Name: **(You may work in pairs)**

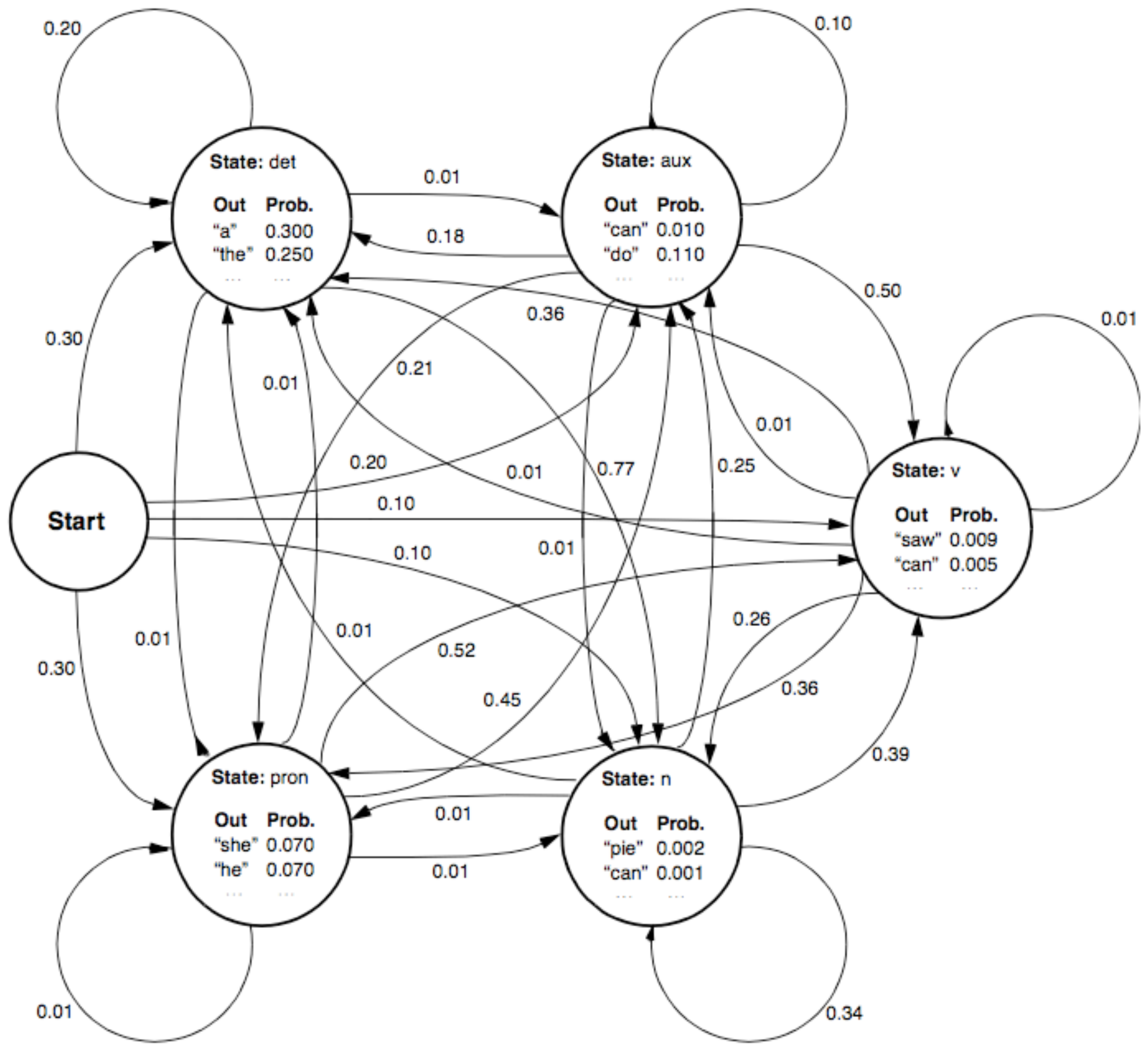
Statistical Lexical Category Disambiguation

Tutorial 4

The Parser Files

- We'll begin by looking at two simple tagger implementations in Prolog:
  `/proj/courses/comppsych/Tutorial4/tagger.pl`
  `/proj/courses/comppsych/Tutorial4/viterbi.pl`
- Copy all the files to your own directory.

The HMM model that is included with the prolog tagger corresponds to the following:



Notes: The probabilities of all transition arcs leaving a state should sum to 1.0. Similarly, the outputs from a state should also sum to 1.0, but here they don't since the lexicon is clearly incomplete.

1. Tagging in Prolog

The basic tagger computes the probabilities of all possible paths, depth first. There is a "write" statement, which displays all the alternatives it considers (note: the tags are in reverse order). To run the tagger on a string of words (note, it doesn't need to be a grammatical string) simply type:

```
?- most_probable_sequence([a,can,can],S).

?- most_probable_sequence([he,can,can,a,can],S).
```

Question: How many paths are considered for the second sentence? Explain why the algorithm considers exactly this many paths.

2. Viterbi in Prolog

The Viterbi algorithm is much more efficient (both in time and space), because it only computes the best path, and thus avoids keeping track of any paths which cannot lead to the best bath. You can try it again with the same sentences:

```
?- most_probable_sequence([a,can,can],S).

?- most_probable_sequence([he,can,can,a,can],S).
```

Question: How many paths are considered for the second sentence? Can you explain why the algorithm considers exactly this many paths. Can you tell which paths are not considered?

3. Tagging with TnT

The TnT tagger was developed at Coli by Thorsten Brants, and is widely used. It's a trigram-based tagger. The installation on the servers has been trained using two English corpora (`wsj` and `susanne`). You will need to log in to the Coli Unix servers from a terminal application, and copy the tutorial files to your (server) home directory:

```
      bash% ssh -l user_name login.coli.uni-sb.de
      bash% cp /proj/courses/comppsych/Tutorial4 .
```

First you will need to add the location of the tagger to your PATH environment variable:

```
      bash% PATH=$PATH:/proj/corpora/tools/bin
      bash% export PATH
```

`TnT` expects the text to be a file containing the text you wish to tag, with one word per line (there are simple unix commands you can use to do this for you).

```
bash% tr -sc 'a-zA-Z'  '\012'  < corpus.txt > text.txt
```

This will generate our input file. Afterwards, running the tagger requires the following:

```
tnt options model text
```

where option can be (`-z0`), and model needs to specify the full path of the model:

```
bash% tnt -z0 /proj/corpora/tools/src/tnt/models/wsj text.txt
```

Try tagging the sentences discussed in the lecture (see file `corpus.txt`) with the `susanne` model, and identify those cases where the tagger assigns the wrong POS tag to the critical word.

```
bash% tnt -z0 /proj/corpora/tools/src/tnt/models/susanne
text.txt
```

Info about the tagset that the susanne corpus uses is in `Susanne-Tagset.pdf`

a) N/V ambiguity:
   - The warehouse prices the beer very modestly.
   - The warehouse prices are cheaper than the rest.
   - The warehouse makes the beer very carefully.
   - The warehouse makes are cheaper than the rest.

b) And additionally the versions that would be disambiguated by number:
   - The warehouse make is cheaper than the rest.
   - The warehouse price is cheaper than the rest.

c) That ambiguity:

- That experienced diplomat(s) would be very helpful bothered the lawyer.
- The lawyer insisted that experienced diplomat(s) would be very helpful.

Tagging with the WSJ

Try tagging the same set of sentences using the `wsj` language model (which differs both in the corpus used to train the tagger, and the tag set used). Are there any interesting differences? Info about the tagset can be found in `P1AKN0`