CSE 515 Multimedia and Web Databases

Phase #3

(Due November 28th 2021, midnight – **no extensions***)*

Description: In this project, you will experiment with

- vector models,
- indexing and search,
- classification, and
- relevance feedback.

NOTES:

- In this phase, images to be inserted into the database will be labeled as `image-X-Y-Z.png`, where
  - $X \in \{cc, con, emboss, jitter, neg, noise01, noise02, original, poster, rot, smooth, stipple\}$ denotes the type of the image,
  - $1 \leq Y \leq 40$ denotes the subject ID, and
  - $1 \leq Z \leq 10$ denotes the image sample ID.
- The tasks in this phase involve the three feature models and similarity/distance functions developed in the previous phases.
- You cannot use existing libraries for classifiers.
- You cannot use existing libraries for LSH and VA-files.

PROJECT TASKS:

- **Task 1:** Implement a program which,
  - given a folder of images, one of the three feature models, and a user specified value of k, computes k latent semantics (if not already computed and stored), and
  - given a second folder of images, associates X labels to the images in the second folder using the classifier selected by the user:
    - an SVM classifer,
    - a decision-tree classifier, or
    - a PPR based clasifier,

    in this latent space.

  Also compute and print false positive and miss rates.

- **Task 2:** Implement a program which,
  - given a folder of images, one of the three feature models, and a user specified value of k, computes k latent semantics (if not already computed and stored), and
  - given a second folder of images, associates Y labels to each image in the second folder using the classifier selected by the user:
    - an SVM classifer,
    - a decision-tree classifier, or
    - a PPR based clasifier,

    in this latent space.

  Also compute and print false positive and miss rates.

- **Task 3:** Implement a program which,
  - given a folder of images, one of the three feature models, and a user specified value of k, computes k latent semantics (if not already computed and stored), and
  - given a second folder of images, associates Z labels to each image in the second folder using the classifier selected by the user:
    - an SVM classifer,
    - a decision-tree classifier, or
    - a PPR based clasifier,

    in this latent space.

  Also compute and print false positive and miss rates.

- **Task 4: Locality-Sensitive Hashing**
  - Implement a Locality Sensitive Hashing (LSH) tool, which takes as input (a) the number of layers, L, (b) the number of hashes per layer, κ, and (c) a set of vectors (generated by other tasks) as input and creates an in-memory index structure containing the given set of vectors. See

    > ”Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions” (by Alexandr Andoni and Piotr Indyk). Communications of the ACM, vol. 51, no. 1, 2008, pp. 117-122.

  - Implement similar image search using this index structure:
    - given a folder of images and one of the three feature models, the images are stored in an LSH data structure (the program also outputs the size of the index structure in bytes), and
    - given image and t, the tool outputs the t most similar images; it also outputs
      - the numbers of buckets searched as well as the unique and overall number of images considered
      - false positive and miss rates.

- **Task 5: VA-Files**
  - Implement a VA-file index tool and associated nearest neighbor search operations. The data structures and relevant algorithms are described in the following two papers:

    > Stephen Blott and Roger Weber, “A Simple Vector-Approximation File for Similarity Search in High-Dimensional Vector Spaces” 1997. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.9708
    >
    > Roger Weber, Hans-Jörg Schek, and Stephen Blott. 1998. “A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces”. In Proceedings of the 24rd International Conf erence on Very Large Data Bases (VLDB ’98), pp. 194-205. 1998.
    > http://dl.acm.org/citation.cfm?id=671192

    Given (a) a parameter b denoting the number of bits per dimensions used for compressing the vector data and (b) a set of vectors (generated by other tasks) as input, the program and creates an in-memory index structure containing the indexes of the given set of vectors. The program also outputs the size of the index structure in bytes.

  - Implement similar image search using this index structure:
    - given a folder of images and one of the three feature models, the images are stored in a VA-file data structure (the program also outputs the size of the index structure in bytes), and
    - given image and t, the tool outputs the t most similar images; it also outputs
      - the numbers of buckets searched as well as the unique and overall number of images considered
      - false positive and miss rates.

- **Task 6: Decision-tree-based relevance feedback:** Implement a decision tree based relevance feedback system to improve nearest neighbor matches, which enables the user to label some of the results returned by the search task as relevant or irrelevant and then returns a new set of ranked results, either by revising the query or by re-ordering the existing results.

- **Task 7: SVM-clasifier-based relevance feedback:** Implement an SVM based relevance feedback system to improve nearest neighbor matches, which enables the user to label some of the results returned by the search task as relevant or irrelevant and then returns a new set of ranked results, either by revising the query or by re-ordering the existing results.

- **Task 8: Query and feedback interface:** Implement a query interface, which allows the user to provide a query, relevant query parameters (including how many results to be returned). Query results are presented to the user in decreasing order of matching.

  The result interface should also allow to user to provide positive and/or negative feedback for the ranked results returned by the system.

  User feedback is than taken into account (*either by revising the query or by re-ordering the results as appropriate*) and a new set of ranked results are returned.

Deliverables:

- Your code (properly commented) and a README file.

- Your outputs for the provided sample inputs.

- A report describing your work and the results.

Please place your code in a directory titled “Code”, the outputs to a directory called “Outputs”, and your report in a directory called “Report”; zip or tar all off them together and submit it through the digital dropbox.