Indexer

This module creates Indri indexes for each given collection.

Input

1. The semi-structured text file (data_store_structure.txt) containing the list of corpora, their specifications, and locations where to create the indexes.
2. Indexing parameter file containing the instructions to Indri software needed to build the index (normalization, stopwords, stemmer, etc.)
3. Root path of the collection
4. Output folder to store log files

Example of data_store_structure.txt:

```
[corpora_specs]
relative_directory=NIST-data
all_languages_included=ka
all_types_included=text;;audio
number_of_queries=0
number_of_corpora=2
created=20210521_112822
created-by=SCRIPTS

[corpus_1]
name=3B/IARPA_MATERIAL_OP2-3B/EVAL
language=ka
type=text

[location]
source_location=text/src

[location]
SentSplitter_location=text/sentSplitter_store/sent-split-v5.0
SentSplitter_version=sent-split-v5.0
SentSplitter_source=text/src
…
[location]
MT_location=text/mt_store/umd-nmt-v7.2_sent-split-v5.0
MT_version=umd-nmt:v7.2
MT_source=text/sentSplitter_store/sent-split-v5.0
...
[index]
index_out_root_location=text/index_store
indexer_version=indexing-umd:v7.9
```

Indexing parameter files are listed in the Appendix.

Output

Output is the set of indexes that are created for each location (corpus) listed in the data_store_structure file. The indexer generates the Indri and Anserini indexes. Along with indexes some auxiliary files are produced as well.

* The input data_store_structure file is modified in-place.

Docker Commands

To run the docker images pass the following environment variables to docker using the '-e' flag:

`INDEX_PARAMS=<filename>`	`filename with the Indri parameters`
`REINDEX=True|False`	`flag to reindex existing indexes`

and volume mounts using '-v' flag:

`-v <data_store_structure>:/media/data/data_store_structure.txt`	`path to data_store_structure`
`-v <dir>:/media/params_dir`	`directory with INDEX_PARAMS file`
`-v <log_dir>:/media/log_dir/`	`directory to save logs`

Examples

```
docker run --rm \
-e "INDEX_PARAMS=indexing_params_v1.0.txt" \
-e "REINDEX=True" \
-v
/storage/proj/ezotkina/indexing_development/data_store_farsi_split.tx
t:/media/data/data_store_structure.txt \
-v
/storage/proj/ezotkina/indexing_development/indexing_params:/media/pa
rams_dir \
-v /storage/data/:/media/data/source \
-v
/storage/proj/ezotkina/indexing_development/farsi_split_log:/media/lo
g_dir \
--name indexing-umd_v7.9 indexing-umd:v7.9
```

System Requirements

- CPU
- RAM

Standalone

Yes

Approach

The indexer generates two types of indexes:

-- Indri. Two kinds of Indri indexes are produced: one with default Indri character normalization (‘indri’) and the second with the custom normalization (‘indri_T_N’ which stands for **T**okenization followed by **N**ormalization). Moses tokenizer is used, custom character normalization is applied.

-- Anserini. Anserini index is produced only for MT output.

Notes

Appendix: Indexing parameters files:

indexing_params_v1.0.txt
```
[general]
skip_location=Language_Identification_location;;Domain_Identifi
cation_location
clean=true

[indri]
memory=512m
normalize=true
### Do not modify the following unless you are sure what you
are doing ###
```

```
### Keep empty fields ###
corpus_class=trectext
field=TEXT,s
offsetannotationhint=unordered
annotations=
metadata=
stemmer=
stopper=
```

indexing_params_porter_stemmer_v2.0.txt

```
#Porter stemmer + no stopwords

[general]
#Skip all locations except MT
skip_location=Language_Identification_location;;Domain_Identifi
cation_location;;source_location;;SentSplitter_location;;Morpoh
ological_Analysis_location;;ASR_location
clean=true

[indri]
memory=512m
normalize=true
### Do not modify the following unless you are sure what you
are doing ###
### Keep empty fields ###
corpus_class=trectext
field=TEXT,s
offsetannotationhint=unordered
annotations=
metadata=
stopper=
stemmer=porter
```