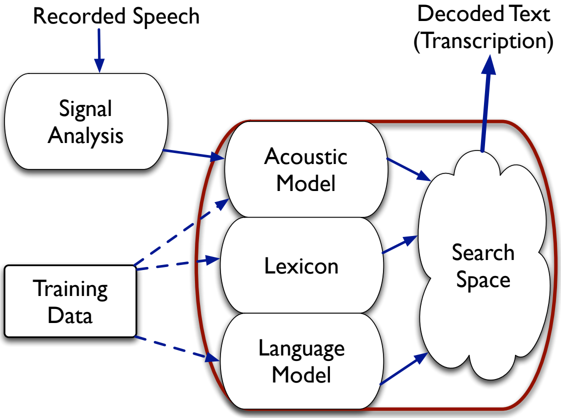
Words: Pronunciations and Language Models

Steve Renals

Automatic Speech Recognition— ASR Lecture 8
9 February 2015

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
- 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

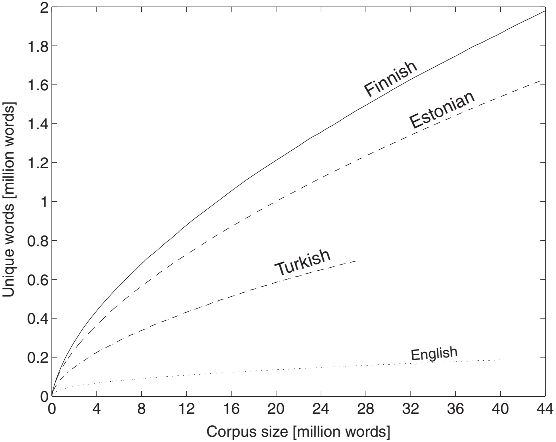
Out-of-vocabulary (OOV) rate

- 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

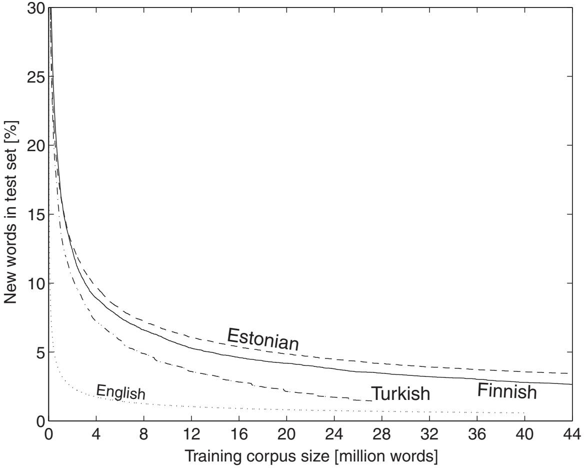
- Many languages are morphologically richer than English: this has a major effect of vocabulary construction and language modelling
- Compounding (eg German): decompose compund words into constituent parts, and carry out pronunciation and language modelling on the decomposed parts
- Highly inflected 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

Vocabulary size for different languages



M. Creutz et al, "Morph-based speech recognition and modeling OOV words across languages", *ACM Trans Speech and Language Processing*, 5(1), art. 3. http://doi.acm.org/10.1145/1322391.1322394

OOV Rate for different languages



M. Creutz et al, "Morph-based speech recognition and modeling OOV words across languages", *ACM Trans Speech and Language Processing*, 5(1), art. 3. http://doi.acm.org/10.1145/1322391.1322394

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
- 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)

Modelling pronunciation variability

- State-of-the-art systems absorb variations in pronunciation in the acoustic models
- Context-dependent acoustic models may be though of as giving broad class representation of word context
- Cross-word context dependent models can implicitly represent cross-word phonological phenomena
- Hain (2002): a carefully constructed single pronunciation dictionary (using most common alignments) can result in a more accurate system than a multiple pronunciation dictionary

Mathematical framework

HMM Framework for speech recognition. Let W be the universe of possible utterances, and X be the observed acoustics, then we want to find:

$$
\begin{aligned}
W^* &= \arg\max_W P(W \mid X) \\
&= \arg\max_W \frac{P(X \mid W)P(W)}{P(X)} \\
&= \arg\max_W P(X \mid W)P(W)
\end{aligned}
$$

Words are composed of a sequence of HMM states Q:

$$
\begin{aligned}
W^* &= \arg\max_W P(X \mid Q, W)P(Q, W) \\
&\simeq \arg\max_W \sum_Q P(X \mid Q)P(Q \mid W)P(W) \\
&\simeq \arg\max_W \max_Q P(X \mid Q)P(Q \mid W)P(W)
\end{aligned}
$$

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

Language modelling

- Basic idea The language model is the prior probability of the word sequence $P(W)$
- Use a language model to disambiguate between similar acoustics when combining linguistic and acoustic evidence
  never mind the nudist play / *never mind the new display*
- Use hand constructed networks in limited domains
- 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

Statistical language models

- For use in speech recognition a language model must be: statistical, have wide coverage, and be compatible with left-to-right search algorithms
- Only a few grammar-based models have met this requirement (eg Chelba and Jelinek, 2000), and do not yet scale as well as simple statistical models
- 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
- 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 $\mathbf{W} = w_1, w_2, \dots w_M$

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

- Bigram approximation—consider only one word of context:

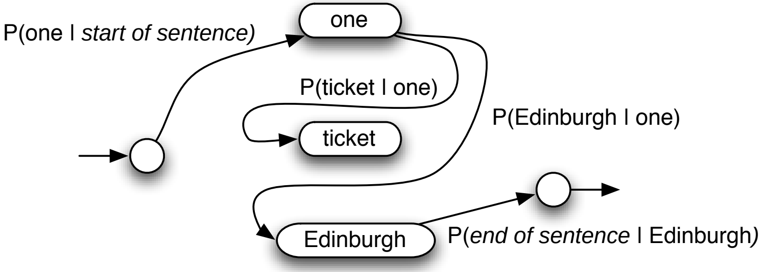
$$P(\mathbf{W}) \simeq P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2) \dots P(w_M \mid w_{M-1})$$

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

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

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

Bigram network



- n-grams can be represented as probabilistic finite state networks
- only some arcs (and nodes) are shown for clarity: in a full model there is an arc from every word to every word
- note the special start and end sentence probabilities

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?
- 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
- For a bigram

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

c is the count threshold, and D is the discount. b_{w_j} 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

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

- Perplexity of a bigram LM

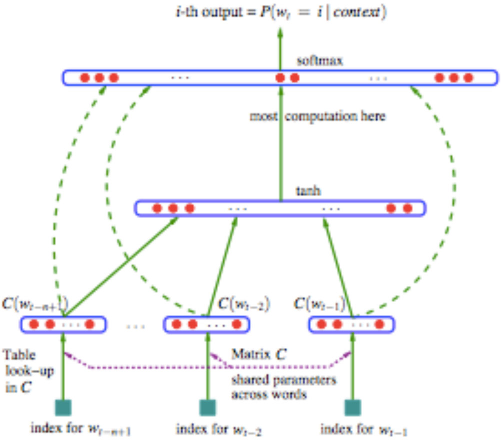
$$PP(W) = (P(w_1)P(w_2|w_1)P(w_3|w_2)\dots P(w_N|w_{N-1}))^{\frac{-1}{N}}$$

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

Practical language modelling

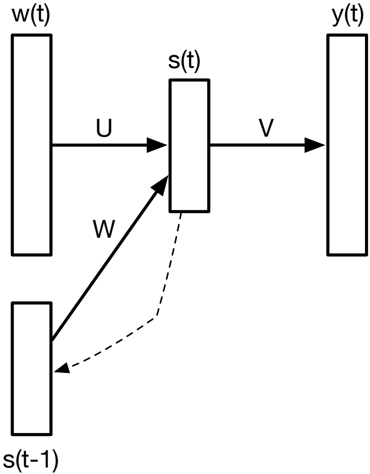
- Work in log probabilities
- The ARPA language model format is commonly used to store n-gram language models (unless they are very big)
- Many toolkits: SRILM, IRSTLM, KenLM, Cambridge-CMU toolkit, ...
- Some research issues:
  - Advanced smoothing
  - Adaptation to new domains
  - Incorporating topic information
  - Long-distance dependencies
  - Distributed representations

Neural Probabilistic Language Model



Bengio 2003

Recurrent Neural Network Language Model



Mikolov et al (2010,2011) - state of the art performance

References

- Jurafsky and Martin, chapter 4
- Fosler-Lussier (2003) - pronunciation modelling tutorial
- Hain (2002) - implicit pronunciation modelling by context-dependent acoustic models
- Bisani and Ney (2008) - Sequitur G2P (`http://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html`)
- Gotoh and Renals (2003) - language modelling tutorial
- Good coverage of n-grams in Manning and Schütze (1999)
- Jelinek (1991) - "Up from trigrams!"
- Chelba and Jelinek (2000) - example of a probabilistic grammar-based language model
- Goodman (2001) - state-of-the-art smoothing for n-grams (Modified Kneser-Ney smoothing)
- Bengio (2003) - Neural probabilistic language model
- Mikolov et al (2011) - strategies for training large scale neural network language models (RNNs)