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
“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 where to implement new features in Joshua
- Went ahead and implemented one
- Simple binary feature firing for specific combination of labels

Rule:

\[[+:NP+CC] || [:,1] \text{পূর্ব} [NNS,2] \text{ and } || [:,1] \text{eastern} [NNS,2]\]

Firing binary feature:

\=> \text{labelCombinationFeature}:+NP+CC_:NNS
New features

• Hey, this is easy, let’s implement a whole lot of features!

• Eeuuh 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

Gideon Maillette de Buy Wenniger and Matt Post

New features, testing and refactoring Joshua
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

I DON'T ALWAYS REFACTOR CODE

BUT WHEN I DO, I MAKE SURE TO BREAK EVERYTHING
Refactoring to the rescue

Gideon Maillette de Buy Wenniger and Matt Post

New features, testing and refactoring Joshua 10/11

Gideon: Refactored the JoshuaConfiguration. Made it into a non-static...

...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.

1 master

Gideon Wenniger authored 11 hours ago

1 parent 6c45fab commit 8621380 src/joshua/decoder/ArgsParser.java View file @ 8621380

Showing 31 changed files with 377 additions and 332 deletions

src/joshua/decoder/ArgsParser.java

... ...
7 7 *
8 8 */
9 9 public class ArgsParser {
10 -
10 +
11 11 private String configFile = null;
12 12 private String testFile = "-";
13 13 ...
16 16 *
17 17 */
18 18 */
19 19 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!