1. Keyword Frequency Count using Hash

The frequency count measures how often specific keywords appear in a set of documents. The `build_frequency_table` function starts by creating a dictionary called `frequency_table`, where each keyword is set to zero. It reads each document, converts the text to lowercase for consistency, and extracts words using a regular expression. For each word, it checks if it exists in the `frequency_table` and increases the count if it does. This continues until all documents are processed. The time complexity of this algorithm is O(n + m), where n is the total number of words and m is the number of keywords. This means it performs efficiently, even with many documents or keywords, because each word is checked in a fixed amount of time.

2. Common Phrase Detection using Tries:

This method stores and recognizes common words using a Trie (prefix tree). Phrases from each document are taken out and added to the Trie framework. A TrieNode has a set that keeps track of the documents in which a phrase appears, a dictionary of its offspring, and a flag to indicate the end of a phrase. The Trie's nodes are visited by each word when a phrase is inserted. The document ID is stored by the node that marks the conclusion of the sentence after it has been fully appended. This guarantees the effective storage of terms from various documents. The Trie is recursively searched for phrases that occur in several documents in order to identify frequent words. Phrases are recognized by looking for several document IDs in the end node, which indicates that they are shared. This technique works well for identifying phrases that overlap in different publications.