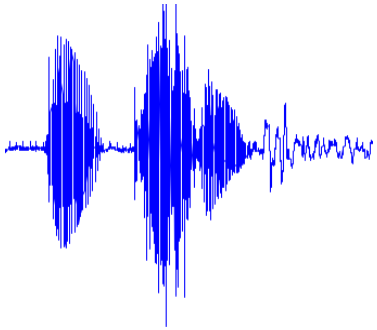
Timeline in the course

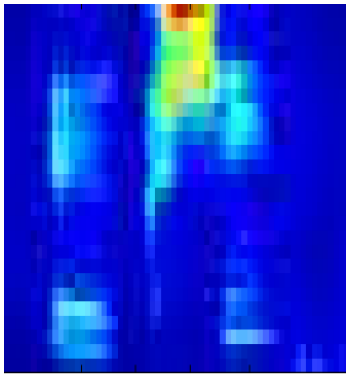
	Meetings Wednesdays	Thursdays or Fridays	Home exercises	Project work status
Week1	Speech features	Classification	Feature classifier	Literature study
	entry test			Meet tutors Wed
Week2	Phoneme modeling	Recognition	Word recognizer	Work plan
				Meet tutors Wed
Week3	Lexicon and language	Language model	Text predictor	Analysis
				Meet tutors Wed
Week4	Continuous speech	LVCSR	Speech recognizer	Experimentation
	advanced search			Meet tutors Wed
Week5	End-to-end ASR	End-to-end	End-to-end recognizer	Preparing reports
				Meet tutors Wed
Week6	Projects1	Projects2		Presentations
Week7	Projects3	Projects4		Report submissio
		Conclusion		

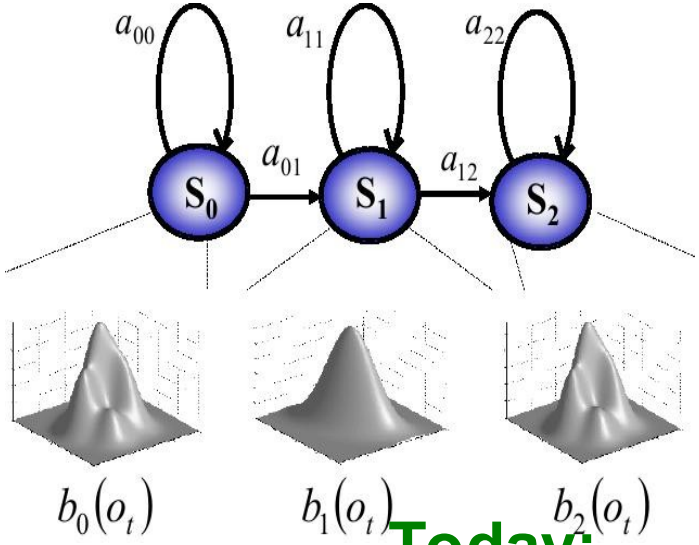
Content today

1. **Recognition of continuous speech**
2. A token passing decoder
3. Measuring and tuning the performance
4. Home exercise: Build a continuous speech recognition system
5. Status of project group works

Speech recognition -from beginning to end







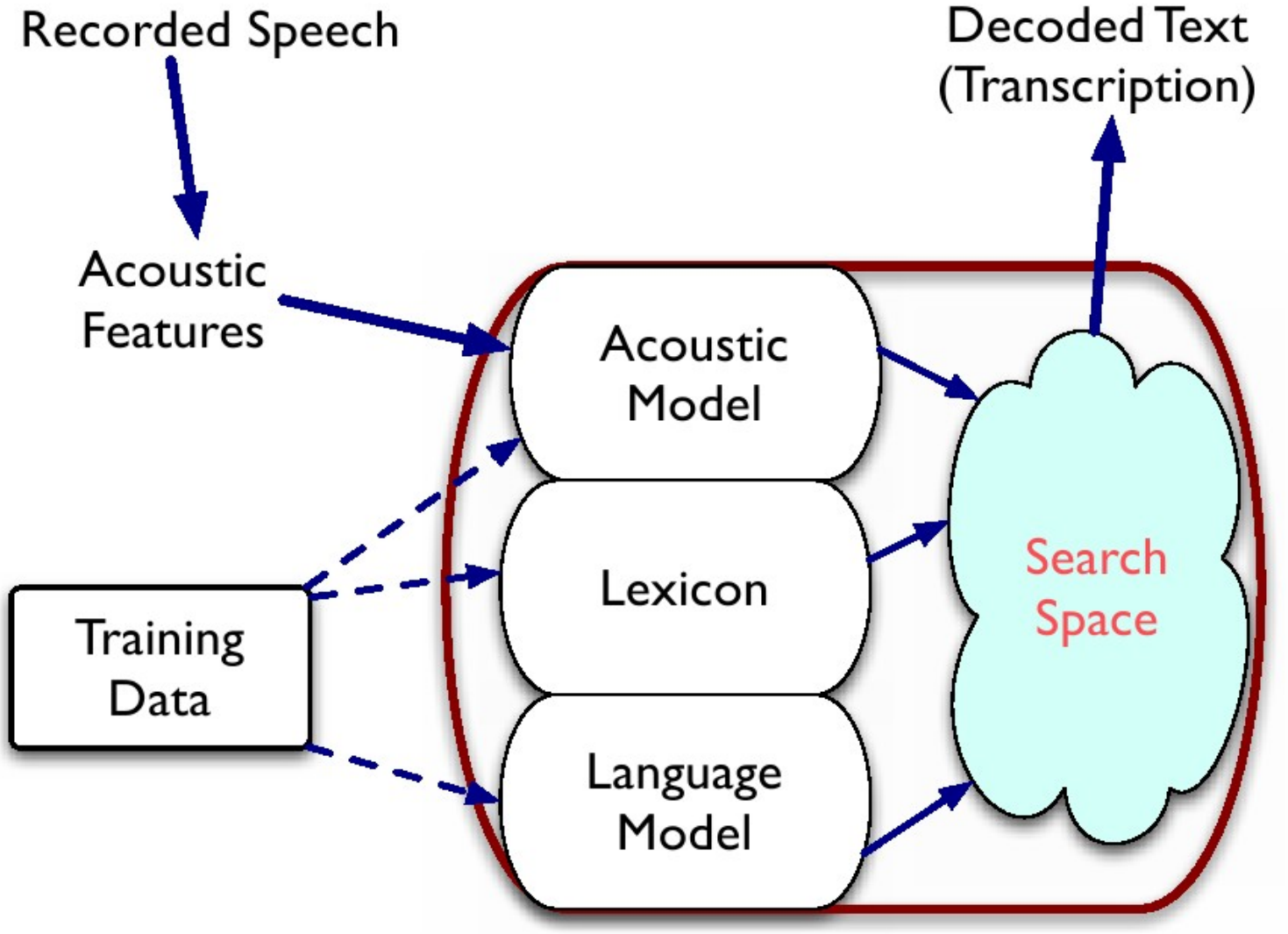
Dictation
Indexing
Translation

Today:

Speech signal → Feature extraction → Acoustic modeling → Decoder → *Recognized text*

Language modeling → Decoder

Decoding and search



Picture by S.Renals

Recognition performance

- Discussion: What factors limit the recognizer's performance?
  - what makes recognition slow?
  - what causes recognition errors?
  - could Lumi or any huge *supercomputer* recognize 100%?
  - how to improve the performance?

ASR definition

- Given a sequence of observations (evidence) from an audio signal,

$$\mathrm{O} = o_1 o_2 \cdots o_T$$

- Determine the underlying word sequence,

$$\mathrm{W} = w_1 w_2 \cdots w_m$$

- Number of words (m) unknown, observation sequence is variable length (T)

ASR solution

- **Goal: Minimize the classification error rate**

- **Solution: Maximize the Posterior Probability**

$$\hat{W} = \arg\max_{W} P(W \mid O)$$

- **Solution requires optimization over all possible word strings!**

Bayes rule in ASR

- **Using Bayes Rule,**

$$P(\mathrm{W} \mid \mathrm{O}) = \frac{P(\mathrm{O} \mid \mathrm{W})P(\mathrm{W})}{P(\mathrm{O})}$$

- **Since P(O) does not impact optimization,**

$$\hat{\mathrm{W}} = \arg\max_{\mathrm{W}} P(\mathrm{W} \mid \mathrm{O})$$

$$= \arg\max_{\mathrm{W}} P(\mathrm{O} \mid \mathrm{W})P(\mathrm{W})$$

From words to states

- **Let's assume words can be represented by a sequence of states, S,**

$$\hat{\mathrm{W}} = \arg\max_{\mathrm{W}} P(\mathrm{O} \mid \mathrm{W}) P(\mathrm{W})$$

$$= \arg\max_{\mathrm{W}} \sum_{S} P(\mathrm{O} \mid \mathrm{S}) P(S \mid W) P(\mathrm{W})$$

- **Words → Phonemes → States**
- **States represent smaller pieces of phonemes**

ASR solution

- **Optimize:** $\hat{W} = \arg\max_{W} \sum_{S} P(O \mid S) P(S \mid W) P(W)$

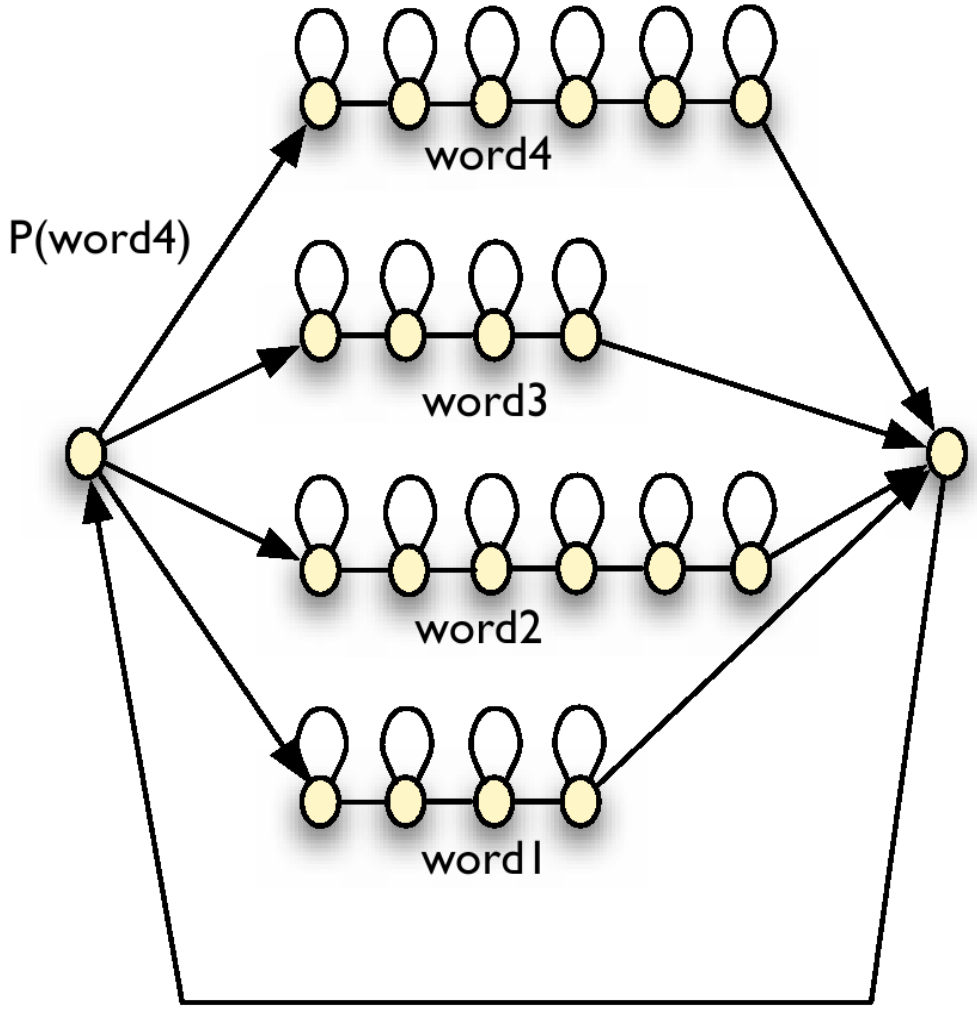
- **Practical Realization,**

O	Observation (feature) sequence
$P(O \mid S)$	Acoustic Model
$P(S \mid W)$	Lexicon / Pronunciation Model
$P(W)$	Language Model

Decoding

- The task is to find **the most probable word sequence**, given our models and the recorded acoustic observations
- **Viterbi search**: Find the most probable **state sequence** (not sum of all sequences)
- An efficient exhaustive search by applying dynamic programming and recursion
- For large vocabulary and continuous speech (**LVCSR), heavy pruning and optimization required!**

From words to continuous speech

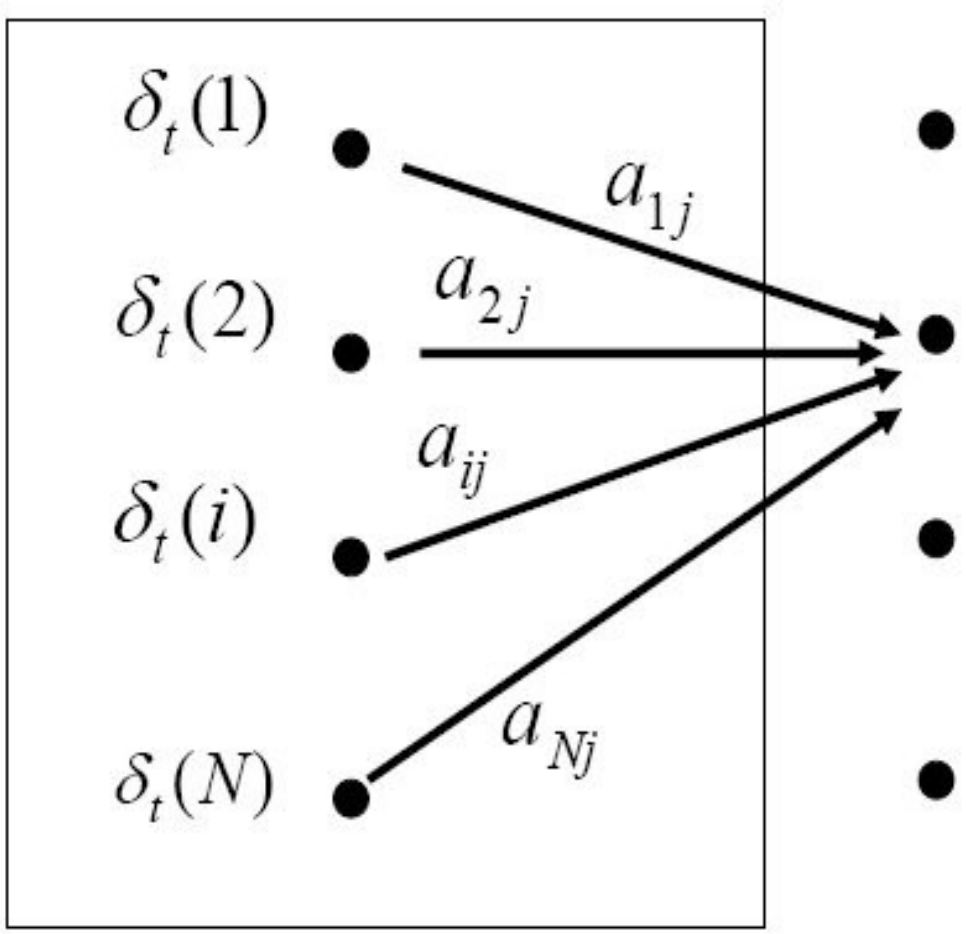


Picture by S.Renals

From words to continuous speech

- **Transition times** from word to word **unknown**
- **No silence** between words
- We must search for transitions to **any word at any time**
- For N-gram N>2, there are many possible word histories
- With a large vocabulary this explodes the search space!
- Decoding must be made more intelligent:
  - prune out **unlikely hypothesis**, asap
  - eliminate any **repeated computations**

Review: Recursion step in Viterbi



Computing the path score:

$$\delta_{t+1}(j) = \max_{1 \le i \le N} [\delta_t(i) a_{ij}] b_j(\mathbf{o}_{t+1})$$

Note: For jumping from word1 to word2:
a(i,j) = P(word2|word1)

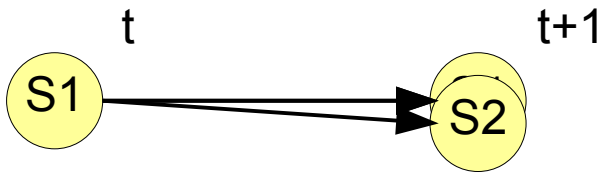
Picture by B.Pellom

Token passing decoder

- **An extension of Viterbi decoding to use language models**
- **Problem in Viterbi:** No history beyond the previous state can affect the decisions! => no language models other than bigrams
- **Extension**: Keep several word histories by carrying the histories as “tokens” through each state. The word histories are used to compute LM probabilities and prune away unlikely tokens.
- **Difference to Viterbi:** Several word histories (tokens) are preserved up to the current state

Step 1. Passing tokens to (t+1)

- Let each state carry several tokens (instead of one best), one for each different word history:



token = <*path history H, path cost P*>

- In the next time step, the state **throws** all its tokens to all its successor states, and updates them
- **Difference to Viterbi:** All possible word histories (the tokens) are preserved up to the current state
- At the end of the utterance: Select the best token
- Tokens mark the fastest path

  - like a colony of ants

Picture by M.Karatay (wikipedia)
Video by K.Schulz

http://tinyurl.com/ohaexm3

Step 2. Update of tokens at (t+1)

- add the new state to the path history
- add the state **transition probability** to the new state
- add the probability of the **acoustic observation** there
- add the new **language model probability**, when entering a new word

<H,P> => <H(t+1),P(t+1)>

H(t+1) = **S(t+1)**

P(t+1) = P(t) * a(t , t+1)

* b(S(t+1), O(t+1))

* p(S(t+1) | H(t))

Step 2. Update of tokens at (t+1)

- add the new state to the path history
- add the state **transition probability** to the new state
- add the probability of the **acoustic observation** there
- add the new **language model probability**, when entering a new word

$$\langle H,P \rangle \Rightarrow \langle H(t+1), P(t+1) \rangle$$

$$H(t+1) = S(t+1)$$

$$P(t+1) = P(t) * \mathbf{a(t, t+1)} * b(S(t+1), O(t+1)) * p(S(t+1) \mid H(t))$$

Step 2. Update of tokens at (t+1)

- add the new state to the path history
- add the state **transition probability** to the new state
- add the probability of the **acoustic observation** there
- add the new **language model probability**, when entering a new word

$$\langle H,P \rangle \Rightarrow \langle H(t+1), P(t+1) \rangle$$

$$H(t+1) = S(t+1)$$

$$P(t+1) = P(t) * a(t, t+1) * b(S(t+1), O(t+1)) * p(S(t+1) \mid H(t))$$

Step 2. Update of tokens at (t+1)

- add the new state to the path history
- add the state **transition probability** to the new state
- add the probability of the **acoustic observation** there
- add the new **language model probability**, when entering a new word

$$<H,P> \Rightarrow <H(t+1),P(t+1)>$$

$$H(t+1) = S(t+1)$$

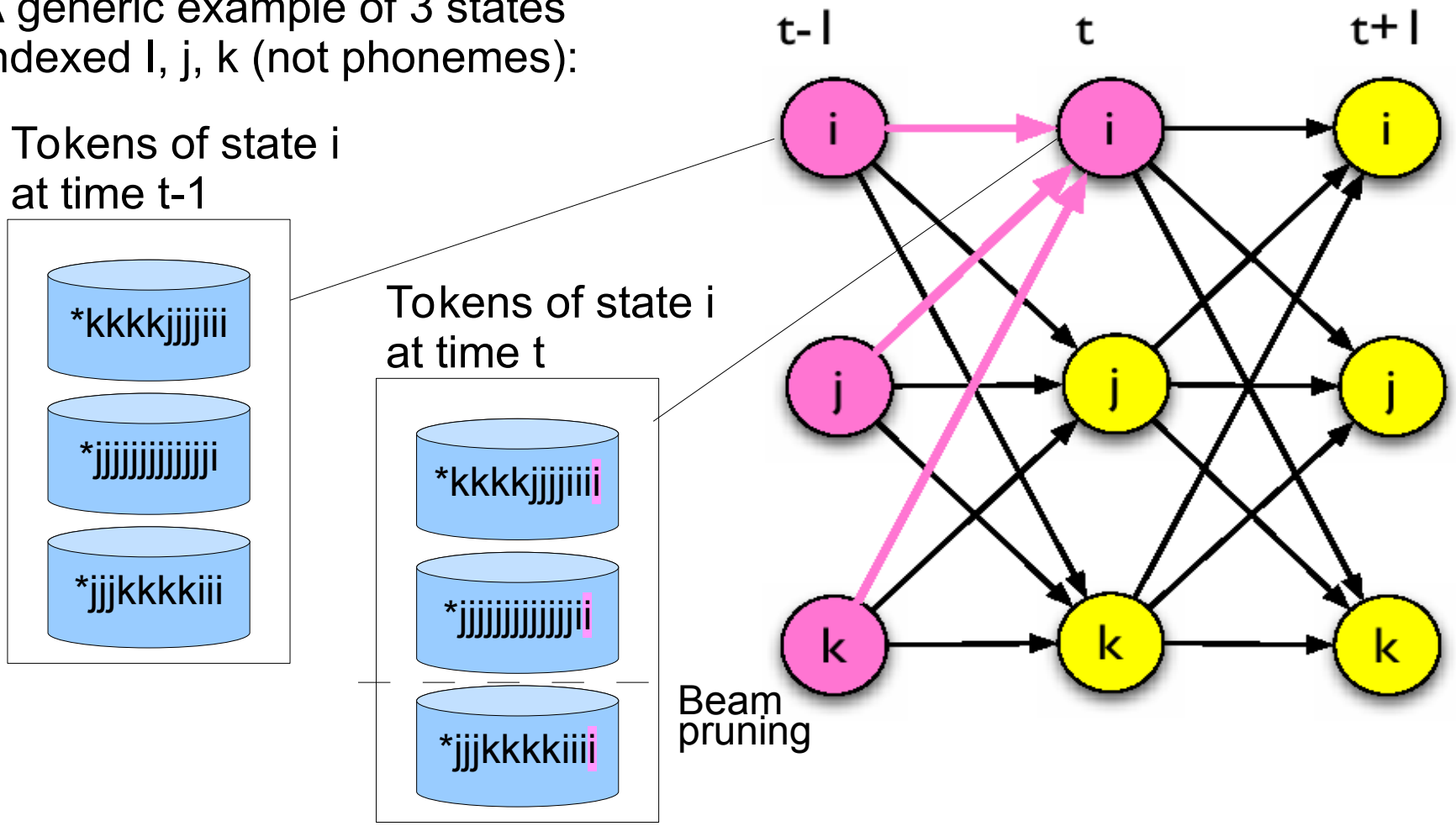
$$P(t+1) = P(t) * a(t, t+1) * b(S(t+1), O(t+1)) * p(S(t+1) \mid H(t))$$

Step 3. Pruning and merging tokens

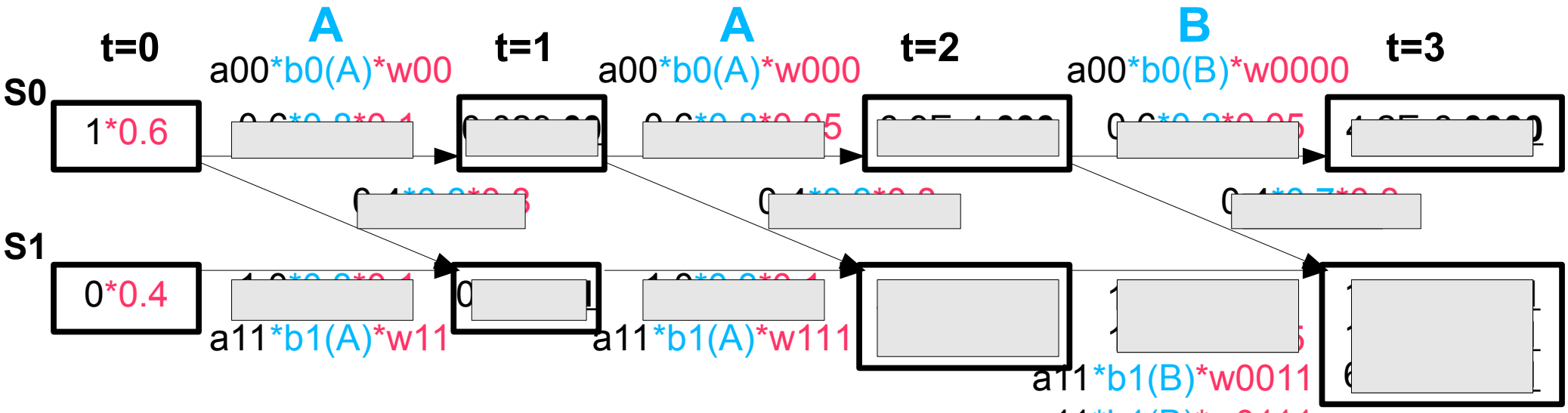
- **Difference to Viterbi:** Conserve all possible word histories (the tokens) up to the current state
- **The amount of tokens increases rapidly**
- Tokens can be merged when the N last words are the same (out of N-gram span) => select the best
- Tokens that have a low score can be pruned in several ways
- At the end of the utterance: Select the best token.

Token passing vs. Viterbi

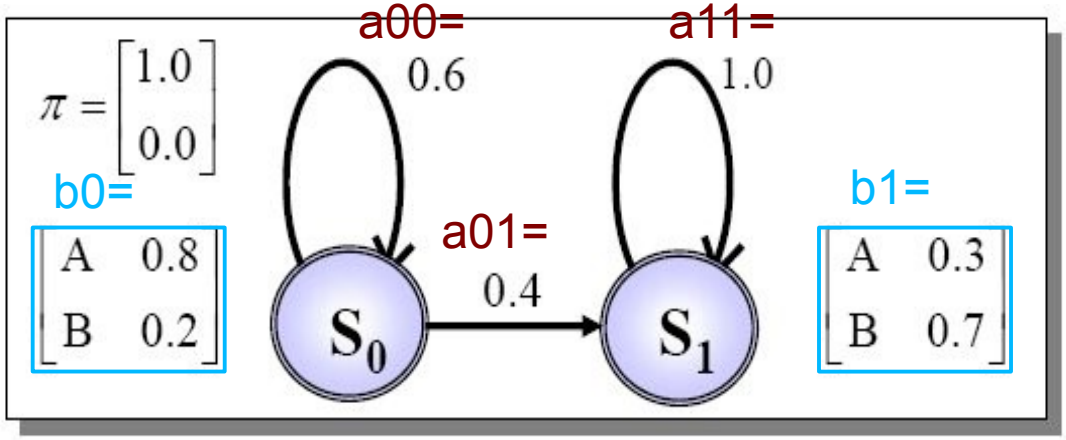
A generic example of 3 states indexed I, j, k (not phonemes):



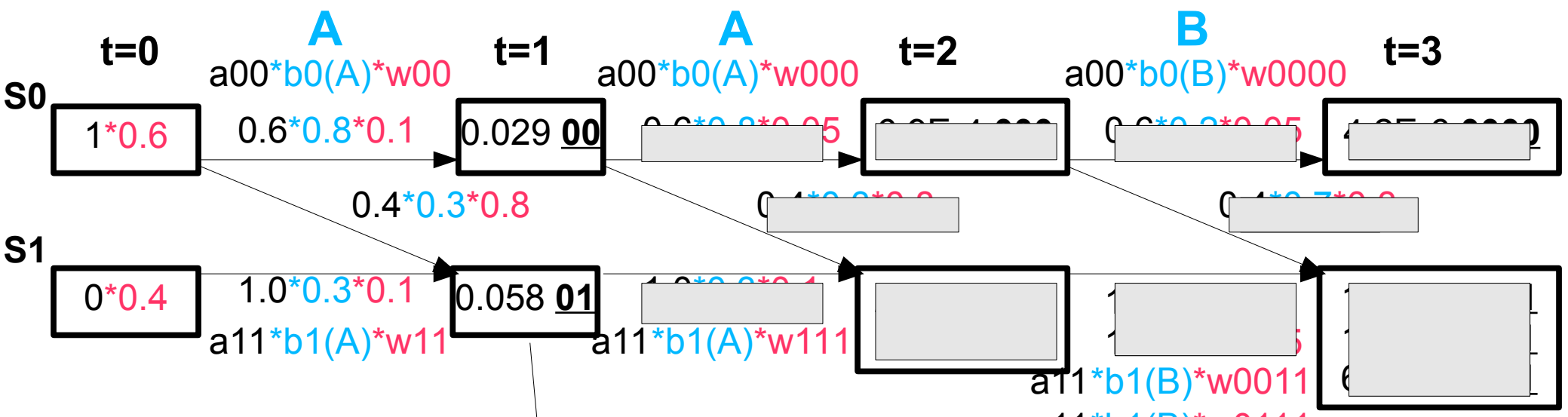
Exercise: Token passing



1-gr	prob	2-gr	prob	3-gr	prob
0	0.6	**00**	0.1	**000**	0.05
1	0.4	**01**	0.8	**001**	0.8
		11	0.1	**011**	0.1
				111	0.05



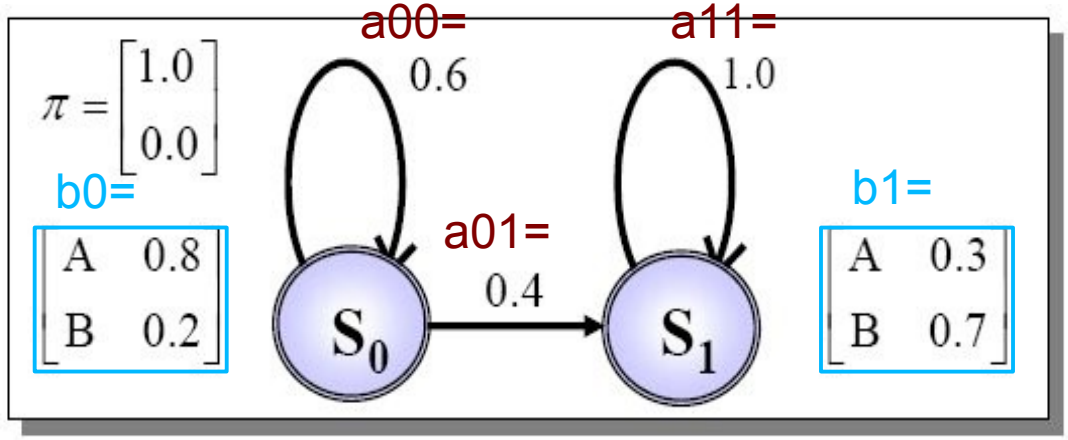
Exercise: Token passing



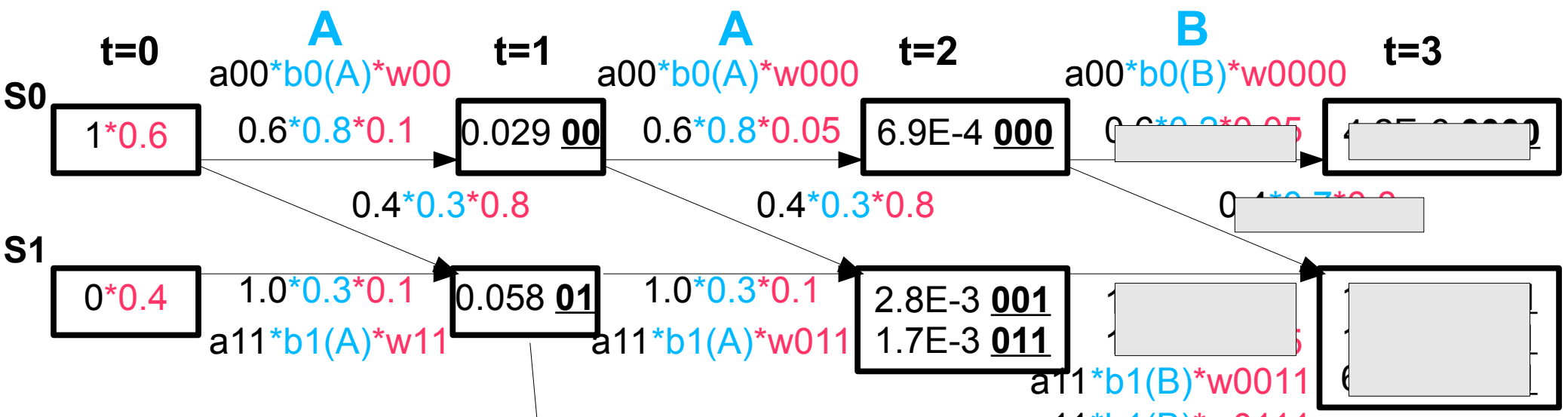
1-gr	prob	2-gr	prob	3-gr	prob
0	0.6	**00**	0.1	**000**	0.05
1	0.4	**01**	0.8	**001**	0.8
		11	0.1	**011**	0.1
				111	0.05

Note1: 6.9E-4 = 0.00069

Note2: 0.058 is max prob, 0.058 **01**
01 is state sequence in:



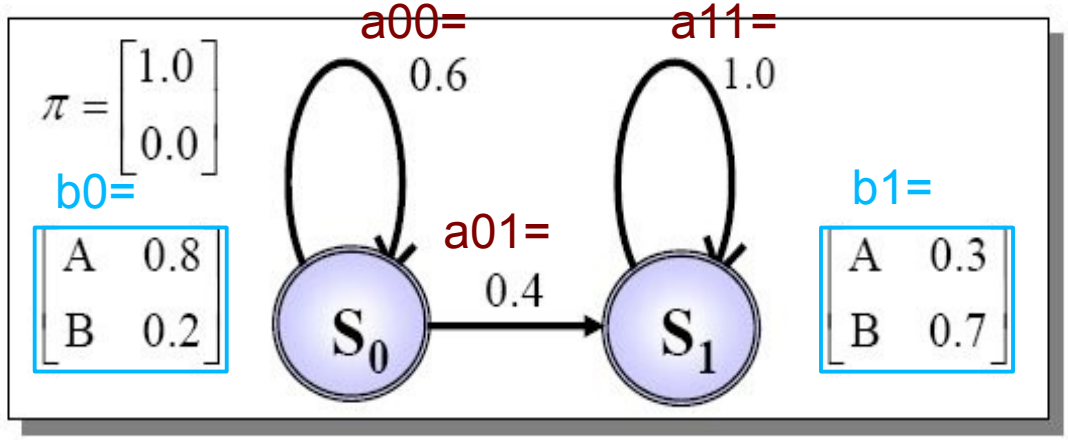
Exercise: Token passing



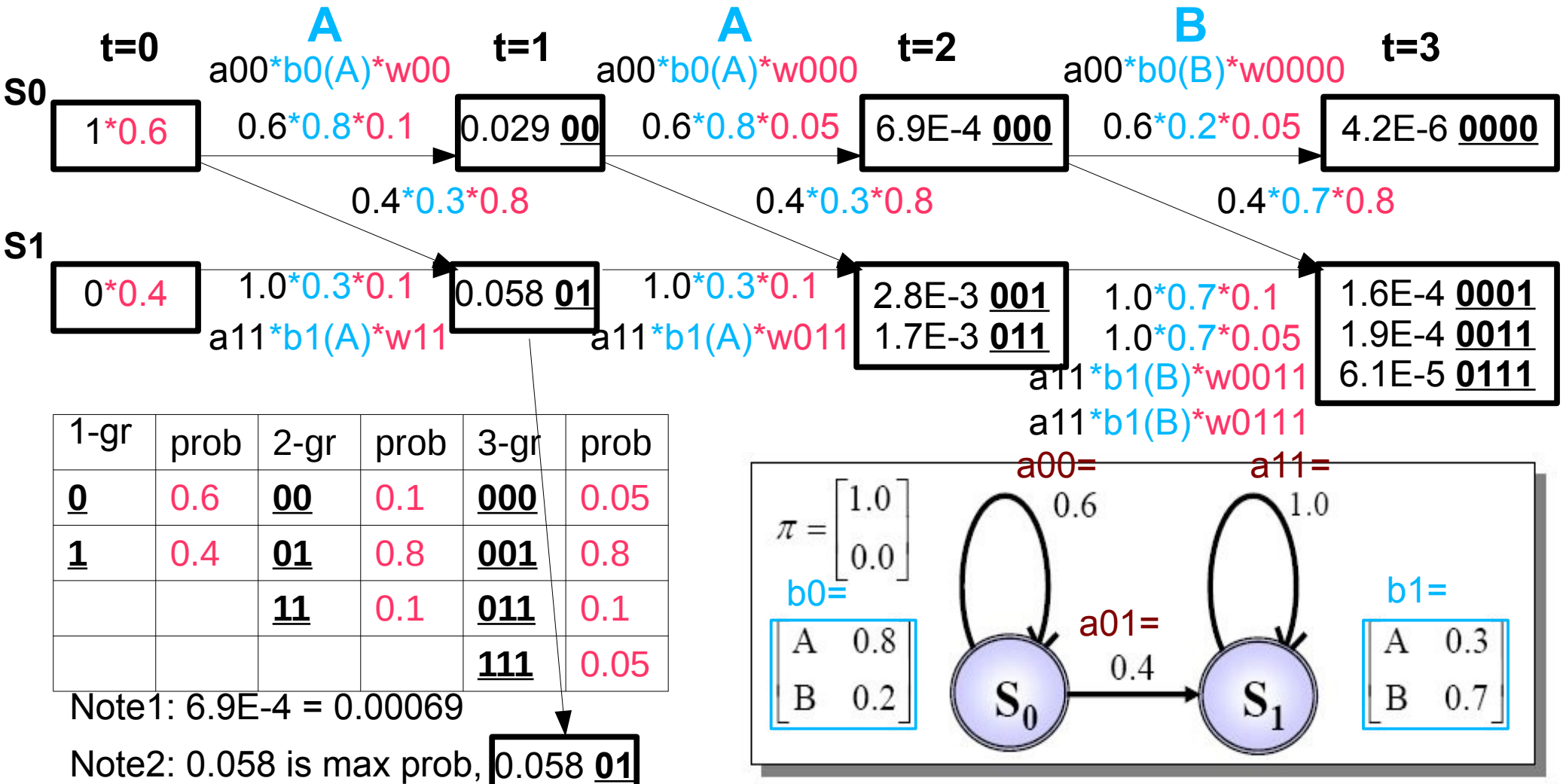
1-gr	prob	2-gr	prob	3-gr	prob
0	0.6	**00**	0.1	**000**	0.05
1	0.4	**01**	0.8	**001**	0.8
		11	0.1	**011**	0.1
				111	0.05

Note1: 6.9E-4 = 0.00069

Note2: 0.058 is max prob, 0.058 **01**
01 is state sequence in:



Exercise: Token passing

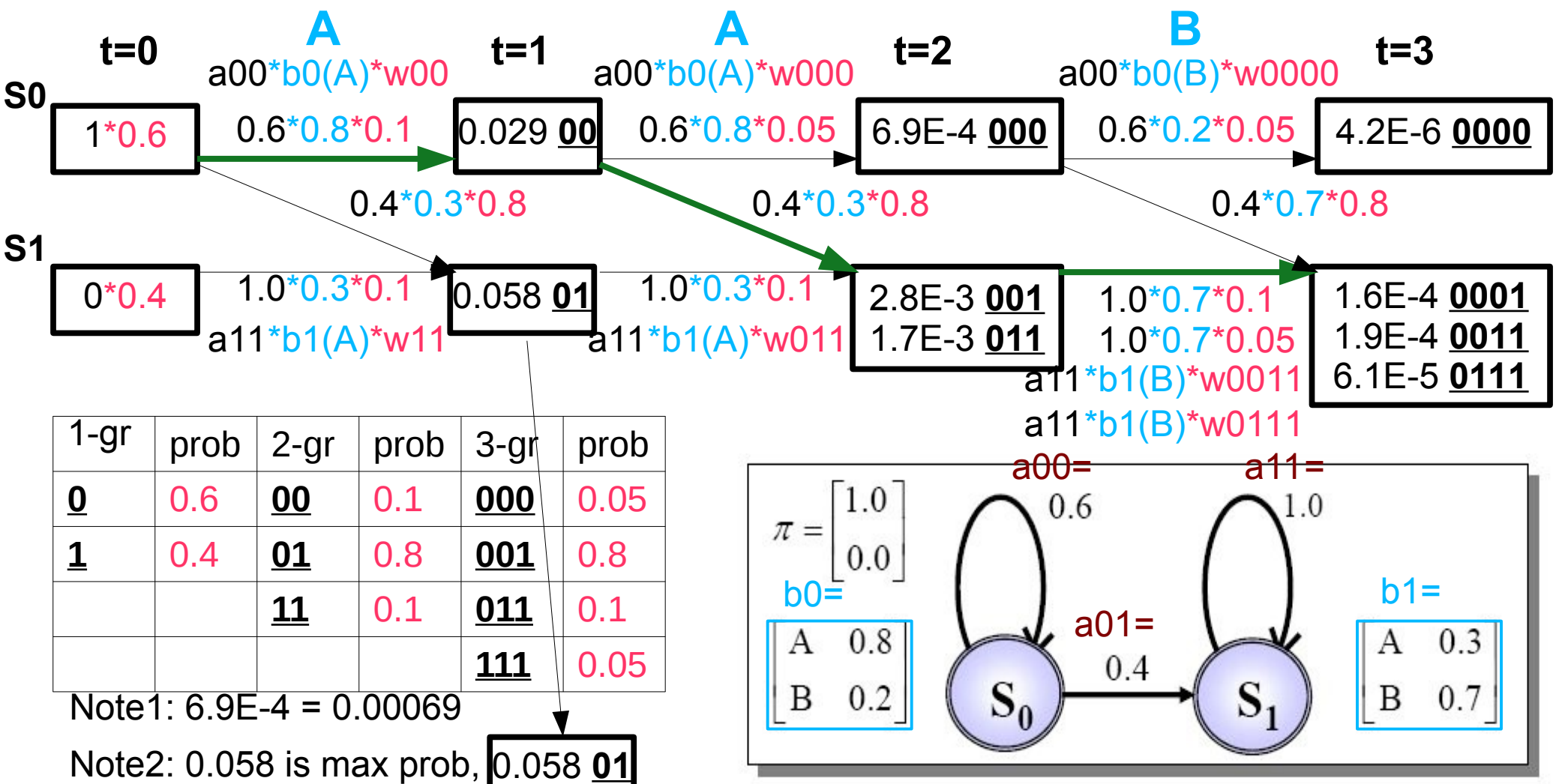


1-gr	prob	2-gr	prob	3-gr	prob
0	0.6	**00**	0.1	**000**	0.05
1	0.4	**01**	0.8	**001**	0.8
		11	0.1	**011**	0.1
				111	0.05

Note1: 6.9E-4 = 0.00069

Note2: 0.058 is max prob, 0.058 **01**
01 is state sequence in:

Exercise: Token passing



1-gr	prob	2-gr	prob	3-gr	prob
0	0.6	**00**	0.1	**000**	0.05
1	0.4	**01**	0.8	**001**	0.8
		11	0.1	**011**	0.1
				111	0.05

Note1: 6.9E-4 = 0.00069

Note2: 0.058 is max prob, 0.058 **01**
01 is state sequence in:

Token passing decoder in LVCSR

Proceed step-wise through the speech sample as in Viterbi.

Let each state carry several tokens to preserve different word histories.

The decoding process in each time step:

1. **Pass** tokens from all states to all successor states
2. **Update** the tokens for each state
3. **Prune** unlikely tokens
4. **Merge** tokens with they have identical N last words

Content today

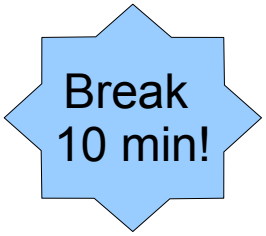
1. Recognition of continuous speech

2. A token passing decoder

3. **Measuring and tuning the performance**

4. Home exercise: Build a continuous speech recognition system

5. Status of project group works



Test what you remember from week 3

Individual test for everyone, now:

1. Go to ***https://kahoot.it*** with your phone/laptop
2. Type in the ID number you see on the screen (also in chat)
3. Give your **REAL (sur)name**
4. Answer the questions by selecting **only one** of the options
   - There may be several right (or wrong) answers, but just pick one
   - About 1 min time per question
5. 1 activity points for everyone + 0.2 per correct answer in time
   - Kahoot score is just for fun, only the correct answers matter

Optimizing the performance

- speed vs. error rate
- amount of pruning applied
- insertions vs. deletions
- weight of language models vs. acoustic models
- look ahead to guess the next active phonemes
- take a held-out development set to optimize, never your final test set!

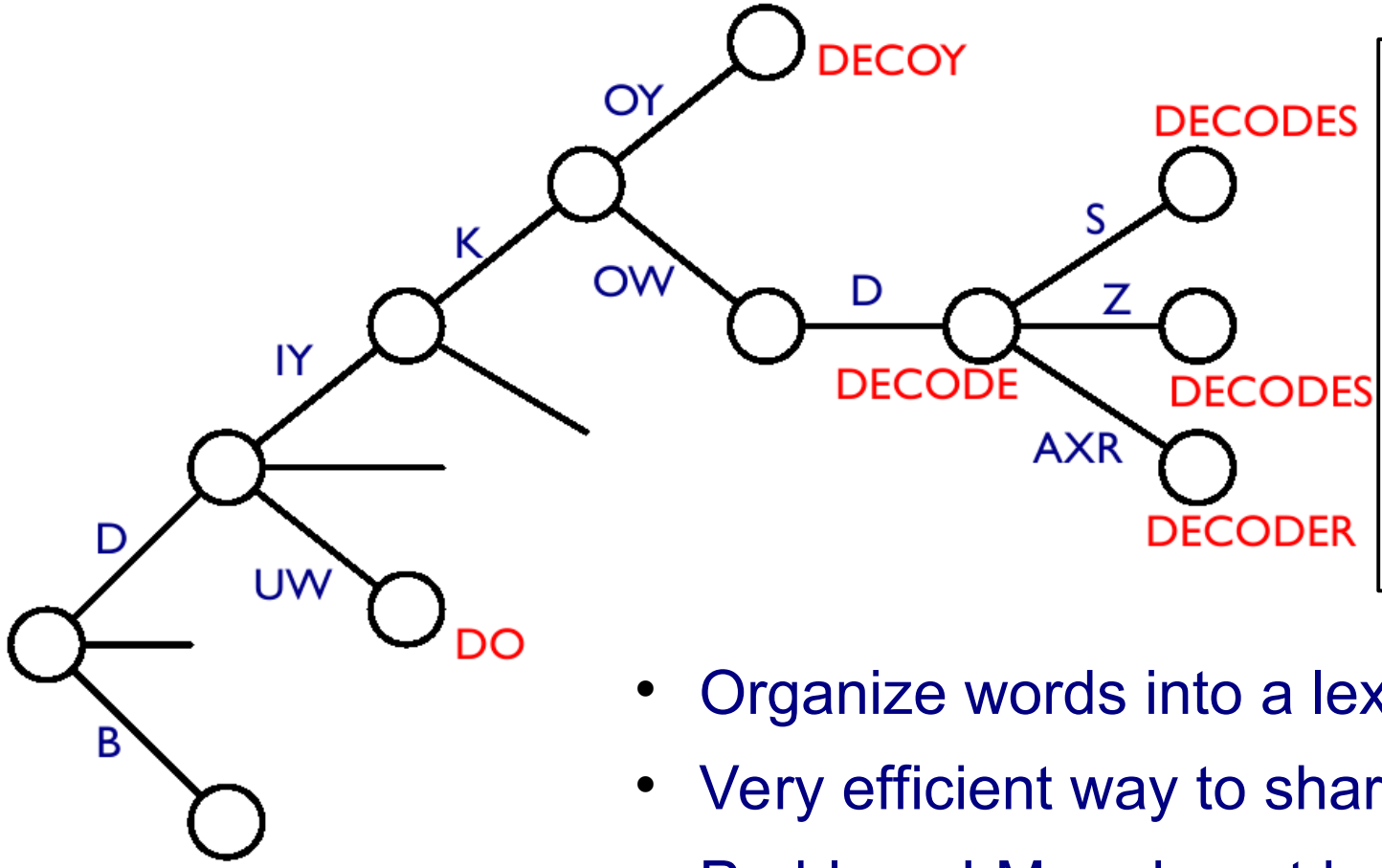
Key issues in large vocabulary search for continuous speech

- The more words, the more paths may start at every time frame
- Cross-word triphones increase the search space a lot
- How to prune paths most efficiently?
- How to efficiently incorporate long span LMs?

Recognition speed

- Discussion: Suggest ways to speed up recognition!
  - Faster acoustic probabilities (GMM, HMM)?
  - Faster lexicon and language models (n-gram)?
  - Faster decoding and search?
  - Which speed-ups affect accuracy?

An idea to share AM computations



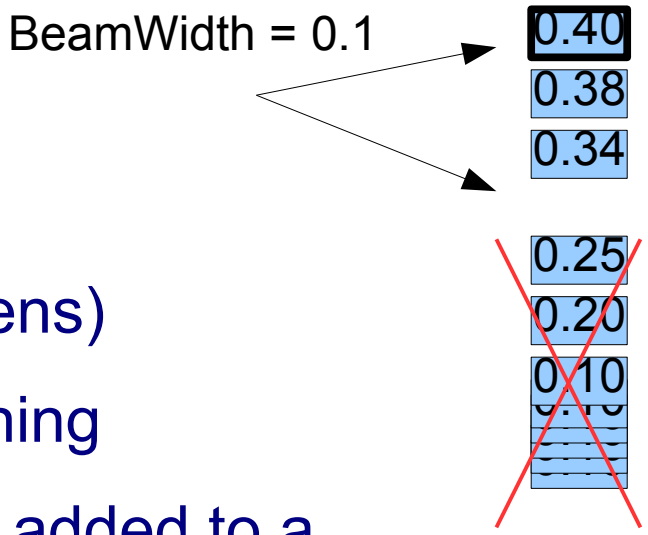
In a standard pronunciation dictionary all words have separate paths:
DECOY: D IY K OY
DECODE: D IY K OW D
DECODES: D IY K OW D S
DECODES: D IY K OW D Z
DECODER: D IY K OW D AXR
DO: D UW

- Organize words into a lexical prefix tree
- Very efficient way to share the computations
- Problem: LM prob. not known before the end
- Solution: Approx. tree by its most likely leave

Picture by S.Renals

Faster decoding by beam pruning

- At every time step prune away paths that are too far (outside the beam = *BW*) from the best one
  - global beam
  - word, phoneme and state beams
- Limit also the amount of active paths (tokens)
- Use both AM and LM probabilities for pruning
- LM can be computed when a new word is added to a hypothesis



Faster decoding by multipass search

- 1. pass: use coarse models to decode an **intermediate result**:
  - N-best list
  - word graph or lattice
- 2. pass: r**escore** the results using more detailed models
  - Higher order or larger N-grams or other LMs
  - Cross-word or longer context AMs
- 2nd pass is typically very fast (small search space)
- Intermediate results may take a lot of space

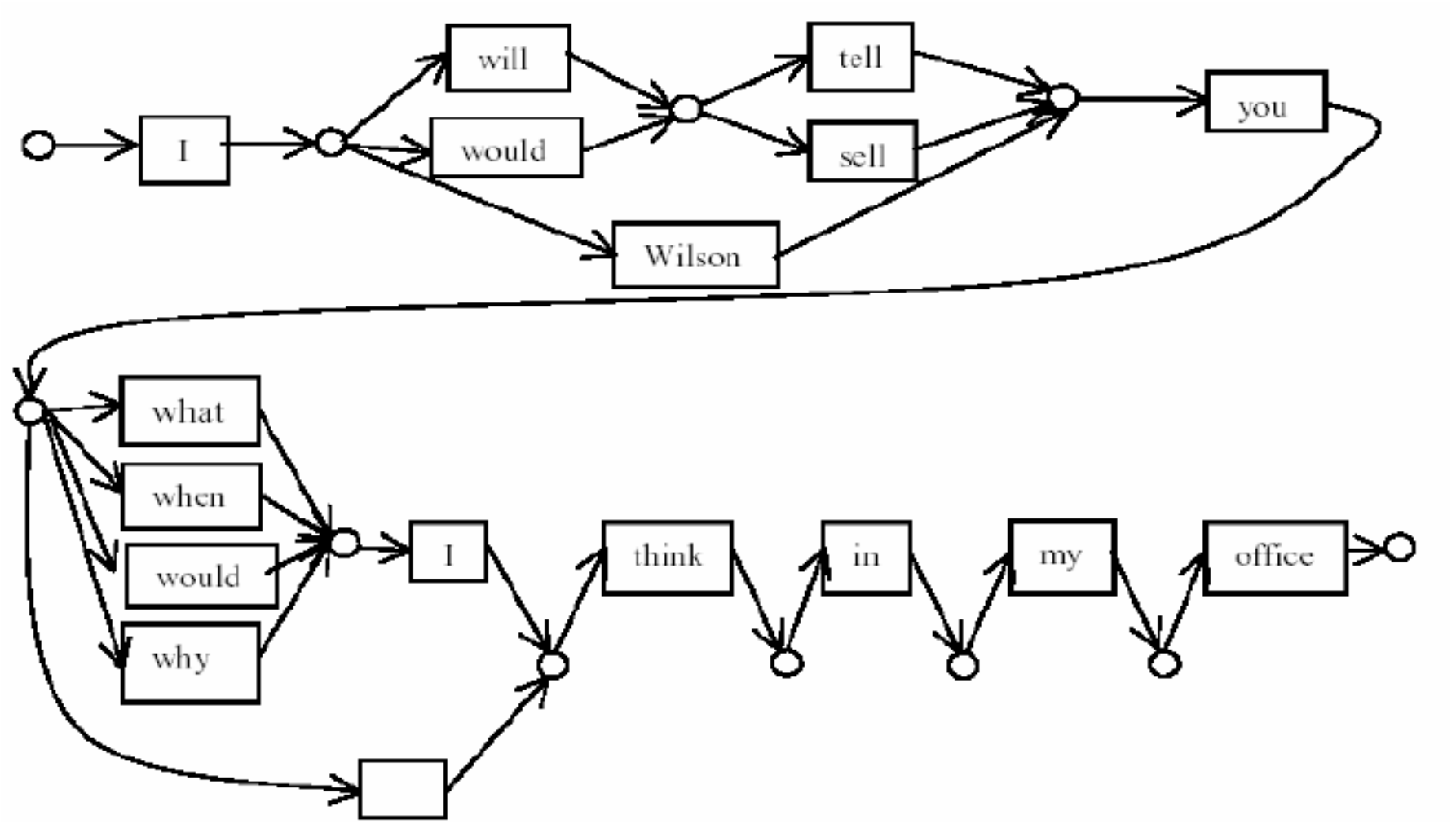
N-best lists

1. I will tell you would I think in my office
2. I will tell you what I think in my office
3. I will tell you when I think in my office
4. I would sell you would I think in my office
5. I would sell you what I think in my office
6. I would sell you when I think in my office
7. I will tell you would I think in my office
8. I will tell you why I think in my office
9. I will tell you what I think on my office
10. I Wilson you I think on my office

- easy to apply long span or bidirectional LMs for rescoring
- the differences are small
- not very compact representation
- Better to organize in a **lattice** or **word graph** structure that shows all good options

Picture by B.Pellom

Word graph representation



ASR tests

- Recognition **error rate**
  - requires speech data and the full ASR run
  - shows which LM improvements are relevant
  - solving confusable word sequences is important
- **Re-scoring** intermediate ASR results
  - list of best hypothesis or full “word lattices” with pre-computed acoustic probs
  - much faster than full ASR runs
  - errors in lattices can not be recovered

WER and LER

- **Letter Error Rate** = minimum number of edit operations needed to get the correct text
- Computed by dynamic programming
- Substitutions + Deletions + Insertions / No. letters
- May be over 100%, if ASR outputs many extra letters
- Likewise, **Word Error Rate** is the number of word errors

Home exercise 4

- Build a continuous speech recognition system and test it!
- Details, instructions and help given in Thursday and Friday meetings
- To be returned before the next Wednesday meeting

Summary of today

- The components of an ASR system, language models
- Decoding for LVCSR
- Testing and tuning the LVCSR system
- **on Thu/Fri:** Large vocabulary continuous speech recognition experiments, using SRILM and HTK
- **Next week:** End-to-end ASR

Feedback

Now: Go to **MyCourses > Lectures** and fill in the feedback for **Lecture4**.

Some of the feedback from the previous week:

+ Kahoot works great for recalling past topics

+ The lecture was quite easy to understand and follow

- The homework description are somewhat vague and hard to understand

- More visual presentations of neural networks

- time management (rush at the end)

Week1 ave time in Study: 3/50h, Exercise: 5/40h, Project: 3/40h (Max:15,10,8)
Week2 **cum** time in Study: **7**/50h, Exercise: **12**/40h, Project: **7**/40h (Max:5,10,6)

Thanks for all the valuable feedback!

Project work receipt

1. Form a group (3 persons)
2. Get a topic
3. Get reading material from M*ycourses* or your group tutor
4. 1st meeting: Specify the topic, start literature study
5. 2nd meeting: Write a work plan
6. **Perform analysis, experiments, and write a report**
7. **Book your presentation time for weeks 6 - 7 (DL Nov 18)**
8. Prepare and keep your 20 min presentation
9. Return the report (DL Dec 12)

This week

A doodle for presentations: https://doodle.com/meeting/participate/id/b4Qv1p0d

Allocation of project work presentations

	Talk 1	Talk 2	Talk 3	Talk 4
Wed 30 Nov 10-12,U6				
Fri 2 Dec 14-16, U5				
Wed 7 Dec 10-12, U9				
Fri 9 Dec 14-16, U5				

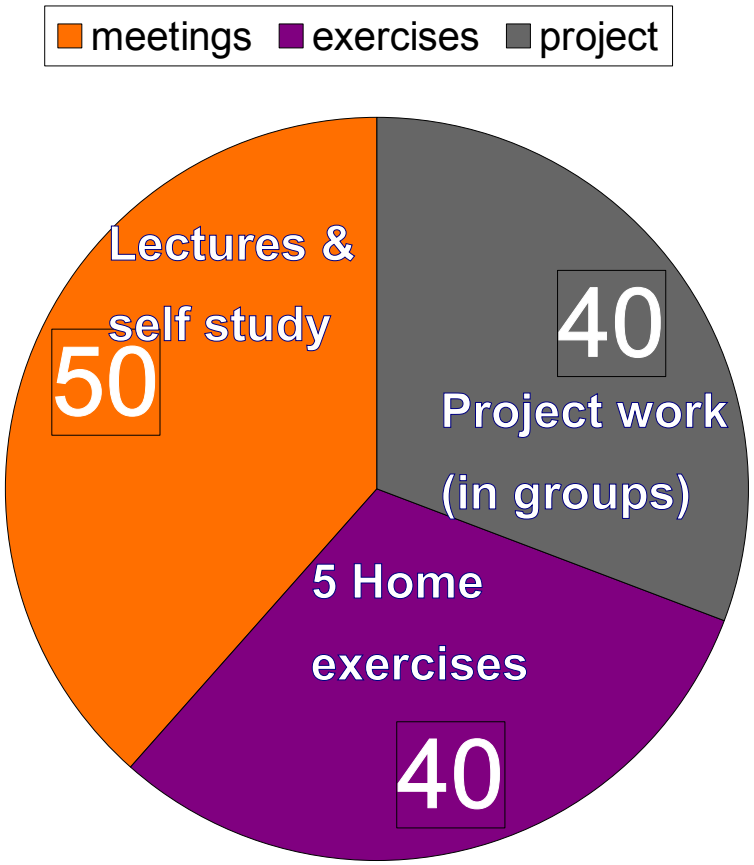
14 groups but 16 slots so some will be left unused.
30 min slots including 20min talk + questions + setup

Final report or even results do not have to be ready for the talk, just present what you have done so far

Fill in the Doodle by 18 Nov

Course Format

"Hour pie": 130h



"Grade pie": 1..5