3. Rules of Thumb

2. Applications and Examples

1. Description of the Finite-State Machine (FSM) Toolkit

Overview
Definition level: Specification of labels, of costs, and of kinds of FSM representations.

FSMDump("out. fsa", out);$fsm out = FSMintersect(fsm1, fsm2);$fsm 1n2 = FSMload("in2.fsa");$fsm 1n1 = FSMload("in1.fsa");

That implements the user program level, $fsm(?)$:

Library level: Library archive of C(++) functions

FSMintersect 1n1. fsm 1n2.fsm >out.fsm

Files or pipelines, $fsm(I)$:

User program level: Programs that read from and write to finite-state machines (FSMs).

The FSM tools construct, combine, minimize, and search weighted

Part I - Finite-State Machine (FSM) Toolkit
Utilities.

**Binary Format**: Compiled representation used by all FSMs

- Symbols files
- Transducer files
- Acceptor files

**Textual Format**: Used for manual inputting and viewing of

FSM File Types
### Graphical Representation (A, ps)

<table>
<thead>
<tr>
<th>PNN</th>
<th>CST</th>
<th>FIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3.3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3.0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5.5</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Textual Representation (A, txt)

Acceptor Files
Graphical Representation (T. ps):

<table>
<thead>
<tr>
<th></th>
<th>FIN</th>
<th>CST</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Textual Representation (T. txt):

Transducer Files
Graphical Representation (4, sps):

Symbols File (4, sym)

Acceptors and Symbols Files

- Green/0.3
- Red/0.5
- Blue/0
- Yellow/0.6

Symbols File (4, sym):

Acceptors and Symbols Files:

- Yellow
- Blue
- Green
- Red

Graphical Representation (4, sps):

Symbols File (4, sym)
Graphical Representation (T.sps):

Symbols File (T.syms):

Transducer File (T.txt)
Drawing of binary representations •

Printing of binary representations •

Compiling, printing, and drawing FSMs •
Graphical Representation

Program: T obtains A, Tsa, B, Tsa > C, Tsa

\[(B + A = C) \quad B \cap A = C\]

Union (Sum)
Graphical Representation

Program: fsmconcat A.fsa B.fsa C.fsa

Equation: \( C = AB \)

Concatenation (Product)
Graphical Representation

Program: fsmsteps  b.fsa > c.fsa

Equation: C = B * B_0 + B_1 + B_2

(Concatenative) Closure
Graphical Representation: •
Program: fsiReverse a.fsa > c.fsa
Equation: $A = \tilde{C}$

Reversal
Graphical Representation:

Program: transit A.fst > C.fst

Equation: $C = A^{-1}$

Inversion
A. fsa

Graphical Representation:

Program: fsprojec \texttt{T} \texttt{.test} \texttt{A}.fsa

Equation: \[ L^T v = A v \]
Graphical Representation:

Program: Transposition A. ftsa > C. ftsa

Epsilon Removal
Graphical Representation:

Program: \texttt{fsmintersect A.fsa B.fsa > C.fsa}

Equation: \( C = A \cup B \)

Intersection (Hadamard Product)
Graphical Representation

Program: fsdfgoogle A.fsa B.fsa C.fsa

Equation: C = A - B

Difference
Graphical Representation:

Program: fsmconnect A.fsa > C.fsa

Connection (Trimming)
Determinization

Graphical Representation: • Program: FSMdeterminize A, fsa > D, fsa
Minimization
Graphical Representation: $\text{M} \cong \text{M'}$

Program: $\text{transm} \text{eq} \text{M, M'}$

Equivalence
Graphical Representation: 

Program: fsmsbestpath [-n] A.tfa > C.tfa

Best Path(s)
Graphical Representation:

Random Path(s)
Graphical Representation:

Pruning

Program: trim prune -cl 0. A. ftsa > C. ftsa

C'ea

A'ea
<table>
<thead>
<tr>
<th>Format</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSMpackedReader</td>
<td>-p</td>
</tr>
<tr>
<td>FSMconstoutputClass</td>
<td>-co</td>
</tr>
<tr>
<td>FSMconstInputexedReaderClass</td>
<td>-ci</td>
</tr>
<tr>
<td>FSMconstexedReaderClass</td>
<td>-ci</td>
</tr>
<tr>
<td>FSMInitexedReaderClass</td>
<td>-c</td>
</tr>
<tr>
<td>FSMoutputexedReaderClass</td>
<td>0</td>
</tr>
<tr>
<td>FSMInputexedReaderClass</td>
<td>I</td>
</tr>
<tr>
<td>FSMexedReaderClass</td>
<td>I</td>
</tr>
<tr>
<td>FSMBasicClass</td>
<td>-q</td>
</tr>
</tbody>
</table>

Program:_FWD <A.fsa >C.fsa
Graphical Representation: FSTs in Speech and Language Processing

Program: Fsttopsort A. Fsa > C. Fsa

Topological Sort
Graphical Representation:

- # of states: 5
- Transducer in class basic
- Program: FSM Information
Program

FSM Information (numeric)
FSM Information (distributions)
Deterministic

Costs non-negative

Costless

No eps

Top sorted w/ eps

Top sorted

Acyclic w/ eps

Acyclic w/ start state

Acyclic

Connected

Arcs cost sorted

Arcs label sorted

FSM Information (Properties)
6. Sentence generation (with pronunciations)
5. Context-dependency in ASR
4. Language normalization
3. String alignment (matching with errors)
2. String matching
1. Keyword detection/recognition

Part 2 - Applications and Examples
else return 0;
}

if(strcmp(token, "int") == 0) return 1;
if(strcmp(token, "signed") == 0) return 1;
if(strcmp(token, "short") == 0) return 1;
if(strcmp(token, "continue") == 0) return 1;
if(strcmp(token, "char") == 0) return 1;

Brute-force search:

{char, const, continue, if, else, short, signed, sizeof}

C identifiers:

Keyword Detection
search (token, keywords, KEYS, sizeof (char *)) {

{ (**) char * (char *) x = strcmp (x, const void *)

int keyword (const void * const x)
{

'"sizeof"
'"strlen"
'"short"
'"int"
'"if"
'"else"
'"continue"
'"const"
'"char"

} = [KEYS] keywords [char * KEYS] [char * KEYS]

#define KEYS 9

Keyword Detection - Tabular Search
Keyword Detection – Automata Search
Keyword Detection – Deterministic Search
Search

Keyword Detection - Minimal Deterministic
Keyword Recognition - Automata Search
Key word Recognition – Deterministic Search
Search

Keyword Recognition – Minimal Deterministic
String matching

Graphical Representation:

Equation: $A \times \mathcal{L} = \mathcal{M}$
String Alignment
\[
\left\{ (q^t)^p + (\bar{q}^t)^p + (\bar{q}^t)^p \right\}_{\text{uni}} = (q^t)^p \\
\text{(insertion)} \quad (q^t)^t \quad + \\
\text{(deletion)} \quad (\bar{q}^t)^r \quad + \\
\text{(substitution)} \quad (\bar{q}^t)^t \quad + \\
0 = (q^t)^p
\]

<table>
<thead>
<tr>
<th>Type</th>
<th>Weight</th>
<th>Phone</th>
<th>Baseform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion</td>
<td>1</td>
<td>pr</td>
<td>e</td>
</tr>
<tr>
<td>Deletion</td>
<td>0</td>
<td>e</td>
<td>d</td>
</tr>
<tr>
<td>Substitution</td>
<td>0</td>
<td>eh</td>
<td>ae</td>
</tr>
</tbody>
</table>

**Minimum string edit distance (prefixes):**

**Symbol edit weights (task-specific):**

**Weighted edit distance:**
Graphical Representation:

Progam:

Equation: \( C = \text{argmin} (A \circ T \circ B) \)

String Alignment by Automata
<table>
<thead>
<tr>
<th>Context-Dependence</th>
<th>0</th>
<th>dx</th>
<th>$\Lambda$, $\Lambda/\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transposition</td>
<td>2</td>
<td>ax r</td>
<td>r eh</td>
</tr>
<tr>
<td>Contraction</td>
<td>3</td>
<td>em</td>
<td>eh m</td>
</tr>
<tr>
<td>Expansion</td>
<td>1</td>
<td>ped pr</td>
<td>p</td>
</tr>
<tr>
<td>Phone(s)</td>
<td>$n(q_{1}, q_{2})$</td>
<td>$q$</td>
<td>$q_{1}$</td>
</tr>
<tr>
<td>Weight(s)</td>
<td>TYPE</td>
<td>BASEFORM(s)</td>
<td>Generalized Weighted Edit Distance</td>
</tr>
</tbody>
</table>
Identity Transducer (ι)

Sentence Fragment Transducer (Λ)

<table>
<thead>
<tr>
<th>Neither · · nor theoretical computer science directions to English Phrase</th>
<th>Neither · · nor computer science directions to English Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>M. Mohri Phrase</td>
<td>M. Mohri Phrase</td>
</tr>
</tbody>
</table>

Language Normalization


\[(\mathcal{L})^{\text{I}}\mathcal{L} = N\]

then:

\[
\forall \mathcal{L} \exists \mathcal{L} - \mathcal{L} = \mathcal{L}
\]

and

\[
(\mathcal{L})^{\text{I}}\mathcal{L} = \mathcal{L}
\]

Let

Method 3: •

\[
(\mathcal{L})^{\text{I}}\mathcal{L} = \mathcal{L}
\]

Method 2: •

\[
\mathcal{L} \exists \mathcal{L} = \mathcal{L}
\]

Method 1: •

With Fragment Transducer and Identity Transducer: Language Normalization Transducer (N)
Triphonic context transducer for two symbols \(x\) and \(y\):

\[ p \rightarrow q \rightarrow aebp \]

CD units: Example: Maps from CI units to context-dependent phone models: Context-Dependency in ASR Modelling
N-Gram Language Model Examples

<table>
<thead>
<tr>
<th>5046499</th>
</tr>
</thead>
<tbody>
<tr>
<td>1679722</td>
</tr>
<tr>
<td>5046501</td>
</tr>
<tr>
<td>506971</td>
</tr>
<tr>
<td>19981</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2000</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>2</td>
</tr>
</tbody>
</table>

N-Gram Statistics

N-Gram

<table>
<thead>
<tr>
<th>3</th>
<th>2</th>
</tr>
</thead>
</table>

Phonetic Output

tsmproject -T tsmprint | tsmprint
| tsmreduced trigram.fsa | tsmcompose lexicon.fst

Phonetic Output

Suspending him without portfolio said it has no cabinet departments.

Analysts are so many the final tally of new accounting firms.

Opennheimer and company believes its main power broker shyn.

Social philosophy professor at Harvard UniversityCharles Brand of

Phonetic Output

<table>
<thead>
<tr>
<th>5046499</th>
</tr>
</thead>
<tbody>
<tr>
<td>1679722</td>
</tr>
<tr>
<td>5046501</td>
</tr>
<tr>
<td>506971</td>
</tr>
<tr>
<td>19981</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2000</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>2</td>
</tr>
</tbody>
</table>

N-Gram Language Model Examples

<table>
<thead>
<tr>
<th>3</th>
<th>2</th>
</tr>
</thead>
</table>
1. Use minimization for space
   (a) ordinary non-determinism
   (b) epsilon transitions

2. Reduce non-determinism for speed
   (d) do not depend on the distribution of costs along a path
   (c) do not depend on the relationship between input and output
   (e) do not depend on the presence or absence of epsilon
   (p) do not depend on the presence or absence of epsilon
   (q) encode information on transitions not states

3. Reduce non-determinism for speed
   (d) do not depend on the distribution of costs along a path
   (c) do not depend on the relationship between input and output
   (e) do not depend on the presence or absence of epsilon
   (p) do not depend on the presence or absence of epsilon
   (q) encode information on transitions not states

Rules of Thumb