LoonyBin: Making Empirical MT Reproducible, Efficient, and Less Annoying

Jonathan Clark
Alon Lavie
CMU Machine Translation Lunch
Tuesday, January 19, 2009
Outline

• A (Brief) Guilt Trip
• What goes into a LoonyBin workflow
• What goes on in a LoonyBin workflow
• What comes out of a LoonyBin workflow
• The MT Toolpack for LoonyBin
The Guilt Trip

• What does a scientifically responsible experiment look like?
  • Small $\Delta$’s, Reproducible, Detailed Analysis/Logs...

• What do most MT experiments look like?
What MT Research Pipelines Look like in Papers
Actual MT Workflows

**Step 2: EC Format**

Convert symmetric alignments to the elicitation corpus format. Make sure the source and target are correct! For the rule learner's t2ts format, I think the source is supposed to be the strong (i.e. English) parse tree.

```
/barrow/usr2/vamshi/bin/pharoh2ec.pl <tokenized-source> <tokenized-target> <aligned.grow-diag-final> > <ec-file>
```

**Step 3: Config File**

Set up a config file for Vamshi's rule learning system. Remember that the source side is the strong (i.e. English) side. Set `GRA_FORMAT=ONELINE`. If the lexical entries don't matter and you only want the grammar rules, set `PTABLE_FILE=/dev/null` and `LEXICON_FILE=/dev/null`.

**Step 4: Extract Rules**

Run Vamshi's rule learning.

```
sh /barrow/usr2/vamshi/bin/ruleinduction.sh <config-file>
```
if ($_HELP) {
    print "Train Phrase Model"
}

Steps: (--first-step to --last-step)
(1) prepare corpus
(2) run GIZA
(3) align words
(4) learn lexical translation
(5) extract phrases
(6) score phrases
(7) learn reordering model
(8) learn generation model
(9) create decoder config file

For more, please check manual or contact koehn@inf.ed.ac.uk

exit(1);

my $___FACTOR_DELIMITER = $_FACTOR_DELIMITER;
$___FACTOR_DELIMITER = '|' unless ($_FACTOR_DELIMITER);

print STDERR "Using SCRIPTS_ROOTDIR: $SCRIPTS_ROOTDIR"

# supporting binaries from other packages
my $GIZA = "$BINDIR/GIZA++";
my $SNT2COOC = "$BINDIR/snt2cooc.out";
Issues

- Automation
- Reproducibility
- Variability
- Scripting Bugs
- Multiple machines, clusters, and schedulers
- Hard to see Big Picture
- “But that’s not research”
What goes into LoonyBin
Going in

- Knowledge from self 6 months ago
- Knowledge from predecessors about removing the 300-character underscore out of the corpus after 8 years
- Visual representation of input/output files and parameters as a DAG + integrated documentation for each tool
- ALL parameters, etc. are specified in LoonyBin
- Mapping from vertices (tool instances) to machine instances
Obligatory Screenshot

Drag and Drop

Available Tools

Tooltips
Obligatory Screenshot
HyperWorkflows

• **HyperWorkflows**: Shared substructure in experiments

• Encode small variations in a HyperDAG
• **HyperWorkflows**: Shared substructure in experiments

• Encode small variations in a HyperDAG
HyperWorkflows

- **HyperWorkflows**: Shared substructure in experiments
- Encode small variations in a HyperDAG
What goes on and What comes out
What happens where?

- **Design Machine**: Java
  - Manually Copy Bash Script
  - Remote Execution Machine: UNIX
  - Passwordless SSH

- **Home Execution Machine**: UNIX
  - Remote Execution Machine: UNIX
  - Passwordless SSH
Push-button MT for the Masses

- A bash script is generated and copied over
- Check if files and tools are present first
- Sanity checking at each step
- Copying of files (including to HDFS)
- Automatic login to remote machines (via passwordless SSH)
- Scheduler wrappers (e.g. Torque/Condor/SGE)
What comes out

• Artifacts of the workflow in an organized directory structure

• Log with detailed information about data (corpus, alignments, parses, etc.) after pipeline step
  • Simple text format
  • Complete history in each file

• Email/SMS notifications of completion/failure
Example Log Output

5000-tune.it1.AvgWeight.pt_wordcount    -1.76
5000-tune.it1.ExampleTopbestHyp.1 oslo 6-2 -lrb- afp -rrb- - terje roed...
5000-tune.it2.hypotheses.Total    712902
5000-tune.it2.hypotheses.PerSentence    396
5000-tune.it2.hypotheses.AddedTotal    272703
5000-tune.it2.hypotheses.AddedPerSentence    151
5000-tune.it2.Weight.lm    1.55
4250-prune-pt-default.MachineName    gritgw1005.yahooresearchcluster.com
4250-prune-pt-default.TimeElapsed    0:17:17
4250-prune-pt-default.COUNT.Phrase_Records_Read    14561086
4250-prune-pt-default.COUNT.Source_Sides_After_Pruning    176529
4250-prune-pt-default.FileSystemCounters.FILE_BYTES_READ    308358509
LoonyBin Best Practices

- **Lots** of steps. Why?
- Continue on failures
- Interchange components easily
- Record effect of each component on data
MT Toolpack for LoonyBin

- Will be released MT Marathon next week

- Joshua training pipeline including Berkeley aligner and recasing (Jonny and Byung @ JHU)

- Moses training pipeline

- MGIZA/Chaksi (Qin)

- SAMT (Andreas)

- Multi-Metrics -- BLEU/NIST/Meteor/TER (Kenneth)

- LM training via SRILM

- MEMT (Kenneth)
Adding your own tools

class Lowercase(Tool):

    def getName(self):
        return 'Machine Translation/Output/Unescape English'

    def getDescription(self):
        return "Unescapes -lrb- -rrb- " ` ` and splits hyphens"

    def getRequiredPaths(self):
        return ['gale-p4-scripts']

    def getParamNames(self):
        return []

    def getInputNames(self, params):
        return [ ('eFileIn', 'file containing English data') ]

    def getOutputNames(self, params):
        return [ ('eFileOut', 'unescape file with English data') ]

    def getCommands(self, params, inputs, outputs):
        return [ ('unescapeEnglish.py < %eFileIn%s%inputs + %>' > %eFileOut%s%outputs) ]

    def getPostAnalyzers(self, params, inputs, outputs):
        return [ ]
Conclusion

• Make your life easier
• Be a more responsible scientist
Questions?
Post-Mortem: Directory Structure

- (picture of directory structure)
Post-Mortem: Future Work

• Default parameters -- Next few months
• Asynchronous DAG execution (currently all steps are run in serial) -- Next year
• Workflow monitoring and modification during execution -- Next several years
Post-Mortem: Recommendations

- Store your workflow files in SVN
- Store your log files in SVN -- experimental data is useful long after we get annoyed with size of data files!
- Log the SVN revision of frequently changing tools in your Loon logs -- Build them from SVN every time to ensure you’re executing that version