Large Scale Inference of Deterministic Transductions

Tenjinno Problem 1

Alexander Clark
Royal Holloway, University of London
alexc@cs.rhul.ac.uk

Learning FSA

- Two families of algorithms for learning finite state automata
  - Deterministic automata
    - Construct Prefix tree automaton
    - Merge states and recursively determinise
  - Non-deterministic
    - Random ergodic FSA/HMM
    - Em based algorithm for training

Learning FST

- Two families of algorithms for learning finite state transducers
  - Deterministic transducers (OSTIA)
    - Construct Prefix tree automaton
    - Add outputs to leaf transitions
    - Convert to onward transducer
    - Merge, and push output
  - Non-deterministic
    - Random ergodic FST/PHMM/IOHMM
    - Em based algorithm for training

Problem overview

- Say 10,000 states, and alphabet of 1000/1001
- Neither approach will work
  - OSTIA relies on merge ordering
    - Won't work
  - PHMM is computationally intensive
    - Would work (?), but too much computation needed
    - Linear in both lengths, quadratic in states

Overall strategy

1) Learn input language with state-merging
   - Will be errors!
2) Do probabilistic alignment of a non-deterministic model
3) Use the non-determinism to identify errors
   - Split states until it is deterministic(*)
4) Merge the deterministic transducer
5) Finally merge to cover test data.

Phase 0

- Randomly split 95,000/5,000 train and validation set
- Throw away output labels
- Construct PTA
- Terminology:
  - Signature of a state is the set of labels on transitions out of that state

Histogram 0

Signature size	Types	Tokens
0	1	95000
1	1001	860039
2	15920	41278
3	7580	12756
4	4764	6259
5	1955	2250
6	77	84
7	2	2
8	0	0
Total	31300	1017668

Phase 1: learn DFA

- Standard state merging algorithm
  - Custom measure of similarity
- Index based on signatures
  - Compare all states that share three labels
  - Compare all states that share two labels
  - ....
- Need to have number of transitions < number of training pairs

Histogram after merging

Signature size	Types	Tokens
0	1	6520
1	841	2090
2	894	942
3	2059	2362
4	3906	3907
5	2089	2089
6	81	81
7	4	4
8	0	0
Total	9875	17995

Phase 2:Alignment algorithms

- Solve a relaxation of the real problem
- Given large DFA
  - For each transition have a distribution over outputs
  - Use EM algorithm to optimise this
- 3 problems
  - Initialisation
  - EM DP computation
  - Errors in original DFA

Exact alignment problem

- Solve linear word equations
  - T1 + T2 + T3 = “abcd”
- Maybe 50,000 variables
- 100,000 equations
  - Just for length!
- Highly over-constrained, solvable using standard techniques (Krylov subspace acceleration ...)

Initialisation

- Assume correct model
  - If we have a transition t
  - Then the substring output by t must lie in the intersection of the substrings for all output strings
- Initialised with weights drawn from co-occurrence statistics

Example

ABC --> XYZ
ABGE --> XYWW
ADEF --> XYPQR

Transition for A must generate a string that is in the intersection of the substrings of { XYZ, XYWW, XYPQR}

Must be { X, XY, Y or empty string }

Use distribution over these 4 strings;

EM computation

2-d dynamic programming grid
Posterior expectation that a particular transition generates a substring in the output

		X	Y	P	Q	R	
		0	1	2	3	4	5
	0	1	0	0	0	0	0
A	1	0	0	1	0	0	0
D	2	0	0	0	1	0	0
E	3	0	0	0	0	1	0
F	4	0	0	0	0	0	1

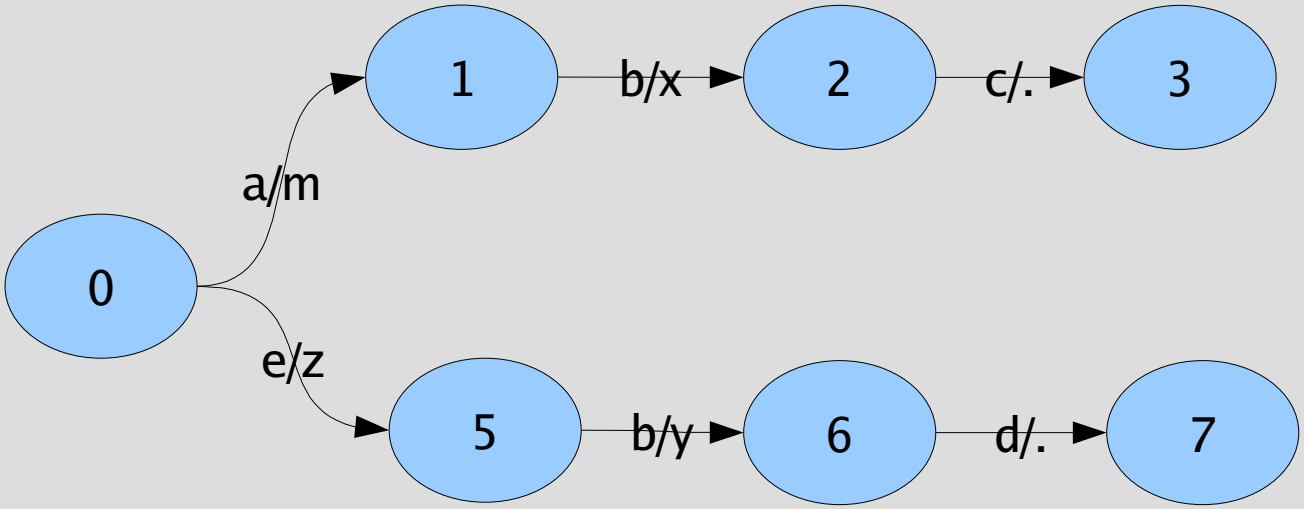
At convergence

- When we train the model to
- Nearly all transitions have a distribution with exactly one string with a non-zero probability.
- But there were 11 transitions with more than one string in output distribution

Phase 3: Splitting states

- Some transitions will have a non-trivial distribution
- This is because they are incorrectly merged and correspond to two different transitions in original transducer
- We can now split them...

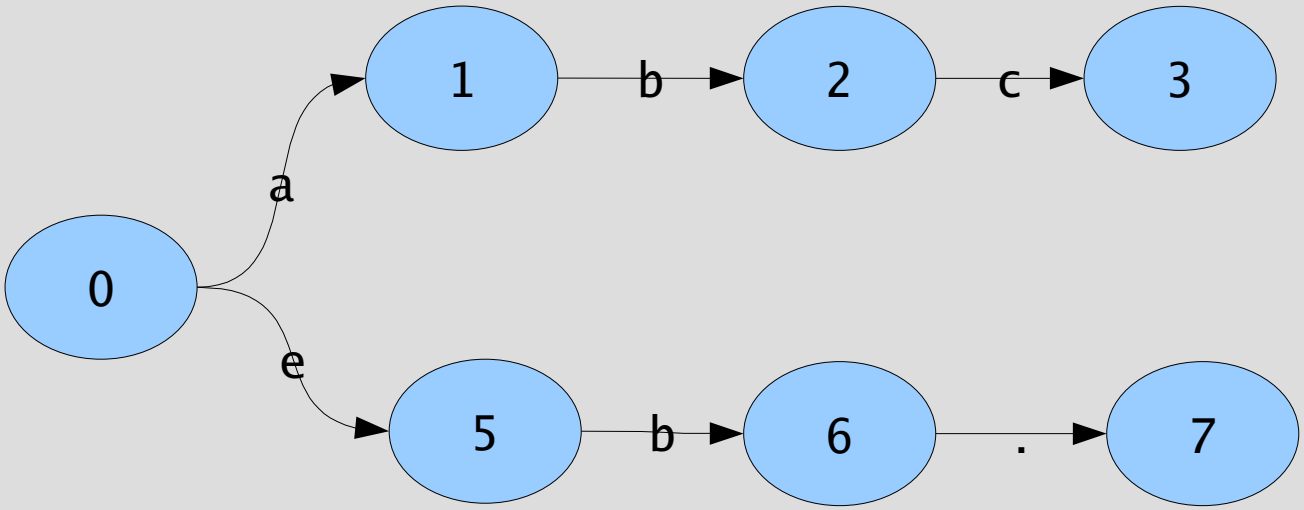
Example (Fragment)



DATA:
abc --> mx
ebd --> zy

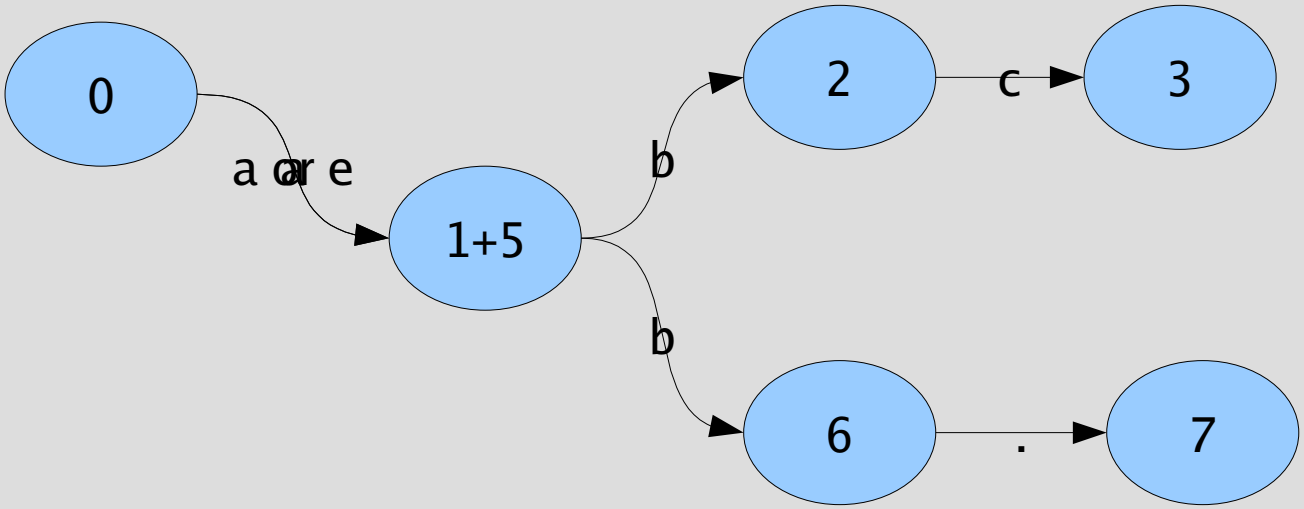
Construct PTA

INPUT DATA:
abc
ebd



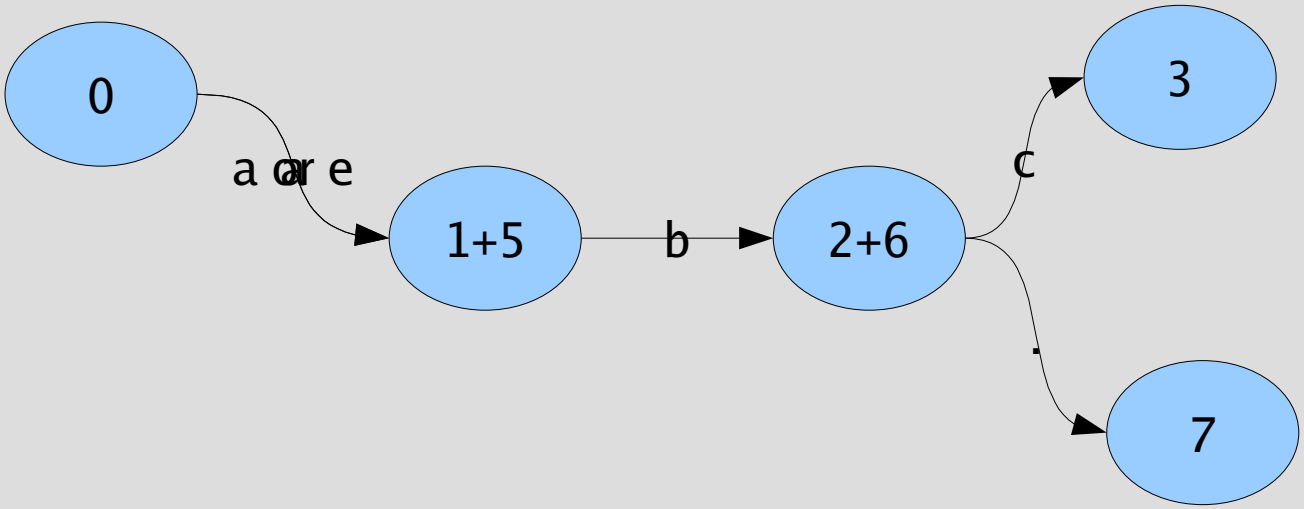
State 1 looks similar to State 5!

Merge incorrectly



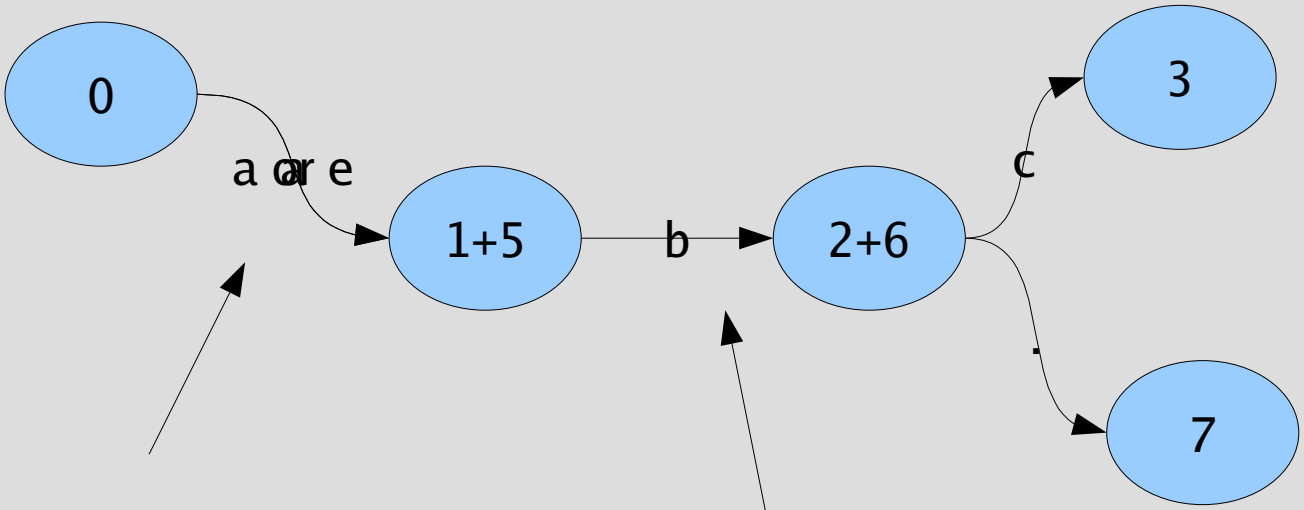
INPUT DATA:
abc
ebd

Determinise



INPUT DATA:
abc
ebd

Align



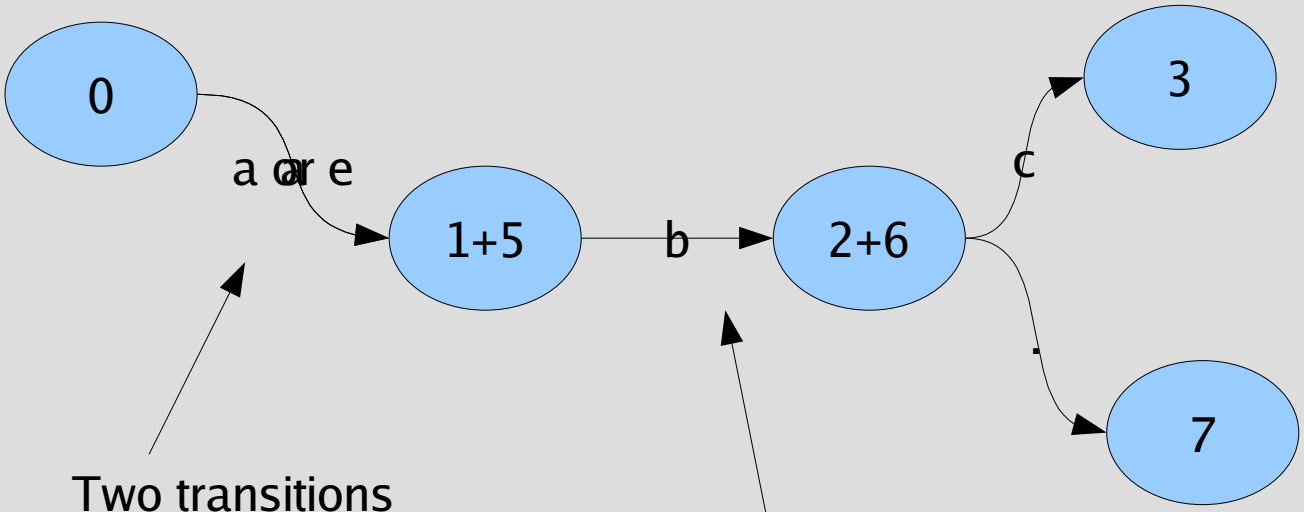
Could generate
m or mx or z or
zy or nothing

Could generate
m, mx, x, z, y,
zy or nothing

INPUT DATA:
abc
ebd

OUTPUT DATA:
abc --> mx
ebd --> zy

At convergence



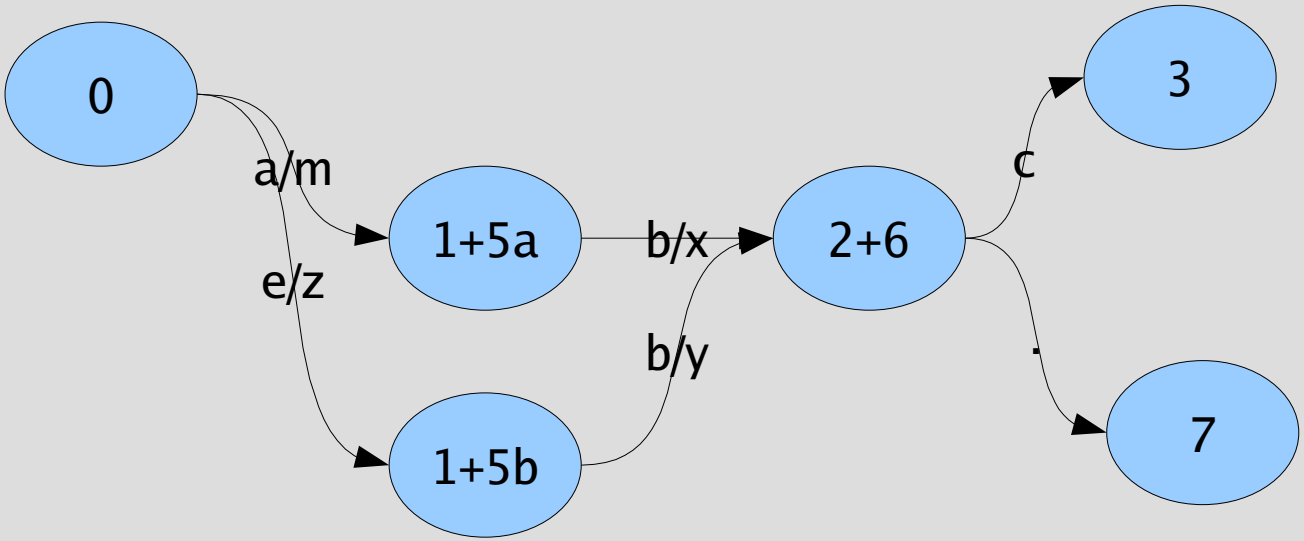
Two transitions
1 a --> m
1 e --> z

One transition
with two outputs
0.5 x
0.5 y

INPUT DATA:
abc
ebd

OUTPUT DATA:
abc --> mx
ebd --> zy

Split to make outputs deterministic



INPUT DATA:
abc
ebd

OUTPUT DATA:
abc --> mx
ebd --> zy

Merging the aligned automaton

- We now have a lot of extra information so we can merge states based not just on the input label but also on the output label.
  - Simple modification to similarity algorithm
  - Some more splitting is necessary

Final steps

- Merge using the training data.
- Given an input string not accepted by transducer
- Identify the state at which it leaves the transducer;
- Merge that state with the best match
  - Always have two arcs for this point.

Final Histogram

Signature size	Types	Tokens
0	1	1
1	5	5
2	23	23
3	1923	1923
4	3904	3904
5	2089	2090
6	81	81
7	4	4
8	0	0
Total	8030	8031

Other techniques

- Restricting the alignment algorithm
  - Tapering the alignment grids
- Aligning based on the PTA
- Segmentation of the output language
- Aligning based on per input letter distributions
- Oracle based techniques
  - Look for long output strings that are correlated with unique input symbols.

Implementation

- Lazy copying/ immutable substrings
- Cached hashcodes
- State merging with union find and generational dereferencing
- Efficient indexes
  - Hashing from signatures to hashsets
- Suffix arrays

Some thoughts on the competition

- These competitions are worthwhile
  - Drive focused research on particular issues
  - Large problems are important
- Some complaints
  - Exact identification is unrealistic
    - Error state identification and final merge are bogus
  - Random instances are too easy
  - Technical limitations of Oracle design limit non-determinism
    - More natural to allow many different possible outputs

CFG Problems

- CFG models computationally intractable
- Most algorithms require global search/alignment algorithms
  - Cubic in both input lengths (some have length 1000!),
  - cubic in number of states (>> 1000)
  - 100,000 examples
  - operations/iteration = 10^{32}
- CFG problems much too large
- Probably wouldn't give an exact answer

Conclusions

- A new(-ish) family of algorithms for FST inference based on
  - Input model induction
  - Alignment
    - EM based method
    - Novel initialisation
  - Splitting
    - Non-determinism in outputs identifies erroneous states
- Large alphabets + random instances are easy