EWA-parser Framework

Components and capabilities

Table of Contents

The issue

- Context: experimental results from (Academic) Economics experiments.
- Data source: a small number of outdated software programs.
- Data format: obscure, inefficient, and redundant.
- Data readability: low.
- Automated post-processing: hard.
- Interoperability: low.

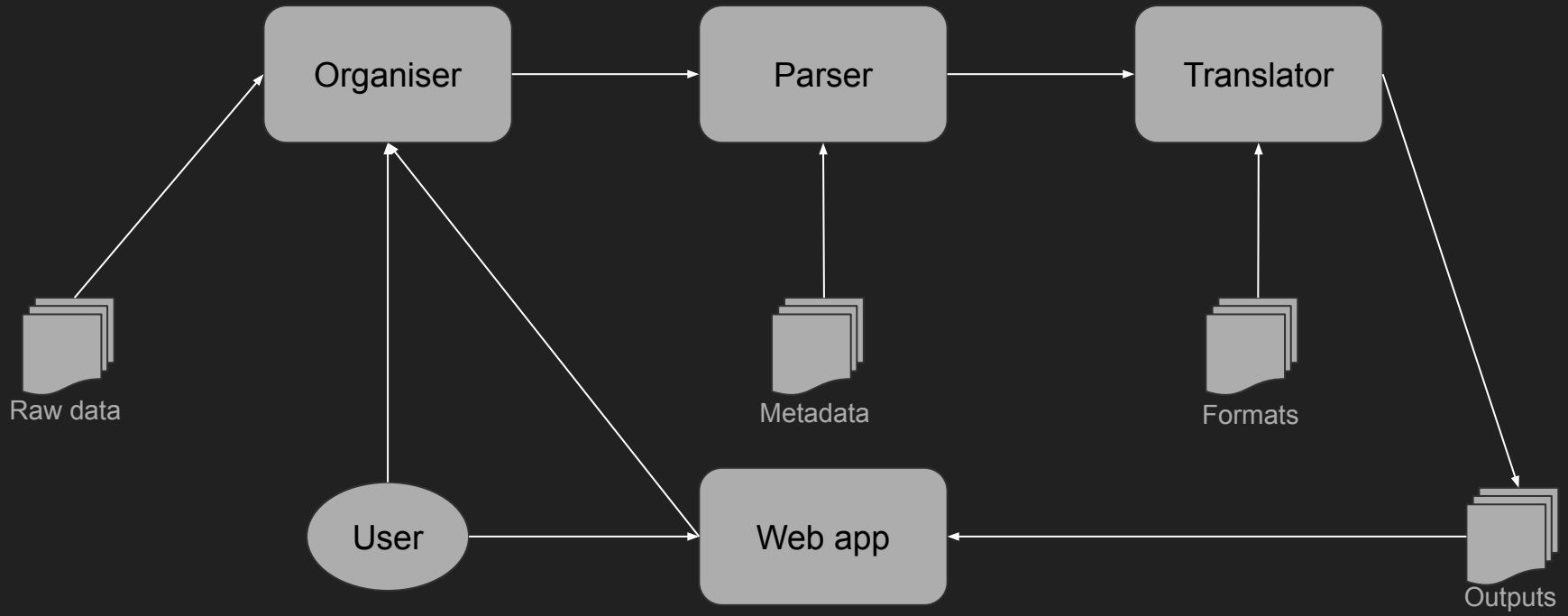
The proposal

- EWA-parser, a modular, pipeline-oriented framework:
  - Pre-processing (data organisation) of data files for compatibility reasons.
  - Data extraction.
  - Data parsing.
  - Data translation.
- Desired properties:
  - Extensibility.
  - Agility (with a caveat).
  - Interoperability.
  - Human readability.
  - Support for further automated analysis.

Design principles

- Modularity: small, incremental steps that can be replaced and plugged in.
- Extensibility:
  - Support for new modules at various stages.
  - Code readability: high-quality code to ensure readability and maintainability.
- Agility concentration: “Have one joint, and keep it well oiled” - Adam Langley
- Cross-platform support.
- Security:
  - Best coding practices.
  - State-of-the-art technologies.
  - Testing

Main architecture



The pipeline: pre-processing and data organisation

1. Supported files: .csv, .xlsx, .xls, and compressed folders (.tar.gz, .tar.xz, .zip).
    a. Everything else is simply ignored without failure to ensure extensibility (i.e., future support).
        i. Exception: executable files (ELF, PE, scripts, etc.) are rejected for security reasons.
2. Extract, and convert everything to standard CSV.
    a. As usual, everything unsupported is ignored.
    b. The data transition from a “raw unorganised” state to a “raw organised” state.

The pipeline: parsing

- If metadata specifying the player ID reuse policy are available, good.
  - Otherwise, we assume player IDs are re-used on a per-session basis.
  - No “session” column → unsupported dataset.
- Equivalence classes: different column names for the same concepts.
  - E.g., “treatment”, “game”, and “paper” (plus case variations).
- Mandatory columns: e.g., Player ID (or player 1 ID), Session, etc.
  - Equivalence classes are used to account for name variations.
- Optional columns: everything that is not mandatory.
- Procedure: the dataset is loaded, and all checks are performed.
  - If anything is amiss, the dataset is rejected as malformed (different from “unsupported”).

The pipeline: translating

- Format selection: either concurrent or single-format translation.
- Support for parallel translation of multiple datasets.
- Split output:
  - Main file with IDs, and experiment info.
  - Constants and payoff matrices (if available) → dedicated files.
- All output files are CSV files.

Output formats (general)

- Different output formats for different use cases:
  - Human readability → Minimising complex IDs, and sacrificing pipeline friendliness.
  - Automated post-processing → Maximising standardisation, and sacrificing readability.
- Naming convention: <id>_<first_author>_<year>.csv for most output formats.
  - Custom naming is supported (e.g., to resolve ambiguity).
- Multiple files:
  - matches.csv → main translated data.
  - payoff_matrix_<n>.csv → payoff matrices for each treatment (if relevant).
  - constants.csv → constant values in a key: value format for optimal storage.

Output formats (Vince’s)

- Experiment ID: features a counter, and a dataset ID.
  - Never to be reused across outputs.
- Dataset ID:
  - Condensed ID that keeps track of main author, and year.
- Treatment, session, and round (standard subdivision of an experiment)
  - Usually counters, or alphanumeric values.
- Player ID(s): unique for each player, to track players.
- Player choice(s): their nature depend on the game.
  - Usually numbers, or option IDs.

Output formats (David’s)

- Experiment: an experiment ID, similar to VInce’s dataset ID.
- Treatment, session, match period: equivalent to treatment, session, round.
- Match group: essentially, an opponent ID, or -1 when irrelevant.
- Supergame period: an identifier to group sessions into “super games”.
- Player: the player ID.
    - “Match group” keeps track of 1 v. 1 matches.
- Role: the role of the player, if available.
- a1~a6: up to six player actions in the same match period (round).
- Observation: the ID of a further relevant match, if relevant.

Data presentation

- For each experiment:
  - Main directory containing a sub-directory for each format, and the original paper (if available).
  - Format sub-directories:
    - matches.csv
    - constants.csv
    - payoff_matrix_<n>.csv
    - Other miscellaneous files (e.g., metadata, notes. etc.).
- Work in progress: web application
  - See next slide.

Web app (future work)

- Plan: build a web application that enables:
  - On-demand processing of single files, or collections of files.
  - Access to results.
- Requirements:
  - A database and a file server to store the results and the temporary artefacts.
  - A GNU/Linux machine for optimal hosting of the web server.
  - A domain name (perhaps a subdomain of dicelab-rhul.org).
  - Time estimate: ~1 month, including design, implementation, testing, and deployment.

Conclusions

- Motivation and issues: unreadable and machine-unfriendly experimental data produced by obsolete standalone software.
- Proposal: EWA-parser, a pipeline-oriented modular system that organises, parses, and translates experimental data to one or more common formats.
- Use:
  - Current: as a software framework.
  - Future: both as a software framework, and as a web service.