USING PEGASUS FOR NLP & ML WORKFLOWS

Marjorie Freedman & Jacob Lichtefeld

Intro to Me and My Work (1)

- 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

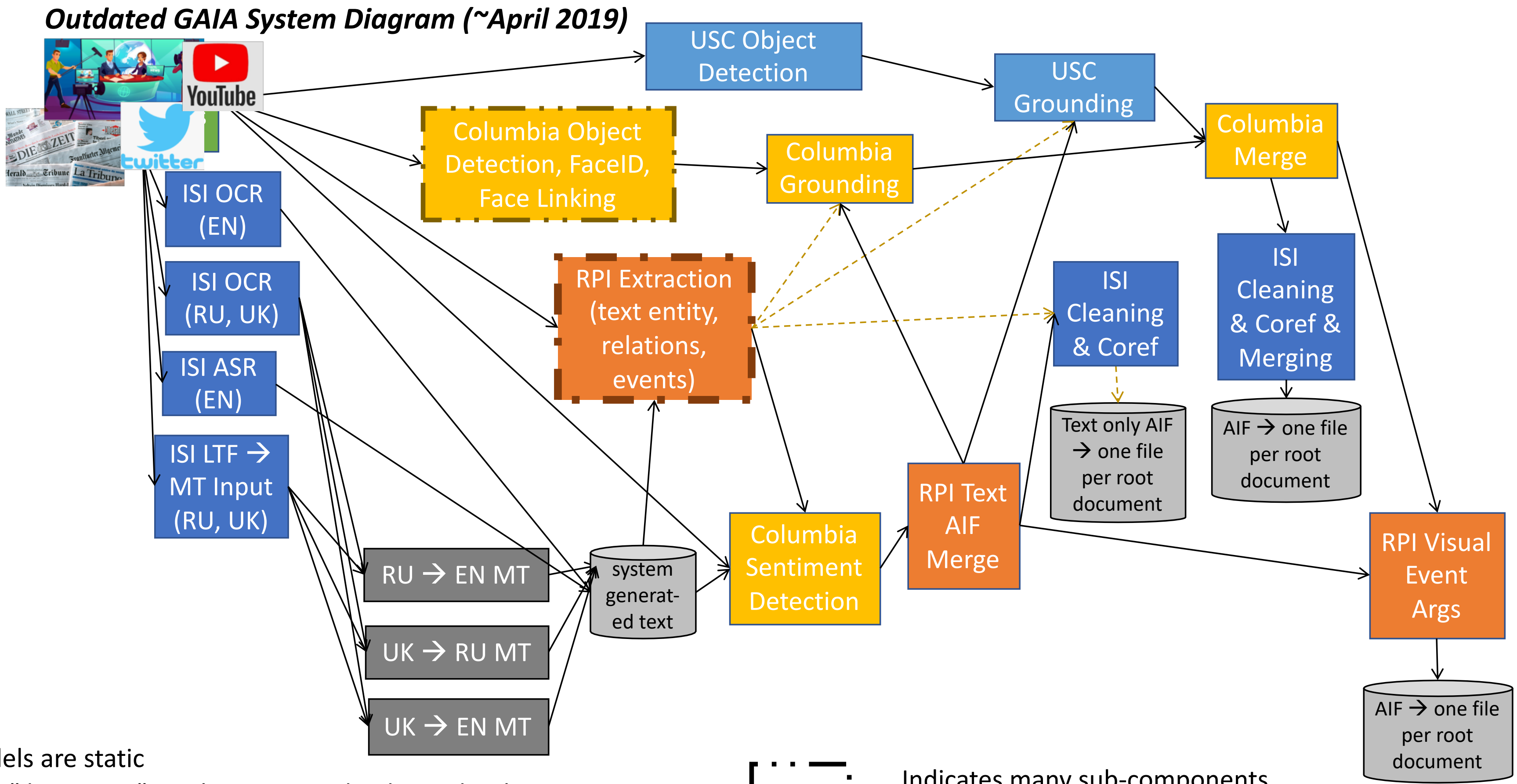
- Several possible pre-trained language models (BERT, RoBERTa, T5, ….)
- Many leaderboards
  - Many with a shared structure (e.g. multiple choice question answering)
  - Each with its own training data
- Research goal: Repeatable framework for
  - Testing research questions, e.g. *relationship between 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

Model / Task	Hellaswag 10%	Hellaswag 25%	Hellaswag 90%	Physicaliqa 10%	Physicaliqa 25%	Physicaliqa 90%	Socialiqa 10%	Socialiqa 25%	Socialiqa 90%
Ensemble All	72.4	76.2	91.9	69.9	72.2	79.7	80.1	82.4	84.0
cn_10k_arc2_0	-	-	-	-	63.7	75.9	66.6	75.8	81.4
cn_10k_arc2_10061880	-	-	-	-	64.6	76.7	69.4	79.7	81.7
cn_10k_arc2_42	-	-	-	-	68.4	74.9	-	77.4	81.4
cn_10k_arc1_0	-	64.2	88.3	-	65.0	76.6	76.0	79.7	81.9
cn_10k_arc1_10061880	61.5	70.2	88.3	-	66.1	76.6	77.0	80.1	82.0
cn_10k_arc1_42	-	63.5	87.8	60.4	65.4	75.1	75.1	79.0	81.7
cn_10k_standard_0	67.3	69.0	89.5	65.5	72.4	**77.5**	**79.4**	80.5	82.6
cn_10k_standard_10061880	68.2	71.8	89.4	62.3	70.7	76.7	77.3	80.5	82.9
cn_10k_standard_42	67.5	69.8	88.6	63.0	70.4	77.1	79.4	**81.3**	82.3
arc2_0	-	-	-	65.2	70.7	-	72.7	78.5	-
arc2_10061880	-	-	-	64.2	67.1	76.1	75.1	79.8	82.2
arc2_42	-	-	-	63.0	68.9	73.1	-	78.4	81.7
arc1_0	-	62.4	88.6	-	66.1	76.3	72.7	79.6	82.3
arc1_10061880	62.0	69.6	88.5	-	-	77.0	75.1	80.1	**83.0**
arc1_42	62.2	69.9	88.6	-	-	76.9	75.5	79.1	82.3
standard_0	**70.1**	**72.9**	89.2	-	**73.7**	73.4	79.1	80.7	82.7
standard_10061880	64.7	72.5	88.9	**69.2**	-	77.2	78.0	80.1	82.4
standard_42	68.0	74.1	**90.0**	-	71.0	76.9	77.5	80.2	82.7

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



- 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

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)