Cluster Computing for Statistical Machine Translation

Qin Gao, Kevin Gimpel, Alok Palikar, Andreas Zollmann
Stephan Vogel, Noah Smith
Language Technologies Institute
Carnegie Mellon University

Supported by NSF Cluster Exploration (CluE)
Outline

- Introduction
  - Statistical Machine Translation
  - How can parallel processing help MT?
  - Roadmap
  - Progress
  - Ongoing work
Hello John ...
The United State government is going to...
Australia is one of a few countries that has relationship with North Korea.
Australia is North Korean has relationship a few...
Sarah Palin's neighbors saluted their hometown girl with Alaskan Amber.
The idea, especially from the Democrats that I know.

.....

Which one to choose?
Components of Statistical Machine Translation

Source Language

Interlingua

Transfer

Analysis

Generation

Target Language
Components of Statistical Machine Translation

Analysis → Transfer → Generation

Interlingua

Source Language

Target Language

Semantic (?)

Syntactic Tree-to-Tree

Syntactic Tree-to-String

String-to-String

Analysis

Generation
Components of Statistical Machine Translation

Tasks for Analysis:
- Text normalization
- Tokenization
- Syntactic Parsing
- Dependency Parsing
- Semantic Parsing

Source Language

Interlingua

Target Language

Text Normalizing

Semantic Parsing

Syntactic Parsing

String-to-String

Syntactic Tree-to-Tree

Semantic (?)

?”
Components of Statistical Machine Translation

Tasks for Transfer:
• Phrase extraction
• Rule extraction
• Feature value estimation

Source Language  Interlingua (?)  Target Language

Analysis  Generation

Phrase extraction  Rule extraction
Components of Statistical Machine Translation

Tasks for Generation:
- Target language modeling
- Minimal Error Rate Training
- Decoding

Source Language

Interlingua (?)

Target Language
**Tasks for Analysis:**
- Text normalization
- Tokenization
- Syntactic Parsing
- Dependency Parsing
- Semantic Parsing

**Tasks for Transfer:**
- Phrase extraction
- Generation rule extraction
- Feature value estimation

**Tasks for Generation:**
- Target language modeling
- Minimal Error Rate Training
- Decoding

---

**Data Driven**

- **Bilingual Data:** ~300 Million Words
- **Phrase Table/Rule Database:** ~500 Million Entries
- **Language Model:** ~700 Million n-grams
- **Monolingual Data:** More than 4.5 Billion Words
INCA: An Integrated Cluster Computing Architecture for Machine Translation

With the help of computer clusters, we want to make the statistical machine translation tasks:

**Faster**
- Parallelize time-consuming components
- Automate the procedure

**Easier**
- Make the experiment management easier
- Make the substitution of components more convenient

Moses is an excellent example how such tool can benefit the research community!
Target: Integrated Cluster Computing Architecture for MT

Provide Cluster-enabled Components

Simplify experiment process with integrated framework

Experiment Management: Reproduce the results
Progress

- POS tagging
- Parsing
- Semantic Role labeling

Data Pre-processing

Word Alignment

- Parallel GIZA++ (To be ported to Hadoop)

Tuning and Decoding

Phrase/Rule Extraction

- Distributed Phrase extraction (Chaski)
- Rule extraction for SAMT

- Distributed Tuning/Decoding framework (Trambo)
Data Preprocessing

- **Examples:**
  - Tokenization
  - Word segmentation (e.g. for Chinese)
  - POS tagging
  - Parsing
  - Semantic role labeling

- Naturally parallelizable, perfect for MapReduce
- However, there are still interesting problems
Error Tolerance?

- Hadoop has the functionality to re-run failed jobs.
  - It works fine for hardware errors or “unpredictable” software errors
  - But it does not work for predictable errors
- E.g: A certain sentence may crash the parser, restart the parser does not help
  - Fail-safe mechanism: Falling back to simple models
  - Currently implemented through individual scripts, aiming at providing the frame work
Parallelized Word Alignment

Algorithm of word alignment: **EM** (Iterative Optimization)
Parallel GIZA++
# Performance of PGIZA++

Comparison of Speed and BLEU score using PGIZA++ and GIZA++

<table>
<thead>
<tr>
<th></th>
<th>Running Time</th>
<th>BLEU Score (Tuning)</th>
<th>BLEU Score (Testing)</th>
<th>CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIZA++</td>
<td>169h</td>
<td>32.34</td>
<td>29.43</td>
<td>2</td>
</tr>
<tr>
<td>PGIZA++</td>
<td>39h</td>
<td>32.20</td>
<td>30.14</td>
<td>11</td>
</tr>
</tbody>
</table>

Normalization time in PGIZA++

<table>
<thead>
<tr>
<th></th>
<th>Per-Iteration</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>47.0 min</td>
<td>235 min (3.9 h)</td>
</tr>
<tr>
<td>HMM</td>
<td>31.8 min</td>
<td>159 min (2.6 h)</td>
</tr>
<tr>
<td>Model 3/4</td>
<td>25.2 min</td>
<td>151 min (2.5 h)</td>
</tr>
</tbody>
</table>

Corpus Size: 10 million sentence pairs, Chinese-English
I/O bottleneck

- Currently the tool run on NFS
- I/O of counts becomes the major bottleneck
- The plan is to port it to Hadoop
- The locality of HDFS should help
Ongoing Work: Distributed Asynchronous Online EM

- Master node runs MapReduce jobs to perform E-step on mini-batches of data
- When each mini-batch completes, master node updates parameters, starts a new mini-batch immediately
- E-steps performed using slightly stale parameters → “asynchronous”
Phrase Extraction Tool Chaski

- Chaski
  - Postmen of Inca Empire
  - They run really fast!
Phrase Model Training

- Problems in phrase extraction:
  - Disk space
    - 700GB when extracting phrase up to length 10, on 10 million sentences
  - Sorting phrases
    - External sorting must be used, which adds up to the disk usage
Phrase Model Training (Moses)

1. **Bidirectional Word Alignment**
   - Extract Phrases
   - Phrase Pairs Source → Target
   - Phrase Pairs Target → Source
   - Sort on Source
   - Sort on Target
   - Phrase Translation Feature and Lexicon Translation Feature Source → Target
   - Phrase Translation Feature and Lexicon Translation Feature Target → Source

2. **Reordering**
   - Reorder Info
   - Target → Source
   - Reordering Table

3. **Merge**
   - Merge
   - Phrase Table
Chaski

- Extract phrases
  - Sort on target
  - Score phrases
- Dummy Mapper
  - Sort on source
  - Learn Reordering

- Dummy Mapper
  - Sort on source
  - Score phrases

- Hadoop and HDFS handle large intermediate file and sorting
- Kept away from the merge operation which MapReduce is not good at
Performance of Chaski

<table>
<thead>
<tr>
<th>Time Used (Logged Hours)</th>
<th>Phrase Length 7</th>
<th>Phrase Length 10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Moses</strong></td>
<td>43</td>
<td>128</td>
</tr>
<tr>
<td><strong>M-R 50 Nodes</strong></td>
<td>0.7</td>
<td>1.9</td>
</tr>
<tr>
<td><strong>M-R 100 Nodes</strong></td>
<td>0.57</td>
<td>1.2</td>
</tr>
</tbody>
</table>
Rule Extraction for Syntactic Augmented Machine Translation System

- **Extract** probabilistic synchronous context-free grammar (PSCFG) rules in a Map step from all the target-parsed training sentence pairs
  
  From syntax-annotated phrase pairs...

  S -> Il ne marche pas / It does not work
  
  NP -> Il / It
  
  VB -> marche / work

  ...create rule:

  S -> NP ne VB pas / NP does not VB

- **Merge** and compute weights for them in the Reduce step
Distributed Tuning/Decoding (Trambo)

- Translate one sentence at a time
- Split up decoding into sub-processes
- Collect the output for MERT
Trambo

- Filter the phrase table and language models on a per-sentence basis -- beforehand
  - Each decoder instance loads faster
  - Memory usage is kept in check
- Tuning time (with Moses decoder):
  - 12.5 hrs on desktop machine
  - $\rightarrow$ 70 mins using 50 nodes
Filtering as (temporary?) Solution

- For language model
  - In Trambo, filtering a 11GB language model for 1600 sentences, ends up with 1TB temporary storage, with the limitation of 60 candidate target phrases for each source phrase

- Distributed Language Model?
Conclusion

- Working towards integrated cluster computing architecture for MT by providing cluster-enabled components
  - Data preprocessing
  - Word alignment
  - Phrase/Rule extraction
  - Tuning/Decoding
On-going Work

Research Side
- Distributed Asynchronous Online EM
- Improved phrase/rule extraction
- Efficient access of language models
- User study: Which kinds of automation can benefit researchers most

Engineering Side
- Error tolerant framework through fall-back strategies
- Porting PGIZA++ to Hadoop
- Merge Chaski and SAMT rule extractor into single framework
- Experiment management
- Master script to integrate components

Data preprocessing
- Word Alignment
- Phrase/Rule Extraction
- Tuning/Decoding
- System Integration
Thanks!
Questions & Suggestion?