Digital Information Retrieval

Part II

Students should implement a Retrieval System. In particular, two basic functions must be implemented:

1. Indexing Function: given a collection of documents stored on a dataset, the procedure creates the inverted index (the dictionary and all the postings lists) and stores it permanently (using a text file or a binary file). Due to the long execution time, indexing process should be run in advance
2. Retrieving Function: given a term written by the user through a user interface, the procedure searches for the term into the inverted index, then shows the list of all documents containing such a term.

Students also must also implement at least one feature belonging to the group A and at least one feature belonging to the group B:

Group A:

1. Extending the Retrieving Function to make the procedure able to process conjunctive and disjunctive queries with two or more terms (for each term, the corresponding posting list is retrieved, then all posting lists are intersected or united together)
2. Extending the Retrieving Function to make the procedure able to run Optimized Conjunctive Queries, by processing terms in order of increasing document frequency (just for the Conjunctive Queries)
3. Extending the Indexing Function by adding Stop Words, such that all the terms having a collection frequency higher than a threshold are dropped. Consequently, extends the Retrieving Function such that Stop Words are not searched into the index
4. Extending the Indexing Function by adding Stop List, such that all the terms belonging to such a list are dropped. Consequently, extends the Retrieving Function such that terms belonging to the Stop List are not searched into the index
5. Extending the Indexing Function in order to store into the index not just the term that has been found into the collection but also all the synonymous of the term (expansion at indexing time)
6. Extending the Retrieving Function in order to expand a term with a disjunction of all the synonyms of the term (expansion at searching time). For example, if the user searches for CAR, the system look for (CAR OR AUTOMOBILE)
7. Implementing the Edit Distance for detecting all the mistakes written by the user during query definition. The system should suggest possible alternatives
8. Implementing the K-Gram Overlap for detecting all the mistakes written by the user during query definition. The system should suggest possible alternatives
9. Implementing the Sounded Index for retrieving documents depending on the sound of the terms (and not in the syntax they are written)

Group B

1. Extending the Indexing Function by adding the Porter's Algorithm. Before storing the token into the vocabulary, it must be stemmed according to the rules defined by the Porter Stemmer
2. Extending the Retrieving Function by adding the Skip List in order to reduce the time spent by the Conjunctive Query (from linear to sublinear). You can use a fixed skip span or its value can be defined according to the heuristic described during lecture. Run empirical experiments to see how much the time is reduced
3. Extending the Indexing Function in order to use a BiWord Index (where terms are made by two nouns) by tagging words in noun and non-noun. Extend also the Retrieving Function for retrieving BiWords
4. Extending the Indexing Function in order to use a Positional Index. Even the Retrieving Function should be adapted to use even this new index
5. Making use of a Binary Tree for managing terms into the vocabulary. Extends the Retrieving Function in order to be able to work with WildCard Queries. Build also a Reversed Search Tree for retrieving all terms ending with a suffix
6. Making use of a B-Tree for managing terms into the vocabulary. Extends the Retrieving Function in order to be able to work with WildCard Queries. Build also a Reversed Search Tree for retrieving all terms ending with a suffix
7. Making use of a Permuter Index for managing permuterm into the vocabulary. Extends the Retrieving Function in order to be able to work with WildCard Queries
8. Making use of a K-Gram Index for managing k-grams into the vocabulary. Extends the Retrieving Function in order to be able to work with WildCard Queries
9. Extending the Indexing Function in order to build the index by using the Blocked Sort Based Indexing Algorithm
10. Extending the Indexing Function in order to build the index by using the Single Pass In Memory Algorithm
11. Extending the Indexing Function in order to discard those documents that are similar to others already computed (their terms have been indexed). Use the Shingling Algorithm for computing similarity
12. Extending the Indexing Function in order to discard those documents that are similar to others already computed (their terms have been indexed). Use the MinHash Signature for computing similarity

Optionally, students can also implement one or more of these features as an alternative of the of the Group A and B

1. By using MPI, develop a distributed index (index is built in parallel by several parser, then the posting lists are distributed among several inverters that are in charge of running queries)
2. Extending the Indexing Function in order to manage a compressed version of the Dictionary, for example managing the dictionary like a string or blocks of strings or using the Front Coding
3. Extending the Indexing Function in order to manage a compressed version of the Posting Lists, by using the Variable Byte Encoding or the Gamma Code
4. Extending the architecture of the Retrieval System by adding one or more Parametric Indexes or Zone Indexes and computing the document score

5. Extending the architecture of the Retrieval System by computing the document score making use of the Term Weighting (term frequency, term count, inverse document frequency, bm25 score function)
6. Extending the architecture of the Retrieval System by computing the document score making use of the Vector Space Model and the Cosine Similarity
7. Extending the Retrieval System by using a Tiered Index