Introduction to QBASHER

Version 1.5.124-OS

Microsoft

March 14, 2018

QBASHER is a stand-alone indexing and querying system, based on an inverted file format optimized for short texts. Example short text collections include academic paper titles, song lyric lines, and logs of search queries. QBASHER is capable of supporting word-order-independent query auto-suggest, query classification, and simple retrieval.

QBASHER is designed to handle many billions of records in a single index and to achieve high query processing rates and low latency query responses. Data to be indexed is expected to be in TSV (tab-separated-value) format with a minimum of two columns. Column one contains the document text, column two is expected to contain a numeric quantity representing the static score (popularity, authority, etc) of the document. Extra columns may contain additional application-specific information such as latitude and longitude, object-id within a knowledge base, or a display-oriented form of the document. In normal operation, QBASHER internally reorders documents into order of descending static score, after normalising the scores into the range $0 - 1$. During retrieval, candidates are ranked by a linear combination $\alpha - \theta$ of features, including the static score. One of the features is a BM25 score. In a query classification scenario, different features are used: $\omega - \chi$.

In order to calculate features used in scoring candidates and to perform matching of word prefixes and rank-only words, QBASHER uses the raw document text as its own "per-document-index". In other words, features are checked or calculated using string scanning. This makes sense given the assumed shortness of the documents. To cut down on the amount of scanning required a Bloom filter based on word first letters is used.

QBASHER has three essential components:

QBASHI Indexer executable

QBASHQ-LIB Library implementing the query processing API for both C and C# p-invoke front-ends.

QBASHQ or QBASHQsharpNative Query processing front-ends in C and C# respectively which process queries using the QBASHQ-LIB libraries.

QBASHER supports the following query processing operators:

`"..."`	Phrase match, e.g. `"san francisco"`
`[...]`	Disjunction, e.g. `[cat dog] food`
`~`	Rank-only operator, e.g. `queens of england ~victoria`
`/`	Match a word prefix, e.g. `Seattle /seah`
`<`	Match a line prefix, e.g. `<Faceb`
Phrase within disjunction	e.g. `["mountain lion" "snow leopard"]`
Disjunction within phrase	e.g. `"[metal plastic] lid"`

QBASHER is written in C and can be built with either GCC or Visual Studio 2015. The query processing logic is implemented in a DLL which has an API, designed to be "p-invoked" from a C-sharp main program. An example C-sharp main program is included for illustration and testing purposes.

The `QBASHER/scripts` directory contains a reasonably extensive suite of test scripts, written in `perl5`. These tests must pass prior to checking in any modifications to the code.

1 Getting going with QBASHER

1.1 Obtaining QBASHER

Assuming you are working with a Unix style shell, e.g. on Linux or MacOSx, under Cygwin, or under Ubuntu Bash for Windows 10 ....

```
git clone "https://github.com/Microsoft/QBASHER"
```

1.2 Prerequisites

To use the system you need first to build executables from their sources. For this you need to have installed either `gcc`[1] and `make` or Microsoft Visual Studio 2015.

To use the system you will also need to have `perl 5` installed. Finally, to re-generate the PDF of this document you will need a version of LaTeX.

1.3 Building with gcc / make

If you have `gcc`, `make`, and `perl5` installed and in your path you can just type:

```
cd QBASHER/src    # Change to the src directory of the cloned repository
make cleanest
make
```

On some Linux distributions, you may need to use position independent code to compile the PCRE2 libraries correctly. If this is the case, you can use:

```
make fPIC=1
```

Later, you can save some time by using 'make cleaner' instead of 'make cleanest' – it avoids the rebuilding of the PCRE2 libraries.

Libraries and executables will appear in the QBASHER/src directory.

After successfully building the libraries and executables, you should run the test suite, following the instructions in `QBASHER/scripts/README.txt`. See `QBASHER/test_queries/README.txt` for information about the sets of test queries used by the test suite.

1.4 Building with Visual Studio 2015

To build the executable using Visual Studio 2015 (VS 2015), please navigate to `QBASHER/src/visual_studio` using the file explorer, then double click on `visual_studio.sln`. That should open the VS 2015 application. In the lower part of the toolbar at the top of the VS window, select `Release` and `x64` from the first and second drop-downs. (The QBASHER design assumes 64-bit. You will need to change a number of configuration parameters if you wish to build a Debug version.) Then click on `Rebuild solution` from the `Build` menu.
Assuming the build succeeds you should see:

```
========== Rebuild All: 9 succeeded, 0 failed, 0 skipped ==========
```

in the output pane, and all the EXEs and DLLs will be in:

```
QBASHER/src/visual_studio/x64/Release/
```

You should then run the test suite, following the instructions in `QBASHER/scripts/README.txt`. See `QBASHER/test_queries/README.txt` for information about the sets of test queries used by the test suite.

[1] The MacOS version of `gcc` which is actually `Clang`, see `https://clang.llvm.org/comparison.html` also works.

2 Help on options

If you run any of the .EXE files from the command line, without specifying any options, that EXE will print a list of all available options, each with its default value and a one line description. Options whose names start with "x_" are considered experimental and are more likely to be removed or changed in future releases. At the time of writing (12 Dec 2017) there are some such options which have proved their worth and which have been thoroughly tested, but which have not yet had the 'x_' dropped.

3 Examples of command-line usage

In each of the following examples, it is assumed that the current working directory is one level above the `src` directory, and that there is a `test_data/wikipedia_titles` directory containing a file `QBASH.forward` in TSV format with the records to be indexed in column one, static weights in column two, text for display in column three.

3.1 Simplest indexing of a data set

```
src/visual_studio/x64/Release/QBASHI.exe -index_dir=test_data/wikipedia_titles
```

The `QBASH.forward` file is indexed, resulting in three additional files being created in the same directory: `QBASH.if`, `QBASH.vocab`, `QBASH.doctable`. Remember that `QBASH.forward` is also part of the index and is needed for query processing. Note that, by default, the records in `QBASH.forward` are internally (i.e. no change to the file itself) sorted in descending static score order.

3.2 Running a single query against the simplest index

```
src/visual_studio/x64/Release/QBASHQ.exe
                index_dir=test_data/wikipedia_titles -pq="united states elections"
src/visual_studio/x64/Release/QBASHQ.exe
                index_dir=test_data/wikipedia_titles -pq="united states elections /pr"
src/visual_studio/x64/Release/QBASHQ.exe
               index_dir=test_data/wikipedia_titles -pq="united states elections pr" -auto_partials=ON
```

In the first example, `QBASHQ` retrieves up to eight records which contain all three words and ranks them in descending order of static score:

```
Voter turnout in the United States presidential elections       0.62035
United States elections, 2016   0.60078
List of United States presidential elections by popular vote margin     0.56947
List of third party performances in United States elections     0.51663
List of United States presidential elections by Electoral College margin        0.48337
United States elections, 2014   0.48337
United States House of Representatives elections, 2012  0.47945
United States Senate elections, 2018    0.44031
```

In the second example, the `/pr` indicates that "pr" is to be treated as a word prefix rather than a word. The set of records containing "united", "states", and "elections" are filtered to remove those which don't contain a word beginning with "pr".

```
Voter turnout in the United States presidential elections       0.62035
List of United States presidential elections by popular vote margin     0.56947
List of United States presidential elections by Electoral College margin        0.48337
Electoral vote changes between United States presidential elections     0.37573
Lists of newspaper endorsements in United States presidential elections 0.33072
United States presidential elections in Missouri        0.24853
United States presidential elections    0.18591
Canada and the United States presidential elections     0.18591
```

The third example produces exactly the same results as the second, because the `auto_partials` option treats the last word in the query (unless followed by a space) as though it had a leading slash.

3.3 Running a batch of queries against the simplest index

```
src/visual_studio/x64/Release/QBASHQ.exe
                index_dir=test_data/wikipedia_titles -file_query_batch=test_queries/emulated_log_1k.q
src/visual_studio/x64/Release/QBASHQ.exe
                index_dir=test_data/wikipedia_titles
                -file_query_batch=test_queries/emulated_log_1k.q -query_streams=1
```

Assuming that `test_queries/emulated_log_1k.q` contains a set of queries, one per line, the first example will run all the queries and print the results on standard output. The queries will be run using multiple (ten by default) threads, and the order of results will not in general correspond to the order of queries in the query batch. In the second example, the `query_streams` option runs the queries single-threaded and presents results in the same order as the queries appear in the file.

4 Committing changes to QBASHER

Please update the QBASHER version number recorded near the head of `src/shared/QBASHER_common_definitions.h`. If you change the format of the index, increment the `INDEX_FORMAT` definition in the same place and reset `QBASHER_VERSION` to ".0".

After updating the version number please append a comment to the tail of the file `src/shared/QBASHER_version_history.txt` detailing what changes you made, and why. Follow the format of the previous entries.

Before committing changes, please make sure that at least the BASIC tests in the test suite pass. Then you can use `make git` to issue all the necessary 'git add' commands to ensure that all the necessary items (and none of the derivative items) are added to the commit list. Then do

```
git commit -m "new version number and/or synopsis of your brilliant changes"
git push <options>
```

The 'git commit' will commit your changes to your local repository. The 'git push' can be used to contribute your changes to the github project repository. If you are an authorized contributor 'git push' with no arguments will do the trick. At this stage, I'm not clear on what is required to become an authorized non-Microsoft contributor.