The Machine Translation Toolpack for LoonyBin

Machine Translation and HyperWorkflows

Jonathan Clark, Jonathan Weese, Byung Gyu Ahn, Andreas Zollmann, Qin Gao, Kenneth Heafield, Alon Lavie

4th Machine Translation Marathon
Monday, January 25, 2009
Outline

- A (Brief) Guilt Trip
- LoonyBin
  - What goes in
  - What goes on
  - What comes out
- The MT Toolpack for LoonyBin
- A Few Recommendations
The Guilt Trip

• What does an ideal experiment look like?
  • Small Δ’s, Reproducible, Detailed Analysis/Logs...

• What do most MT experiments look like?
  • Convoluted scripts, lazy evaluation of data analysis...
MT Workflows in Papers
Actual MT Workflows

if ($_HELP) {
    print "Train Phrase Model"
    exit(1);
}

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

print STDERR "Using SCRIPTS_ROOTDIR: $SCRIPTS_ROOTDIR\n";

# 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
What goes into LoonyBin
Going in

- Knowledge from self 6 months ago
- Knowledge from predecessor 8 years ago about removing the 300-character underscore out of the corpus
- Visual representation of input/output files and parameters as a DAG
• **HyperWorkflows:** Shared substructure in experiments

• Encode small variations in a HyperDAG

---

Diagram:

- **Parallel Corpus** → **Filter Corpus** → **Word Alignment** → **Stanford Parser** → **Charniak Parser** → **Build Syntactic Translation Model** → **Moses Phrase Table Training** → **Moses**

- **Target Language Corpus** → **Build Language Model** → **Minimum Error Rate Training** → **Decide Sentences**

- **syntax** → **{syntax-st, syntax-ch, moses}**
**HyperWorkflows**

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

• Encode small variations in a HyperDAG
What goes on
and
What comes out
What happens where?

Design Machine
Java

Home Execution Machine
UNIX

Manually Copy Bash Script

Remote Execution Machine
UNIX

Passwordless SSH

Remote Execution Machine
UNIX

Passwordless SSH
What goes on

- 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000-tune.it1.AvgWeight.pt_wordcount</td>
<td>-1.76</td>
</tr>
<tr>
<td>5000-tune.it1.ExampleTopbestHyp.1 oslo 6-2 -lrb- afp -rrb- - terje roed...</td>
<td></td>
</tr>
<tr>
<td>5000-tune.it2.hypotheses.Total</td>
<td>712902</td>
</tr>
<tr>
<td>5000-tune.it2.hypotheses.PerSentence</td>
<td>396</td>
</tr>
<tr>
<td>5000-tune.it2.hypotheses.AddedTotal</td>
<td>272703</td>
</tr>
<tr>
<td>5000-tune.it2.hypotheses.AddedPerSentence</td>
<td>151</td>
</tr>
<tr>
<td>5000-tune.it2.Weight.lm</td>
<td>1.55</td>
</tr>
<tr>
<td>4250-prune-pt-default.MachineName</td>
<td>gritgw1005.yahooresearchcluster.com</td>
</tr>
<tr>
<td>4250-prune-pt-default.TimeElapsed</td>
<td>0:17:17</td>
</tr>
<tr>
<td>4250-prune-pt-default.COUNT.Phrase_Records_Read</td>
<td>14561086</td>
</tr>
<tr>
<td>4250-prune-pt-default.COUNT.Source_Sides_After_Pruning</td>
<td>176529</td>
</tr>
<tr>
<td>4250-prune-pt-default.FileSystemCounters.FILE_BYTES_READ</td>
<td>308358509</td>
</tr>
</tbody>
</table>
MT Toolpack for LoonyBin

- Includes
  - 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 (Please)

- Just implement a python interface
  - Inputs
  - Outputs
  - Parameters
- How to form UNIX command
- Analyzers (optional, but recommended)
Future Work

• Default parameters -- Short-term
• Asynchronous DAG execution (currently all steps are run in serial) -- Mid-Term
• Workflow monitoring and reprioritization during execution -- Long-term
• Encapsulation of workflows as “tools” (hierarchical tools) -- Long-term
• Automatic file compression -- Long-term
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
LoonyBin Best Practices

- Lots of steps. Why?
  - Continue on failures
  - Interchange components easily
  - Record effect of each component on data
- Workflows can have many granularities!
Conclusion

- Make your life easier
- Make our lives easier
- Be a more responsible scientist
Questions?

loonybin.sourceforge.net