HFST Transducer and Analysis Visualization

Jacob M. Springer
Swarthmore College

Introduction

The problem

- Large .lexc files are difficult to read.
- Beginners have trouble figuring out where to start when jumping into a project
- It is difficult to trace a transducer by hand when a form analyses incorrectly.

The solution

- I built a visualization tool to easily comprehend the structure of a .lexc file.
- The tool traces analyses of a word through the graph, highlighting the path in red.

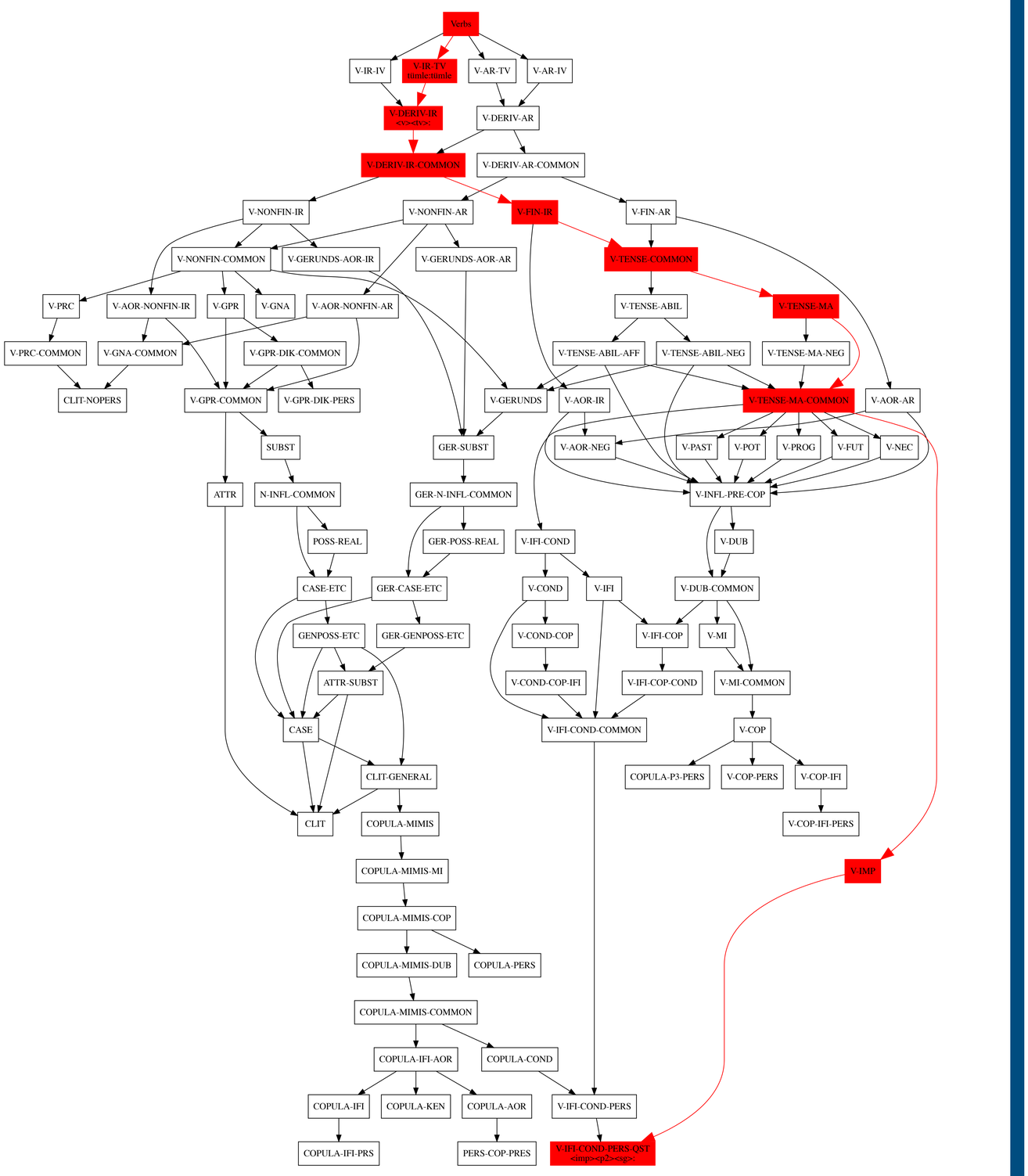
Implementation

The script `dapertium-trace.py`:

- Has arguments: [.lexc file] [output file] [word analysis] [word form] {--root [root node to display]}
- Parses the .lexc file into a graph
- Traces a given analysis path given a word analysis and corresponding word form
- Renders a graph to a file using Graphviz

Example (tümle)

```
$ python3 dapertium-trace.py apertium-tur.tur.lexc tur-hl/tümle
    'tümle<v><tv><imp><p2><sg>' tümle --root Verbs
```



Features and Problems

Can be used to:

- Quickly summarize large rulesets
- Track down bugs in the transducer
- Help beginners understand the transducer system

Known problems:

- Does not correctly handle TWOL rules
- Should be implemented using a state machine instead of a graph search algorithm
- Does not escape characters entirely correctly

Future Work

- Extend system to parse other file formats
- Improve the visualization to render more data more effectively
- Automatically detect possible analyses of a given stem
- Detect ambiguity and redundancy

Evaluation

Tested successfully on the following languages:

- Avaric, Bashkir, Berik, Buriat, Kazakh, Kyrgyz, Tuvinian, Turkish
- Anticipated to work on any .lexc file

Code and documentation can be found at github.com/jakespringer/dapertium