**LyricsIR - Boolean IR System Documentation**

**Overview**

A Boolean Information Retrieval (IR) engine designed to search through a collection of song lyrics organized by artist folders and individual song files. The system builds and uses an inverted index structure where each word maps to documents and positions. It supports keyword-based search with Boolean logic (AND, OR, NOT) as well as exact phrase search using double quotation marks (" ").

**Folder Structure**

The lyrics are stored in the following format:

Lyrics/

├── Artist1/

│ ├── Song1.txt

│ ├── Song2.txt

├── Artist2/

│ ├── SongA.txt

│ ├── SongB.txt

Each .txt file contains the full lyrics of a song.

**Features**

* **Boolean Query Support**: AND, OR, NOT keyword logic.
* **Phrase Search**: Quoted queries using double quotation marks for exact phrase matches.
* **Stopword Filtering**: Common English stopwords (e.g., "the", "is", "a") are filtered out to enhance search precision.
* **Normalized Queries**: Input is case-insensitive and non-alphanumeric characters are removed.

**How It Works**

**1. Preprocessing**

* All text is converted to lowercase.
* Non-alphanumeric characters are removed.
* Tokens (words) are extracted.
* Stopwords are removed to reduce noise in search results.

**2. Inverted Index Construction**

* The system creates an inverted index where:
  + Each term maps to a dictionary of document IDs and positions where the term appears.
* Each document is assigned a unique integer ID and mapped to its "Artist/SongTitle" string.

**3. Boolean Search Logic**

* **Single term**: Returns documents that contain a single word.
* **Negation**: Returns documents that do not contain a specific word.
* **Binary logic**:
  + A and B → documents that contain both terms
  + A or B → documents that contain either term A or B

**4. Phrase Search**

* Triggered when user input is enclosed in quotation marks, e.g.:
  + "you are beautiful"
* Checks if all words appear in the exact same order and position across the document.

**Stopword Filtering**

Stopwords are filtered in both indexing and querying. This removes high-frequency, low-value words like:

* a, an, the, is, are, was, were, and, or, in, on, of, for, to, etc.

Phrase search bypasses stopword filtering to preserve exact matches.

**Running the System**

1. Place your lyrics inside the Lyrics/ArtistName/SongTitle.txt structure.
2. Run the Python program.
3. Enter a query at the prompt, e.g.:
   * love
   * love and peace / love or peace
   * not war
   * "you are beautiful"

**Example Queries and Results**

| **Query** | **Description** | **Matches** |
| --- | --- | --- |
| love | Finds all lyrics with the word "love" | Songs mentioning love |
| not love | Finds lyrics without the word "love" | All others |
| peace and hope | Both words must appear | Intersection of results |
| "let it be" | Exact phrase match | Songs with that lyric phrase |

**Limitations**

* Does not support nested Boolean logic or parentheses (e.g. love and (peace or hope)).
* Phrase search is exact-match only.

**Summary**

The system implements a Boolean Information Retrieval model that allows users to search for lyrics using keyword queries or exact phrases. It builds an inverted index to map words to document positions, applies stopword filtering to reduce noise, and supports logical operations like AND, OR, and NOT for accurate and efficient retrieval.

These implemented features allow for fast and accurate retrieval of relevant song lyrics based on user-defined Boolean queries or exact phrases.