Document Retrieval System

This project implements a Document Retrieval System using the Extended Boolean model. The system allows users to search for documents using Boolean logic with AND, OR, and NOT operations.

Features

- **Extended Boolean Search**: Supports complex queries with AND, OR, and NOT operations.
- **Text Preprocessing**: Tokenization, stop word removal, and stemming.
- **Term-Document Matrix**: Efficient representation of documents for quick retrieval.

Installation

1. **Clone the repository**:

   ```
   git clone https://github.com/KhadimHussainDev/document-retrieval-system.git
   cd document-retrieval-system
   ```

2. **Install dependencies**:

   ```
   pip install -r requirements.txt
   ```

3. **Run the Django server**:

   ```
   python manage.py runserver
   ```

Usage

1. **Navigate to the search page**: Open your web browser and go to `http://127.0.0.1:8000/search/`.

2. **Enter a search query**: Use Boolean logic to search for documents. Example queries:

   - `apple and cat`
   - `apple or cat`
   - `apple and cat not dog`

3. **View search results**: The results will display the documents that match the query along with their relevance scores.

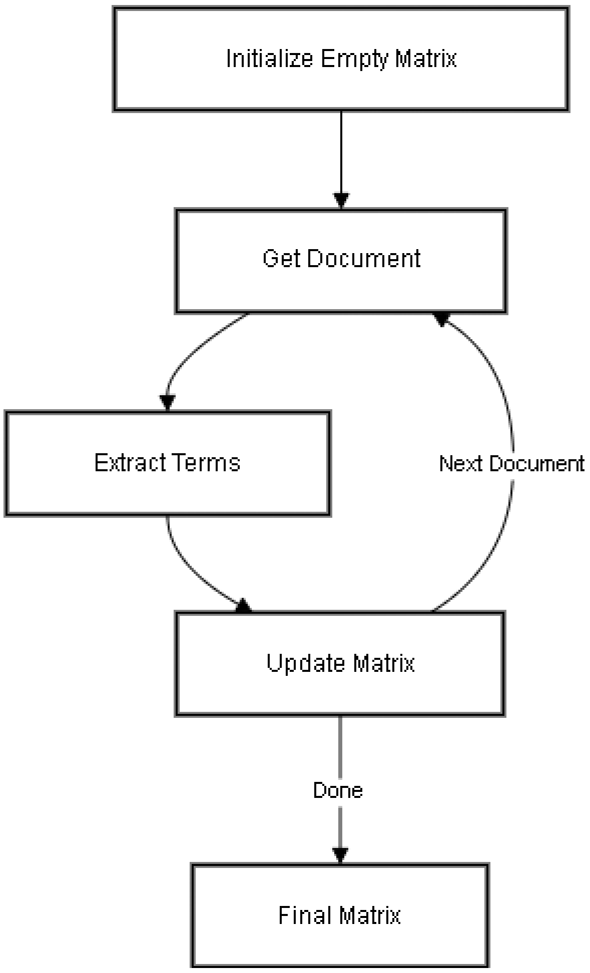
Code Explanation

`create_term_document_matrix`

This function creates a term-document matrix, which is a binary representation of the presence of terms in documents.

```python
def create_term_document_matrix(documents):
    term_document_matrix = defaultdict(lambda: [0] * len(documents))
    for doc_index, doc in enumerate(documents):
        terms = preprocess_text(doc.content)
        for term in terms:
            term_document_matrix[term][doc_index] = 1
    return term_document_matrix
```

- **Input**: A list of document objects.
- **Output**: A dictionary where keys are terms and values are lists indicating the presence (1) or absence (0) of the term in each document.
- **Process**:
    - Iterate over each document.
    - Preprocess the document content to tokenize, remove stop words, and stem the words.
    - Update the term-document matrix to indicate the presence of each term in the document.



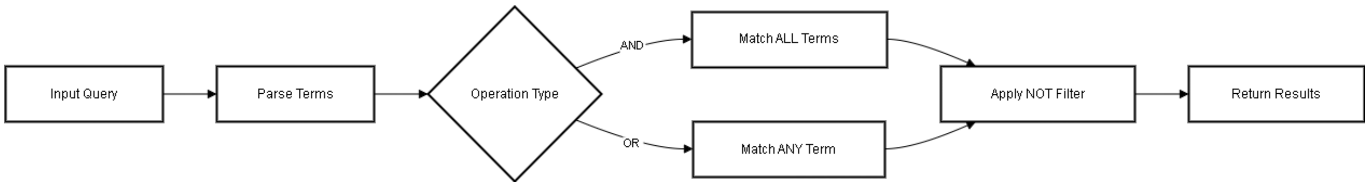
`search_documents_extended_boolean`

This function implements the Extended Boolean search with AND, OR, and NOT operations.

```python
def search_documents_extended_boolean(query, documents, term_document_matrix):
    query_terms = preprocess_text(query)

    # Split query into AND, OR, NOT terms
    and_terms = []
    or_terms = []
    not_terms = []
    current_terms = and_terms
    for term in query_terms:
        if term == 'and':
            current_terms = and_terms
        elif term == 'or':
            current_terms = or_terms
        elif term == 'not':
            current_terms = not_terms
        else:
            current_terms.append(term)

    # Find documents for each term
    and_docs = [term_document_matrix[term] for term in and_terms if term in
term_document_matrix]
    or_docs = [term_document_matrix[term] for term in or_terms if term in
term_document_matrix]
    not_docs = [term_document_matrix[term] for term in not_terms if term in
term_document_matrix]

    # Apply AND logic
    if and_docs:
        and_result = set.intersection(*map(set, and_docs))
    else:
        and_result = set(range(len(documents)))

    # Apply OR logic
    if or_docs:
        or_result = set.union(*map(set, or_docs))
    else:
        or_result = set()

    # Apply NOT logic
    if not_docs:
        not_result = set.union(*map(set, not_docs))
    else:
        not_result = set()

    # Combine results
    final_result = (and_result | or_result) - not_result

    ranked_docs = [(documents[doc_index], 1) for doc_index in final_result]
    return ranked_docs
```

- **Input**:

    - `query`: The search query string.
    - `documents`: A list of document objects.
    - `term_document_matrix`: The term-document matrix created by `create_term_document_matrix`.
- **Output**: A list of tuples containing documents and their relevance scores.
- **Process**:
    - Preprocess the query to tokenize, remove stop words, and stem the words.
    - Split the query into AND, OR, and NOT terms.
    - Find the documents that contain each term.
    - Apply AND logic to find documents that contain all AND terms.
    - Apply OR logic to find documents that contain any OR terms.
    - Apply NOT logic to exclude documents that contain any NOT terms.
    - Combine the results and rank the documents based on their relevance.



Contributing

Contributions are welcome! Please open an issue or submit a pull request for any improvements or bug fixes.

License

This project is licensed under the MIT License.