Project 1: Information Retrieval System Summary

Group Members

- Dongbing Han (dh3071)
- Ziang Xu (zx2462)

- **Files listed:**
  - README.pdf: General overview of the implemented algorithm and functions.
  - query.py: Compilation of the execution code for the information retrieval system program.
  - transcript.pdf: Highlighted the pertinent outcomes from the program's execution across the three test cases, alongside their corresponding precision values.

- **References:**
  - Google Search Engine ID: 62edd23ecda9c4041
  - Json API Key: AIzaSyC5v2l8vS4TZEIgsOeyrLUeBATHkmKBf58

- **Running Instruction:**
  - General search command:
    - python3 main.py <google_search_api_key> <search_engine_id> <target_precision> <query>
    - Reminder: When <query> consists of multiple words, ensure they are enclosed in quotes (e.g., "per se").
  - Look for information on the Per Se restaurant in New York City with target precision 0.8
    - python3 query.py AIzaSyC5v2l8vS4TZEIgsOeyrLUeBATHkmKBf58 62edd23ecda9c4041 0.8 "per se"
  - Look for information on 23andMe cofounder Anne Wojcicki with target precision 0.8
    - python3 query.py AIzaSyC5v2l8vS4TZEIgsOeyrLUeBATHkmKBf58 62edd23ecda9c4041 0.8 wojcicki
  - Look for information on COVID-19 cases with target precision 0.8
    - python3 query.py AIzaSyC5v2l8vS4TZEIgsOeyrLUeBATHkmKBf58 62edd23ecda9c4041 0.8 cases

- **Internal design Description:**
  - Retrieve Google Results: The main function responsible for this task is `call_google_api()`, which is executed once per iteration. It accepts the query, along with the necessary credentials (API key and search engine ID), and proceeds to make a call to the Custom Search Engine API. Once the results are obtained, the `getFormattedData()` function steps in to format each result and store them as dictionary objects, segregated by title, link, and snippet.
  - Obtain Relevance Feedback: Users are prompted to distinguish between relevant and irrelevant results through the terminal, facilitated by the `getLabeledData()` method. Recognized true labels encompass 'y', 'Y', 't', and 'T', while recognized false labels include

'n', 'N', 'f', and 'F'. Users provide feedback regarding the relevance of each result. After gathering relevance feedback, `getPrecisionScore()` verifies if the desired precision has been met. If the desired precision is attained, the program concludes after presenting this outcome to the user. If not, it proceeds to the next step.

- Compute Augmented Query: After receiving relevance feedback, the system progresses to compute the augmented query within the `build_inverted_list()`, `rocchio()`, and expand_query_list()` methods.
  - `build_inverted_list`: Constructs an inverted index from the labeled results, mapping terms to the documents they appear in.
  - `rocchio`: Implements the Rocchio algorithm to adjust query weights based on relevance feedback. It calculates weights for terms in the query, considering both relevant and irrelevant documents.
  - `expand_query_list`: Expands the original query by selecting additional terms based on their weights calculated by the Rocchio algorithm. This aims to improve subsequent search queries by incorporating terms likely to be relevant.

- **External libraries Reference:**
  - 18.6. Relevance feedback - Rocchio's algorithm with mathematical formulation
    - https://www.youtube.com/watch?v=h-O_PAcHT4Q
  - The Rocchio algorithm for relevance feedback
    - https://nlp.stanford.edu/IR-book/html/htmledition/the-underlying-theory-1.html

- **Query-modification method Description:**
  - The process begins by using the NLTK module with the`stop_words='english'` parameters to tokenize and remove English stop words from the corpus. This step serves to extract meaningful words as building blocks for further analysis. Then all the corpus are transformed by both relevant and non-relevant document corpora into TF-IDF vectors, utilizing tokenization through the `build_inverted_list` function.

  - Subsequently, employing Rocchio's algorithm with fixed parameters ALPHA = 1, BETA = 0.75, and GAMMA = 0.15, the augmented query vector is computed. This vector incorporates feedback from the relevant and non-relevant documents.

  - Following the generation of the augmented query, the `expand_query_list` function is invoked to identify words with the highest TF-IDF index. The two highest-ranking words, not already present in the query, are selected to augment the query. This process is iterated to refine the query further.

- **Additional Notes:**