Query Analyzer (QA)

The Query Analyzer module uses input queries in IARPA format, parses them, and produces queries in the custom format described below.

Input

Query list: original IARPA query list format

```
query_id        query_string
query7169       "strategic decision","early childhood education"+
query5967       vomit[syn:to throw up]
query5718       sweat[syn:perspiration]
query8193       father,uncle
...
```

Output

(please also indicate if this component modifies the input in-place)

Three JSON files per each query are generated: queryN, queryN.lft, queryN.rgt.
*The inputs are not modified in-place

```
{
    "IARPA_query": "\"strategic decision\",\"early childhood
education\"+",
    "query_id": "query7169",
    "language": {
        "ISO_639_1": "bg"
    },
    "queries": [
        {
            "type": "words",
            "indri": "#combine (strategic decision early childhood
education)"
        },
      {
            "type": "phrases-simple_conjunction_075",
            "indri": "#weight (0.75 #band (strategic decision early
childhood education #uw2 (strategic decision)  #uw2 (early
childhood education) ) 0.25 #combine (strategic decision early
childhood education #uw2 (strategic decision)  #uw2 (early
childhood education) ) )"
…
{
            "type": "EdiNMT_words_part_flat",
            "indri": "#combine (стратегическо решение за
образованието в ранна детска възраст)"
        },
…
{
            "type": "PSQ_tokenized_normalized+berk+transl",
            "indri": "#combine (  #wsyn( 0.41636162510056324
стратегически  0.17048471440064364 стратегическо
0.13228077232502014 стратегическа
…
0.00225295763833463 обучение  0.0015154423576241901 образователен
0.0013032804275568035 обучението  0.0010406037522352773
образователния ))",
…
```

If query is of conjunction type, two separate JSON files are generated - one for the left part (*.lft) and one for the right part (*.rgt). Each part is parsed as a separate non-conjunction query in this case. If query is of non-conjunction type, empty *.lft and *.rgt are created.

Docker Commands

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

`QUERY=<filename>`	`filename with the list of input queries`
`EXPANSION=True|False`	`Query expansion`
`EdiNMT=<hostname:8081>`	`Edinburgh NMT server to translate query (null to disable)`
`LNG=<language code>`	`language ISO 639-1 code (tl, sw, kk, etc.)`

and volume mounts using '-v' flag:

`-v <input_dir>:/media/in_dir/`	`directory with input query file`
`-v <output_dir>:/media/out_dir/`	`directory to save output`
`-v <log_dir>:/media/log_dir/`	`directory to save logs`

One extra parameter, --network <docker bridge network name>, can be used to utilize the query translation feature (described below).

Examples

```
docker run --rm \
-e "QUERY=georgian_query2_list.tsv" \
-e "LNG=ka" \
-e "EXPANSION=true" \
-e "EdiNMT=null" \
-v /storage/proj/ezotkina/query_analyzer_development:/media/in_dir/ \
-v /storage/proj/ezotkina/query_analyzer_development/ka_output:/media/out_dir/
\
-v /storage/proj/ezotkina/query_analyzer_development/ka_log:/media/log_dir/
\
--name query-analyzer-umd-v17.0 query-analyzer-umd:v17.0
```

The above example shows how to run QA without the query translation option.

The following steps show how to run QA with the query translation feature:

1) setup docker network giving, e.g. ‘edinetwork’, name:

```
docker network create edinetwork
```

2) start Edinburgh NMT connecting the container to the ‘edinetwork’ network using --network docker flag:

```
docker run --gpus all --rm \
-e TYPE=text \
-e MODE=accurate \
-e SRC=en -e TGT=ka -e DEVICES=0 -e USE_JSON=True
--name edinmt \
--network edinetwork \
scriptsmt/systems:v22.1.2 serve
```

3) run query analyzer connecting the container to the same ‘edinetwork’:

```
docker run --rm \
--network edinetwork \
-e "QUERY=georgian_query2_list.tsv" \
-e "LNG=ka" \
-e "EXPANSION=true" \
-e "EdiNMT=edinmt:8081" \
-v /storage/proj/ezotkina/query_analyzer_development:/media/in_dir/ \
-v /storage/proj/ezotkina/query_analyzer_development/ka_output:/media/out_dir/
\
-v /storage/proj/ezotkina/query_analyzer_development/ka_log:/media/log_dir/
\
--name query-analyzer-umd-v17.0 query-analyzer-umd:v17.0
```

The Edinburg NMT docker container is running as a server, QA sends requests (English terms to translate) to it. MT docker container name is used as a hostname for QA `EdiNMT` parameter.

System Requirements

- CPU
- RAM

Standalone

Yes

Approach

QA gets a query in IARPA format and parses it according to the MQL specifications. ANTLR Java library (https://www.antlr.org/about.html) is used to parse CFG. The parser provides easy access/extraction of different parts of the query (words, constraints, etc.). By using those different parts the query can be built in in any format we need. For each query, many different 'query types' (or query variations) are generated. Each custom 'query type' is an Indri query per se (since one of our search engines is Indri).

Examples of query variations that QA generates:
-- bag of words query including the words from synonyms, hypernyms, event frames and example_of constraints, or
-- only bag of words query and no synonyms, hypernyms, event frames and example_of part of the queries are used, or
-- bag of words plus query expansion from New York Times/CC News collections, or
-- query terms translated from English to the target language by Edinburgh MT docker component, or
-- PSQ queries: query terms translated by using TPM with assigned weights,
-- etc.

To generate Indri queries, the other local resources are used such as:
1) Translation Probability Matrix (TPM) provided by UMD MT team
2) (Indri) Indexed New York Times collection and Common Crawl News 2018-2019 collection for the query expansion
3) (Edinburg) MT server that translates query from English to the target language

1) and 2) are included into the QA docker component.
3) is an external resource which is Edinburgh MT docker image (part of the deliverables).

Along with Indri query, auxiliary information is stored -- for example, all translations (extracted from the TPM) for each term that are used in the custom search engine.

Query in custom JSON format is the input to the Matcher docker component.