Contents

1 Quick tutorial on generating L2R feature files

1.1 Preliminaries

- Make sure ElasticSearch is running and the indices (datasets) are up
- for TREC (4+5) and NTCIR 2 certain input files are needed: topics file, documents files, relevance scores file. The document file contains the document text + id and is needed for the features which are not calculated by the ES engine but rather "on-the-fly" (all necessary files I have precomputed and can be found under: TODO)
- Switch to: `$ cd your_git_folder/moving/Code/elastify`
- Make sure you have the latest git master updates (assuming you are on the master branch): `$ git pull origin`
- Activate the virtualenv for elastify by running the following command: `$ source EVE/bin/active`
- Install the changes locally: `$ python setup.py install`
- There is one script for the datasets without goldstandard (GS) and one for datasets providing goldstandard
  - trainify_nogs (no goldstandard, e.g. economics)
  - trainify_no (with goldstandard, e.g. trec or ntcir)

1.2 Preprocessing input files

Before one can run the elastify scripts for a dataset with GS provided, we need to preprocess the input files. Here are the requirements for the formats (remove header columns beforehand!):

- goldstandard file: `topicID \t docID \t score`
- topics (=queries) files: `topicID \t topicText`
- document files: `docID \t docText`

The python lib pandas can be a very handy tool for preprocessing data.

1.3 Note on the parameters

1.3.1 without GS

- queryfile: query file with one query per line
- -i/–index: the index
- -f/–field: field of the index

- -d/–doctype: doctype of the index
- -I/–gold_index: the index used for the goldstandard retrieval
- -g/–gold-strategy: the strategy for the goldstandard (default: fulltext)
- -s/–strategies: the strategies to use
- -S/–size: Number of documents per query for non-gold strategies
- -l/–lindex: the index for the language models
- -F/–lfield: the field for the language model index
- -B/–batches: batches for the goldstandard
- -o/–outfile: path to write the output
- -T/–type: type of output (l2r, txt, etc)
- -w/–w2vmodel: path to the word2vec model file
- -m/–mu: mu parameter for mk language model score
- -x/–let_mu: mu parameter for letor language model with dirichlet smoothing
- -y/–let_alpha: alpha parameter for letor language model with jelinek-mercer smoothing
- -z/–let_delta: delta parameter for letor language model with absolute discounting smoothing

1.3.2 with GS

- queryfile: query file with one query per line in format given above
- docsfile: file containing all documents with corresponding id in format given above
- goldfile: file containing all relevances scores in format given above
- -i/–index: the index
- -f/–field: field of the index
- -d/–doctype: doctype of the index
- -I/–gold_index: the index used for the goldstandard retrieval
- -g/–gold-strategy: the strategy for the goldstandard (default: fulltext)
- -s/–strategies: the strategies to use
- -S/–size: Number of documents per query for non-gold strategies
- -l/–lindex: the index for the language models
- -F/–lfield: the field for the language model index
- -B/–batches: batches for the goldstandard
- -o/–outfile: path to write the output
- -T/–type: type of output (l2r, txt, etc)
- -w/–w2vmodel: path to the word2vec model file
- -m/–mu: mu parameter for mk language model score
- -x/–let_mu: mu parameter for letor language model with dirichlet smoothing
- -y/–let_alpha: alpha parameter for letor language model with jelinek-mercer smoothing
- -z/–let_delta: delta parameter for letor language model with absolute discounting smoothing

1.4 Example calls

1.4.1 Example: NTCIR2 with titles

```
gstrainify /path/to/topics/topics.txt /path/to/titles/titles.txt

/path/to/relscores/rel_scores.txt -i ntcir_titles -f title -d publication

-s mk sm letor ntcir_cfidf ntcir_hcfidf ntcir_bm25 -l ntcir_fulltext

-F fulltext -m 10.0 -x 2000.0 -y 0.1 -z 0.7

-w /path/to/w2vmodel/GoogleNews-vectors-negative300.bin -T l2r

-o /path/to/output/titles_features.txt
```

1.4.2 Example: TREC 4+5 with fulltext

```
gstrainify /path/to/topics/topics.txt /path/to/fulltext/fulltext.txt
/path/to/relscores/rel_scores.txt -i trec_fulltext -f fulltext -d
publication -s mk sm letor trec_cfidf_f trec_hcfidf_f trec_bm25_f -l trec_fulltext
-F fulltext -m 10.0 -x 2000.0 -y 0.1 -z 0.7 -w
/path/to/w2vmodel/GoogleNews-vectors-negative300.bin
-T l2r -o /path/to/output/fulltext_features.txt
```

1.4.3 Example: ZBWEconomics with titles

```
nogstrainify -i economics -f title -d publication -g fulltext -I economics -s mk sm letor
cfidf hcfidf bm25 -S 100 -l economics -m 10.0 -x 2000.0 -y 0.1 -z 0.7
-w /path/to/w2vmodel/GoogleNews-vectors-negative300.bin
-T l2r -o /path/to/output/feature_file.txt
```

1.5 Learning to Rank

As soon as the feature files have been generated, one can start to run RankLib experiments. Here I provide sample calls (Link to RankLib How-To):

```
java -jar RankLib.jar -train path/to/feature/file.txt -ranker 1
-metric2t MAP -tvs 0.75 -kcv 5
```

That will use RankNet (*ranker 1*), **m**ean **A**verage **P**recision as optimization and evaluation metric, a training-validation split of 0.75-0.25 (*tvs*) and 5-fold cross-validation (*kcv*)

If we want to use a subset of the features, one has to provide an additional feature file (one line = nr of one feature):

```
java -jar RankLib.jar -train path/to/feature/file.txt -ranker 1
-feature path/to/featuresubset/file.txt -metric2t MAP -tvs 0.75 -kcv 5
```

1.6 Creating the Correlation-based feature subset scores

For finding a meaningful subset of the whole feature list, I wrote a script which evaluates all possible combinations of all sizes of a feature set and reports the best scoring feature subset for each size. The file can be found in the moving git under Code/elastify/feat_sel/cc.py . The following parameters exist

- input: the input feature file in RankLib format
- -F/–features [optional]: file with feature names in right order line by line, if not provided the column nrs will be used to identify the features
- -f/–from_nr [optional]: size of subsets to start with, default = 1
- -t/–to_nr [optional]: size of subsets to end with (including), default = 0 (0 will be evaluated to length of feature list)
- -o/–output [optional]: output file path to write the results to, default=stdout
- -O/–type [optional], the format output type, default='csv', choices=['csv','tsv']

Example call:

```
python cc.py path/to/feature/file.txt --features path/to/features.txt -f 28 -t 29
```