than using traditional C-style functions like strtok().

How it Works:

1. Stream Creation: The string to be tokenized is used to initialize a std::stringstream object.
2. Token Extraction: The stream extraction operator (>>) automatically splits the input on whitespace.
3. Token Processing: Each extracted token can be further processed to remove punctuation and normalize case.

Advantages over C-style strtok():

- Safety: Doesn't modify the original string (unlike strtok() which modifies the input string)
- Reentrant: Can be used in multi-threaded environments (unlike strtok() which uses static variables)
- Flexibility: Can easily combine with other STL algorithms and containers
- No Buffer Management: Handles memory allocation automatically

Implementation in Our Project:

In our SentimentClassifier::tokenizeText method, the tokenization process follows these steps:

1. Convert the custom DSString to a C-string using .c_str()
2. Create a std::stringstream initialized with this C-string
3. Extract words from the stream one by one using the >> operator
4. Clean each word by removing non-alphabetic characters and converting to lowercase
5. Store valid tokens in a std::vector<DSString> for further sentiment analysis