Developer Overview of SynthaCorpus

Microsoft

July 17, 2017

The SynthaCorpus project is written in C11 and perl 5. Development has been carried out under cygwin on Windows 10, using gcc version 5.4.0 and perl 5.22. The C executables can also be built under Visual Studio 2015 and solution/project files are included in the distribution to facilitate this. Successful building and basic operation under Bash on Windows-10 has been confirmed though more thorough testing is needed. I don't expect major problems porting to other environments. However, because large data sizes lead to heavy memory demands, there is a strong assumption that the executables will be built for and run on 64-bit architecture.

In the following it is assumed that the root of the project directory structure is at .../SynthaCorpus. All references to directory paths are relative to that.

1 Licenses, Authorship and Copyright

The bulk of the code in this project was written by Microsoft, with David Hawking as principal author and contributions from Bodo Billerbeck. That code is copyright Microsoft and is distributed under MIT licence. See LICENSE.TXT.

Third-party open-source code and content is also included as follows:

Project Gutenberg books - the directory ProjectGutenberg contains 49 popular books in text format, downloaded from http://www.gutenberg.org/. These are books which are no longer subject to copyright. They are provided as a small test set to illustrate and verify the operation of project SynthaCorpus.

Fowler Noll Vo hash function - this is a hashing function which we have consistently found to be fast and effective. The code is at src/imported/Fowler-Noll-Vo-hash. Both fnv.c and fnv.h in that directory include a licence allowing free distribution. See http://www.isthe.com/chongo/tech/comp/fnv/.

Tiny Mersenne Twister random number generator - this is a fast random number generator with very good properties. We chose to use the tiny version (still with excellent properties), to allow the possibility of corpus generation being built into an indexer being studied. The code is at src/imported/TinyMT_cutdown. The file readme.html contains the license. See http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/TINYMT/index.html

2 Project dependencies

Successful building and operation of the SynthaCorpus project requires installation of the following packages. Version numbers shown are the ones used. Other versions may be suitable.

- gcc 5.4.0 and make 4.2.1 OR VisualStudio 2015
- perl 5.22
- gnuplot 5.0 – for displaying extracted corpus properties

- `pdflatex 2.9.4902` – for compiling documentation (if needed)
- `acroread 2017 version` OR other PDF viewer – for viewing documentation and plots.

3 Building executables and documents

3.1 With gcc

```
cd src
make cleanest
make
```

builds all the executables. One warning is emitted, complaining about a function which is not referenced but which may be in future. Currently, the executables remain in `src`.

3.2 With VS 2015

Solution files are stored in sub-directories of `src`. E.g. in `src/corpusGenerator`

Properties are only set for the `Release`, `x64` combination. Changes to defaults for this combination include:

- Project/General: Use multi-byte characters
- C/C++/Preprocessor: Define symbols `WIN64` and `_CRT_SECURE_NO_WARNINGS`
- C/C++/Precompiled headers: Not using precompiled headers
- C/C++/Advanced: Compile as C

Open the solution file with VS 2015 and click Build/Rebuild.

The executable will be found in e.g. `src/corpusGenerator/x64/Releases/corpusGenerator.exe`

3.3 Building PDF documents

```
cd doc
rm *.pdf
make
```

4 Notes on coding

Project `SynthaCorpus` has been developed in the course of scientific research rather than engineered to commercial standards. In many cases, but not always, there is quite extensive internal documentation at the head of modules or critical functions. Unfortunately, there is inconsistency in variable naming. The process of moving from the use_of_underscores to "camelCase" is regrettably incomplete.

Be warned that the principal author is highly enamoured of the `unless` clause in perl. Most scripts contain many examples of:

```
die "message"
   unless <condition>;
```

The C programs make very extensive use of "dahash" hash tables, defined in `src/utils/dahash.[ch]`. These hash tables avoid the use of buckets. Their size is constrained to be a power of two and collisions are handled by relatively prime rehash. Once a table reaches a specified percentage of capacity, the table size is doubled behind the scenes. The hash function is the Fowler Noll Vo one.

Also defined is a mechanism for dynamic arrays – `src/utils/dynamicArrays.[ch]`. Accessing an array element beyond allocated storage, causes the array to grow using a configurable strategy.

C programs make extensive use of memory mapping through wrappers in `src/utils/general.[ch]` which use either the Windows style `CreateFileMapping()/MapViewOfFile()` (used if the WIN64 symbol is defined) or the Unix interface `mmap()`.

All programs assume the use of UTF-8. We recommend the use of the iconv library `https://www.gnu.org/software/libiconv/` for converting other character sets into UTF-8. Functions for dealing with UTF-8 are in `src/characterSetHandling/unicode.[ch]`. Also in that directory is a perl script for updating Unicode tables from the database at `http://unicode.org/`.

5 Layout of Project Directories

The project root directory contains the following files and directories:

`doc` - Contains the LaTeX source and PDFs of project documentation.

`Experiments` - This is the directory under which all the output produced by scripts and executables will be stored. If it doesn't exist the main perl scripts will create it.

`LICENCE.TXT` - See beginning of this document.

`ProjectGutenberg` - Text versions of 49 books used to illustrate and validate the executables in this project.

`src` - All the perl and C sources. Currently executables are built in or under this directory.

6 Desirable future developments

1. Code tidyup - variable names, internal documentation, module structure. [Boring but useful]
2. Additional document formats - or maybe converters from other formats into STARC format? [Boring but useful]
3. Piecewise growth models - Currently scaling up from a small corpus assumes the use of only a simple linear model for word frequency distribution. [A challenge and very useful contribution.]
4. Extending the supported term dependency models to include term repetition within a document, and term co-occurrence as well as n-grams. [A challenge and very useful contribution.]
5. Efficiency monitoring and improvement, particularly reducing memory demands without exploding runtime. [Could increase scope of application.]
6. Reduce memory demands of `corpusGenerator.exe` by using more memory-efficient representations of Markov transition matrices. [Could increase scope of application.]

7 Using `SynthaCorpus`

Please see the companion User Overview document.