

# New features, testing and refactoring joshua

Gideon Maillette de Buy Wenniger  
Project lead by: Matt Post

gemdbw AT gmail.com  
<http://staff.science.uva.nl/~gemaille/>

Statistical Language Processing and Learning Lab  
Institute for Logic Language and Computation  
University of Amsterdam, the Netherlands



September 14, 2013

# Food for discussion

*“Don't leave "broken windows" (bad designs, wrong decisions, or poor code) unrepaired. Fix each one as soon as it is discovered. If there is insufficient time to fix it properly, then board it up. Perhaps you can comment out the offending code, or display a "Not Implemented" message, or substitute dummy data instead. Take some action to prevent further damage and to show that you're on top of the situation. ” – Andy Hunt and Dave Thomas*

*(From 'The Pragmatic Programmer')*

# New features

- Project start : Matt is implementing Lattice (Batch) Mira in Joshua
- Showed were to implement new features in Joshua
- Went ahead and implemented one
- Simple binary feature firing for specific combination of labels

Rule:

[**:+NP+CC**] ||| [[:1] পূর্ব [NNS,2] ও ||| [[:1] eastern [NNS,2] and

Firing binary feature:

=> labelCombinationFeature\_**:+NP+CC**\_:\_**NNS**

# New features

- Hey, this is easy, let's implement a whole lot of features!



- Eeuh yes, very good, but does it actually work?

# Feature Functions Test : Motivation

- Feature function applied? Total weight must change!
- Manual testing many features = tedious + buggy
- Solution : automated testing

# Feature Functions Test : Tools and approach

- JUnit
- Rerouting input/output
- Run1: Extract list new features + weights decoder output
- Re-run2: New features specified in config
- Test check : weight pairs have different weights over both runs

# Feature Functions Test

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with a test class `Joshua decoder FeatureFunctionsTest`.
- JUnit Runner:** Shows the test execution results: "Finished after 8.272 seconds", "Runs: 1/1", "Errors: 0", "Failures: 0".
- Code Editor:** Displays the source code for `FeatureFunctionsTest`. The code includes a `testFeatureFunctions` method that:
  - Prints the working directory.
  - Creates a basic test configuration file.
  - Runs the decoder with the basic configuration to produce `decoderOutput1`.
  - Writes a new configuration file based on the first run's extra features.
  - Re-runs the decoder with the new configuration to produce `decoderOutput2`.
  - Asserts that both runs produce the same number of total weights and that the decoder weights are unequal.
- Console:** Shows the output of the test, including feature labels and a list of 10 `NBestList` entries with their weights for two runs.

```
public void testFeatureFunctions(String featureName) {
    System.out.println("Working directory : " + FileUtility.getWorkingDirectory());
    createTestFilesBasicTest(featureName);
    // First run the decoder without extra features weights specified:
    // they fire but should have no effect on the total weight
    DecoderOutput decoderOutput1 = runDecoder(JOSHUA_CONFIG_FILE_NAME);

    // write the new configuration file based on the list of extra features found in the first run
    writeJoshuaExtraFeaturesConfigFile(featureName, decoderOutput1.getExtraFeaturesList());
    // Re-run the experiment, using the new configuration file with weights for the extra features
    //JoshuaConfiguration.reset();
    StatefulFF.resetGlobalStateIndex();
    DecoderOutput decoderOutput2 = runDecoder(JOSHUA_EXTRA_FEATURES_CONFIG_FILE_NAME);

    assertEquals("Both decoder runs produce same number of total weights", decoderOutput1, decoderOutput2);
    assertTrue("Both decoder runs produce unequal decoder weights", decoderOutput1, decoderOutput2);
}
```

```
<terminated> FeatureFunctionsTest [JUnit] /usr/lib/jvm/jdk1.7.0/bin/java (Sep 13, 2013 10:42:14 PM)
feature: labelcombinationfeature_100V
feature: labelcombinationfeature_1PP+VBZ
feature: labelcombinationfeature_1PP
feature: labelcombinationfeature_1VBD+VBN_1_VBN
feature: labelcombinationfeature_1_VBN
NBestList entry: 0 weight run1: -23.334 weight run2: -19.534
NBestList entry: 1 weight run1: -23.562 weight run2: -19.782
NBestList entry: 2 weight run1: -24.258 weight run2: -20.458
NBestList entry: 3 weight run1: -24.642 weight run2: -20.842
NBestList entry: 4 weight run1: -24.719 weight run2: -20.919
NBestList entry: 5 weight run1: -25.138 weight run2: -21.338
NBestList entry: 6 weight run1: -25.262 weight run2: -21.462
NBestList entry: 7 weight run1: -25.387 weight run2: -21.587
NBestList entry: 8 weight run1: -25.441 weight run2: -21.641
NBestList entry: 9 weight run1: -26.274 weight run2: -22.474
```

# Challenges encountered along the way

- Globally shared configuration object: problems multiple consecutive decoder runs
- First attempt: reset methods
- But : how does user know/expect reset method must be called, and where?
  - ⇒ fix in a lame way
- In fact: empirical evidence global variables are evil



# Refactoring to the rescue



# Refactoring to the rescue

The screenshot shows a GitHub pull request interface. At the top, the repository is identified as 'gweniger / joshua', forked from 'joshua-decoder/joshua'. The pull request title is 'Gideon : Refactored the JoshuaConfiguration. Made it into a non-stati...'. The description explains that the refactoring changes the configuration from a global static class to a non-static class, which is passed as an object to other classes. This change is intended to prevent unexpected behavior when multiple instances of Joshua are run in the same VM or multiple instances are created within a single VM. A TODO note mentions that 'StatefullFF' still has global state and should also be refactored. The pull request is authored by Gideon Wenniger 11 hours ago. Below the description, a diff view for the file 'src/joshua/decoder/ArgsParser.java' is shown, highlighting changes to the class definition and its constructor.

gweniger / joshua  
forked from joshua-decoder/joshua

Unwatch 1 Star 0 For

**Gideon : Refactored the JoshuaConfiguration. Made it into a non-stati...** [Browse code](#)

..c class, that is

passed as an object to the classes that need it rather than being accessed as a global object, which leads to all kinds of unexpected behavior when running multiple instances of Joshua within the same VM or alternatively multiple instances after each other within a java test (e.g. executing the decoder twice). Generally it can be argued that global state is almost always suboptimal, unless there are very convincing arguments to adopt it - rather than just convenience - which seems to be not the case here.

TODO: StatefullFF, amongst other classes still has global state. This should also be refactored to make the whole enterprise completely without requiring reset methods.

master

Gideon Wenniger authored 11 hours ago 1 parent 6c45fab commit 8621380e1af3d9af674fae9b29a4ba7b423d94b

Showing 31 changed files with 377 additions and 332 deletions. [Show Diff Stats](#)

src/joshua/decoder/ArgsParser.java [View file @ 8621380](#)

```
... .. @@ -7,7 +7,7 @@
7 7 *
8 8 */
9 9 public class ArgsParser {
10 -
10 +
11 11     private String configFile = null;
12 12     private String testFile = "-";
13 13
... .. @@ -17,8 +17,9 @@
17 17 *
18 18     * @param args
19 19     */
20 - public ArgsParser(String[] args) {
```

# Conclusions

- Started implementation new sparse features
- Generic test shows: do features actually fire and change weight?
- Big refactoring enables more effective testing with eclipse
- Foundation smooth implementation more features
- Combination with Lattice (Batch) Mira : awesomeness!