I. Installation and preparation

I.1 Unpack the MarcXimiL archive

Use your favorite archive opener.

I.2 Choose a collection

The MARC XML collection files must be stored in the `var` subdirectory. A few sample collections are included in the distribution. Here we'll use the `2k_a.xml` file containing 1990 original records and 10 artificial duplicates, but feel free to try another.

I.3 Choose your strategy

The `etc` subdirectory contains various configuration files that we have used in the past. For a first experiment it's better to pick one of these and start designing your own when you've become comfortable with the program.

II. Executing MarcXimiL from the command line

MarcXimiL is executed from the command line, which means you will have to open a command window (Terminal, DOS Prompt or other depending on your operating system). The main script is `similarity.py`, located in the `bin` subdirectory – make sure it's your working directory in order to run it (either by going there using the `cd` command or by opening a command shell directly there if your system allows it). You might also want to add the Python directory to your PATH if it is not already included. Otherwise you will have to type longer commands such as `c:\python2.7\bin\python similarity.py` instead of `python similarity.py` or `similarity.py`.

The command line options have changed significantly with respect to the original version of 2009.

II.1 The big red reset switch

```
similarity.py -r
```

This command will delete all data from MarcXimiL's internal database (strategies, XML data and output files are retained). After that you're ready for a fresh start.

II.2 So-called dummy mode: 1-step analysis

This mode works mostly like the original MarcXimiL version of 2009, but with added options (including all parameters inside the strategy file is no longer required). One single command will take care of loading the MARC XML data and execute the comparison. By default the program will use the strategy `similarity_config__ubiquist2_sxic`, another one can be specified with the `-s` switch.

`similarity.py` **`-1`** `2k_a.xml` # load file `var/2k_a.xml`, compare, and write results to `var/log/one_results.txt`

`similarity.py -1 "2k_a.xml smallcoll.xml"` # compares the two collections, results in `var/log/one_results.txt`

`similarity.py -1 2k_a.xml -z xml` # output format : xml

`similarity.py -1 2k_a.xml -n 1000` # writes the 1000 first results instead of default (500)

`similarity.py -1 2k_a.xml -s similarity_config__digrams_ic` # uses alternate similarity strategy

II.2 Advanced usage

Version 0.3 has introduced a more advanced workflow where reading data, comparing records and writing the results are handled separately. It makes MarcXimiL more flexible: for example, one can re-use the same data (which could be slow to load) in various contexts.

II.2.1 Loading data

First you need to load the MARC XML collection into the internal database using the `-l` switch. It will be stored in a virtual collection that can be used later. Let's call it `articles`. MarcXimiL will want a strategy parameter in order to properly parse the records and extract the desired fields. Let's use `similarity_config__ubiquist2_sxic.py`.

```
similarity.py -l articles -f 2k_a.xml -s
similarity_config__ubiquist2_sxic
```

You can list the collections that are currently stored in the internal database:
`similarity.py -i` # list indexes in `var/indexed_colls.list`

You can also remove the collection from the internal database if needed:
`similarity.py -k "coll:articles"` # kill collection `articles`

It is also possible to add records (here the `smallcoll.xml` file) into an existing internal collection. This operation used to be unsafe (especially with the ic/luhn strategies) but it should be OK as of version 0.3.6.

```
similarity.py -u articles -f smallcoll.xml -s
similarity_config__ubiquist2
```

II.2.2 Executing the comparisons

The `-c` switch executes the chosen comparison strategy on one or two internal collections. In general, the same strategy will be used as for the loading step. However, it is not mandatory: the only requirement is that the internal name of the fields (as given in the `record_structure` section) must be the same.

```
similarity.py -c articles -o results1 -s
similarity_config__ubiquist2_sxic      # compare (one collection with itself)

similarity.py -c "articles preprints" -o out.dat -s
similarity_config__ubiquist2_sxic      # compare (two collections)
```

The output (as specified by the `-o` switch) can be a file or a table in the internal database, depending on the output parameters in the strategy file.

II.2.3 Results

If the output was sent to a file, just look at it. Otherwise, you can extract the results stored in the internal database:

```
similarity.py -w results1 -o res.txt  -s
similarity_config__ubiquist2_sxic    # write tabulated output in var/log/res.txt
```

You can list the results set in your database:

```
similarity.py -j                      # list results sets in var/result_sets.list
```

You can also delete one set of results or merge several sets into a new one:

```
similarity.py -k "res:results1"                     # kill result_set results1

similarity.py -m "results1 results2" -o results3    # merges result sets
```