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.

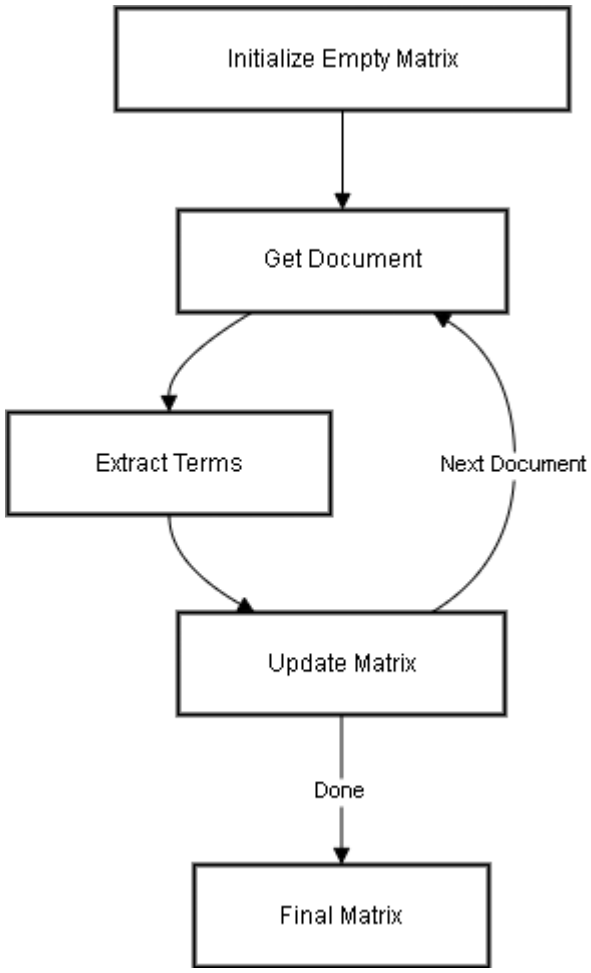
Code Explanation

`create_term_document_matrix`

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

```
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.

```
def search_documents_extended_boolean(query, documents, term_document_matrix):
    """
    Implements Extended Boolean search with AND, OR, NOT operations.
    Query format: "term1 and term2 not term3" or "term1 or term2 not term3"

    Args:
        query (str): The search query (e.g., "apple and mangos not juice")
        documents (list): List of documents to search through
        term_document_matrix (dict): Term-document matrix from
create_term_document_matrix

    Returns:
        list: List of matching documents
    """
    # Convert query to lowercase and split into parts
    query_parts = query.lower().split()

    # Initialize variables to store terms
    positive_terms = []
    negative_terms = []
    operation = 'and'  # default operation

    # Parse query
    i = 0
    while i < len(query_parts):
        term = query_parts[i]

        if term in ('and', 'or'):
            operation = term
            i += 1
            continue

        if term == 'not':
            if i + 1 < len(query_parts):
                negative_terms.append(query_parts[i + 1])
                i += 2
            else:
                i += 1
            continue

        positive_terms.append(term)
        i += 1

    # Find matching documents
    matching_docs = []

    for doc_idx, doc in enumerate(documents):
        # Check positive terms based on operation
        matches_positive = False

        if operation == 'and':
            # All positive terms must be present
```

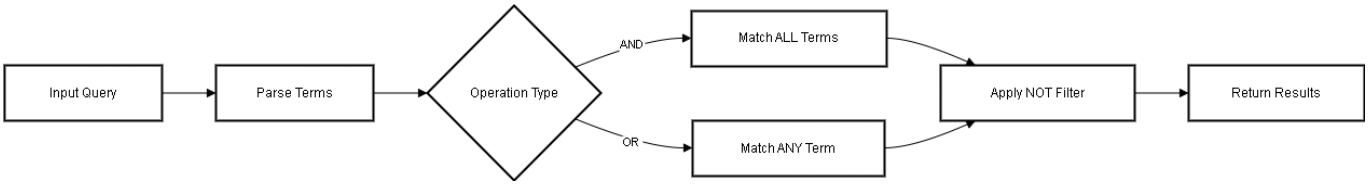
```
            matches_positive = all(
                term in term_document_matrix and
                term_document_matrix[term][doc_idx] == 1
                for term in positive_terms
            )
        else:  # OR operation
            # At least one positive term must be present
            matches_positive = any(
                term in term_document_matrix and
                term_document_matrix[term][doc_idx] == 1
                for term in positive_terms
            )

        # Check negative terms (NOT operation)
        # Document must not contain any negative terms
        matches_negative = all(
            term not in term_document_matrix or
            term_document_matrix[term][doc_idx] == 0
            for term in negative_terms
        )

        # Document must match positive terms criteria and not contain negative
terms
        if matches_positive and matches_negative:
            matching_docs.append(doc)

    return [(doc, 1.0) for doc in matching_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