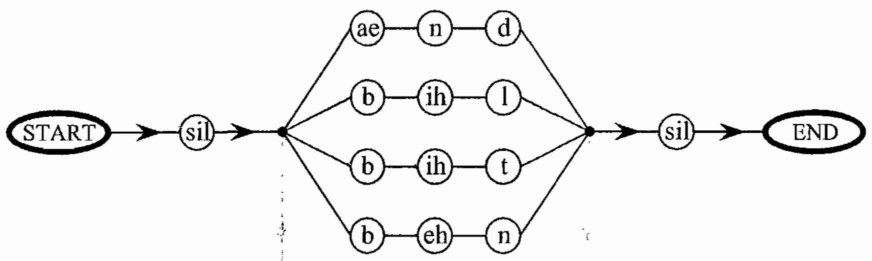
Prof P C Woodland and Dr W J Byrne

Solutions to 4F11 Speech and Language Processing, 2009

1. *Speech Recognition Search*

(a) Taking as an example the simple 4 word (AND, BILL, BIT, BEN) from lectures then use: [15%]



To use a unigram language model: Add the language model probability (e.g. $\log(P_{BEN})$) to the log-likelihood of the token at the start of each word.

(b) The Viterbi algorithm can be applied to the above structure (it is an HMM) noting that between models/words the transition probabilities come from the language model and not the internal phone level transition probabilities.

For an N-state HMM, with observation distributions $b_i(o_t)$ and state transition probabilities $a_{i,j}$ the Viterbi algorithm is as follows.

Initialization:
$\phi_1(0) = 1.0$
$\phi_j(0) = 0.0$ for $1 < j < N$ and $\phi_1(t) = 0.0$ for $1 \leq t \leq T$

Recursion:

```
for $t = 1, 2, \ldots, T$
... for $j = 2, 3, \ldots, N-1$
......compute $\phi_j(t) = \max_{1 \leq k < N} [\phi_k(t-1) a_{kj}] b_j(\mathbf{o}_t)$
......store the predecessor node: $\text{pred}_k(t)$
```

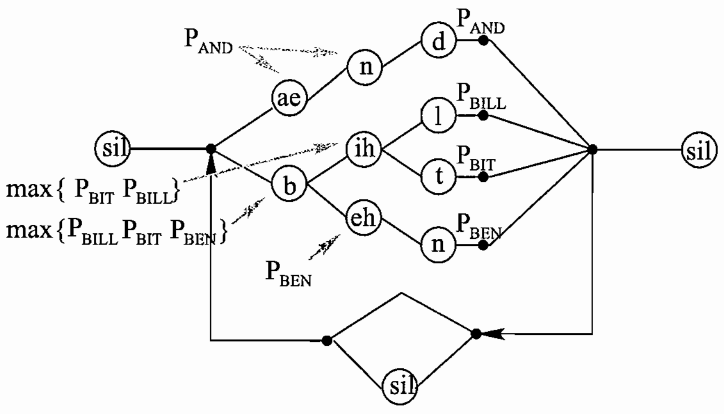
Termination:

$$p(\mathbf{O}, \mathbf{X}^* | \lambda) = \max_{1<k<N} \phi_k(T) a_{kN}$$

The sequence of models and hence the sequence of words can be found by back-tracing along the best state-level path. It is only necessary to store this path at word boundaries (the token-passing implementation includes explicit ways to do this) for efficiency. Note that an explanation based on token-passing is also a valid solution here! [20%]

(c) Beam search. At each frame store the most likely token (log likelihood to that point). Set the log-likelihood of any token/path that is more than B (a log-likelihood difference) from the best path to $-\infty$. These paths then do not need to be extended. In practice keep an active list of models and only process the active models. Note that when paths are extended from an active model need to activate all connected models. By using only a (small) sub-set of active models the computational load is very considerably reduced both for model-internal token propagation and output probability computation. [15%]

(d)(i)



The LM probability is applied at the end of each word or the maximum language model probability at each node in the network is applied (as shown). [15%]

(d)(ii) For word-internal models, simply use the contexts in place of monophone labels when performing tree-structuring. Not so much tree-structuring is possible since can only share if the complete triphone context is the same. [15%]

(d)(iii) Major issue is that in the bigram the identity of the current-word is not unique at the start and therefore it isn't clear how to apply the LM between the end of one word and the start of the next. There are a number of solutions that are possible. One involves the use of multiple tokens to store different histories, and then incrementally applying the bigram probabilities throughout the tree. Another is dynamically growing the network and starting a new network for different language model contexts. A further set of solutions is to use a multi-pass solution in which a simpler language model is first used to generate a lattice or N-best list: however this method will likely lead to many search errors and would not normally be used for a bigram LM. [20%]

2. *HMM Output Distributions and Cross-Word Triphones*

(a)(i) Single Gaussian Full Covariance Matrix.

$$b_j(\boldsymbol{o}_t) = \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{o}_t-\boldsymbol{\mu}_j)'\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{o}_t-\boldsymbol{\mu}_j)}$$

where $\boldsymbol{\Sigma}$ is a full covariance matrix.

There are d mean parameters and $\frac{d(d+1)}{2}$ covariance parameters. Computation is appromixately $d^2 + d$ multiply-adds.

This allows the correlations between feature elements to be explicitly modelled. If the feature vector has strong correlations and there is enough training data it will improve performance over single Gaussian diagonal but not well model non-Gaussian data. [20%]

(a)(ii) Gaussian Mixture Diagonal Covariance Matrix. This is more flexible in modelling data as it can model weak correlations, multi-modality and other non-Gaussian effects. Normally applied to e.g. cepstral data for which the data is globally fairly well de-correlated.

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{M} c_{jm} b_{jm}(\mathbf{o}_t) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$$

c_{jm} is the component weight, or prior, and here $\boldsymbol{\Sigma}_{jm}$ is a diagonal covariance matrix. For this to be a pdf it is necessary that

$$\sum_{m=1}^{M} c_{jm} = 1 \quad \text{and} \quad c_{jm} \geq 0$$

The total number of parameters are $2dM$ for the means and variances and $M - 1$ mixture weights. Computation (apart from the log-add operations to add together the log pdfs of the individual components) is approximately $2dM$ multiply-adds.

(b)(i) Cross-Word Triphones. Convert each monophone to a cross-word context-dependent phone model (model depends on the immediate left and right phone context as well as the phone itself and the context extends across word boundaries). It allows co-articulation to be explicitly modelled. This complicates training as the number of models and potential parameters. For a single Gaussian cross-word triphone system co-articulation effects are enforced by the model constraints. For a monophone system with a mixture distribution the co-articulation effects are modelled by different Gaussian mixture components with no frame-to-frame constraints. [20%]

(b)(ii) The number of models is greatly increased and some type of smoothing or parameter sharing is reuqired. The training data is very unevenly distributed and to get the correct trade-off between allocation of parameters and available data need to reduce the number of contexts. Note that there will be some contexts with no training data. A common method is state-tying via phonetic decision trees are used. These group contexts so that models can be robustly estimated for the grouped contexts and unseen triphones can be dealt with. The phonetic decision tree is grown automatically from training in a top-down fashion with questions chosen so as to maximise an approximate likelihood of the training data. The questions for splits are chosen from a pool of linguistoc questions which yield generalisation ability. The advantage of this method is that it generalises including linguistic information and allows unseen contexts to be properly dealt with. It is widely used in state-of-the-art systems. [30%]

(b)(iii) Changing to cross-word triphones (from monophones) decoding is greatly complicated (the network to search has a very increase in the number of nodes to enfore the cross-word contraints) and computational cost increases substantially (and storgae since there will usually be far more parameters even with tying) although pruning can be very effective in such networks. [10%]

3. *Machine Translation and HMM alignment models*

(a) Some reasons why machine translation is difficult include: [10%]

- Translation should respect document domain and genre. These differ in
  - Specialized vocabularies
  - Stylistic differences, e.g. active vs. passive voice
  - Use of headlines, passage and chapter numbers, web addresses, proper names, ...
  - Variable sentence lengths, e.g. weather forecasts vs. news stories. Consequently the same translation system cannot be used for all genres
- Translation must depend on word sense, which varies across languages depending on context
- Morphologically rich languages may require complex morphological analysis to be integrated into translation, and surface variability due to morphological variants can lead to sparsity of individual tokens in text translations
- Movement due to characteristic word order and lead to exponential complexity growth with sentence length

(b) The BLEU score is calculated as follows (key points to mention, can be less detailed): [20%]

- Set N to be the order of the highest n-gram to be considered; here N=4
- For each sentence i, and for $n = 1, \dots, N$, gather the following n-gram counts:
  - c_n^i : the number of hypothesized n-grams
  - $\bar{c}_n^i$: the number of correct n-grams, where the contribution of each distinct n-gram is *clipped* to the maximum number of occurrences in any one reference
  - Compute the precision for each n-gram , $n = 1, \dots, N$: $p_n = (\sum_i \bar{c}_n^i) / (\sum_i c_n^i)$
  - Calculate the Brevity Penalty
    - Compute the shortest reference length : $r = \sum_i \min\{|E_{(1)}^i|, |E_{(2)}^i|, |E_{(3)}^i|, |E_{(4)}^i|\}$
    - Compute the hypothesis length : $c = \sum_i |E^i|$

$$BP = \begin{cases} 1 & c > r \\ \exp\left(1 - \frac{r}{c}\right) & c <= r \end{cases}$$

  - The BLEU score is

$$\text{BLEU} = BP * \exp\{ \sum_{n=1}^{N} \log \frac{p_n}{N} \}$$

(c)(i) To find the HMM alignment likelihood $P(f_1^J, a_1^J | e_1^I)$. [10%]

$$\begin{aligned}
P(f_1^J, a_1^J | e_1^I) &= P(f_1^J, a_1^J, J | e_1^I, I) \\
&= P(f_1^J | a_1^J, J, e_1^I, I) P(a_1^J, | J, e_1^I, I) P(J | e_1^I, I) \\
&= P(f_1^J | a_1^J, J, e_1^I) P(a_1^J | J, e_1^I, I) P(J | e_1^I, I) \\
&= \prod_{j=1}^{J} p_t(f_j | e_{a_j}) P_{HMM}(a_j | a_{j-1}, I) p_L(J | I)
\end{aligned}$$

(c)(ii) A recursive relationship for the forward probability $\alpha_j(i) = P(a_j = i, f_1^j | e_1^I)$ is [20%]

$$\alpha_j(u) = \sum_{i'} p_T(f_j | e_i) p_{HMM}(a_j = i | a_{j-1} = i') \alpha_{j-1}(i')$$

(c)(iii) The corresponding backward probability is defined as $\beta_j(i) = P(f_{j+1}^J | a_j = i, e_1^I)$. Using the relationship [20%]

$$P(a_j = i, f_1^J | e_1^I) = P(a_j = i, f_1^j | e_1^I) P(f_{j+1}^J | a_j = i, e_1^I)$$

it follows that

$$P(a_j = i, f_1^J | e_1^I) = \alpha_j(i) \beta_j(i)$$

(c)(iv) An expression for the alignment link posterior probability $P(a_j = i | f_1^J, e_1^I)$ in terms of the forward and backward probabilities can be found as follows. Since $P(a_j = i | f_1^J, e_1^I) = \frac{P(a_j = i, f_1^J | e_1^I)}{P(f_1^J | e_1^I)}$ and $P(f_1^J | e_1^I) = \sum_i \alpha_J(i)$ it follows that [20%]

$$P(a_j = i | f_1^J, e_1^I) = \frac{\alpha_j(i) \beta_j(i)}{\sum_i \alpha_J(i)}$$

4. *N-gram Language Models and Weighted Finite State Networks*

(a) In order to obtain the probability of a word sequence one takes only the current word and the most recent $N-1$ words into account. The estimation of these probabilities is based on frequency of occurrence in the training data. If the value of N increases we are likely to obtain better probability estimates for sentences. However, if we have a vocabulary size of V we need to be able to give probability estimates for V^N possible word combinations. Due to limited size of the training corpus many of these words combinations will have a frequency of 0, even though they represent a valid sequence of words, and smmothing/discounting/back-off methods need to be used. [15%]

(b)(i) $d(r)$ is the discount coefficient. Its role is to reduce the counts of seen events to allow some probability mass for unseen events. In this case it is applied directly to the relative-frequency based bigram probability. $\alpha(\cdot)$ is the back-off weight. It is chosen to ensure that $\sum_{j=1}^{V} \hat{P}(w_j|w_i) = 1$ for the vocabulary of size V. C is the N-gram cut-off point t (i.e. only N -grams that occur more frequently than this are retained in the final model). The value of C also controls the size of the resulting language model. [20%]

(b)(ii) One form of discounting is required.

One form is Good Turing discounting. The discounting formula is

$$d(r) = \frac{(r+1)n_{r+1}}{rn_r}$$

and it is used to scale the counts of N-gram events to allow some probability mass for unseen events. Here n_r is the number of N-grams occurring r times. [20%]

Another form is Absolute discounting. Here

$$d(r) = \frac{(r-b)}{r}$$

Typically $b = \frac{n_1}{(n_1+2n_2)}$. The discounting is applied to all counts. This is, of course, equivalent to simply subtracting the constant b from each count.

(c) An algorithm to construct a WFST for an exact implementation of the bigram language model is:

There is one state for every word, plus a unigram back-off stateϵ :

$$Q = \{(w_1), \ldots, (w_V), \epsilon\}$$

There is an arc for each pair of words w and w' for which $f(w,w') > C$

$$(w) \overset{w'/p(w'|w)}{\longrightarrow} (w')$$

There is a back-off arc (with failure transition ϕ) from every word state (w) to the backoff state ϵ

$$(w) \xrightarrow{\phi/\,\alpha(w)} \epsilon$$

There is a unigram arc from the back-off state ϵ to every word state (w')

$$\epsilon \xrightarrow{w'/\hat{P}(w')} (w')$$

The scores are either probabilities or negative log likelihoods, depending on the semi-ring. [30%]

(d) Extending this algorithm to construct a WFST which implements a back-off trigram language model. The main complication is the increase in the number of transducer states. In the bigram, only single word histories need to be retained so the state set is Q, as above. To build a WFST to implement a trigram $P(w_j|w_i, w_k)$, states of the form (w_i, w_k) are needed whenever there is a w_i for which $f(w_i, w_j, w_k) > C$. A somewhat more complex backoff structure is needed so that backoff from trigram to bigram to unigram is done correctly. [15%]