Instructions for Using the Experimental Testbed v.2

Eleftherios Spyromitros-Xioufis, espyromi@iti.gr
Multimedia Lab, ITI/CERTH, Thessaloniki, Greece

July 23, 2013

1 Basic Info[1]

We provide a library of java classes that can be used to reproduce the experimental results of our paper titled: "**Exploring performance trade-offs in large-scale image search**", currently under review. Below we briefly describe the basic components of this library and Section 2 gives specific details for running each experiment.

The basic components are the following:

- Classes of the `experimental_data_generation` package are used to extract the different types of VLAD vectors that are used in the different experimental sections. Most of them take as input a folder with pre-computed features for a set of images and create a BDB (Berkley Data Base) store that contains the VLAD vectors (along with the corresponding image identifiers) for these images.
- `dimensionality_reduction.PCAProjection` class takes as input a BDB store that contains full-dimensional VLAD vectors and a pre-computed PCA matrix and produces a set of BDB stores that contain PCA-projected (and optionally whitened) VLAD vectors.
- `utilities.IndexTransformation` class is used to transform an existing index (BDB store) of PCA-projected VLAD vectors into either of 3 types of indices: a) an index of lower-dimensional PCA-projected VLAD vectors (by truncating and re-normalizing them), b) an ADC index or c) an IVFADC index.
- `evaluation.EvaluationFromFile` is the evaluation class. It takes as input an evaluation setup file (described in Subsection 1.3) which provides the required information for performing the evaluation and generates an evaluation output file (described in Subsection 1.4) with the results of the

[1]For the pdf links to work correctly, this documentation file should be opened from its default location inside the folder where lsis.jar has been extracted.

evaluation. We provide sample evaluation setup and output files for all types of experiments in evaluation_setup_files and evaluation_output_files respectively.

1.1 Code for Feature Extraction / Pre-computed Features

`examples.SURForSIFTExtraction` can be used for the extraction of SURF and SIFT features from a set of images using various settings. Additionally we supply pre-computed SURF and SIFT features for the images of the Holidays dataset in data/features/.

1.2 Code for Learning / Pre-computed Learning Files

In data/learning_files/ you can find pre-computed learning files (codebooks, PCA matrices, etc.). Additionally we provide code for the generation of new learning files. Below is the list of classes that are used in various learning steps:

- `codebook_generation.SampleLocalFeatures` can be used for generating random samples from sets of local feature files.
- `codebook_generation.SimpleKMeansWithOutput` is a slightly modified (in order to generate additional output) version of Weka's SimpleKMeans.java class that is used for learning a k-means quantizer. k-means clustering is used for learning a)visual vocabularies, b)product quantizers and c)coarse quantizers.
- `dimensionality_reduction.PCALearning` can be used for learning PCA matrices including means, eigenvectors and eigenvalues (used when whitening is applied). The implementation is based upon and uses the EJML library.
- `product_quantizationProductQuantizerQLearning` can be used for learning product quantizers.
- `product_quantizationCoarseQuantizerQLearning` can be used for learning coarse quantizers.

1.3 Evaluation Setup Files

An evaluation setup file is a csv file that contains (in each row) the required info for performing an image retrieval evaluation. It contains the following attributes (see evaluation setup files in evaluation_setup_files for examples):

1. **collectionName**: One of {Holidays/Oxford/Paris}.
2. **groundTruth**: Full path to the file or folder that contains the ground truth for this collection. E.g. data/ground_truth/ground_truth_holidays.txt for Holidays.

3. **length**: Length of plain VLAD vectors.

4. **collectionPlainBDB**: Full path to the BDB store that contains the plain (not quantized) VLAD vectors of the collection. E.g. data/BDB_stores/-exp1/BDB_4096_surf_l2.

5. **distractorsPlainBDB**: Full path to the BDB store that contains the plain (not quantized) VLAD vectors of the distractors (you can use an arbitrary string to evaluate without distractors). E.g. data/BDB_stores/-exp1/BDB_4096_surf_l2_distractors.

6. **numDistractors**: The number of distractors to be merged with the collection or 0 to perform evaluation without distractors.

7. **indexType**: The type of the index, one of plain/adc/ivfadc. When adc or ivfadc are selected, additional parameters need to be specified.

8. **collectionAdcBDB***: Full path to the BDB store that contains the PQ codes of the collection vectors.

9. **distractorsAdcBDB***: Full path to the BDB store that contains the PQ codes of the distractor vectors.

10. **productQuantizer***: Full path to the product quantizer.

11. m*: m parameter (number of subquantizers) of the product quantizer.

12. k_s*: k_s parameter (centroids of each subquantizer) of the product quantizer.

13. **randomTransformation***: whether to perform random orthogonal transformation.

14. **coarseQuantizer****: Full path to the coarse quantizer.

15. k_c**: k_c parameter (number of centroids) of the coarse quantizer.

*Parameters used when adc or ivfadc index types are selected.
**Parameters used when ivfadc index type is selected.

1.4 Evaluation Output Files

Each row of an evaluation output type contains the results of the evaluation specified in the respective row of the corresponding evaluation setup file. It consists of the following attributes (see evaluation output files in evaluation_-output_files for examples):

1. **collectionName**: One of {Holidays/Oxford/Paris}.

2. **numDistractors**: Number of distractors that were merged with the collection for this evaluation.

3. **length**: Length of plain VLAD vectors.
4. **mAP**: Mean Average Precision.
5. **recall@100**: Recall at 100.
6. **collection look-up/search time**: Time in milliseconds required for name look-up/search on the collection images.
7. **distractors look-up/search time**: Time in milliseconds required for name look-up/search on the distractor images.

1.5 Installation and Setup

In order to be able to run the examples of the following section, follow these steps:

1. Download lsis.jar and extract the contents of the jar in a folder of your choice (e.g. C:/lsis).
2. Download the following compressed data files:
   - Pre-computed SURF and SIFT features for the images of the Holidays collection: features.zip
   - A selection of pre-computed learning files: learning_files.zip
   - A selection of pre-computed indexes: BDB_stores.zip

   and extract their contents in the /data subfolder of the folder where you extracted the contents of lsis.jar (e.g. C:/lsis/data).
3. cd to C:/lsis/

You can consult the developer documentation (Javadoc) for more details about the provided classes.

2 Experiment Details

2.1 Comparison of Local Features

Class for generating the data:
`experimental_data_creation.BestFeatureData`
Short description:
Given a folder with pre-computed SURF or SIFT features and two codebook files (with 64 centroids) (one learned using original and one learned using Power+L2-normalized features), generates two BDB stores of full-dimensional VLAD vectors.
Inputs:

1. Full path to the folder that contains the raw feature files in text or binary format. You can use the pre-computed features in data/features/.

2. Full path to the folder where the BDB stores will be created.

3. The type of features (sift or surf)

4. Full path to a codebook file with 64 centroids learned using L2-normalized features of the specified type. You can use the pre-computed codebooks in data/learning_files/codebooks/exp1/.

5. Full path to a codebook file with 64 centroids learned using Power+L2-normalized features of the specified type. You can use the pre-computed codebooks in data/learning_files/codebooks/exp1/.

Example Command:

```
java -cp "lib/*" experimental_data_creation.BestFeatureData "/
    data/features/surf/" "data/BDB_stores/exp1/" surf "data/
    learning_files/codebooks/exp1/surf_l2_64c.arff" "data/
    learning_files/codebooks/exp1/surf_power+l2_64c.arff" >
    exp1_log.txt
```

Supplied Learning files:
Four visual vocabularies with $k = 64$ centroids, each one learned using a different type of features (SURF_{l2}, SURF_{sr}, SIFT_{l2}, SIFT_{sr}) can be found in data/learning_files/codebooks/exp1/). These vocabularies were learned using two samples (one for SURF and one for SIFT) of approximately 100K features from the images in Flickr100K. These samples can be found in data/learning_files/feature_samples/.
Evaluation setup file: exp1_eval.txt
Evaluation output file: exp1_results.txt
Pre-computed BDB stores: data/BDB_stores/exp1

2.2 Feature Filtering

2.2.1 Based on Feature Intrinsic Structure

Class for generating the data:
`experimental_data_creation.FeatureFilteringIntrinsicData`
Short description:
Given a folder with pre-computed SURF features and a codebook file with 64 centroids learned using L2-normalized SURF features applies the selected feature filtering method to retain the desired percentage of features and generates a BDB store of full-dimensional VLAD vectors.
Inputs:

1. Full path to the folder that contains the raw SURF feature files in text or binary format. You can use the pre-computed features in data/features/.

2. Full path to the folder where the BDB stores will be created.

3. Full path to a codebook file with 64 centroids learned using L2-normalized SURF features. You can use the pre-computed codebook file data/learning_files/codebooks/exp1/surf_l2_64c.arff.

4. Percentage of features to be retained (valid percentages are: 0.5, 0.8, 0.9, 0.95).

5. Filtering method to be applied (random/entropy/variance).

   Additional parameters when the entropy-based filtering method is selected:

6. Number of equal-width bins to be used for discretization. We used 128 in our experiments.

7. Full path to the serialized discretization filter file. The filter used in our experiments can be found here: data/learning_files/exp2/sample_100K_s1_surf_l2_EW_128.filter.obj.

8. Full path to the serialized un-discretized Instances object. You can use this file: data/learning_files/exp2/sample_100_s1_surf.instances.obj.

PCA-projected vectors can be generated using the `dimensionality_reduction.PCAProjection` class and the pre-computed PCA matrix data/learning_files/pca/single_voc/pca_surf_k64_4096to1024.txt.
Example Command (full vectors):

```
java -cp ".;lib/*" experimental_data_creation.
    FeatureFilteringIntrinsicData "/data/features/surf/" "data
    /BDB_stores/exp2/" "data/learning_files/codebooks/exp1/
    surf_l2_64c.arff" 0.95 entropy 128 "data/learning_files/
    exp2/sample_100K_s1_surf_l2_EW_128.filter.obj" "data/
    learning_files/exp2/sample_100K_s1_surf.instances.obj" >
    exp2_log.txt
```

Example Command (pca-projected vectors):

```
java -cp ".;lib/*" dimensionality_reduction.PCAProjection "
    data/BDB_stores/exp2/BDB_4096_random_0.95" 4096 1491 "data
    /learning_files/pca/single_voc/pca_surf_k64_4096to1024.txt
    " "false" 128 > exp2_log.txt
```

Evaluation setup file: exp2_eval.txt
Evaluation output file: exp2_results.txt
Pre-computed BDB stores: data/BDB_stores/exp2

2.2.2 Based on Feature-Vocabulary Relation

Class for generating the data:
`experimental_data_creation.FeatureFilteringIntrinsicData`
Short description:
Given a folder with pre-computed SURF features and a codebook file with 64 centroids learned using L2-normalized SURF features applies the selected feature filtering method to retain the desired percentage of features and generates a BDB store of full-dimensional VLAD vectors.
Inputs:

1. Full path to the folder that contains the raw SURF feature files in text or binary format. You can use the pre-computed features in data/features/.
2. Full path to the folder where the BDB stores will be created.
3. Full path to a codebook file with 64 centroids learned using L2-normalized SURF features. You can use the pre-computed codebook file data/learning_files/codebooks/exp1/surf_l2_64c.arff.
4. Percentage of features to be retained (valid percentages are: 0.85, 0.90, 0.95).
5. Filtering method to be applied (dist/ratio/std).

   Additional parameters when the dist filtering method is selected:

6. Full path to the file containing the thresholds for each centroid and percentile for the dist method. The file used in our experiments can be found here: .

PCA-projected vectors can be generated using the `dimensionality_reduction.PCAProjection` class and the pre-computed PCA matrix data/learning_files/pca/single_voc/pca_surf_k64_4096o1024.txt.
Example Command (full vectors):

```
java -cp ".;lib/*" experimental_data_creation.
    FeatureFilteringIntrinsicData "/data/features/surf/" "data
    /BDB_stores/exp3/" "data/learning_files/codebooks/exp1/
    surf_l2_64c.arff" 0.90 dist "data/learning_files/exp3/
    percentiles.txt" > exp3_log.txt
```

Example Command (pca-projected vectors):

```
java -cp ".;lib/*" dimensionality_reduction.PCAProjection "
    data/BDB_stores/exp3/BDB_4096_dist_0.9" 4096 1491 "data/
    learning_files/pca/single_voc/pca_surf_k64_4096to1024.txt"
     "false" 128 > exp3_log.txt
```

Evaluation setup file: exp3_eval.txt
Evaluation output file: exp3_results.txt
Pre-computed BDB stores: data/BDB_stores/exp3

2.3 Aggregation Strategies

Class for generating the data:
`experimental_data_creation.SumVsMeanAggregationData`
Short description:
Given a folder with pre-computed SURF features and a codebook file with 64 centroids learned using L2-normalized SURF features generates a BDB store of mean-aggregated full-dimensional VLAD vectors.
Inputs:

1. Full path to the folder that contains the raw SURF feature files in text or binary format. You can use the pre-computed features in data/features/.
2. Full path to the folder where the BDB stores will be created.
3. Full path to a codebook file with 64 centroids learned using L2-normalized SURF features. You can use the pre-computed codebook file data/learning_files/codebooks/exp1/surf_l2_64c.arff.

PCA-projected vectors can be generated using the `dimensionality_reduction.PCAProjection` class and the pre-computed PCA matrix data/learning_files/pca/single_voc/pca_surf_k64_4096o1024.txt.
Example Command (full vectors):

```
java -cp ".;lib/*" experimental_data_creation.
    FeatureFilteringIntrinsicData "/data/features/surf/" "data
    /BDB_stores/exp4/" "data/learning_files/codebooks/exp1/
    surf_l2_64c.arff" > exp4_log.txt
```

Example Command (pca-projected vectors):

```
java -cp ".;lib/*" dimensionality_reduction.PCAProjection "
    data/BDB_stores/exp4/BDB_4096_mean" 4096 1491 "data/
    learning_files/pca/single_voc/pca_surf_k64_4096to1024.txt"
     "false" 128 > exp4_log.txt
```

Evaluation setup file: exp4_eval.txt
Evaluation output file: exp4_results.txt
Pre-computed BDB stores: data/BDB_stores/exp4

2.4 Vocabulary Size, PCA and Whitening

Class for generating the data:
`experimental_data_creation.BestVocSizeData`
Short description:
Given a folder with pre-computed SURF features and a comma separated list of codebook files learned using L2-normalized SURF features generates multiple BDB stores of full-dimensional VLAD vectors.
Inputs:

1. Full path to the folder that contains the raw SURF feature files in text or binary format. You can use the pre-computed features in data/features/.
2. Full path to the folder where the BDB stores will be created.
3. Comma separated list of full paths to the codebook files learned using L2-normalized SURF features. You can use the pre-computed codebook files in data/learning_files/codebooks/exp5.

PCA-projected (with and without whitening) vectors can be generated using the `dimensionality_reduction.PCAProjection` class and the pre-computed PCA matrices in data/learning_files/pca/single_voc.
Example Command (full vectors):

```
java -cp ".;lib/*" experimental_data_creation.BestVocSizeData
    "/data/features/surf/" "data/BDB_stores/exp5/"
"data/learning_files/codebooks/exp5/surf_l2_16c.arff,data/
    learning_files/codebooks/exp5/surf_l2_32c.arff,data/
    learning_files/codebooks/exp1/surf_l2_64c.arff"
"16,32,64" > exp5_log.txt
```

Example Command (pca-projected and whitened vectors):

```
java -cp ".;lib/*" dimensionality_reduction.PCAProjection "
    data/BDB_stores/exp5/BDB_2048" 2048 1491 "data/
    learning_files/pca/single_voc/pca_surf_k32_2048to1024.txt"
     "true" 128 > exp5_log.txt
```

Evaluation setup file: exp5_eval.txt
Evaluation output file: exp5_results.txt
Pre-computed BDB stores: data/BDB_stores/exp5

2.5 Multiple Vocabularies

Class for generating the data:
`experimental_data_creation.MultiVocData`
Short description:
Given a folder with pre-computed SURF features and a codebook list file (that contains full paths and sizes of multiple vocabularies learned using L2-normalized SURF features) generates a BDB store of multiple vocabulary aggregated full-dimensional VLAD vectors.
Inputs:

1. Full path to the folder that contains the raw SURF feature files in text or binary format. You can use the pre-computed features in data/features/.
2. Full path to the folder where the BDB stores will be created.
3. Full path to a text file that contains the list of codebook files (full paths and sizes) that will be used for the generation of the vectors. You can find such a text file and the corresponding codebooks in: data/learning_files/codebooks/exp6.

PCA-projected (with and without whitening)vectors can be generated using the `dimensionality_reduction.PCAProjection` class and the pre-computed PCA matrix data/learning_files/pca/mvoc/pca_surf_kx128_32768to1024.txt.
Example Command (full vectors):

```
java -cp ".;lib/*" experimental_data_creation.MultiVocData "/
    data/features/surf/" "data/BDB_stores/exp6/" "data/
    learning_files/codebooks/exp6/4x128.txt" 4 > exp6_log.txt
```

Example Command (pca-projected vectors):

```
java -cp ".;lib/*" dimensionality_reduction.PCAProjection "
    data/BDB_stores/exp6/BDB_32768_4x128" 32768
1491 "data/learning_files/pca/mvoc/pca_surf_4x128_32768to1024.
    txt" "true" 128 > exp6_log.txt
```

Supplied Learning files:
–

Evaluation setup file: exp6_eval.txt
Evaluation output file: exp6_results.txt
Pre-computed BDB stores: data/BDB_stores/exp6

2.6 Product Quantization Parameters

Class for generating the data:
`utilities.IndexTransformation`
Short description:
The datasets used in this experiment can be generated by transforming the already extracted VLAD vectors into Product Quantization codes using the `utilities.IndexTransformation` class and the quantizers (product/coarse) provided in data/learning_files/quantizers.
Command to transform 128 dimensional vectors to 48 dimensional vectors:

```
java -cp ".;lib/*" utilities.IndexTransformation "data/
    BDB_stores/exp6/BDB_32768_4x128to128w/" "data/BDB_stores/
    exp7/BDB_32768_4x128to48w/" 128 48 1491
small > exp7_log.txt
```

Example to transform 128 dimensional vectors to ADC index:

```
java -cp ".;lib/*" utilities.IndexTransformation "data/
    BDB_stores/exp6/BDB_32768_4x128to128w/"
"data/BDB_stores/exp7/BDB_32768_4x128to48w_pq8x10/" 128 48
    1491 adc "data/learning_files/quantizers/pq_48_8x10_50000.
    txt" 8 1024 false > exp7_log.txt
```

Example to transform 128 dimensional vectors to ADC index:

```
java -cp ".;lib/*" utilities.IndexTransformation "data/
    BDB_stores/exp6/BDB_32768_4x128to128w/" "data/BDB_stores/
    exp7/BDB_32768_4x128to48w_rpq8x10_ivf1024/" 128 48 1491
```

```
ivfadc "data/learning_files/quantizers/
rpq_48_8x10_50000_c1024.txt" 8 1024 false "data/
learning_files/quantizers/qc_k1024_50000.txt" 1024 >
exp7_log.txt
```

Evaluation setup file: exp7_eval.txt
Evaluation output file: exp7_results.txt
Pre-computed BDB stores: data/BDB_stores/exp7