Covarep

A Cooperative Voice Analysis Repository for Speech Technologies

Gilles Degottex[1], John Kane[2], Thomas Drugman[3], Tuomo Raitio[4], Stefan Scherer[5]

April 23, 2014

1 Aims of the project

Over the past few decades a vast array of advanced speech processing algorithms have been developed, often offering significant improvements over the existing state-of-the-art. Such algorithms can have a reasonably high degree of complexity and, hence, can be difficult to accurately re-implement based on article descriptions. Another issue is the so-called 'bug magnet effect' with re-implementations frequently having significant differences from the original ones. The consequence of all this has been that many promising developments have been under-exploited or discarded, with researchers tending to stick to conventional analysis methods.

By developing Covarep we are hoping to address this by encouraging authors to include original implementations of their algorithms, thus resulting in a single de facto version for the speech community to refer to. We envisage a range of benefits to the repository:

1) Reproducible research: Covarep will allow fairer comparison of algorithms in published articles.

2) Encouraged usage: the free availability of these algorithms will encourage researchers from a wide range of speech-related disciplines to exploit them for their own applications.

3) Feedback: as a GitHub project users will be able to offer comments on algorithms, report bugs, suggest improvements etc.

[1]University of Crete, Heraklion, Greece.
[2]Trinity College Dublin, Dublin, Ireland.
[3]University of Mons, Mons, Belgium.
[4]Aalto University, Espoo, Finland.
[5]University of Southern California, Los Angeles, USA.

2 License

Getting contributing institutions to agree to a homogenous Intellectual Property (IP) policy would be close to impossible. As a result Covarep is a repository and not a toolbox, and each algorithm will have its own licence associated with it. Though flexible to different licence types, contributions will need to have a licence which is compatible (or *linkable*) with the General Public License Version 3 (GPLv3) (see diagram at `http://en.wikipedia.org/wiki/GPL#Compatibility_and_multi-licensing`) (e.g. LPGL, MIT, Apache or similar). The LGPL licence is advised in order to be more industry friendly.

3 Contribution

3.1 Scope

Contributions are welcome from a wide range of speech processing areas, including (but not limited to): Speech analysis, synthesis, conversion, transformation, enhancement, glottal source/voice quality analysis, etc.

It can be novel methods as well as existing methods.

3.2 Requirements

In order to achieve a reasonable standard of consistency and homogeneity across algorithms, the following elements are required:

- Only published work can be added to the repository.
- The code must be available as open source.
- Algorithms should be coded in Matlab, however we strongly encourage authors to make the code compatible with Octave in order to maximize usability.
- Contributions have to comply with a Coding Convention (see in the directory documentation/CodingConvention.txt and documentation/CodingConvention_FunctionTemplate.txt). However, only for normalizing the inputs/outputs and the documentation. There is no restriction for the content of the functions (though, comments are obviously encouraged).
- Each method exist only once in the whole repository. Thus, redundancy has to be minimized between the methods existing in the repository and the submission.

Note that the original author of a method (the author of the article) does not necessarily correspond to the author of the implementation! Anybody is free to send his/her implementation of a published method which does not exist yet in Covarep.

3.3 Submission process

The instructions for submitting contributions are given at: `http://covarep.github.io/covarep`. In order to avoid confusion, there is no other source of information about the submission process.

3.4 Improvements of methods

We welcome improvements of methods already existing in Covarep (e.g. replacement of the traditional computation of the Linear Prediction residual by a more robust one which can be exploited in many functions in Covarep). However, the default behavior of a function has to correspond to the original article description. Thus, improvements can be added in existing functions, but they have to appear as a non-default option (please see `sinusoidal/sin_analysis.m` for a complete example of option management).

4 Coding

The code as to be fast enough to be run on big corpora of speech utterances. However, incomprehensible implementations that save only a negligible CPU time, compared to a clear and pedagogic code, should be avoided.

4.1 Hosting

The Covarep project is currently hosted on GitHub (`https://github.com/covarep/covarep`). Please have a look at the help of the GitHub website in order to rise an issue or discuss any point.

4.2 Dependencies

- In order to minimize redundancy, one method exists only once in the whole repository. Issues can be discussed on the GitHub project if the implementation of an existing method is questionable.
- There is no requirement for the Matlab or Octave version.
- Covarep is dependent on the *Voicebox* Matlab toolbox (see `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`)
- The code has to run on Windows, Mac and Linux platforms.

4.3 Testing and bug tracking

A HOWTO example file is necessary to exemplify each method. Basic regression tests can be also submitted or developed by the maintainers. A basic function should allow the user to run regression tests to verify that all the methods implemented in Covarep are working properly on his/her machine.

4.4 Code maintenance

4.4.1 Versioning of the project

If the revision number changes (x.x.X), it means that only bugs have been fixed. If the minor number changes (x.X.x), it means that bugs have been fixed and new features have been added, without breaking compatibility with former versions of identical major number. If the major number changes (X.x.x), it means that new features appeared and the compatibility with former version is not ensured.

4.4.2 Project merge and Branching policy

The Branch-When-Needed system: maintainers commit their day-to-day work on the master branch.

> **Rule 1** The master must compile and pass regression tests at all times. Committers who violate this rule are publically humiliated.
>
> **Rule 2** A single commit (changeset) must not be so large so as to discourage peer-review.
>
> **Rule 3** If rules 1 and 2 come into conflict (i.e. it's impossible to make a series of small commits without disrupting the trunk), then the user should create a branch and commit a series of smaller changesets there. This allows peer-review without disrupting the stability of the master branch.

4.4.3 Release instructions

- Import the novel functionalities, bug fixes, etc.
- Compile the Covarep.pdf document if Covarep.tex changed. And commit the new Covarep.pdf.
- Change the version in the README.txt file to the final version number (e.g. 1.0.2). This commit has to be the very last modification of the repository before tagging the master with the release tag.
- Create an annotated tag (Can use the GitHub interface for this purpose). Follow descriptions of previous releases.
- List novelties in the Release's description. (can use "git log –pretty=oneline tag1...tag2" in order to list the commits made between the previous release and the new one.
- Change the version in the README.txt file to "master (after ¡current version¿)" (e.g. "master (after 1.0.2)")