Lexicon and Language Model

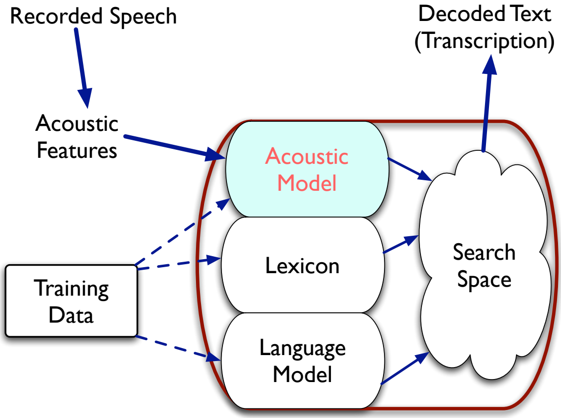
Steve Renals

Automatic Speech Recognition – ASR Lecture 10
15 February 2018

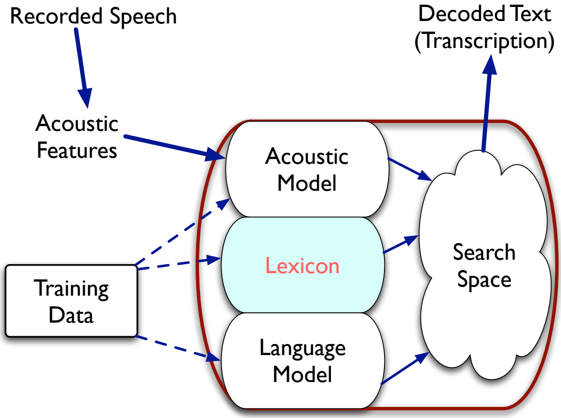
Three levels of model

- **Acoustic model** $P(X \mid Q)$
  Probability of the acoustics given the phone states: context-dependent HMMs using state clustering, phonetic decision trees, etc.
- **Pronunciation model** $P(Q \mid W)$
  Probability of the phone states given the words; may be as simple a dictionary of pronunciations, or a more complex model
- **Language model** $P(W)$
  Probability of a sequence of words. Typically an n-gram

HMM Speech Recognition



HMM Speech Recognition



Pronunciation dictionary

- Words and their pronunciations provide the link between sub-word HMMs and language models
- Written by human experts
- Typically based on phones

Pronunciation dictionary

- Words and their pronunciations provide the link between sub-word HMMs and language models
- Written by human experts
- Typically based on phones
- Constructing a dictionary involves
  1. Selection of the words in the dictionary—want to ensure high coverage of words in test data
  2. Representation of the pronunciation(s) of each word

Pronunciation dictionary

- Words and their pronunciations provide the link between sub-word HMMs and language models
- Written by human experts
- Typically based on phones
- Constructing a dictionary involves
  1. Selection of the words in the dictionary—want to ensure high coverage of words in test data
  2. Representation of the pronunciation(s) of each word
- Explicit modelling of pronunciation variation

Out-of-vocabulary (OOV) rate

- OOV rate: percent of word tokens in test data that are not contained in the ASR system dictionary
- Training vocabulary requires pronunciations for *all* words in training data (since training requires an HMM to be constructed for each training utterance)
- Select the recognition vocabulary to minimize the OOV rate (by testing on development data)
- Recognition vocabulary may be different to training vocabulary
- Empirical result: each OOV word results in 1.5–2 extra errors ($>$1 due to the loss of contextual information)

Multilingual aspects

- Many languages are morphologically richer than English: this has a major effect of vocabulary construction and language modelling
- Compounding (eg German): decompose compound words into constituent parts, and carry out pronunciation and language modelling on the decomposed parts
- Highly inflected languages (eg Arabic, Slavic languages): specific components for modelling inflection (eg factored language models)
- Inflecting and compounding languages (eg Finnish)
- All approaches aim to reduce ASR errors by reducing the OOV rate through modelling at the morph level; also addresses data sparsity

Single and multiple pronunciations

- Words may have multiple pronunciations:
  1. Accent, dialect: *tomato*, *zebra*
     global changes to dictionary based on consistent pronunciation variations
  2. Phonological phenomena: *handbag*/ `h ae m b ae g`
     I can't stay / `[ah k ae n s t ay]`
  3. Part of speech: *project*, *excuse*

Single and multiple pronunciations

- Words may have multiple pronunciations:
  1. Accent, dialect: *tomato*, *zebra*
     global changes to dictionary based on consistent pronunciation variations
  2. Phonological phenomena: *handbag*/ `h ae m b ae g`
     I can't stay / `[ah k ae n s t ay]`
  3. Part of speech: *project*, *excuse*
- This seems to imply many pronunciations per word, including:
  1. Global transform based on speaker characteristics
  2. Context-dependent pronunciation models, encoding of phonological phenomena

Single and multiple pronunciations

- Words may have multiple pronunciations:
  1. Accent, dialect: *tomato*, *zebra*
     global changes to dictionary based on consistent pronunciation variations
  2. Phonological phenomena: *handbag*/ `h ae m b ae g`
     I can't stay / `[ah k ae n s t ay]`
  3. Part of speech: *project*, *excuse*
- This seems to imply many pronunciations per word, including:
  1. Global transform based on speaker characteristics
  2. Context-dependent pronunciation models, encoding of phonological phenomena
- **BUT** state-of-the-art large vocabulary systems average about 1.1 pronunciations per word: most words have a single pronunciation

Consistency vs Fidelity

- Empirical finding: adding pronunciation variants can result in reduced accuracy
- Adding pronunciations gives more "flexibility" to word models and increases the number of potential ambiguities—more possible state sequences to match the observed acoustics

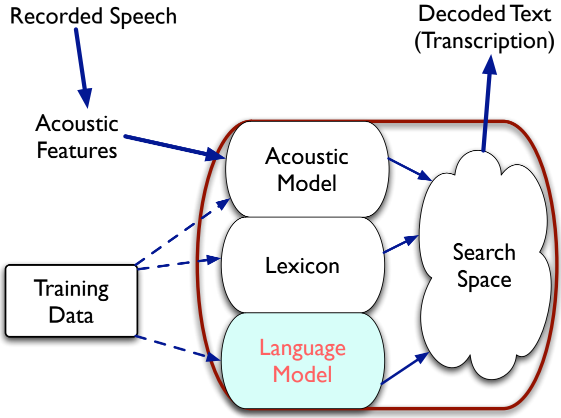
Consistency vs Fidelity

- Empirical finding: adding pronunciation variants can result in reduced accuracy
- Adding pronunciations gives more "flexibility" to word models and increases the number of potential ambiguities—more possible state sequences to match the observed acoustics
- Speech recognition uses a consistent rather than a faithful representation of pronunciations
- A consistent representation requires only that the same word has the same phonemic representation (possibly with alternates): the training data need only be transcribed at the word level
- A faithful phonemic representation requires a detailed phonetic transcription of the training speech (much too expensive for large training data sets)

Current topics in pronunciation modelling

- Automatic learning of pronunciation variations or alternative pronunciations for some words – e.g. learning probability distribution over possible pronunciations generated by grapheme-to-phoneme models
  - Automatic learning of pronunciations of new words based on an initial seed lexicon
- Joint learning of the inventory of subword units and the pronunciation lexicon
- Sub-phonetic / articulatory feature model
- Grapheme-based modelling: model at the character level and remove the problem of pronunciation modelling entirely

HMM Speech Recognition



Statistical language models

- Basic idea The language model is the prior probability of the word sequence $P(W)$
- Statistical language models: cover "ungrammatical" utterances, computationally efficient, trainable from huge amounts of data, can assign a probability to a sentence fragment as well as a whole sentence
- Until very recently **n-grams** were the state-of-the-art language model for ASR
  - Unsophisticated, linguistically implausible
  - Short, finite context
  - Model solely at the shallow word level
  - But: wide coverage, able to deal with "ungrammatical" strings, statistical and scaleable
- In an n-gram, the probability of a word depends only on the identity of that word and of the preceding n-1 words. These short sequences of n words are called n-grams.

Bigram language model

- Word sequence $\mathbf{W} = w_1, w_2, \ldots w_M$

$$P(\mathbf{W}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2) \\ \ldots P(w_M \mid w_1, w_2, \ldots w_{M-1})$$

- Bigram approximation—consider only one word of context:

$$P(\mathbf{W}) \simeq P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2) \ldots P(w_M \mid w_{M-1})$$

Bigram language model

- Word sequence $\mathbf{W} = w_1, w_2, \ldots w_M$

$$P(\mathbf{W}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2) \ldots P(w_M \mid w_1, w_2, \ldots w_{M-1})$$

- Bigram approximation—consider only one word of context:

$$P(\mathbf{W}) \simeq P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2) \ldots P(w_M \mid w_{M-1})$$

- Parameters of a bigram are the conditional probabilities $P(w_j \mid w_i)$
- Maximum likelihood estimates by counting:

$$P(w_j|w_i) \sim \frac{c(w_i, w_j)}{c(w_i)}$$

where $c(w_i, w_j)$ is the number of observations of w_i followed by w_j, and $c(w_i)$ is the number of observations of w_i (irrespective of what follows)

The zero probability problem

- Maximum likelihood estimation is based on counts of words in the training data
- If a n-gram is not observed, it will have a count of 0—and the maximum likelihood probability estimate will be 0
- The zero probability problem: just because something does not occur in the training data does not mean that it will not occur
- As n grows larger, so the data grow sparser, and the more zero counts there will be

The zero probability problem

- Maximum likelihood estimation is based on counts of words in the training data
- If a n-gram is not observed, it will have a count of 0—and the maximum likelihood probability estimate will be 0
- The zero probability problem: just because something does not occur in the training data does not mean that it will not occur
- As n grows larger, so the data grow sparser, and the more zero counts there will be
- Solution: smooth the probability estimates so that unobserved events do not have a zero probability
- Since probabilities sum to 1, this means that some probability is redistributed from observed to unobserved n-grams

Smoothing language models

- What is the probability of an unseen n-gram?

Smoothing language models

- What is the probability of an unseen n-gram?
- Add-one smoothing: add one to all counts and renormalize.
  - "Discounts" non-zero counts and redistributes to zero counts
  - Since most n-grams are unseen (for large n more types than tokens!) this gives too much probability to unseen n-grams (discussed in Manning and Schütze)
- Absolute discounting: subtract a constant from the observed (non-zero count) n-grams, and redistribute this subtracted probability over the unseen n-grams (zero counts)
- Kneser-Ney smoothing: family of smoothing methods based on absolute discounting that are at the state of the art (Goodman, 2001)

Backing off

- How is the probability distributed over unseen events?
- Basic idea: estimate the probability of an unseen n-gram using the (n-1)-gram estimate
- Use successively less context: trigram $\rightarrow$ bigram $\rightarrow$ unigram
- Back-off models redistribute the probability "freed" by discounting the n-gram counts

Backing off

- How is the probability distributed over unseen events?
- Basic idea: estimate the probability of an unseen n-gram using the (n-1)-gram estimate
- Use successively less context: trigram $\rightarrow$ bigram $\rightarrow$ unigram
- Back-off models redistribute the probability "freed" by discounting the n-gram counts
- For a bigram

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

c is the count threshold, and D is the discount. b_{w_i} is the backoff weight required for normalization

Interpolation

- Basic idea: Mix the probability estimates from all the estimators: estimate the trigram probability by mixing together trigram, bigram, unigram estimates
- Simple interpolation

$$\hat{P}(w_n \mid w_{n-2}, w_{n-1}) = \lambda_3 P(w_n \mid w_{n-2}, w_{n-1}) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_1 P(w_n)$$

  With $\sum_i \lambda_i = 1$

- Interpolation with coefficients conditioned on the context

$$\hat{P}(w_n \mid w_{n-2}, w_{n-1}) = \lambda_3(w_{n-2}, w_{n-1}) P(w_n \mid w_{n-2}, w_{n-1}) + \lambda_2(w_{n-2}, w_{n-1}) P(w_n \mid w_{n-1}) + \lambda_1(w_{n-2}, w_{n-1}) P(w_n)$$

- Set λ values to maximise the likelihood of the interpolated language model generating a *held-out* corpus (possible to use EM to do this)

Perplexity

- Measure the quality of a language model by how well it predicts a test set W (i.e. estimated probability of word sequence)
- Perplexity ($PP(W)$) – inverse probability of the test set W, normalized by the number of words N

$$PP(W) = P(W)^{\frac{-1}{N}} = P(w_1 w_2 \ldots w_N)^{\frac{-1}{N}}$$

- Perplexity of a bigram LM

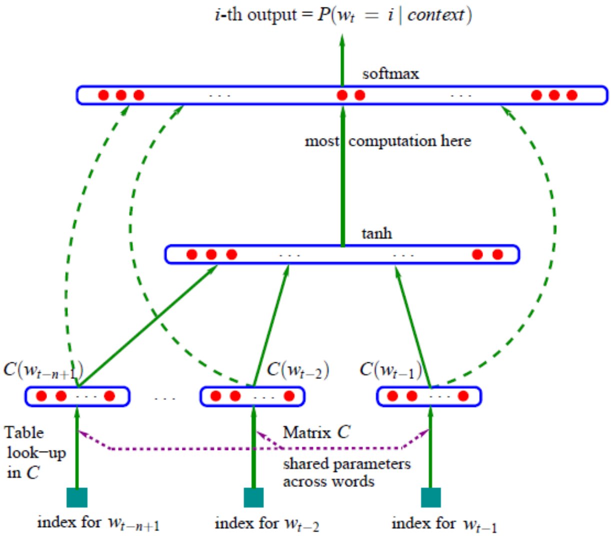
$$PP(W) = (P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_N|w_{N-1}))^{\frac{-1}{N}}$$

- Example perplexities for different n-gram LMs trained on Wall St Journal (38M words)
  - Unigram – 962
  - Bigram – 170
  - Trigram – 109

Distributed representation for language modelling

- Each word is associated with a learned *distributed representation* (feature vector)
- Use a neural network to estimate the conditional probability of the next word given the the distributed representations of the context words
- Learn the distributed representations and the weights of the conditional probability estimate jointly by maximising the log likelihood of the training data
- Similar words (distributionally) will have similar feature vectors — small change in feature vector will result in small change in probability estimate (since the NN is a smooth function)

Neural Probabilistic Language Model



Bengio et al (2006)

Neural Probabilistic Language Model

- Train using stochastic gradient ascent to maximise log likelihood
- Number of free parameters (weights) scales
  - Linearly with vocabulary size
  - Linearly with context size
- Can be (linearly) interpolated with n-gram model
- Perplexity results on AP News (14M words training). $|V| = 18k$

model	**n**	**perplexity**
NPLM(100,60)	6	109
n-gram (KN)	3	127
n-gram (KN)	4	119
n-gram (KN)	5	117

Shortlists

- Reduce computation by only including the s most frequent words at the output — the *shortlist* (S) (full vocabulary still used for context)
- Use an n-gram model to estimate probabilities of words not in the shortlist
- Neural network thus redistributes probability for the words in the shortlist

$$P_S(h_t) = \sum_{w \in S} P(w|h_t)$$

$$P(w_t|h_t) = \begin{cases} P_{NN}(w_t|h_t)P_S(h_t) & \text{if} w_t \in S \\ P_{KN}(w_t|h_t) & \text{else} \end{cases}$$

- In a $|V| = 50k$ task a 1024 word shortlist covers 89% of 4-grams, 4096 words covers 97%

NPLM — ASR results

Speech recognition results on Switchboard

7M / 12M / 27M words in domain data.

500M words background data (broadcast news)

Vocab size $|V| = 51k$, Shortlist size $|S| = 12k$

WER/%

in-domain words	7M	12M	27M
KN (in-domain)	25.3	23.0	20.0
NN (in-domain)	24.5	22.2	19.1
KN (+b/g)	24.1	22.3	19.3
NN (+b/g)	23.7	21.8	18.9

Summary

- Pronunciation dictionaries
- n-gram language models
- Neural network language models

Reading

- Jurafsky and Martin, chapter 4
- Y Bengio et al (2006), "Neural probabilistic language models" (sections 6.1, 6.2, 6.3, 6.6, 6.7, 6.8), Studies in Fuzziness and Soft Computing Volume 194, Springer, chapter 6. http://link.springer.com/chapter/10.1007/3-540-33486-6_6