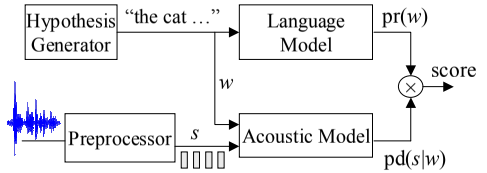
Lecture 18

Language Modelling

- Language Modelling
  - Bigrams and Trigrams
  - Trigram Estimation
- Trigram Recognition
- Dynamic Models
- Recogniser Performance
  - Performance enhancement techniques

Language Modelling



- Purpose: to estimate the probability of a word occurring in a sentence given the previous *N*–1 words. These are called *N*-gram probabilities.
  - Unigrams give the probability of a word in the absence of any context.
  - Bigrams give the probability of a word as a function of the preceding word in the sentence.
  - Trigrams give the probability as a function of the preceding two words.
    - 10,000 words $\Rightarrow 10^8$ different bigrams and 10^{12} different trigrams! Most trigrams will not appear at all in the training data.
    - Use trigrams or quadrams for common word sequences and bigrams or even unigrams for less common sequences.
- A good language model allows unlikely word sequences to be eliminated from the search: saves time.

Estimating Trigrams

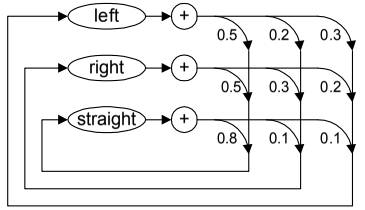
- Common Trigrams
  - If trigram "abc" occurs *n* times out of a total of *N* trigrams "ab?" in the training data, then estimate prob("c" | "ab") as *n*/*N*
- Rare Trigrams
  - Consider all the rare trigrams whose true probability is 0.5/*N*
    - Some will occur once and have an *overestimated* probability of 1/*N*
    - Others will not occur at all and have an *underestimated* probability of 0/*N*
  - *Discounting*: for trigrams with n = 3, … 6 make the probability smaller than *n*/*N* to compensate for the overestimation.
  - *Backing-off*: for trigrams with *n* = 0, 1 or 2 use the bigram probability:
    - estimate prob("*c*" | "*ab*") as prob("*c*" | "*b*")×*k*
    - *k* is chosen to ensure that

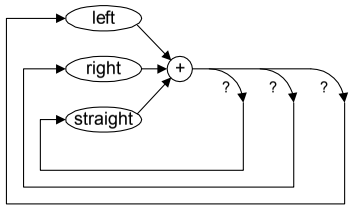
$$\sum_{\text{all } w} \text{prob}(\text{“}w\text{”} \mid \text{“}ab\text{”}) = 1$$

    - If necessary, the bigram probabilities can be estimated from unigrams in the same way.

Bigram Recognition

- Task: Giving directions for maze
- "left" can be followed by "straight", "left" or "right" in that order of frequency: these are the *bigram* probabilities.
- Each word model appears only once.
- Inter-word gap model, ⊕, must be duplicated for each word to avoid losing context information.

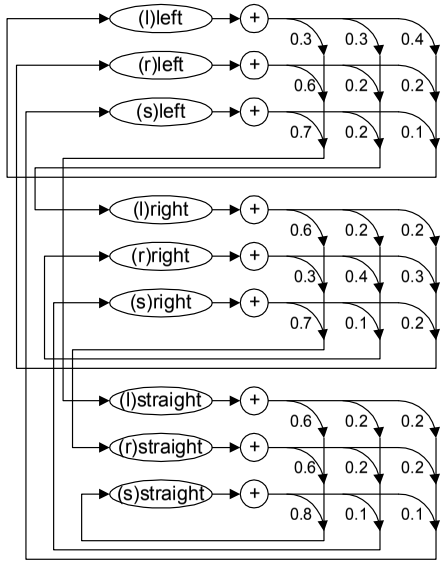




- We can't use bigram probabilities if we merge the ⊕ states (lower diagram)
- All information about the past is incorporated into the HMM state
  - Retain more information ⇒ more states

Trigram Recognition

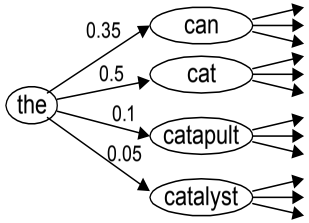
- Word probabilities depend on 2 previous words
  - probability of "left" is
    - 0.4 when following "left left"
    - 0.2 when following "right left"
- Need N^2 word models and N^3 transition probs



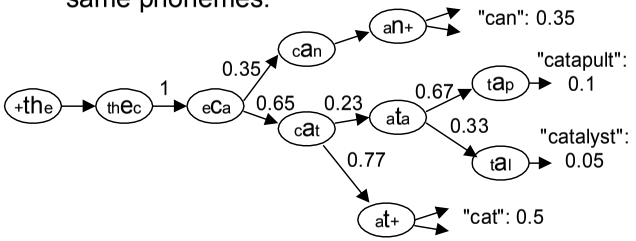
Dynamic Models

- We create storage for model states only when they are needed
  - Each model state corresponds to a particular position within a particular word with a particular *context* (= the previous word for trigrams)
  - Pruning means that only a small fraction of the possible states exist at any time.
- Algorithm: Repeat for each input frame:
  - List all states that the current input frame might be in: the *active states*
  - Allocate storage for each *distinct* state that the next frame might be in:
    - For states in the middle of a word, the next frame can only be in (a) the same state as the current frame or (b) the next state in the word
    - For states at the end of a word, the next frame might be the first frame of any new word that is allowed by the language model.
  - Calculate the cumulative path probability for the next frame belonging to each of the newly allocated states.
  - Delete any state whose cumulative path probability is more than the beam width lower than the best one. Remaining states are the active states for the next frame.

Sharing States Between Words

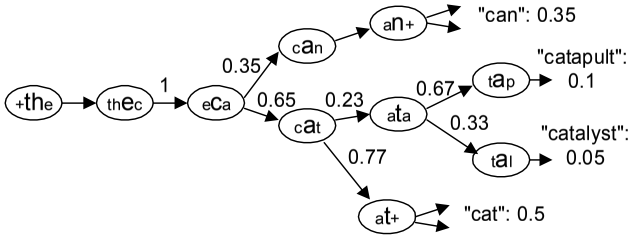


- If several words begin with the same sequence of phonemes it is wasteful to create distinct states for each one. Instead we create a branching tree for all words starting with the same phonemes.

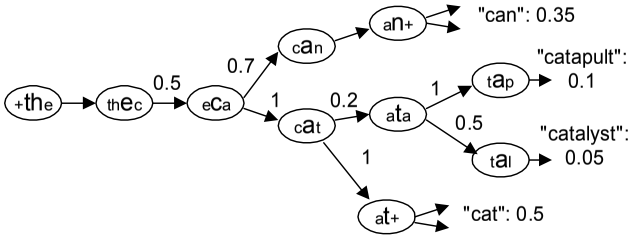


- The last state of a word now branches to only 43^2 different places instead of to every word in the vocabulary.

Applying Probabilistic Constraints



- We want to apply probabilistic constraints as early as possible ⇒ for each node:
  - Find the highest exit transition probability, h.
  - Divide all exit probabilities by h and multiply all entrance probabilities by h.



- Low probabilities are pushed leftwards and pruning occurs earlier.

Performance

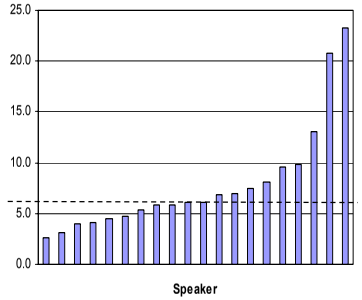
- Tested on readings from the Wall Street Journal
  - 65,000 words in the vocabulary
  - 66 hours of speech used for training
    - Processing is much less than real time ⇒ hundreds of hours of CPU time
- Using trigrams: best performance has 8.2% wrong words.
- Using quadrams: best performance has 7.9% wrong words.
- Huge amount of memory + CPU time

sheep & goats

- Average error rate hides variation between speakers
  - Some speakers are very easy to recognise

  - While others are much harder

  - Average performance is made much worse by a few goats:



Speaker Adaptation

- Can adjust the recogniser to suit a particular speaker
  - *Supervised* Adaptation: computer must be told each time it makes an error
  - *Unsupervised* Adapation: computer assumes that most decisions are correct
  - *Incremental* Adaptation: adaptation goes on continuously.
- We need to adapt the entire recogniser based on a small amount of speech $\Rightarrow$ use a form of general transformation
  - Linear or non-linear warping of the frequency axis
  - Linear transformation of the feature vector (i.e. multiplying by a matrix)
  - Different linear transformation of the feature vector for different states (using the same transformation for similar sounding states)
    - Reduces error rate from 7.9% to 7.2% when the recogniser is told when a new speaker has started talking.

Channel Variability

- Telephone-based systems must cope with a wide range of microphones
  - Microphones vary greatly in their frequency response.
- The speech spectrum is *multiplied* by the channel frequency response $\Rightarrow$ the speech cepstrum has the channel cepstral response *added* to it.
- Can correct for this by subtracting the long-term average of the cepstrum from the cepstrum calculated for each frame.
  - called *cepstral mean subtraction*
  - makes the recogniser immune to variations in channel frequency response
  - under matched training and recognition conditions the use of cepstral mean subtraction increases the error rate slightly.

Background Noise

- Three approaches to coping with background noise:
  - Try to remove the noise without distorting the speech: speech enhancement
  - Choose a set of features that are inherently noise resistant (e.g. positions of peaks in the spectrum)
  - Extend the statistical model of speech to model the noise as well.
    - If we have 1000 states in our speech models and 4 states in our noise model we can think of this as a single 4000-state model of both speech and noise
    - The average power spectrum of one of these combined speech+noise states is just the sum of the average speech power spectrum and the average noise power spectrum.
    - The recogniser uses the 4000-state model and simultaneously recognises both speech and noise
    - Number of states in the combined model is the product of the number of states in each model separately ⇒ can't afford to have many states in the noise model.

Task independence

- Language models are specific to one application
  - language model for Wall Street Journal will not work well for engineering literature
  - N-gram language models are crude and brute force: cannot generalise to new words.
- Language models based on classes of related words would generalise more easily
  - difficult to choose the classes;
    - could be names of colours, makes of car etc
    - could be nouns, adjectives, verbs
  - many words need to be in more than one class

Unscripted Speech

- Most recogniser evaluation uses people reading from prepared texts; performance is much worse with casual speech.
  - mumbling: less clear articulation
  - large speed variations
  - false starts, repetitions and corrections

Confusable Terms

- **Synthesisers**
  - **Diphone** - goes from the middle of one phoneme to the middle of the next. (43^2)
  - **Demisyllable** - the first or last half of a syllable
- **Recognizers**
  - **Triphone** - a hidden markov model for a phoneme typically containing three states. There are 43^2 different triphone models for each phoneme: one for each possible pair of preceding and following phonemes.
  - **Biphone** - a hidden markov model that only depends on one of its adjoining phonemes.
  - **Unigram** - the probability of a word occuring
  - **Bigram** - the probability of a word occuring as a function of the previous word in the sentence. For a vocabulary size of V, each word will have V different bigram probabilities: a total of V^2 probabilities.
  - **Trigram** - the probability of a word occuring as a function of the two previous words in the sentence. Each word has V^2 different trigram probabilities: a total of V^3 values in all.

High performance recognizers normally use *triphones* and *trigrams*.