Timeline in the course

	Meetings		Home exercises	Project work
	Wednesdays	**Thursdays**		**status**
Week1	Speech features and classification		1.Feature classifier	Literature study
Week2	Phoneme modeling and recognition		2.Word recognizer	Work plan
Week3	Lexicon and language modeling		3.Text predictor	Analysis
Week4	Continuous speech and advanced search		4.Speech recognizer	Experimentation
Week5	End-to-end ASR		5.End-to-end recognizer	Preparing reports
Week6	Projects1	Projects2		Presentations
Week7	Projects3	Projects4		Report submissior
		Conclusion		

Learning goals for this week

➡ **1. Phonemes, HMM**

   - remember from last week

2. Vocabulary

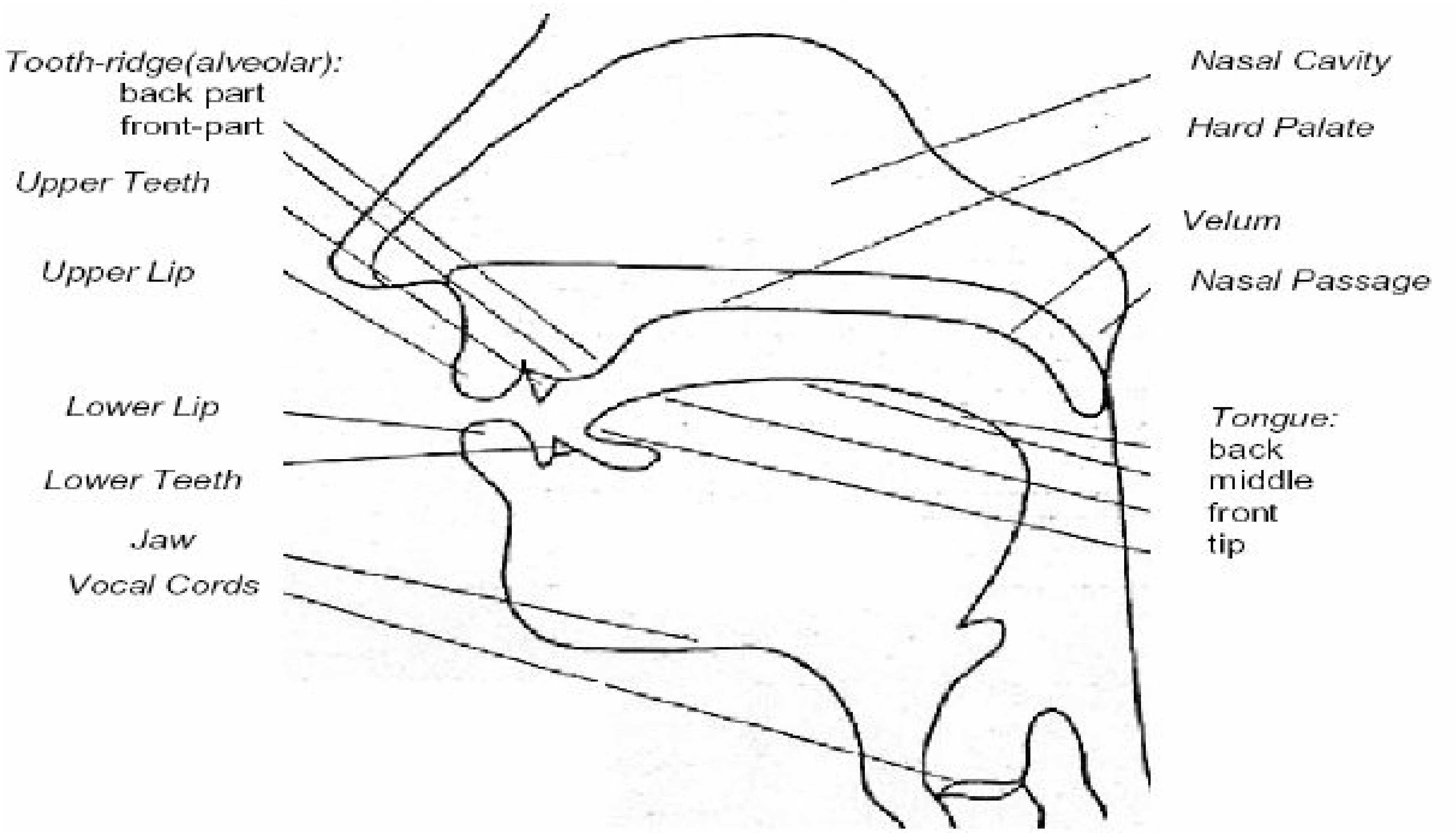
   - know how to compose the recognition lexicon

3. Statistical language model (LM)

   - know how to construct statistical LMs
   - know pros and cons of statistical LMs

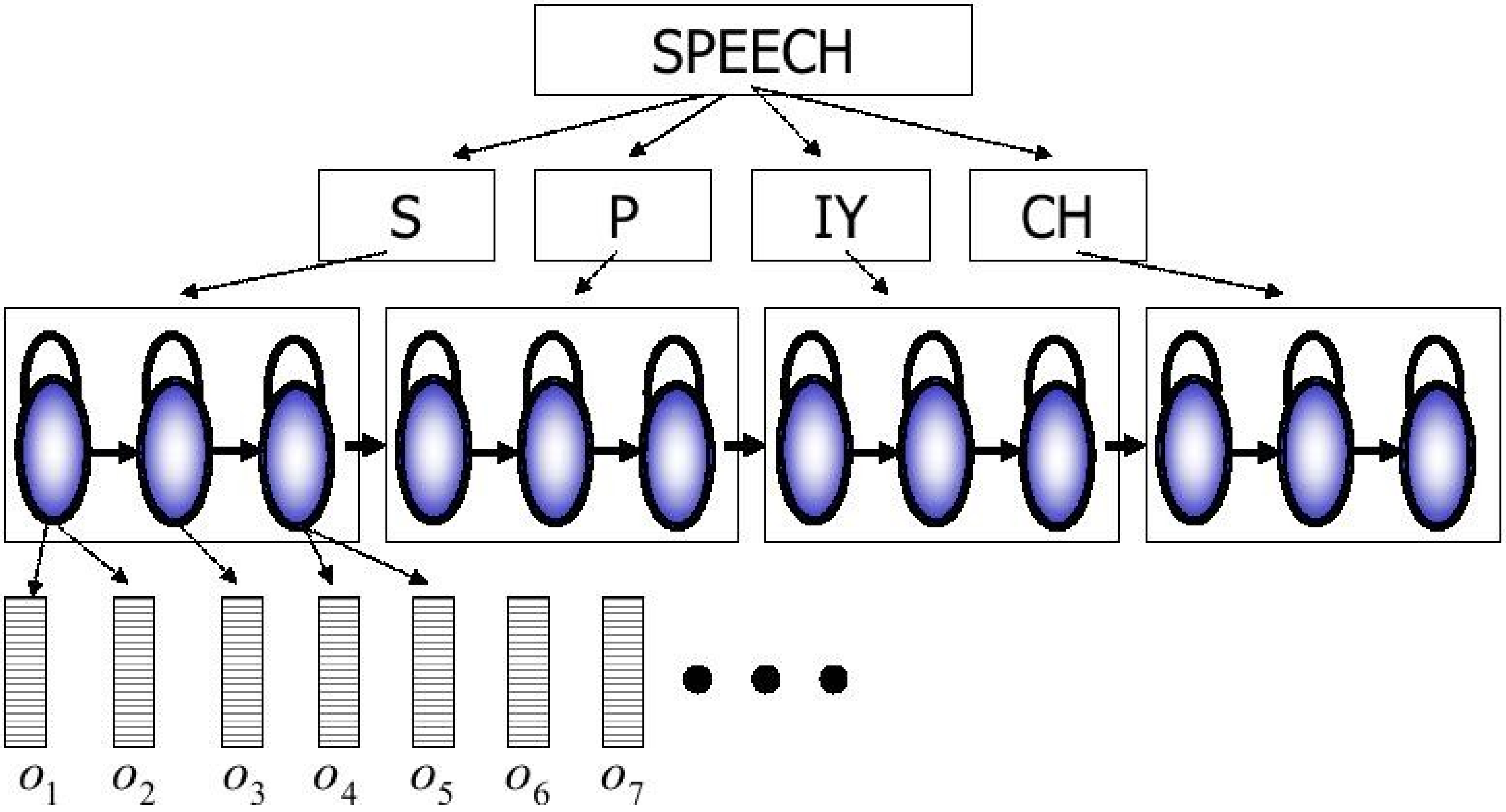
4. Neural network language model (NNLM)

Review: Production of speech sounds



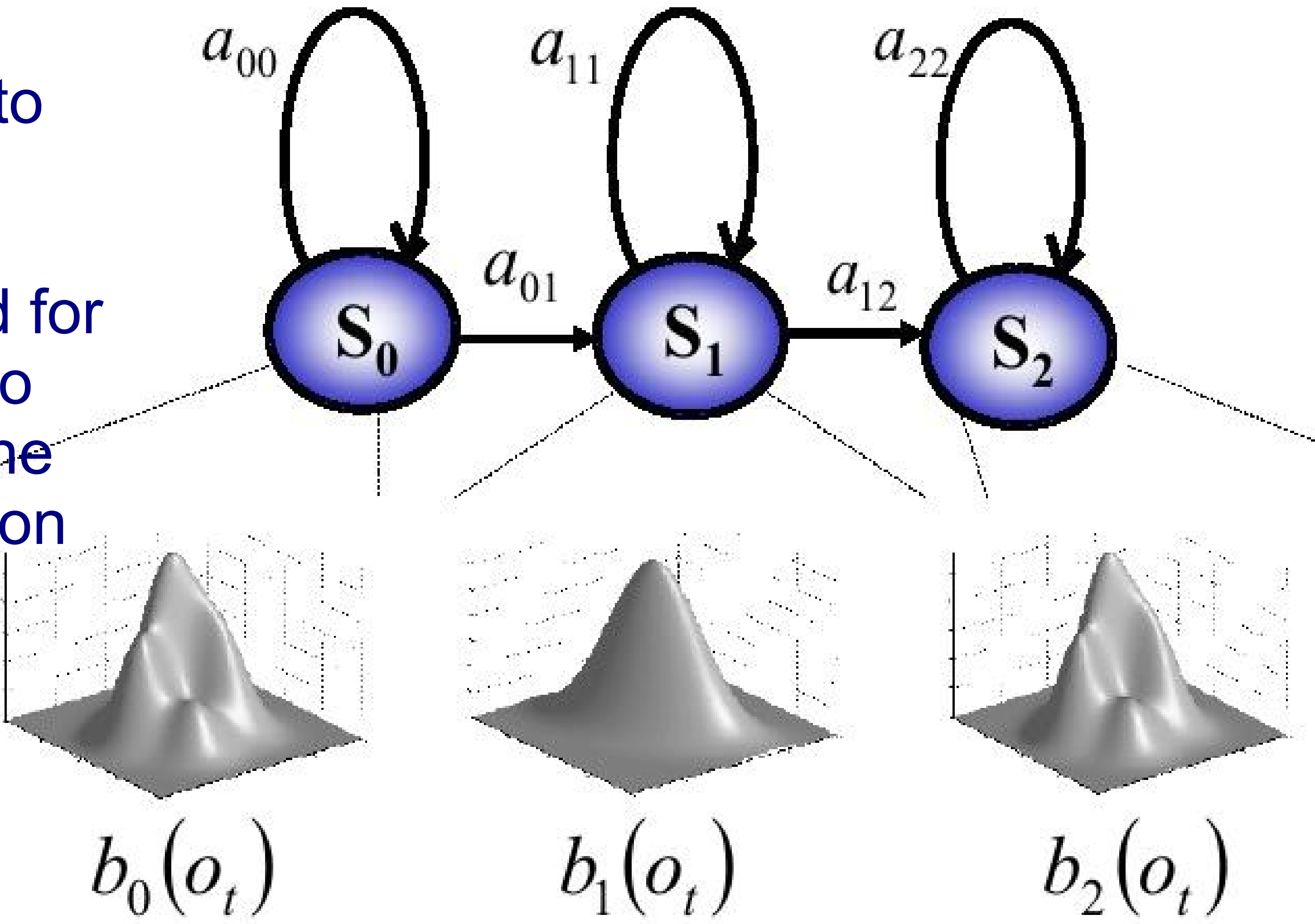
Picture from Huang's text book (2001)

Review: HMM as a phoneme model



Review: GMM-HMM system

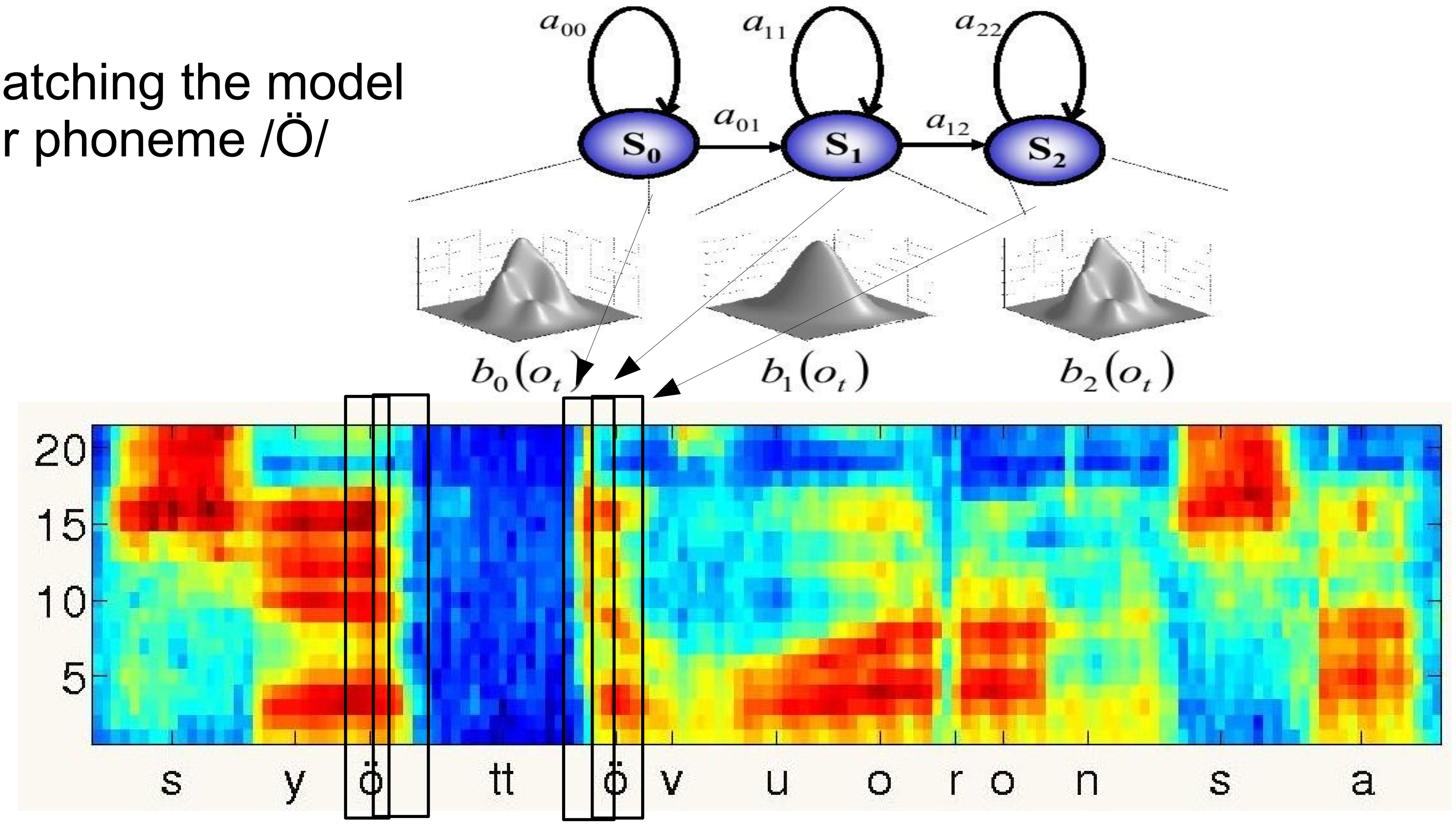
- Each state emits sounds according to its GMM model
- This generative model can be used for **text-to-speech**, too
- The higher *a(ii)*, the longer is the duration



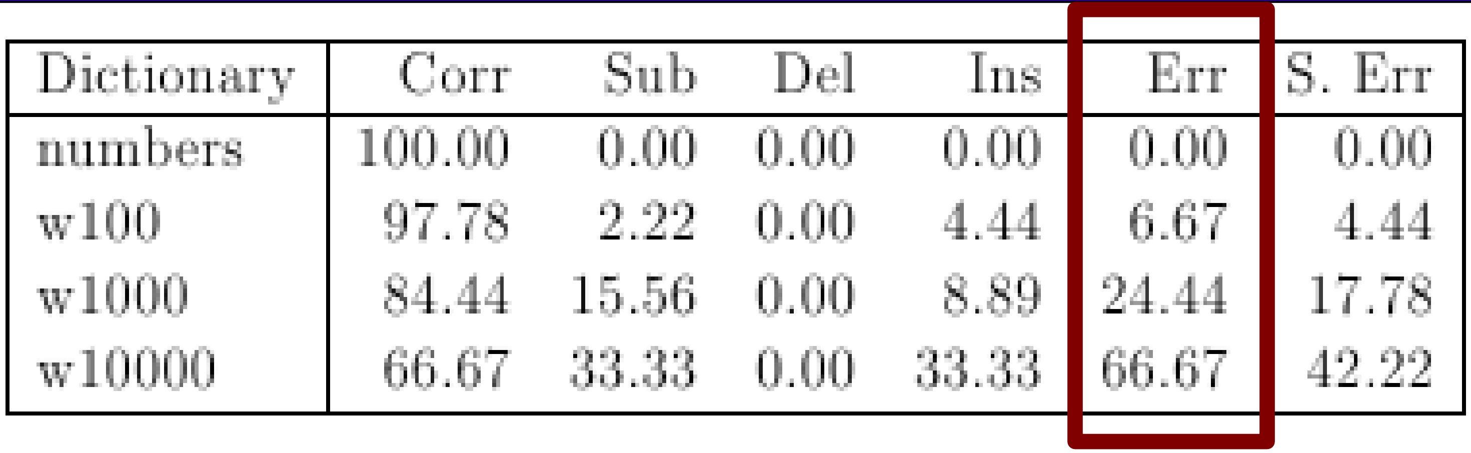
Picture by B.Pellom

Review: An example of a GMM-HMM system

Matching the model for phoneme /Ö/



Result of isolated word recognition?



Dictionary	Corr	Sub	Del	Ins	Err	S. Err
numbers	100.00	0.00	0.00	0.00	0.00	0.00
w100	97.78	2.22	0.00	4.44	6.67	4.44
w1000	84.44	15.56	0.00	8.89	24.44	17.78
w10000	66.67	33.33	0.00	33.33	66.67	42.22

Taulukko 1: Word error rates using different dictionaries

- Rapid increase of errors for large vocabulary
- Real speech: (tens/hundreds) thousands of words...
- Continuous speech: much more difficult, because the words are glued together

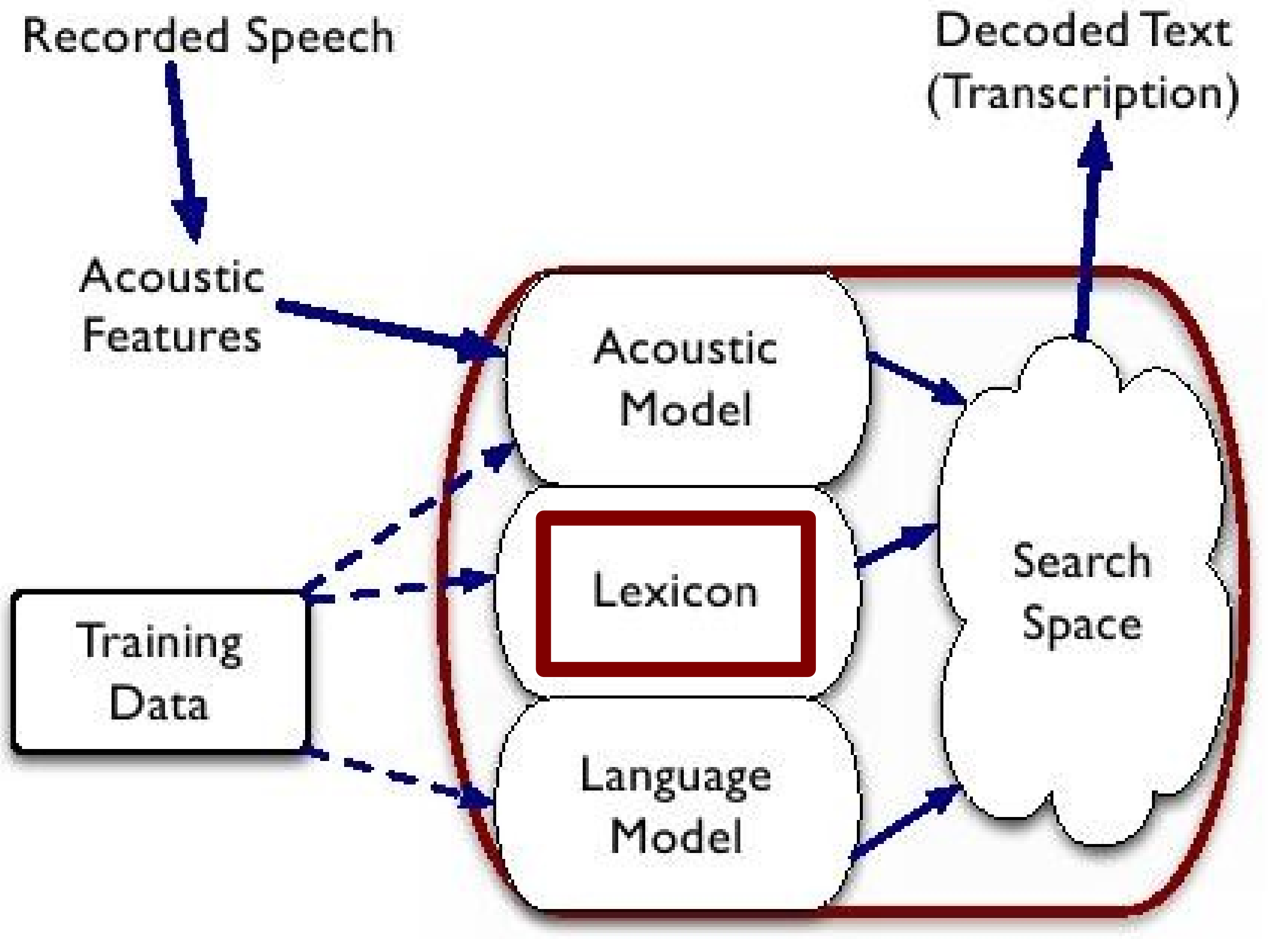
Content this week

1. Phonemes, HMM
2. **Vocabulary**
3. Statistical (n-gram) language model
4. Neural network language model (NNLM)
5. Home exercise: (3) Build a language model for recognition of continuous speech!
6. Status of project group works

What is speech recognition?

- **Find the most likely word or word sequence given the acoustic signal and our models!**
- **Language model** defines words and how likely they occur together
- **Lexicon** defines the word set and how the words are formed from sound units
- **Acoustic model** defines the sound units independent of speaker and recording conditions

Vocabulary = Lexicon



Small vocabulary

- Only listed words will appear in the task
- Only listed words will be recognized, others will always cause errors!
- Applications
  - Number dialling, name dialling
  - Command and control interfaces
  - Menu based services
- Prior probabilities can be added

one
two
three
four
five
six
seven
eight
nine
zero

Pronunciation

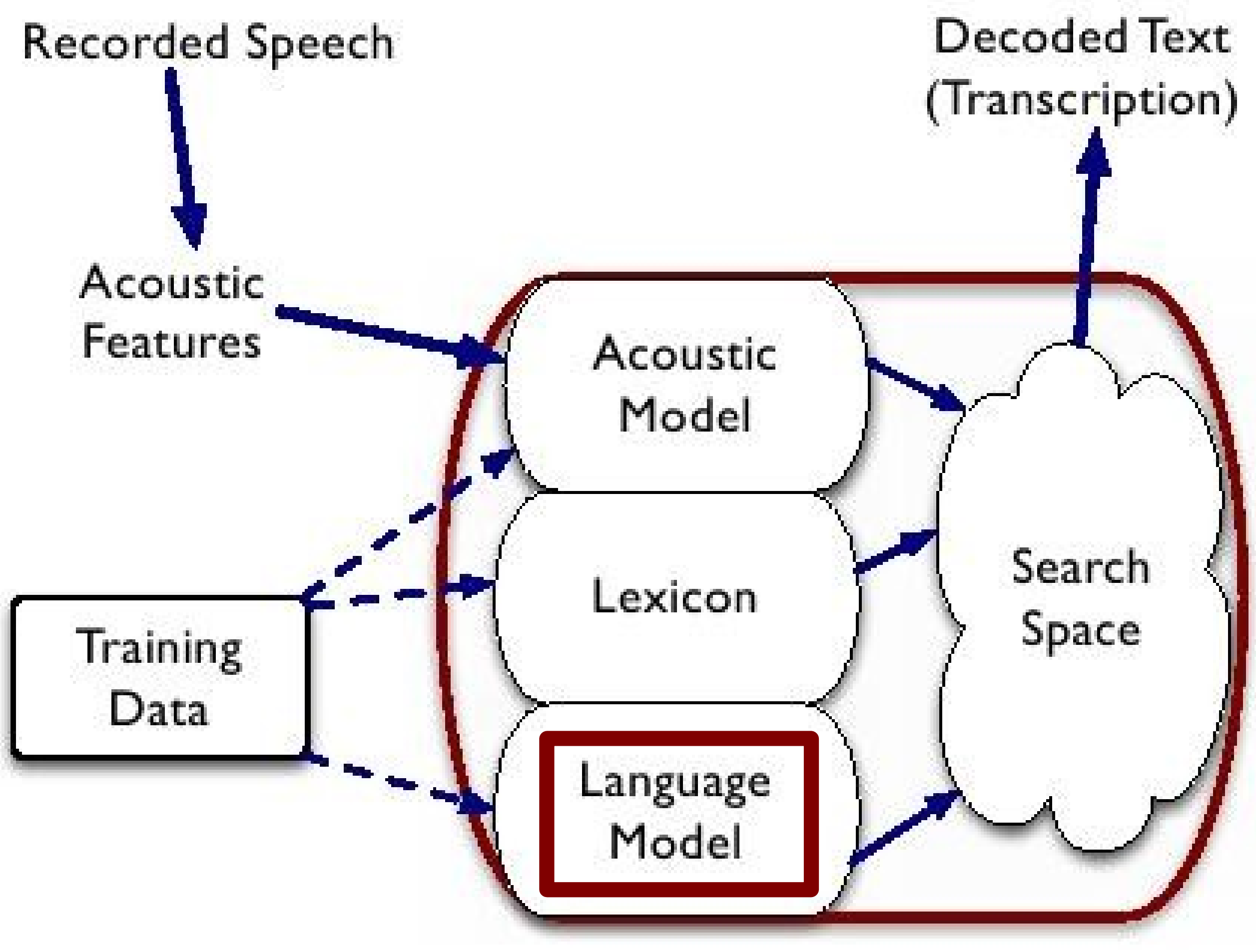
- A **lexicon** or pronunciation dictionary tells how words are pronounced
- Each word is described as a sequence of phonemes (or triphones)
- Problems to think about:

1. *One word may have several pronunciations (with priors)*, does it matter?
2. *Several words may have the same pronunciation,* does it matter?
3. *How to get pronunciations for new words?*
4. *Adding rare words or pronunciations decreases ASR performance.* Why?

one	w ah n
two	t uw
three	th r iy
tomato(0.5)	t ax m ey t ow
tomato(0.5)	t ax m aa t ow
too	t uw

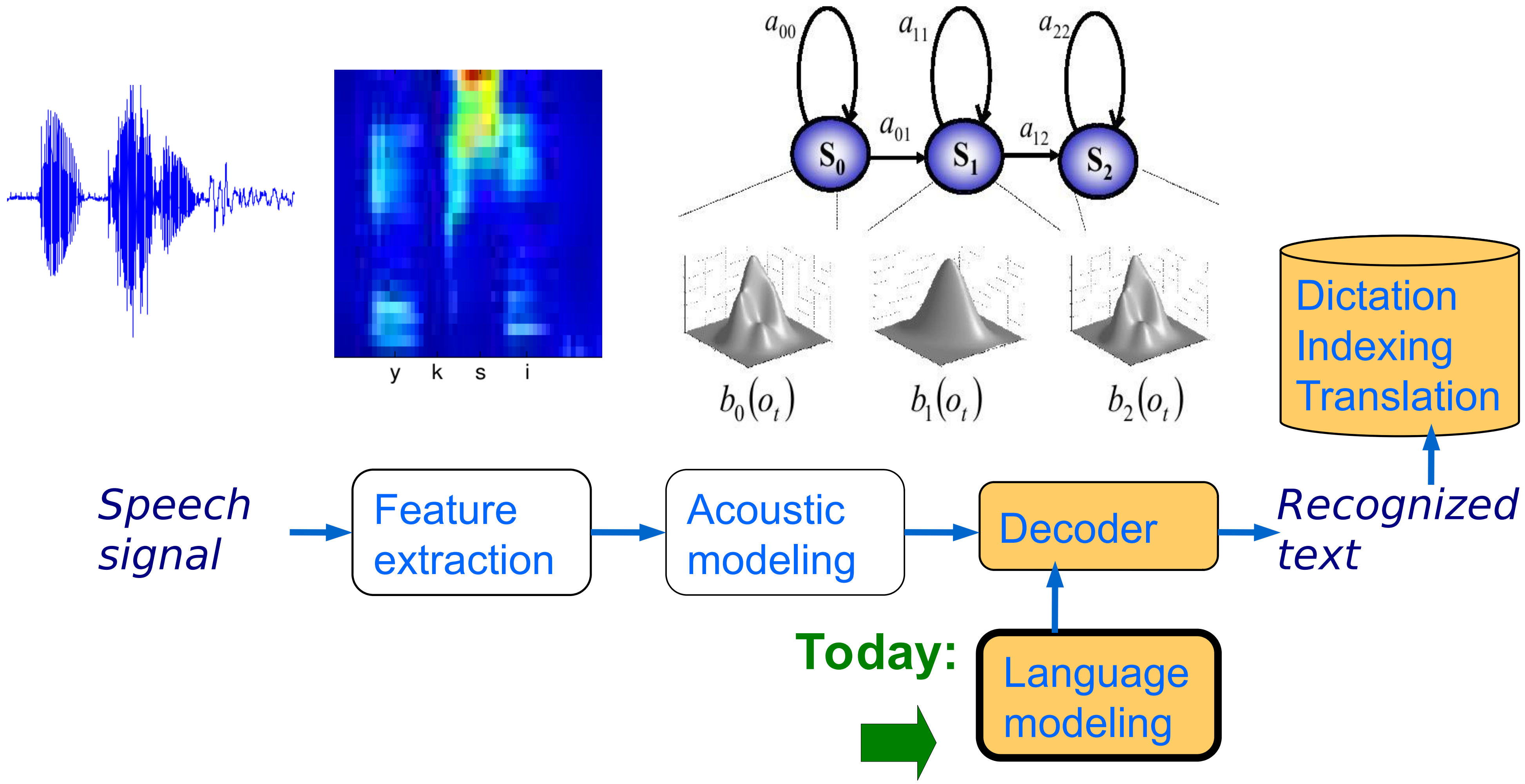
Content today

Language model



Picture by S.Renals

Speech recognition -from beginning to end



What is speech recognition?

- **Find the most likely word or word sequence given the acoustic signal and our models!**
- **Language model** defines words and how likely they occur together
- **Lexicon** defines the word set and how the words are formed from sound units
- **Acoustic model** defines the sound units independent of speaker and recording conditions

Language model

- Assigns a prior probability to word sequences
- Reduces search space and ambiguity
- Resolve homonymes:
  - `Write a letter to Mr. Wright right away`
- Power vs. flexibility
- A good review and comparison of the latest methods:
  - *“A bit of progress in language modeling”,* extended version (2001) by Joshua T. Goodman
  - *www.research.microsoft.com/~joshuago/longcombine.pdf*

When humans fail: popular misheard lyrics

- "**Gladly, the cross-eyed bear.**" /"Gladly The Cross I'd Bear." Traditional Hymn
- "**There's a bathroom on the right.**"/"There's a bad moon on the rise." Bad Moon Rising, Creedence Clearwater
- "**Excuse me while I kiss this guy.**"/"Excuse me while I kiss the sky." Purple Haze, Jimi Hendrix
- "**Dead ants are my friends; they're blowin' in the wind.**"/"The answer my friend is blowin' in the wind." Blowin' In The Wind, Bob Dylan
- "**The girl with colitis goes by.**"/"The girl with kaleidoscope eyes." Lucy in the Sky With Diamonds, The Beatles
- "**She's got a chicken to ride.**"/"She's got a ticket to ride." Ticket to Ride, The Beatles
- "**Are you going to starve an old friend?**"/"Are you going to Scarborough Fair?" Scarborough Fair, Simon and Garfunkel
- "**What a nice surprise when you're out of ice.**"/"What a nice surprise bring your alibis." Hotel California, Eagles
- "**Hope the city voted for you.**"/"Hopelessly devoted to you." Hopelessly Devoted to You, Grease
- "**I'm a pool hall ace.**"/"My poor heart aches." Every Step You Take, The Police

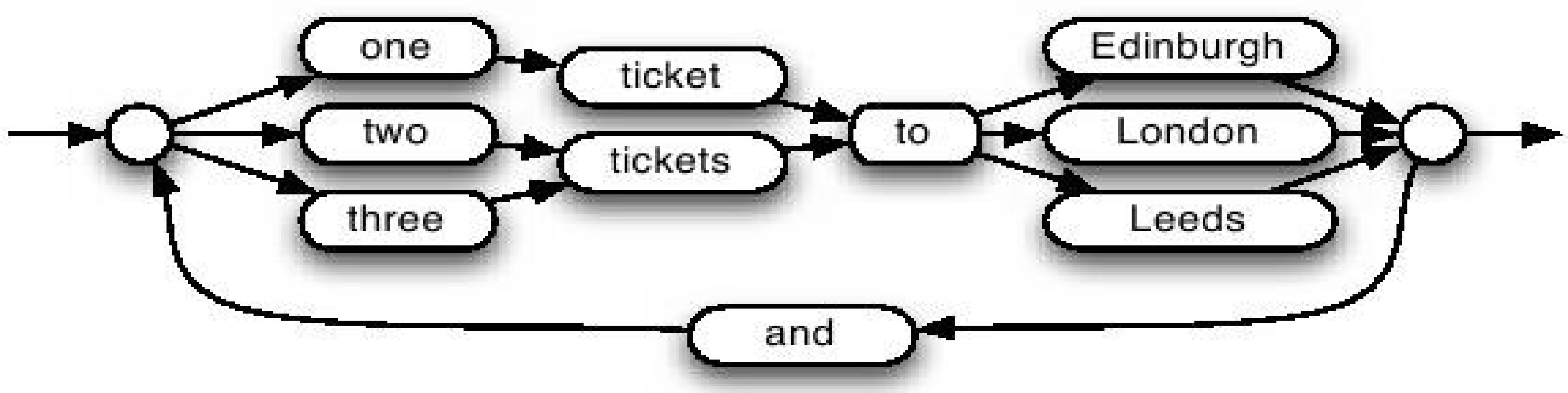
Why humans fail? Suggestions?

Examples from: http://www.fun-with-words.com/

Applications of Statistical LMs

1. Spelling correction, text input
2. Optical character recognition, e.g. scanning old books
3. Automatic speech recognition
4. Statistical machine translation
5. Information retrieval
6. Text-to-speech
7. ...(Can you think of any other? – Suggest!)
8. ...

Simple finite-state network grammar



- Limited domain models, constructed by hand
- Only a limited set of sentences are recognized
- Significant reduction of the recognition task

Picture by S.Renals

HTK example: LM of spoken travel phrases

```
$GENPLACE = ( ( railway station ) | ( hotel ) |  the bus station ) | ( the airport ) );
$GEOPLACE =  ( london ) | ( brussels ) | ( tokyo ) | ( beijing ) | ( helsinki );
$FOOD =  ( chicken ) | ( beef ) | ( fish ) | ( ham ) | ( cheese ) | ( eggs ) | ( salad );
$DRINK = ( coffee ) | ( tea ) | ( juice ) | ( water ) | ( beer ) | ( whiskey ) | ( vodka );
( STARTSIL (
( how much is a ticket to $GEOPLACE ) |
( how do i get to ( $GENPLACE | $GEOPLACE ) |
( could i have [ some ] ( $FOOD | $DRINK ) [ please ] ) |
( may i have a ( glass | cup | bottle ) of $DRINK ) |
( a glass of $DRINK [ please ] )
) ENDSIL )
```

HTK example: LM of spoken travel phrases

EMIME project (2010): https://www.youtube.com/watch?v=wqv7uYAyAQ0

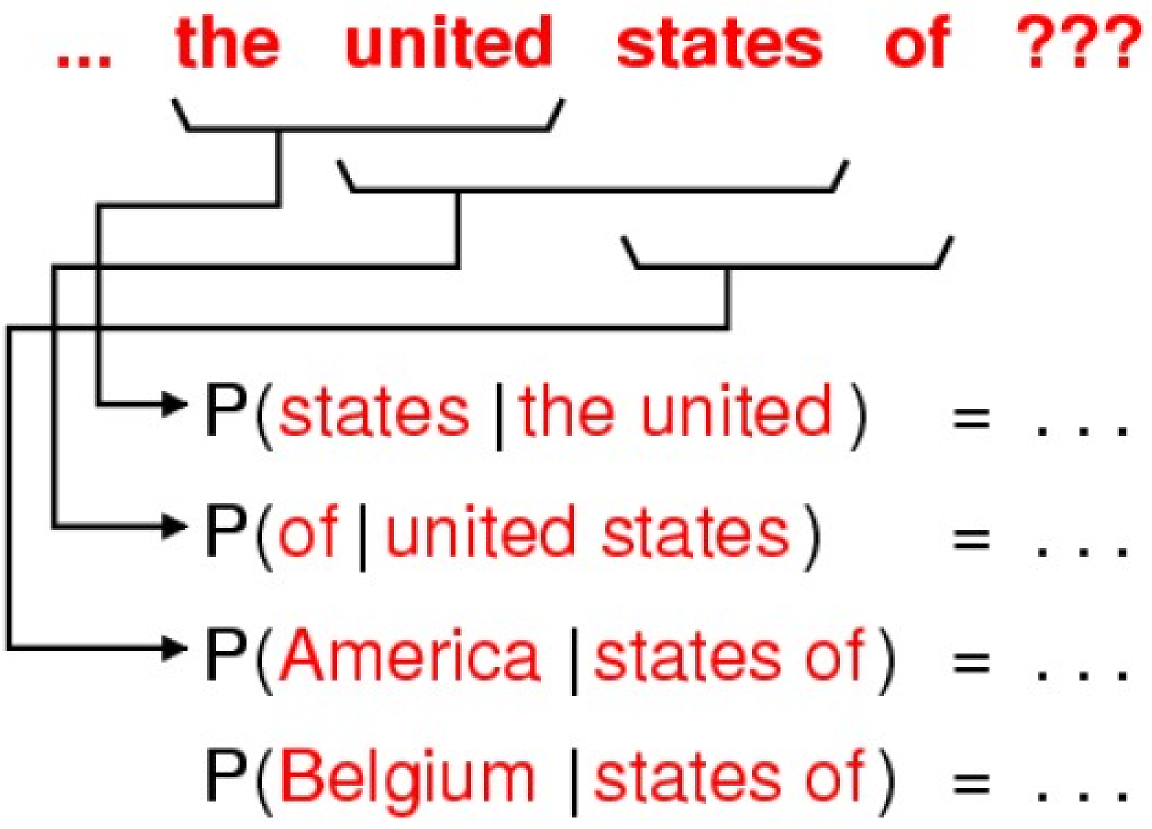
```
$PAIKKAAN = Kyotoon | Hokkaidoon | (Lontooseen) | (Brysseliin) |
(Edinburghiin) | (Tokioon) | (Pekingiin) | (Helsinkiin);

$RUOKAA = (kanaa) | (naudanlihaa) | (kalaa) | (kinkkua) | (makkaraa) |
(juustoa) | (munia) | (salaattia) | (vihanneksia);

$JUOMAA = (kahvia) | (teetä) | (mehua) | (vissyä) | ( vissy vettä )| (vettä) |
(olutta) | (punaviiniä) | (valkoviiniä) | (viskiä) | (vodkaa) | (rommia);

(STARTSIL(
(paljonko maksaa lippu $PAIKKAAN ) |
(miten pääsen ($GEOPAIKKAAN)) |
(saisinko  ($RUOKAA | $JUOMAA) [kiitos]) |
(Saisinko (lasillisen | kupillisen | pullollisen) $JUOMAA) |
(lasillinen $JUOMAA [kiitos])
) ENDSIL )
```

N-gram language model

- N can be 1,2,3,4,...
- Generative model which can be used to produce synthetic sentences
- Statistical, scalable, can deal with ungrammatical sequences
- Suitable for left-to-right search
- Suits well for languages of rigid word order

N-gram models

- E.g. trigram = 3-gram:
- Word occurrence depends only on its immediate short context
- A conditional probability of word given its context
- Estimated from a large text corpus (count the contexts!)



... the united states of ???
P(states | the united) = ...
P(of | united states) = ...
P(America | states of) = ...
P(Belgium | states of) = ...

Estimation of N-gram model

$$P(w_i \mid w_j) = \frac{c(w_j, w_i)}{c(w_j)}$$

$$\frac{c(\text{“eggplant stew”})}{c(\text{“eggplant”})}$$

- Bigram example:
  - Start from a maximum likelihood estimate
  - probability of *P(“stew” | “eggplant”)* is computed from **counts** of *“eggplant stew”* and *“eggplant”*
  - works well only for frequent bigrams
    - Why not for rare bigrams?

Zero probability problem

- If an N-gram is not seen in the corpus, it will get probability = 0
- The higher N, the sparser data, and the more zero counts there will be
- 20K words => 400M 2-grams => 8000G 3-grams, so even a gigaword corpus has MANY zero counts!
- Smoothing: Redistribute some probability mass from seen N-grams to unseen ones

Smoothing methods

1. **Add-one**: Add 1 to each count and normalize => gives too much probability to unseen N-grams
2. **Absolute discounting**: Subtract a constant from all counts and redistribute this to unseen ones using N-1 gram probs and back-off (normalization) weights
3. Best: **Kneser-Ney smoothing**: Instead of the number of occurrences, weigh the back-offs by the **number of contexts** the word appears in
4. Instead of only back-off cases, **interpolate** all N-gram counts with N-1 counts

Estimation of N-gram model

$$P(w_i \mid w_j) = \frac{c(w_j, w_i)}{c(w_j)}$$

$$\frac{c(\text{“eggplant stew”})}{c(\text{“eggplant”})}$$

- Bigram example:
  - Start from a maximum likelihood estimate
  - probability of *P(“stew” | “eggplant”)* is computed from **counts** of *“eggplant stew”* and *“eggplant”*
  - works well only for frequent bigrams
    - Why not for good rare bigrams?

Backing off

$$
\begin{aligned}
P(w_i \mid w_j) &= \frac{c(w_j, w_i)}{c(w_j)} && \text{if } c(w_j, w_i) > c \\
&= P(w_i) b_{w_j} && otherwise
\end{aligned}
$$

- Divide the room of rare bigrams, e.g. *"eggplant francisco"*, in proportion to the unigram ***P("francisco")***
- The sum of all these rare bigrams *"eggplant [word j]"* is ***b("eggplant")*** which is called the **back-off weight**

Absolute discounting and backing off

$$P(w_i \mid w_j) = \frac{c(w_j, w_i) - D}{c(w_j)} \quad \text{if } c(w_j, w_i) > c$$
$$= P(w_i) b_{w_j} \qquad \textit{otherwise}$$

- If bigram is common: Subtract constant *D* from the count
- If not: Back off to the unigram probability normalized by the back-off weight
- Similarly back off all rare N-grams to N-1 grams

Kneser-Ney smoothing

$$P(w_i \mid w_j) = \frac{c(w_j, w_i) - D}{c(w_j)} \quad \text{if } c(w_j, w_i) > c$$
$$= \mathbf{V}(w_i) b_{w_j} \qquad \textit{otherwise}$$

- Instead of the number of occurrences, weigh the back-offs by the **number of contexts** ***V****(word)* the word appears in:
  - In this case the context is the previous word: how many **different** previous words the corpus has for each word
  - E.g. *P(Stew | EggPlant)* is high, because stew occurs in many contexts
  - But *P(Francisco | EggPlant)* is low, because Francisco is common, but only in “San Francisco”

Picture by B.Pellom

Smoothing by interpolation

$$P(w_i \mid w_j) = \frac{c(w_j, w_i) - D}{c(w_j)}$$

$$+ \; P(w_i) b_{w_j}$$

- Like backing off, but always compute the probability as a **linear combination** (weighted average) with lower order (N-1)gram probabilities
- Improves the probabilities of rare N-grams
- Discounts (D) (and interpolation weights) can be separately optimized for each N using a held-out data

Weaknesses of n-gram models ?

The French boy lived in the capital city, __? (Paris)

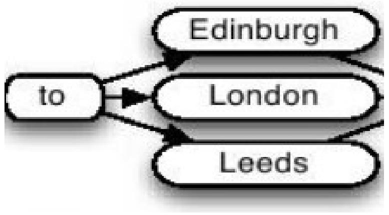
1. Only a fixed length word history is used
   - no long range dependencies
   - depends on word order
2. Probabilities are assigned to fixed sequences of words, can only generalize via smoothing or fixed word classes
   - no latent variables, distributed features or vector space
3. Large-vocabulary and long-span representations are inefficient and unreliable, because most long word sequences are rare
4. Skip n-grams are rarely used, because they would make decoding computationally hard

Testing the language model ?

1. Compute the **log-likelihood** of the words and sentences
2. **Perplexity**, the average number of word choices
3. **Entropy**, the average number of bits-per-word
4. Recognition **error rate**
5. **Re-scoring** intermediate ASR results, “word lattices” with pre-computed acoustic probs

Text-only tests

- Compute the **log-likelihood** of the words and sentences
  - use held-out test data



- **Perplexity**, the average number of word choices
  - inverse of the geom. average word probability $(p(w_1,)..p(w_N))^{-1/N}$
  - low perplexity indicates stronger model (or easier data)
- **Entropy**, the average number of bits-per-word
  - $\Sigma_x\, p(x) \log_2 p(x)$
  - logarithm of the perplexity
- Fast to compute, careful LM normalization required
- Indicates ASR improvements but no guarantees
- Can not compare over different vocabularies

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, “word lattices” with pre-computed acoustic probs
  - much faster than full ASR runs
  - errors in lattices can not be recovered

Software for statistical LMs

- CMU/Cambridge Statistical LM toolkit
  - *www.speech.cs.cmu.edu/SLM_info.html*
  - Easy to use, but some limitations
- **SRI Statistical Language Model Toolkit**
  - *www.speech.sri.com/projects/srilm/*
  - State-of-the-art, well maintained, used in our course
- HTK (some support for low order N-grams)
- Morfessor and VariKN made at TKK
  - `www.cis.hut.fi/projects/{speech,morpho}/`
  - Split words into morphemes, train variable length N-grams

More advanced language models

- Skip n-gram
- Cache n-gram
- Interpolated n-gram
- Topic model, mixture n-gram
- Class LM, Sub-word LM
- Maximum Entropy LM
- Neural Network LM

Test what you learned today!

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
3. Give your **surname** (to get the activity points)
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 time/score is just for fun, only the answers matter

Content this week

1. Phonemes, HMM
2. Vocabulary
3. Statistical language model
4. Neural Network language model
5. **Home exercise**: (3) Build a language model for recognition of continuous speech!
6. Status of project group works

Home exercise 3

- Build a language model for large vocabulary speech recognition!
- Instructions and help given in Zulip by Anssi Moisio (Part1) and Ekaterina Voskoboinik (Part2)
- **Part 1: Language modeling by SRILM toolkit** *http://www.speech.sri.com/projects/srilm/*
- **Part 2: Understanding simple NNLMs**
- To be returned by Wednesday **next week**
- H1: done, feedback in process
- H2: Remember the DL this Wednesday

Home exercise 3: Part 1

- Language modeling by SRILM toolkit
- Goal:
  - understand what n-gram LMs are
  - learn to train n-gram LMs using the SRILM toolkit
  - learn to evaluate n-gram LMs using a text data
- Steps:

  1) Compute n-gram counts and prepare a vocabulary from a text corpus
  2) Create n-gram LM using Kneser-Ney smoothing
  3) Evaluate the LM using text data
  4) Compare word and subword n-gram LMs

Home exercise 3: Part 2

- Goal: Understanding simple NNLMs and the difference between FFNN and RNN
- Compute the last word for a sequence of words using two pre-trained neural language models: FFNN and RNN
- Explain how and why the predictions from FFNN and RNN differ and how both differ from n-grams?
- Remember to submit the answers for both Part1 (n-grams) and Part2 (NNLM) in the same pdf file

Feedback

Now: Go to **MyCourses > Lectures** and fill in the feedback for **week 3**.

Some pics of the feedback from the previous week:

+ the two exercises were helpful to understand HMM

+ quizzes are good, especially going through the right answers

- Kahoot is a bit difficult sometimes

- provide exercise sessions in person

- instructions on project work are very vague

Ave weekly time spent (until week2 / total course target):

Study: 4/50h, Exercise: 6/40h, Project: 5/40h (Max: 6,14,12 One: 15,24,30)

Thanks for all the valuable feedback!

Project work receipt

1. Form a group (3 persons)
2. Get a topic (DL week 1)
3. Get reading material from *Mycourses* or your group tutor
4. 1st meeting: Specify the topic, start literature study (DL week 2)
5. 2nd meeting: Write a work plan (DL week 3)
6. **3rd - 5th meetings: Perform analysis, experiments and write the report**
7. Book your presentation time for weeks 6 – 7 (DL week 4)
8. Prepare and keep your 15 min presentation
9. Return the report (DL week 7)

Start now

Submit your work plan this week!

Final project report

1. Abstract: (your working plan)
2. Introduction: (your literature review)
   - Remember to cite every article you read
3. Experiments: Describe what you did
4. Results: Describe the results you got
5. Conclusion: Your conclusion of the work
6. References: (list of articles that you read)