USING PEGASUS FOR NLP & ML WORKFLOWS

Marjorie Freedman & Jacob Lichtefeld
• Research: Extract & organize information from multimedia sources

• A few common workflows
  – Training & Testing modeling
    • Train N versions of a model (different parameters, training data, etc.)
    • Test those models on a test set
    • Measure performance
  – Take many pretrained models and apply them, in sequence to a data set
    • Example: Broadcast news in a non-English language
      – Speech recognition → Machine Translation → Named Entity Recognition → Event Identification → Event → Entity relations ...
      – Event and object detection from video

• HPC to
  – Process many files quickly, independently (embarrassingly parallel)
  – ML “grid search”
For Us ... Why Pegasus

• Why Pegasus
  – Common work flows need to be re-run often
    • Automating the steps makes it easier to repeat
  – Replicability is good
  – Our HPC is set up for sharing → you can use the most machines if you keep your jobs short
    • This leads to pipelines with checkpoints

• What we’ve done: A Wrapper to make it easier for us to use Pegasus
Example 1: Fine Tuning Pre-trained Language Models for Leaderboards

• Several possible pre-trained language models (BERT, RoBERTa, T5, ....)

• Many leaderboards
  – Many with a shared structure (e.g. multiple choice question answering)
  – Each with its own training data

• Research goal: Repeatable framework for
  – Testing research questions, e.g. relationship between accuracy and size of training data
  – Optimizing parameters for a particular condition

• Result: Simple set-up run many times
  – For a specific Model, Training Data Sample, Set of Parameters
    • Train model
    • Run inference on development data
    • Score
  – Possibly, ensemble models and
    • Run inference on development data
    • Score
  – Aggregate into a table

<table>
<thead>
<tr>
<th>Model / Task</th>
<th>Hellaswag 10%</th>
<th>Hellaswag 25%</th>
<th>Hellaswag 50%</th>
<th>PhysicalQA 10%</th>
<th>PhysicalQA 25%</th>
<th>PhysicalQA 50%</th>
<th>SocialiQA 10%</th>
<th>SocialiQA 25%</th>
<th>SocialiQA 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensemble All</td>
<td>72.0</td>
<td>72.2</td>
<td>69.9</td>
<td>72.2</td>
<td>78.7</td>
<td>80.1</td>
<td>82.4</td>
<td>84.0</td>
<td></td>
</tr>
<tr>
<td>cn_10k_arc2_0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cn_10k_arc2_10061880</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>64.6</td>
<td>74.7</td>
<td>70.6</td>
<td>79.7</td>
<td>81.4</td>
</tr>
<tr>
<td>cn_10k_arc2_42</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>66.0</td>
<td>74.9</td>
<td>70.9</td>
<td>79.4</td>
<td>81.8</td>
<td></td>
</tr>
<tr>
<td>cn_10k_arc1_0</td>
<td>-</td>
<td>64.1</td>
<td>68.3</td>
<td>65.0</td>
<td>76.6</td>
<td>76.0</td>
<td>79.7</td>
<td>81.9</td>
<td></td>
</tr>
<tr>
<td>cn_10k_arc1_10061880</td>
<td>44.5</td>
<td>70.2</td>
<td>88.3</td>
<td>66.1</td>
<td>76.4</td>
<td>77.0</td>
<td>80.1</td>
<td>82.0</td>
<td></td>
</tr>
<tr>
<td>cn_10k_arc1_42</td>
<td>-</td>
<td>64.1</td>
<td>87.8</td>
<td>65.4</td>
<td>74.1</td>
<td>75.1</td>
<td>79.8</td>
<td>81.7</td>
<td></td>
</tr>
<tr>
<td>cn_10k_standard_0</td>
<td>67.3</td>
<td>69.0</td>
<td>69.5</td>
<td>64.5</td>
<td>72.4</td>
<td>74.8</td>
<td>79.4</td>
<td>80.5</td>
<td>82.6</td>
</tr>
<tr>
<td>cn_10k_standard_10061880</td>
<td>68.2</td>
<td>71.8</td>
<td>89.4</td>
<td>62.0</td>
<td>70.7</td>
<td>76.7</td>
<td>77.3</td>
<td>80.5</td>
<td>82.9</td>
</tr>
<tr>
<td>cn_10k_standard_42</td>
<td>67.5</td>
<td>69.6</td>
<td>88.6</td>
<td>63.0</td>
<td>70.4</td>
<td>77.1</td>
<td>79.4</td>
<td>81.8</td>
<td>83.3</td>
</tr>
<tr>
<td>arc2_0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>65.2</td>
<td>70.7</td>
<td>73.7</td>
<td>78.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>arc2_10061880</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>64.2</td>
<td>67.1</td>
<td>74.1</td>
<td>75.4</td>
<td>79.8</td>
<td>82.2</td>
</tr>
<tr>
<td>arc2_42</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>64.0</td>
<td>68.9</td>
<td>73.1</td>
<td>78.4</td>
<td>81.7</td>
<td>83.7</td>
</tr>
<tr>
<td>arc1_0</td>
<td>-</td>
<td>62.4</td>
<td>88.6</td>
<td>66.1</td>
<td>74.3</td>
<td>75.7</td>
<td>79.6</td>
<td>82.3</td>
<td>-</td>
</tr>
<tr>
<td>arc1_10061880</td>
<td>62.0</td>
<td>69.6</td>
<td>88.5</td>
<td>-</td>
<td>77.0</td>
<td>75.1</td>
<td>80.1</td>
<td>82.0</td>
<td>-</td>
</tr>
<tr>
<td>arc1_42</td>
<td>52.2</td>
<td>71.6</td>
<td>88.6</td>
<td>-</td>
<td>74.9</td>
<td>77.5</td>
<td>80.1</td>
<td>82.3</td>
<td>-</td>
</tr>
<tr>
<td>standard_0</td>
<td>76.1</td>
<td>72.8</td>
<td>89.2</td>
<td>67.2</td>
<td>74.4</td>
<td>75.4</td>
<td>79.1</td>
<td>80.7</td>
<td>82.7</td>
</tr>
<tr>
<td>standard_10061880</td>
<td>66.7</td>
<td>72.5</td>
<td>88.9</td>
<td>69.3</td>
<td>77.2</td>
<td>80.0</td>
<td>80.1</td>
<td>82.4</td>
<td>-</td>
</tr>
<tr>
<td>standard_42</td>
<td>58.0</td>
<td>74.1</td>
<td>88.0</td>
<td>-</td>
<td>71.0</td>
<td>76.9</td>
<td>80.1</td>
<td>82.7</td>
<td>-</td>
</tr>
</tbody>
</table>
Example 2: ADAM Learning Framework

• ADAM learns in a “child-like” manner learning graph patterns over a graph of simulated perceptual output
  – An experiment tests a curriculum designed to teach a specific concept (e.g. cookies are edible and circular; frisbees are circular but not edible)
  – Graph matching is slow
  – Our cluster is set up provide access to more nodes if each job is under an hour

• Set up ADAM learning framework to
  – Parallelize at the curriculum level
  – Restart after an hour (checkpointing)
  – Measure performance at the end of the curriculum
Example 3: Multimedia Document Processing

Outdated GAIA System Diagram (~April 2019)

- Models are static
- Each "document" can be processed independently
- Different processing applies to different documents
- File output passed between decoders
- An individual researcher is likely working on only one component, but needs to run full pipeline to test changes

Indicates many sub-components
What the Wrapper Provides

• One Function Call Setup for
  – Configuration Files
    • Including SAGA Cluster as a compute and storage site
  – Execution of Python Script on SAGA
    • Allows for easy use of venv (conda) on the cluster
    • Configuration of per job resource request via SLURM
    • Includes generating bash script to be the Pegasus job
    • Automatic configuration of transformations including reuse discovery

• File Checkpoint System
  – Avoid rerunning already completed jobs without interfering with the parameters structure already use to configure a python job

• Submission Bash Script to plan and execute a workflow

• Directory Structure creation on SAGA NAS for job execution
Why VISTA Wrapper?

• Simplifies user end workflow creation by abstracting away
  – Transformation Catalog
  – Slurm Resource Requests
  – Checkpoint Files to allow for data reuse

• Wrapper doesn’t require changing existing python script configuration via parameter files
  – Future work involves a way to connect the parameters files to Pegasus’ file management for complex workflows including non-SAGA cluster compute environments (e.g. AWS, Google Cloud)