Labs: an introduction

Peter Bell

Automatic Speech Recognition— ASR Lecture 5
30 January 2023

Labs

- Start this week, running for 8 weeks. Only labs 1-5 are compulsory.
- Work on the exercises with a lab partner
- Ask the demonstrator for help whenever you need it
- If working outside scheduled lab times, ask for help on Piazza
- If you need to find a partner feel free to post on Piazza or post privately and we will try to pair you up
- It's normal to continue with the same partner for all labs and the coursework, but not obligatory
- Attendance will be taken by the demonstrator

Lab times

Labs take place in an AT 4.12

- - Lab 01: Tuesdays 11.10 (Andrea)
  - Lab 02: Tuesdays 12.10 (Electra)
  - Lab 03: Thursdays 12.10 (Christoph/Ramon)
  - Lab 04: Thursdays 13.10 (Zeyu)
- Attend one lab per week
- You should attend the same lab as your partner
- To swap, use self-service timetabling: https://www.ed.ac.uk/timetabling-examinations/timetabling/personalised-timetables





Technical setup

- Labs use the DICE computing platform: see https://computing.help.inf.ed.ac.uk/linux if you are not familiar with it.
- Get the labs from a Github repository https://github.com/ZhaoZeyu1995/asr_labs (linked from Learn and the course web page).
- Setup instructions are in the repository README.
- Feel free to ask for technical support on Piazza ahead of your lab
- The exercises have been tested on the Lab PCs.
- Instructions are available for running on your own machine via Remote Desktop (XRDP) or using SSH tunnels

Lab exercises

- Skeleton code is provided in a Jupyter notebook. You will write and run your code within the notebook
  - If you prefer, you can work in pure Python you prefer to work with a different editor
- There is one notebook for each lab (except labs 3 and 4 share a notebook).
- Solutions will be available one week after the lab
- You will need to update the repo to receive subsequent weeks' exercises and the solutions, as well as any corrections or bug fixes

Asessment

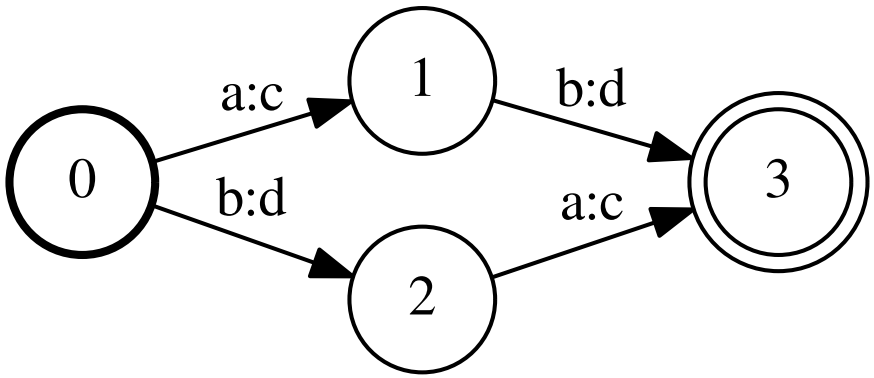
Lab solutions are worth 10% of the course mark

- 2 marks available for each lab
  - No submission – 0 marks
  - A weak/partial submission – 1 mark
  - A good effort – 2 marks
- Marks over the first five labs will be aggregated and scaled in accordance with the University's common marking scheme
- You will need to submit your solutions by Monday morning in the *second week* following the lab
- Only one person from each pair need submit a solution
- Submission will use CodeGrade, linked from the Learn page

HMMs and WFSTs

- The labs will use OpenFst `http://openfst.org` to build and manipulate HMMs represented as weighted finite-state transducers (WFSTs)
- You will first build WFSTs for various phone and word structures, compute forward probabilities and implement your own Viterbi decoder
- A WFST consists of states and directed arcs between them
- Each arc has an input label, and output label and (optionally) a weight (or cost)
- Labels can be blank (epsilon): written ε or `<eps>`
- The WFST *transduces* a sequence of input labels to a sequence of output labels (and optionally applies a weight to this)

WFST example

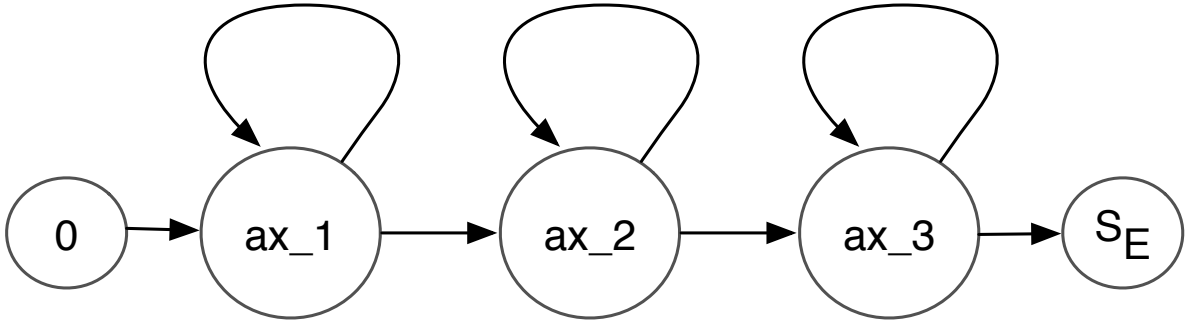


A very simple transducer mapping the string "ab" to "cd" and the string "ba" to "dc". The initial state is shown in bold; the final state is shown with a double circle.

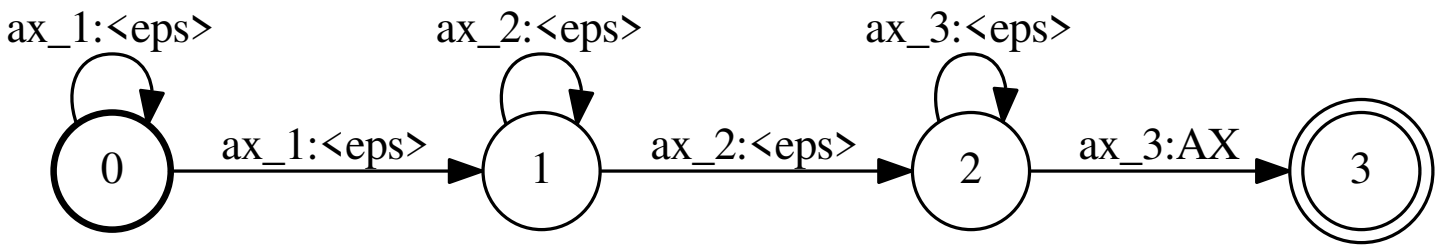
HMMs as WFSTs

- Easier to think of the HMM emitting states as being on the arcs
- Input labels used to denote state ID
- Output labels can be used to encode symbols to be output by the recogniser, such as words or phones
- Internally, both input and output labels are stored as an integer index, with a *Symbol Table* mapping between the two
- The blank (epsilon label) is given index 0 by convention
- We will cover WFSTs in much more detail later in the course

HMMs as WFSTs



Conventional HMM for phone "AX" with three emitting states



WFST representation of the HMM. Note the output label "AX"

Documentation

The lab exercises are intended to be self-contained, but you can find additional documentation:

- About OpenFst in general at https://www.openfst.org/twiki/bin/view/FST/WebHome
- About the Python interface at https://www.openfst.org/twiki/bin/view/FST/PythonExtension
- *NEW* Our own technical documentation at https://openfst-python-documentation.readthedocs.io/