Case Study: Recognition Strategy Library

Kevin Talmadge, Professor Richard Zanibbi
Computer Science Department, Rochester Institute of Technology

Introduction

RSLib is a data provenance tool for pattern recognition research.

- "Run once, evaluate many times"

Purpose: Provide an easy means to

- Record intermediate interpretations
- Produce trace graph of the system
- Analyze and reuse saved interpretations

RSLib Program Elements

- **Strategy**: Object that manages execution and stores provenance information.
- **Interpretation**: Dictionary containing the interpretation data at each decision point.
- **Decision Function**: An individual decision point enqueued and run as part of a strategy. Operates using the current interpretation set and produces a new set.
- **Reporting Function**: Performs reporting operations after running a strategy.

Additions

Several contributions were made to enhance the functionality of the library:

- Saving and loading interpretation sets as-is
- Dynamically adding sub-decision points within a decision function
- Producing the set of unique interpretations
- Computing the intersection and difference of two interpretation sets

Test System

Math recognition system has three stages:

- Segmentation - Group strokes to form symbols
- Classification - Decide symbol labels
- Parsing - Determine symbol layout

Example Code

Decision Point Wrapper Function

```
def segmentFn(eq, interp, strategy):M# Run the
    segmenter.
eq.lei_CROHME2013_segment()

# Save the result in the interpretation.
interp.segments = eq.segments

# Return the updated interpretation.
return interp
```

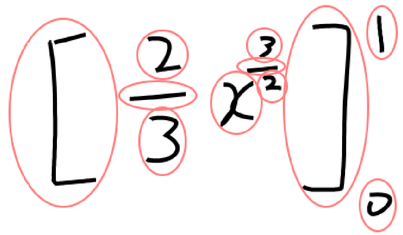
Creating and Running a Strategy

```
# Create the strategy and add decisison points.
strat = rsl.Strategy(iType)

strat.append((segmentFn, "segmentation", [eq]))
strat.append((classFn, "classification",
    [O_eq))
strat.append((parseFn, "parsing", []))

# Run the strategy.
strategy.run()
```

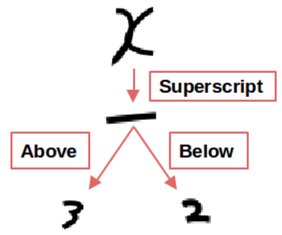
Producing a Report

```
# Get report function (a closure).
reportFn = parseReport(filename, strategy)

# Report the 'parsing' decision point.
strategy.reportInterps(``parsing'', reportFn)
```

Example Expression



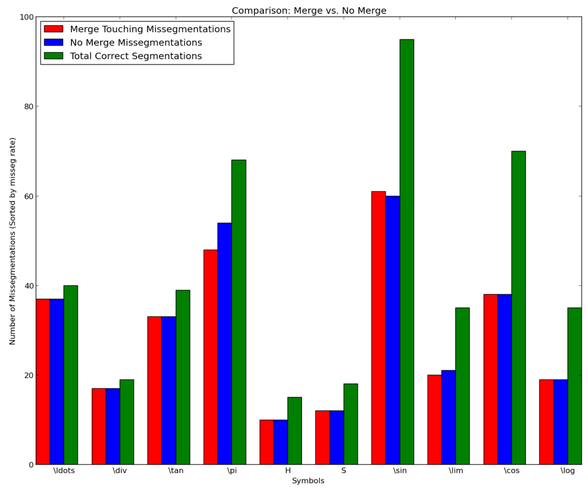
Segmentation:
Grouping Strokes into Symbols



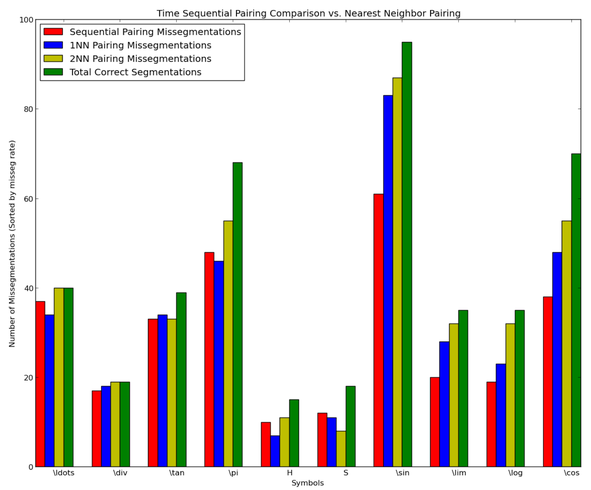
Parsing
Identifying Symbol Layout

Results

Comparison Experiments



Merge vs. No Merge



Time Order vs. Nearest Neighbor Pairing

Case Study

Comparison Experiments: Merging overlapping strokes in the segmentation stage

- Segmentation stage has a preprocessing step where all touching strokes are merged.
- Analyze results with merge, analyze results without merge, compare.
- Produce quantitative evidence of benefit or merge step.

Comparison: Time sequential vs. nearest neighbor pairing for segmentation

- Segmentation algorithm steps through strokes in time order, decides to merge or split
- Pairing by nearest neighbor may provide better results

References

[1] R. Zanibbi, et al., Decision-based Specification and Comparison of Table Recognition Algorithms., in: Machine Learning in Document Analysis and Recognition, 2008

[2] H. Mouchre, et. al ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions., in: ICDAR, IEEE, 2013

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1016815. This material is alse based upon work supported by the RIT Center for Student Innovation's SURF program.