Solutions: 4F11 Speech and Language Processing, 2012

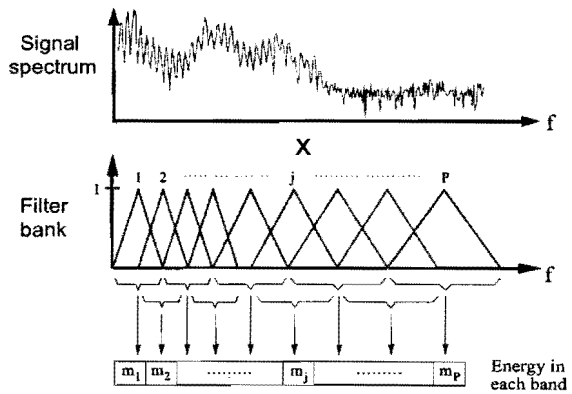
1. *Speech Analysis and Speech Recognition Front-End*

   (a) The desirable attributes are

   - Reduce the raw bit rate to something manageable
   - Remove information that does not discriminate between words/sounds
   - Retain all information that disriminates between words/sounds
   - Transform feature vector to be suitable for the classifier being used. [15%]

   (b) The steps in producing an MFCC feature vector are:

   - Block processing, i.e. taking a frame of the speech signal every 10ms. This allows to use the assumption of quasi-stationary speech segments.
   - Windowing using a Hamming window of 25ms This introduces less distortion (side-lobes) than a rectangular window.
   - DFT of the windowed signal to obtain the magnitude spectrum
   - Filtering using a triangular filterbank on the basis of the Mel scale. A typical number of filters is 19-24 for 8 kHz sampled data. This provides a smoothed representation of the spectrum while taking into account that the speech in lower frequency regions is perceptually more important as shown below:



   - Take the log of the filterbank coefficients This converts the multiplicative relationship in the frequency domain between excitation and source into an additive one.
   - Take the DCT of the log filterbank coefficients. Only retain a smaller number of MFCC elements, a typical value is 12. The zeroth coefficient is a measure of the frame energy. The formula for the DCT is

$$c_n = \sqrt{\frac{2}{P}} \sum_{i=1}^{P} m_i \cos\left[\frac{n(i-\frac{1}{2})\pi}{P}\right]$$

where P is the number of filterbank channels and m_i is the energy in the ith channel. [30%]

(c) Linear prediction uses an all-pole model of speech. The frequency response of the all-pole file (the linear prediction spectrum) can be converted to a cepstral representation by finding the log spectrum, and then taking the inverse DFT (or also by a direct recursion from the predictor coefficients).The key differences to MFCCs is that a linear frequency representation is used and an all-pole model of the speech (& hence the inherent assumptions) is used for smoothing, rather than the Mel scale filterbank which provides smoothing for MFCCs. [15%]

(d) The delta and delta-delta coefficients allow to incorporate dynamic information about neighbouring feature vectors in time into the current vector. Define a static feature vector $\boldsymbol{y}_t$ (for example 12 MFCCs as outlined in (b)), then the delta parameters are computed by

$$\Delta \boldsymbol{y}_t = \frac{\sum_{\tau=1}^{D} \tau(\boldsymbol{y}_{t+\tau} - \boldsymbol{y}_{t-\tau})}{2\sum_{\tau=1}^{D} \tau^2}$$

?which is a linear regression using $2D$ data points. This yields the same number of delta coefficients as in the 'static' MFCCs. The delta-delta or acceleration coefficients are obtained by computing the regression values on the first order differentials. The encoding of temporal information into the feature vector is commonly used to counteract the conditional independence of the observation vectors given the state in HMMs (a very poor assumption but one that gives HMMs the efficiency to be used with very large data sets). The size of the feature vector is substantially increased (three-fold) using this technique, consequently increasing the computational complexity. [20%]

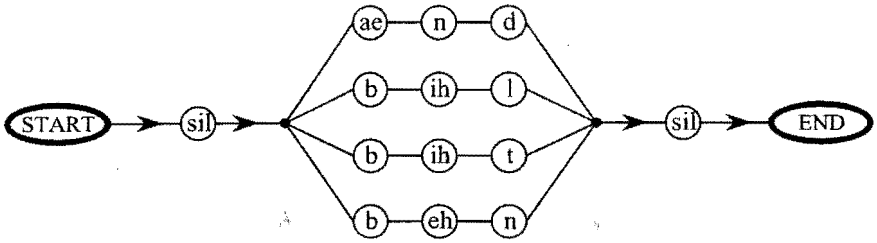
(e) This was not discussed in lectures. The frequency response of a constant linear channel will be multiplicative in the frequency domain and hence additive in the log spectral domain and hence the MFCC domain (since taking the DCT is a linear operation). Hence for channel compensation we need to estimate/remove the constant value. Methods are based on the idea of applying a high pass filter to remove very slow changes in cepstra - and often this can be implemented by simply removing the average value of each cepstral coefficient across a call (which needs to be estimated to be able to run with limited latency). Of course this doesn't affect the values of the delta and delta-delta coefficients. [20%]

Examiner's comments:

The question covered Mel frequency cepstral coefficients, differential coefficients and a part that would have been new to the students on cepstral mean normalisation. Most of this material is fairly straightforward so it was disappointing that there were no excellent answers.

2. *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) To include a bigram language model with a linear lexicon simply need to put the bigram probabilities on the network links between the word ends and the word begin nodes. A back-off node can also be used for improved efficiency. [10%]

(c) 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:
$\phi_1(0) = 1.0$
$\phi_j(0) = 0.0 \quad \text{for} \quad 1 < j < N$ and $\phi_1(t) = 0.0 \quad \text{for} \quad 1 \leq t \leq T$

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

Termination:

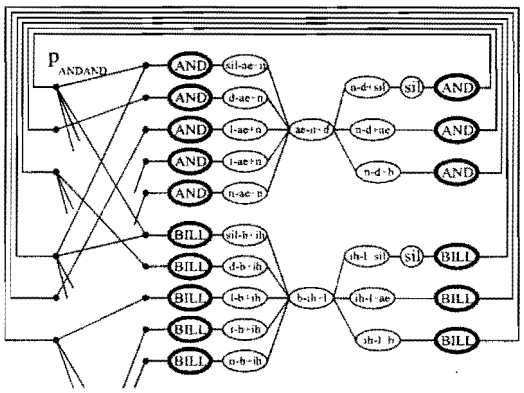
$$p(\mathbf{O}, \mathbf{X}^*|\lambda) = \max_{1<k<N} \phi_k(T) a_{kN}$$

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%]

(d) 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%]

(e)(i) Triphones are context-dependent models of phones in which the model used for a particular phone depends on both the left and right context. Cross-word triphones include context from neighbouring words in determining the HMM to be used. They model co-articulation effects both within and across word boundaries. However they can greatly increase the complexity of the search network. [15%]

(e)(ii) In general need to duplicate the first phone of each word and the last phone of each word (& and single phone words become expanded into n^3 versions if all triphones exist. Note that clustering techniques used in triphone training reduce the actual amount of expansion needed. the simple 4-word vocabulary used in lectures becomes [15%]



(e)(iii) There are several possible approaches to reducing the computational load for cross-word triphones (these were only hinted at in lectures). One approach is to have a multi-pass search in which lattices or n-best lists are created from a simpler system (e.g. using word-internal triphones and a bigram) and then only the reduced search space represented by the lattice is rescored. An alternative is to use a tree-based lexicon - the simple approach to this needs to be modified for cross-word

triphones if a static network is used, but the idea can be used if the network is generated dynamically. Another valid approach is to use WFST based techniques to optimise the complete network for cross-word triphones (although this can interact with pruning strategies that rely on knowledge of word-position). [10%]

Examiner's comment:

. This question covered basic Viterbi decoding, beam search and cross-word triphones. Most candidates knew the basic material covered although the impact of cross-word triphones on recognition search was not in general well-answered.

3. *Machine Translation Assessment and HMM alignment models*

(a) Automatic metrics attempt to capture human preferences in translation quality. The aim is to replace human judgements of translation quality, which are expensive and difficult to obtain. These metrics are used in system optimisation and in measuring progress in system development.

Speech recognition is 'easier' than machine translation in that reordering effects are minor and a single reference transcription usually suffices. Metrics used in translation have to allow for reordering as well as the possibility of multiple, equally valid, translation references and hypotheses. [20%]

(b) Calculation of the BLEU score is described in Lecture 13.

- For each sentence i, and for $n = 1, \ldots, 4$, 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 order, $n = 1, \ldots, N$: $p_n = (\sum_i \bar{c}_n^i) / (\sum_i c_n^i)$
- The BLEU score is

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

(calculation of BP, the Brevity Penalty, is ignored.)

If it was possible to determine automatically where pairs of words or phrases are synonyms, perhaps by consulting an on-line dictionary, the computation of correct n-grams could be modified to allow full or partial credit for hypothesised n-grams which contain synonyms of the reference translations. [20%]

(c)(i) The forward and backward probabilities are defined as $\alpha_j(i) = P(a_j = i, f_1^j | e_1^I)$ and $\beta_j(i) = P(f_{j+1}^J | a_j = i, e_1^I)$. [10%]

(c)(ii)

$$\begin{aligned} P(a_j = i, f_1^J | e_1^I) &= P(a_j = i, f_1^j | e_1^I)\, P(f_{j+1}^J | a_j = i, e_1^I) \\ &= \alpha_j(i)\beta_j(i) \end{aligned}$$

Noting that $P(f_1^J | e_1^I) = \sum_i P(a_J = i, f_1^J | e_1^I) = \sum_i \alpha_J(i)$, [20%]

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

(c)(iii)

$$\#(f \leftrightarrow e) = \sum_{r=1}^{R} \sum_{j=1}^{J^{(r)}} \sum_{i=1}^{I^{(r)}} 1(e = e_i^{(r)})\, 1(f = f_j^{(r)})\, \underbrace{P(a_j^{(r)} = i \mid E^{(r)}, F^{(r)})}_{P(e_i^{(r)} \leftrightarrow f_j^{(r)} | E^{(r)}, F^{(r)})}$$

$$P_T(f|e) = \frac{\#(f \leftrightarrow e)}{\sum_{f'} \#(f' \leftrightarrow e)}$$

[10%]

(c)(iv) The distribution $P_T(f|e)$ could be made to dependend on the context of e based on each sentence that e appears in. Following the approach in triphone models, e could be replaced by $l - e - r$, where l and r are the words preceeding and following e. Clustering techniques used for context-dependent acoustic models with discrete observation distributions could be used here. The distribution $p(a_j|a_{j-1}, I)$ could also be made context dependent, e.g. so that context of word e_{j-1} could be introduced into the alignment model. [20%]

4. *Weighted Finite State Networks*

(a) A complete path through an acceptor can be written as $p = e_1 \cdots e_{n_p}$, where :
- the path p consists of n_p edges, $e_1 \ldots e_{n_p}$
- the path starts at state $i_p = s(e_1)$ where i_p is an initial state
- the path ends at state $f_p = f(e_{n_p})$ where f_p is a final state

The arc weights and initial and final weights combine to form the path weight

$$w(p) = \lambda(i_p) \otimes w(e_1) \otimes \cdots \otimes w(e_{n_p}) \otimes \rho(f_p)$$

- Initial weights and final weights : $\lambda(i_p)$ and $\rho(f_p)$
- $\otimes$ is the product of two weights

To assign a weight to a string, the WFSA combines the weights of all paths which might have accepted a string, as follows:
- Let x be a string constructed from symbols in the input alphabet Σ : $x \in \Sigma^*$
- Let $P(x)$ be the set of complete paths which generate x, i.e. $x = i(e_1) \cdots i(e_{n_p})$
- Let $\oplus$ be the *sum* of two weight values
- The cost assigned to the string x by the transducer is [20%]

$$\bigoplus_{p \in P(x)} \underbrace{\lambda(i_p) \otimes (\otimes_{j=1}^{n_p} w(e_j)) \otimes \rho(f_p)}_{w(p)}$$

(b)(i)

$$[\![C]\!](x) = [\![A]\!](x) \oplus [\![B]\!](x)$$

[10%]

(b)(ii)

$$[\![C]\!](x) = [\![A]\!](x) \otimes [\![B]\!](x)$$

[10%]

(b)(iii)

$$[\![C]\!](x) = \bigoplus_{x_1, x_2 : x = x_1 x_2} [\![A]\!](x_1) \otimes [\![B]\!](x_2)$$

[10%]

(c)(i)

$$\prod_{i=1}^{n} P(u_i, c_i | u_1^{i-1}, c_1^{i-1}) = \prod_{i=1}^{n} P(u_i | c_i, u_1^{i-1}, c_1^{i-1}) P(c_i | u_1^{i-1}, c_1^{i-1}) \approx \prod_{i=1}^{n} P(u_i | c_i) P(c_i | c_{i-1})$$

[10%]

(c)(ii) The distribution $p(c_i|c_{i-1})$ could be realised as an bigram language model. The vocabulary of this LM would be over cased words, i.e. a mix of upper and lower case words. The model parameters can be estimated using any of the backoff and discounting schemes discussed in lecture.

The distribution $p_T(u|c)$ is degenerate, i.e. it is equal to 1 if u is the lower case version of c, and it is equal to zero otherwise. No estimation is needed, although the corpus vocabulary can determine which words c for which the distribution $p_T(u|c)$ can be defined. [20%]

(c)(iii) Defining the following automata:
- An unweighted WFSA U to accept the string u_1^n
- An unweighted transducer T that maps each cased word c to its uncased form u
- A weighted accepted G which implements the bigram language model $p(c_i|c_{i-1})$ with log scores in the tropical semiring

The cased sequence can be produced by carrying out the composition $G \circ T \circ U$, projecting on the input, and finding the short path through the resulting acceptor. [20%]

Examiner's comment:

This question covered the use of semi-ring operations and a model used for true casing in machine translation. In general it was done well