Using Chapel for Natural Language Processing And Interaction

Brian Guarraci
CTO @ Cricket Health
Motivation

• Augment chat bot Human-created rulesets with data
  • ChatScript provides a powerful rule engine, but making Human-created rules is unscalable and limited
  • Use Chapel as a power-tool to create datasets which can be plugged into ChatScript engine
• Focus on two main types of custom datasets
  • Chord: Use word2vec for language support
  • Chriple: Use RDF triple stores for knowledge
Chord: Chapel + Word2Vec

• Word embeddings are vectors computed with a Neural Network Language Model (NNLM)

• Each word vector characterizes the associated word in relation to training data and other words in the vocabulary

• Vectors have interesting and useful NLP features
  • King - Man + Woman = Queen
  • Tokyo - Japan + France = Paris

• Replace Human-derived rules for certain NLP tasks
Chord: Path to Distributed

• First: Port Google’s single-locale classic word2vec and validate

• Second: Port classic model to a multi-locale model
  • Maintain single-locale performance in multi-locale version
  • Preserve Asynchronous SGD (race conditions by design)
  • Encapsulate globals to ensure locale-local only access
  • Experiment with dmapped and other distributed memory strategies to find a fast method for cross machine data sharing
Chord: Path to Distributed

• Distributed models require periodic model sharing across locales

• Naïve dmapped approach is very slow due to model specific behavior yielding excessive cross-machine data transfers

• Use a variant of Google’s Downpour SGD

  • Reserve some locales as “parameter locales” and others as compute locales which train on data shards

  • Each compute locale diverges with it’s training data and updates the parameter locales after each training iteration

  • Use AdaGrad to perform model updates on param locales
Chord: Architecture

Locales are partitioned into param and compute roles.
Chord: Single vs Multi-Locale

Multi-locale configuration:
- 8 locales: single parameter locale with seven compute locales
- Machine type: EC2 m4.2xlarge (8 vCPU 16GB RAM)

Multi-locale version > 3x faster with similar accuracy (eventually).
Chripple: Chapel + Triple Store

• Keep it simple to learn what’s useful

• Naïve implementation inspired by TripleBit
  • Reasonably memory efficient
  • Predicate-based hash partitions on locales

• CHASM (from Chearch) stack-based integer query language
  • Supports essential distributed query primitives (AND/OR)
  • Supports sub-graph extraction
Chriple: Architecture

Locale Predicate Hash Partition

Predicate Hash Table

Predicate Entry

Subject-Object Index

Object-Subject Index

64-bit Index Entry

32-bit ObjectId | 32-bit Subject ID
Chriple: Distributed Queries

Top-level Query \( Q_{\text{top}} \)

Partition Queries

Predicate Partitions (locales)

In-memory partition holds results from partition queries.
Chripple: Current Results

- Memory requirements
  - ~16 bytes per triple
  - 2B triples require ~64GB RAM across cluster
- Performance (8 x EC2 m4.2xlarge [8 vCPU 32GB RAM])
  - 1.1M inserts / s (~137K / locale)
  - 40K reads / s [via parallel iterator] (~5K / locale)
## AllegroGraph Benchmark

<table>
<thead>
<tr>
<th>Load Test</th>
<th># Triples</th>
<th>Time</th>
<th>Load Rate (T/Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM(8000)*</td>
<td>1.106 Billion</td>
<td>36min, 49 sec</td>
<td>500,679</td>
</tr>
<tr>
<td>LUBM(8000)****</td>
<td>1.106 Billion</td>
<td>48min, 30 sec</td>
<td>379,947</td>
</tr>
<tr>
<td>LUBM(160,000)*</td>
<td>22.12 Billion</td>
<td>12 hrs, 18m, 16s</td>
<td>499,188</td>
</tr>
<tr>
<td>AllegroGraph Pre-release**</td>
<td>310.269 Billion</td>
<td>78 hrs, 9m, 23s</td>
<td>1,102,737</td>
</tr>
<tr>
<td>AllegroGraph Pre-release***</td>
<td>1.009 Trillion</td>
<td>338 hrs, 5m</td>
<td>829,556</td>
</tr>
</tbody>
</table>

http://franz.com/agraph/allegrograph/agraph_benchmarks.lhtml
Conclusion

• Work in progress
  • Many opportunities for optimization
  • Useful for generating data and experimentation
• Code is available on Github
  • https://github.com/briangu/chord
  • https://github.com/briangu/chriple