```python
# Importing necessary libraries
import os
import re
from collections import defaultdict


# Define preprocess function first
def preprocess(text):
    return re.findall(r'\b\w+\b', text.lower())


# Mounting google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
def load_documents(folder_path):
    docs = {}
    for filename in os.listdir(folder_path):
        if filename.endswith('.txt'):
            with open(os.path.join(folder_path, filename), 'r', encoding='utf-8') a
                docs[filename] = preprocess(file.read())
    return docs


def load_queries(query_file_path):
    with open(query_file_path, 'r', encoding='utf-8') as file:
        return [line.strip() for line in file.readlines()]


def compute_statistics(docs):
    doc_count = len(docs)
    term_freq = defaultdict(lambda: defaultdict(int))
    doc_freq = defaultdict(int)

    for doc_id, words in docs.items():
        unique_words = set(words)
        for word in words:
            term_freq[doc_id][word] += 1
        for word in unique_words:
            doc_freq[word] += 1

    return term_freq, doc_freq, doc_count


def compute_scores(query, term_freq, doc_freq, doc_count):
    scores = {}
    for doc_id in term_freq:
        score = 1.0
        for term in query:
            tf = term_freq[doc_id].get(term, 0)
            df = doc_freq.get(term, 0)
            p_relevant = (tf + 1) / (sum(term_freq[doc_id].values()) + len(doc_freq))
            p_not_relevant = (df + 1) / (doc_count - df + len(doc_freq))
            score *= (p_relevant / p_not_relevant)
        scores[doc_id] = score
    return scores


def retrieve(folder_path, query_file_path):
    docs = load_documents(folder_path)
    queries = load_queries(query_file_path)
    term_freq, doc_freq, doc_count = compute_statistics(docs)

    for query in queries:
        query_terms = preprocess(query)
        scores = compute_scores(query_terms, term_freq, doc_freq, doc_count)
        ranked_docs = sorted(scores.items(), key=lambda x: x[1], reverse=True)
        print(f"Query: {query}")
        for doc_id, score in ranked_docs:
            print(f"Document: {doc_id}, Score: {score:.4f}")
        print()

folder_path = '/content/drive/MyDrive/Tech400IR/wk5'
query_file_path = os.path.join(folder_path, 'query.txt')

retrieve(folder_path, query_file_path)
```

```
Query: The city beneath the sea was a marvel of human achievement, but it held dark secr
Document: query.txt, Score: 0.0234

Query: On the edge of the universe, explorers discovered a doorway to another reality.
Document: query.txt, Score: 0.7616

Query: A forgotten map found in an old library led them on a quest for an artifact of gr
Document: query.txt, Score: 0.0000

Query: A child, lost in the woods, found a hidden path that led to a world no one had ev
Document: query.txt, Score: 0.0000

Query: In a small village at the foot of the mountains, legends spoke of a hidden treasu
Document: query.txt, Score: 0.0006

Query: The explorers set sail on a vast ocean, chasing the horizon as it melted into the
Document: query.txt, Score: 0.0027

Query: The city beneath the sea was a marvel of human achievement, but it held dark secr
Document: query.txt, Score: 0.0000

Query: A scientist working late into the night discovered something that would change th
Document: query.txt, Score: 0.0015

Query: On the edge of the universe, explorers discovered a doorway to another reality. T
Document: query.txt, Score: 0.0729
```

```
Query: A scientist working late into the night discovered something that would change th
Document: query.txt, Score: 0.0094
```

```python
def retrieve_and_save(folder_path, query_file_path, output_file_path):
    docs = load_documents(folder_path)
    queries = load_queries(query_file_path)
    term_freq, doc_freq, doc_count = compute_statistics(docs)

    with open(output_file_path, 'w', encoding='utf-8') as output_file:
        for query in queries:
            query_terms = preprocess(query)
            scores = compute_scores(query_terms, term_freq, doc_freq, doc_count)
            ranked_docs = sorted(scores.items(), key=lambda x: x[1], reverse=True)
            output_file.write(f"Query: {query}\n")
            for doc_id, score in ranked_docs:
                output_file.write(f"Document: {doc_id}, Score: {score:.4f}\n")
            output_file.write("\n")


output_file_path = '/content/drive/MyDrive/Tech400IR/wk5/output.txt'


retrieve_and_save(folder_path, query_file_path, output_file_path)


print(f"Output has been saved to {output_file_path}")
```

```
Output has been saved to /content/drive/MyDrive/Tech400IR/wk5/output.txt
```