Introduction to OpenFst

Younes samih
Computational Linguistics
Introduction to OpenFst

• The use of Weighted Finite-State Transducers (WFSTs) for language processing.

• OpenFst Library, an open-source project developed by contributors from Google Research and NYU’s Courant Institute.

• This is a brief tutorial to get acquainted with OpenFST (how to create fsts, print them draw them, etc...).
Example: Automata / Acceptor

Graphical Representation (A.ps):

Acceptor File (A.txt):
0 0 red .5
0 1 green .3
1 2 blue
1 2 yellow .6
2 .8

Symbols File (A.syms):
red 1
green 2
blue 3
yellow 4
Example: Automata / Acceptor (Python)

```python
#Example 01 FST Acceptor

import fst

a = fst.Acceptor()
a.add_arc(0, 0, 'red', .5)
a.add_arc(0, 1, 'green', .3)
a.add_arc(1, 2, 'blue', .0)
a.add_arc(1, 2, 'yellow', .6)
a[2].final = .8

a
```

![Diagram](attachment:image.png)
Example Transducer

Graphical Representation (T.ps):

Transducer File (T.txt):
0 0 red yellow .5
0 1 green blue .3
1 2 blue green
1 2 yellow red .6
2 .8

Symbols File (T.syms):
red 1
green 2
blue 3
yellow 4
Example Transducer (Python)

t = fst.Transducer()
t.add_arc(0, 0, 'red', 'Yellow', .5)
t.add_arc(0, 1, 'green', 'blue', .3)
t.add_arc(1, 2, 'blue', 'green', .0)
t.add_arc(1, 2, 'yellow', 'red', .6)
t[2].final = .8

t
red: Yellow/0.5

diagram: [Diagram image of a state transition graph]
FST Operations

- Unary Operations
  - fstunaryop in.fst out.fst
  - fstunaryop < in.fst > out.fst

- Binary Operations
  - fstbinaryop in1.fst in2.fst out.fst
  - fstbinaryop - in2.fst < in1.fst > out.fst
Rational Operations: Sum

- From 0: green/0.3 to 1
- From 1: blue/yellow/0.6 to 2/0.8
- From 2: green/0.4 to 1
- From 0: blue/1.2 to 2
Rational Operations: Sum

Graph:
- Node 6 connects to nodes 0 and 3 with edges labeled ε.
- Node 0 connects to node 1 with an edge labeled red/0.5.
- Node 0 connects to node 5 with an edge labeled green/0.3.
- Node 1 connects to node 2 with edges labeled blue and yellow/0.6.
- Node 3 connects to node 4 with an edge labeled green/0.4.
- Node 4 connects to node 5 with an edge labeled blue/1.2.
Rational Operations: Product

A.fsa

B.fsa

C.fsa
Rational Operations: Closure

B.fsa

C.fsa
Binary Operations: Composition
Shortest-Path Algorithm
Creating FSTs Using Text Files from the Shell

# arc format: src dest ilabel olabel [weight]
# final state format: state [weight]
# lines may occur in any order except initial state must be first line
# unspecified weights default to 0.0 (for the library-default Weight type)

$ cat >text.fst <<EOF
0 1 a x .5
0 1 b y 1.5
1 2 c z 2.5
2 3.5
EOF
Symbol table

$ cat > isyms.txt <<EOF
<eps> 0
a 1
b 2
c 3
EOF

$ cat > osyms.txt <<EOF
<eps> 0
x 1
y 2
z 3
EOF
Creation of binary Fst from text file

# Creates binary Fst from text file.
# The symbolic labels will be converted into integers using the symbol table files.
$ fstcompile --isymbols=isyms.txt osymbols=osyms.txt text.fst binary.fst
Accessing FSTs: Printing, Drawing, Summarizing

- # Print FST using symbol table files.
- $ fstprint --isymbols=isyms.txt--
osymbols=osyms.txt.txt binary.fst text.fst
Drawing an FST

- # Draw FST using symbol table files and Graphviz dot:
  
  ```bash
  $ fstdraw --isymbols=isyms.txt
  --osymbols=osyms.txt binary.fst binary.dot
  $ dot -Tps binary.dot >binary.ps
  ```