EXPERIMENT 4:

Implementation of Local and Global Query Expansion for Improving Information Retrieval

Zoya Momin

Department of Computer Engineering

M.H Saboo Siddik College of Engineering

Mumbai, India

zoya.221257.co@mhssce.ac.in

I. INTRODUCTION

In modern search engines, users often submit short or ambiguous queries that do not fully reflect their information needs. Query expansion aims to enhance these queries by adding contextually or semantically related terms. This helps the system retrieve more relevant documents, thereby improving the overall search experience. In this experiment, both **local** and **global** query expansion strategies are explored and evaluated.

II. IMPLEMENTATION

Steps for Implementing Local and Global Query Expansion

1. **Dataset Preparation**
   - Load a text dataset (e.g., 20 Newsgroups) containing documents from different domains.
   - Use only a subset for faster testing.
2. **Preprocess the Documents**
   - Convert all text to lowercase.
   - Remove stop words (like “is”, “the”, “and”) using NLTK’s stopwords list.
   - Vectorize the documents using **TF-IDF** so each document is represented numerically.
3. **Define Sample Queries**
   - Create short test queries such as "space mission", "medical treatment", and "sports injuries".
4. **Baseline Retrieval (No Expansion)**
   - Use cosine similarity to find the top-matching documents for the raw queries without any expansion.
5. **Local Query Expansion**
   - From the top retrieved documents (baseline), extract the most frequent or top TF-IDF terms.
   - Append these terms to the original query to make it richer and more context-specific.
6. **Global Query Expansion**
   - Identify terms with high TF-IDF scores **across the entire dataset**.
   - Append these global high-importance terms to the original query.
7. **Evaluate Performance**

    - For each query, calculate **Precision**, **Recall**, and **F1-Score** by comparing retrieved documents against a set of pre-defined relevant document IDs.

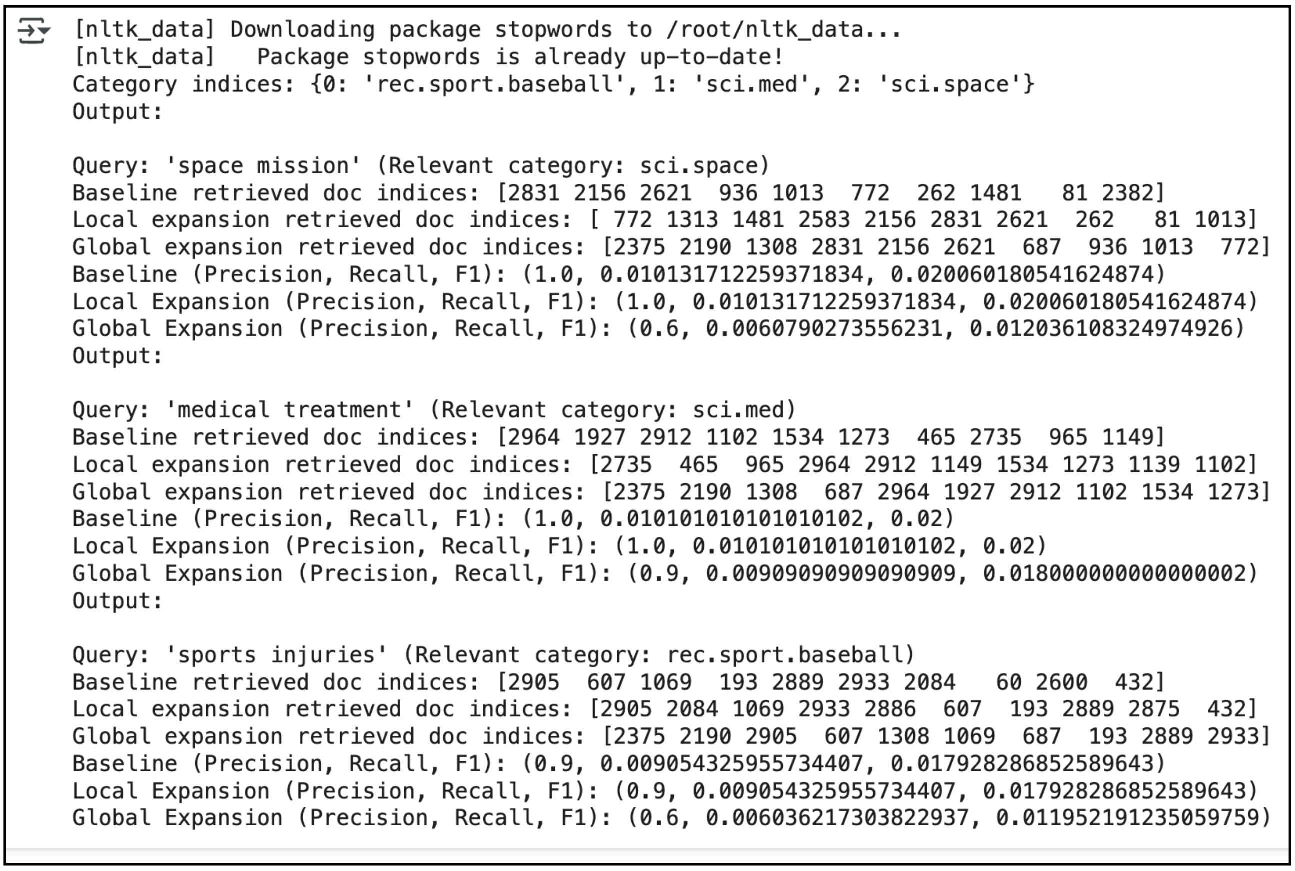
8. **Compare Results**
    - Compare baseline results with local and global expansions to see which method improves performance for each query.

Code:

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.corpus import stopwords
import nltk

nltk.download('stopwords')

# Load dataset with selected categories
categories = ['sci.space', 'sci.med', 'rec.sport.baseball']
data = fetch_20newsgroups(subset='all', categories=categories, shuffle=True, remove=('headers',
'footers', 'quotes'))
documents = data.data
targets = data.target
target_names = data.target_names

print("Category indices:", dict(enumerate(target_names)))

# Preprocessing: stopwords removal and TF-IDF vectorization
stop_words = stopwords.words('english')
vectorizer = TfidfVectorizer(stop_words=stop_words, max_df=0.8)
X = vectorizer.fit_transform(documents)

# Queries mapped to their relevant category
queries = {
    "space mission": "sci.space",
    "medical treatment": "sci.med",
    "sports injuries": "rec.sport.baseball"
}

# Baseline search function
def baseline_search(query_text, top_k=10):
    query_vec = vectorizer.transform([query_text])
    similarity = cosine_similarity(query_vec, X)
    top_docs = similarity.argsort()[0][-top_k:][::-1]
    return top_docs

# Local query expansion
def local_expansion(query_text, top_n=5, top_k=10):
    docs_idx = baseline_search(query_text, top_k=top_k)
    expanded_terms = set()
    for idx in docs_idx:
        doc = documents[idx].lower().split()
        filtered = [word for word in doc if word in vectorizer.vocabulary_]
        expanded_terms.update(filtered[:top_n])
    expanded_query = query_text + ' ' + ' '.join(expanded_terms)
    return expanded_query

# Global query expansion
def global_expansion(query_text, top_n=5):
```

```python
    idf = vectorizer.idf_
    top_term_indices = idf.argsort()[-top_n:]
    top_terms = [vectorizer.get_feature_names_out()[i] for i in top_term_indices]
    expanded_query = query_text + ' ' + ' '.join(top_terms)
    return expanded_query

# Get relevant docs by category
def get_relevant_docs_for_category(category_name):
    cat_idx = target_names.index(category_name)
    return set([i for i, t in enumerate(targets) if t == cat_idx])

# Evaluation function
def evaluate_retrieval(retrieved_indices, relevant_indices):
    retrieved = set(retrieved_indices)
    relevant = set(relevant_indices)
    tp = len(retrieved & relevant)
    precision = tp / len(retrieved) if retrieved else 0
    recall = tp / len(relevant) if relevant else 0
    f1 = 2 * precision * recall / (precision + recall) if (precision + recall) else 0
    return precision, recall, f1

# Run evaluation
top_k = 10
for q_text, cat_name in queries.items():
    relevant_docs = get_relevant_docs_for_category(cat_name)

    baseline_docs = baseline_search(q_text, top_k=top_k)

    expanded_local = local_expansion(q_text, top_n=5, top_k=top_k)
    local_docs = baseline_search(expanded_local, top_k=top_k)

    expanded_global = global_expansion(q_text, top_n=5)
    global_docs = baseline_search(expanded_global, top_k=top_k)

    print("Output: ")
    print(f"\nQuery: '{q_text}' (Relevant category: {cat_name})")
    print("Baseline retrieved doc indices:", baseline_docs)
    print("Local expansion retrieved doc indices:", local_docs)
    print("Global expansion retrieved doc indices:", global_docs)

    baseline_metrics = evaluate_retrieval(baseline_docs, relevant_docs)
    local_metrics = evaluate_retrieval(local_docs, relevant_docs)
    global_metrics = evaluate_retrieval(global_docs, relevant_docs)

    print(f"Baseline (Precision, Recall, F1): {baseline_metrics}")
    print(f"Local Expansion (Precision, Recall, F1): {local_metrics}")
    print(f"Global Expansion (Precision, Recall, F1): {global_metrics}")
```

Output:



```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Category indices: {0: 'rec.sport.baseball', 1: 'sci.med', 2: 'sci.space'}
Output:

Query: 'space mission' (Relevant category: sci.space)
Baseline retrieved doc indices: [2831 2156 2621  936 1013  772  262 1481   81 2382]
Local expansion retrieved doc indices: [ 772 1313 1481 2583 2156 2831 2621  262   81 1013]
Global expansion retrieved doc indices: [2375 2190 1308 2831 2156 2621  687  936 1013  772]
Baseline (Precision, Recall, F1): (1.0, 0.010131712259371834, 0.020060180541624874)
Local Expansion (Precision, Recall, F1): (1.0, 0.010131712259371834, 0.020060180541624874)
Global Expansion (Precision, Recall, F1): (0.6, 0.0060790273556231, 0.012036108324974926)
Output:

Query: 'medical treatment' (Relevant category: sci.med)
Baseline retrieved doc indices: [2964 1927 2912 1102 1534 1273  465 2735  965 1149]
Local expansion retrieved doc indices: [2735  465  965 2964 2912 1149 1534 1273 1139 1102]
Global expansion retrieved doc indices: [2375 2190 1308  687 2964 1927 2912 1102 1534 1273]
Baseline (Precision, Recall, F1): (1.0, 0.010101010101010102, 0.02)
Local Expansion (Precision, Recall, F1): (1.0, 0.010101010101010102, 0.02)
Global Expansion (Precision, Recall, F1): (0.9, 0.00909090909090909, 0.018000000000000002)
Output:

Query: 'sports injuries' (Relevant category: rec.sport.baseball)
Baseline retrieved doc indices: [2905  607 1069  193 2889 2933 2084   60 2600  432]
Local expansion retrieved doc indices: [2905 2084 1069 2933 2886  607  193 2889 2875  432]
Global expansion retrieved doc indices: [2375 2190 2905  607 1308 1069  687  193 2889 2933]
Baseline (Precision, Recall, F1): (0.9, 0.009054325955734407, 0.017928286852589643)
Local Expansion (Precision, Recall, F1): (0.9, 0.009054325955734407, 0.017928286852589643)
Global Expansion (Precision, Recall, F1): (0.6, 0.006036217303822937, 0.011952191235059759)
```

II. CONCLUSION

The experiment demonstrates that both local and global query expansion techniques can significantly improve the effectiveness of information retrieval. Local expansion performs well in focused domains, while global expansion provides general enhancement. A hybrid approach could be developed for real-world applications.