OpenEval

Simplifying Collaboration on Evaluating Natural Language Processing Problems

Joshua Camp, Paul Gibbons, Ryan Kelch, Deepak Shine, Dhruv Vajpeyi

Sponsored by the University of Illinois Cognitive Computation Group

Under the Supervision of Prof. Dan Roth

Daniel Khashabi, Christos Christodoulopoulos, Prof. Mark Sammons

Introduction

Teams working on machine learning problems have historically had several issues relating to evaluating their systems: spending time individually developing evaluation frameworks for tasks, comparing results over time, and keeping evaluations consistent among teams. OpenEval is a system designed to address these problems.

Main Objectives and System Overview

Before we begin a discussion of the actual system, we must define a few terms.

- A **task** is a specific AI problem such as Part of Speech Tagging or Named-Entity Recognition.
- A **task-variant** is a modification to the task. For example, part-of-speech tagging can be done on tokenized sentences or on raw text.
- A **solver** is a piece of software developed to solve a target task. For example, for the task of Part of Speech Tagging, the solver would receive sentences as input, and assign a part of speech to each word.
- A **dataset** is set of (input, output) pairs. In the part-of-speech tagging example, the input would be a sentence, and the output would be an ordered list of part-of-speech tags.
- A **configuration** is what the user runs. It encapsulates a task, a task-variant, a dataset.
- A **run** is specific instance of running the configuration. Users can run their configuration any number of items to see how their solver improves over time.

In developing this system we set out to build a centralized, easy-to-use platform for groups to evaluate their models. All the user needs to do to evaluate their solver is host it on a thin server, which we provide. Then, on the web interface, they need to select their desired task and dataset to test their solver (Figure 1). After their solver finishes processing the dataset, the user can view the results (Figure 2).

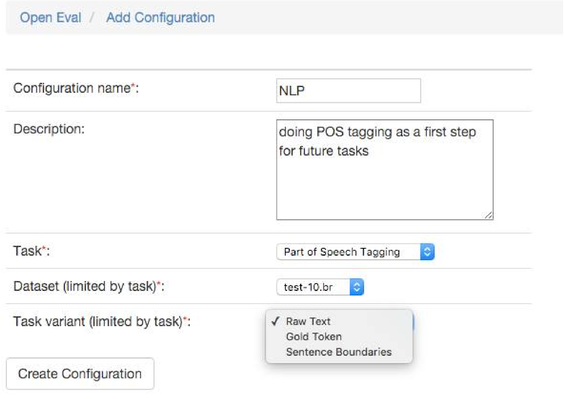

Figure 1: Adding a Run Configuration

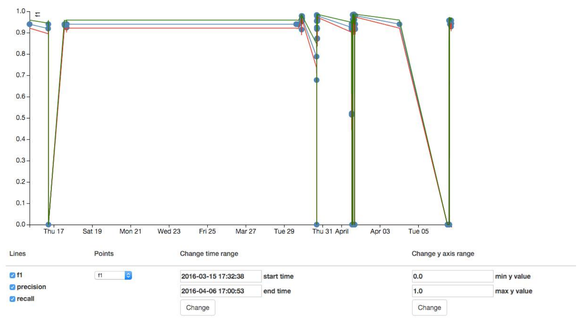

Figure 2: Results of Previous Evaluations

Technical Details

- Our project backend is implemented as a set of modules that pass around and process the instances of the dataset. These modules include
  - Database Interface: Allows the backend to store and retrieve configurations, evaluation records and datasets in the MySQL database.
  - Redactors: Processes the dataset and removes the values to be predicted.
  - Learner Endpoint: A thin server provided to the client. This is run independently to allow our system to send test instances to the solver and retrieve solved instances back.
  - Learner Interface: Sends test instances to the Learner Endpoint and retrieves predictions.
  - Evaluators: Creates an evaluation (results of accuracy and efficiency) given the solver's predictions and the original dataset.
  - Core: Connects all other modules by passing the data between them.
- The actual datasets are stored and passed around as lists of TextAnnotation objects. A TextAnnotation is a central data structure created by the IllinoisCogComp team that serves as a container for solved or predicted values of an NLP problem.
- Play is the web application framework we used, which allowed for fast, and easy development.
- The primary back-end language used is Java.
- The web Interface is written in Javascript, JQuery, HTML, and the Twirl Template Engine.
- Database management service used is MySQL.

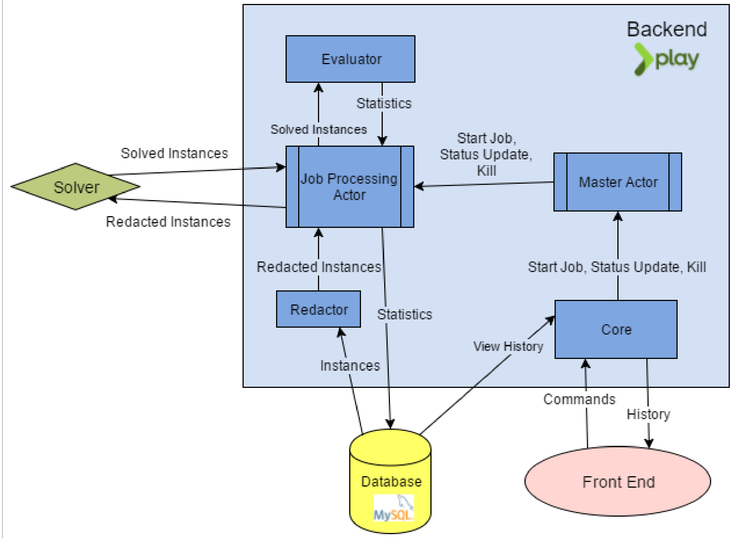

Figure 3: Work Flow

Future Additions

- Allow users to see the diff between their instances with the correct instances.
- The system can always be expanded to cover more AI problems. All this requires is uploading correct datasets and creating correct evaluators.
- Uploading datasets and evaluators instead of using the built-in ones in the system.
- Evaluate the solver on random sets of test data, to decrease over fitting the test data.

The code available at:
`https://github.com/CogComp/open-eval`