Prof P Woodland

Solutions: 4F11 Speech and Language Processing, 2016

1. *HMM acoustic models*

   (a) HMM assumptions:

   - The observations accurately represent the signal. Speech is assumed to be stationary over the length of the frame. Frames are usually around 25msecs, so for many speech sounds this is not a bad assumption.
   - Observations are independent given the state that generated it. Previous and following observations do not affect the likelihood. This is not true for speech, speech has a high degree of continuity.
   - Between state transition probabilities are constant. The probability of from one state to another is independent of the observations and previously visited states. This is not a good model for speech.

   [10%]

   (b) MFCCs. Take the log energies of the mel-spaced filter-bank energies m_i and apply a discrete cosine transform. Desirable properties: use of mel-filter spacing approximates the resolution of the human ear; MFCCs separate out spectral shape from log energy. This reduces the dimensionality of the feature vector and (mainly) diagonalises it so that diagonal covariance matrices are more appropriate.

   Differential coefficients. Add time differentials of MFCCs and the normalised log energy (or c_0). These can be added with a regression formula and the differential is taken as the best straight line through this window. The second differentials can be added in a similar way. This adds information within a state on time evolution and overcomes (to an extent) the state-conditional independence assumption. [20%]

   (c) The following steps can be used:

   - Using flat start to initialise models (all means the same, common variances, define state transitions).
   - Build single Gaussian monophone models (typically 4 re-estimation iterations).
   - "Clone" monophones for every cross-word triphone context seen in training data.
   - Build single Gaussian unclustered triphone models.
   - Perform state-level decision-tree clustering: generates initial single-component state-clustered triphones.
   - Train single-component models (typically 4 re-estimation iterations).
   - Increase num mix components (iterative component splitting) & retrain at each level for desired complexity.

   Final system is a multi-component state-clustered cross-word triphone system. [20%]

(d) State clustering using phonetic decision trees. A tree grown at the state level (for each of three states left to right models normally used). Simple binary decision tree, i.e. yes/no at each node. There are questions about groups of phonetic classes at each node. The trees are automatically constructed by choosing at each stage the question and node that gives the largest increase in approximate training data log likelihood from making the split. Stopping is when there isn't any node with a large enough gain in log likelihood and/or by a occupancy threshold. The tree divides all contexts into equivalence classes and the leaf nodes represent these (for all triphones including those that don't occur in training). An HMM output distribution is built for each clustered state and the data from all contexts shared.

Advantages: there is no need to back-off for unseen triphones; and allows expert knowledge to be incorporated; and allows any degree of context dependency to be simply incorporated. Disadvantages: requires expert knowledge to specify question set (but not chosen questions!); tree building uses locally optimal decisions. [30%]

(e) This is the standard way that decision tree clustering is performed, but the problem here isn't covered in lectures. So the problem asked for is to estimate the value of $\mathbf{\Sigma}$ and N at any node. To estimate $\mathbf{\Sigma}$ need to have sufficient statistics for the data associated with the node. If there are a set of contexts C associated with the node and we have access to the means and variances for each context and occupation counts then the required value is (for each dimension $\mathbf{o}_d$ of the diagonal covariance matrix) $\mathbb{E}[\mathbf{o}_d{}^2] - (\mathbb{E}[\mathbf{o}_d])^2$ for the data associated with the node.

If there are a set of contexts C associated with the node, each with covariance $\mathbf{\Sigma}_i$, mean μ_i and occupation count N_i we have with a total count N ($= \sum N_i$) in the node then the required value is

$$\frac{1}{N}\sum_{i\in C} N_i[\mathbf{\Sigma}_i + \mu_i\mu_i{}^{\mathsf{T}}] - \left[\frac{1}{N}\sum_{i\in C} N_i\mu_i\right]\left[\frac{1}{N}\sum_{i\in C} N_i\mu_i\right]^{\mathsf{T}}$$

If diagonal covariances are assumed then only need the diagonal values in the above. The main assumption here is that during the clustering and merging statistics that the values of the state/frame alignments don't change so that the original statistics can be combined as above. [20%]

Examiner's Comment: Bookwork in parts (a) and (b) covered well. The training of acoustic models from scratch in (c) required putting information from various parts the course together and was less well done. The standard decision tree based state clustering approach was generally fairly well described. However, only a couple of candidates made good attempts at the final part of the question (not covered in lectures) which required understanding of how to obtain the correct sufficient statistics for state clustering at any level of tree building from the untied model single Gaussian statistics.

2. *N-gram language models*

(a) Language models for speech recognition predict the probability of the next word from a history. It is usual to restrict the size of the history to the previous $N-1$ words. This is the N-gram language model. Thus

$$P(w(k)|w(1)...w(k-1)) \approx P(w(k)|w(k-N+1)...w(k-1))$$

Most frequently used are the bigram ($N=2$) and trigram ($N=3$) and 4-gram ($N=4$) LMs. They are effective because they are simple to use in the search, they capture the most important local dependencies (including semantics and syntax) and can be trained on large amounts of data. [15%]

(b) Maximum likelihood estimation isn't used directly since this would be based purely on relative frequency estimates and so any N-grams not occurring in training would give rise to zero probability estimates which in turn would lead to recognition errors.

Hence need to assign some probability "mass" to unseen events by reducing the counts from seen events (discounting). Then for seen events, relative-frequency based estimates of the discounted N-gram counts are used. The N-gram estimate is modified to be e.g. for bigram

$$\hat{P}(w_k|w_i,w_j) \quad = \quad d(f(w_i,w_j,w_k))\frac{f(w_i,w_j,w_k)}{f(w_i,w_j)}$$

where $d(r)$ is a *discount coefficient*. The amount by which the maximum likelihood estimate is altered depends on the count, r, of the N -gram. Typical discounting schemes include Good-Turing and absolute discounting.

Backing off is the use of a more general distribution suitably normalised (using a back-off weight) when the N-gram is not seen (often enough) in training.

$$\hat{P}(w_j|w_i) = \begin{cases} d(f(w_i,w_j))\frac{f(w_i,w_j)}{f(w_i)} & \text{if } f(w_i,w_j) > C \\ \alpha(w_i)\hat{P}(w_j) & \text{otherwise} \end{cases}$$

$\alpha(w_i)$ is the *back-off* weight, it is chosen to ensure that

$$\sum_{j=1}^{V} \hat{P}(w_j|w_i) \quad = \quad 1$$

and C is the N-gram cut-off point (i.e. only N-grams that occur more frequently than this are retained in the final model). [30%]

(c) Trigram search using context-dependent phone HMMs. The language model probabilities are included in the phone level search network. The word models are

formed by connecting the phone models in accordance with the pronunciation dictionary. Language model probabilities are included between word transitions. This is fairly straightforward for a bigram model and doesn't greatly increase the size of the network (although more connections between nodes). For a trigram the possibilities are

- Add trigram probabilities into the full static search network. However for non-backed-off trigrams need to remember the two-word context in the search and this can lead to a very large search network. Note that the network can be optimised using weighted finite state transducer operations to reduce the size.
- Use a multi-pass search in which as a first pass a network is generated with a bigram model and then from the resulting word lattices (or N-best lists), language model rescoring applied with the trigram (or other) language model.

There are are also several other possibilities. These include various types of dynamic network decoder (incremental generation of the search network) and various hybrid approaches. Network optimisation can also use WFSTs. These are not covered in detail in lectures but would receive credit. [20%]

(d)(i) Use of a bigram will reduce the size but also increase the perplexity and hence the error rate. However that the search architecture is simpler for a bigram. For a well-trained trigram might reduce the error rate by about 15%. [10%]

(d)(ii) This is the usual way of controlling size and can lead to large reductions in size since N-grams that occur only once or twice are in the majority. Typically large reductions in size can he achieved with very little increase in perplexity and sometimes none in word error rate. [10%]

(d)(iii) This is a form of what is known as entropy-based pruning (not covered in lectures), and is more effective still than count-based pruning in reducing size, since it least disturbs the probability estimates in the model. The "significantly" higher implies that there is a threshold that determines how much of an increase in model entropy will be allowed. The method can have only very small increase in perplexity and word error rate. [15%]

Examiner's Comment: The most popular question. The portions on search algorithms (for a trigram language model and context dependent phone hidden Markov models) was the least well done part of the question. The questions on limiting the number of parameters in part (d) had rather mixed attempts with only a very few candidates able to discuss the entropy-based pruning covered in the final part (not covered in lectures).

3. *Weighted Finite State Acceptors (WFSAs)*

(a) A weighted acceptor over a finite alphabet Σ is a finite directed graph with a set of nodes Q and a set of arcs (edges) E. A path through the acceptor can be written as a sequence of edges $p = e_q \cdots e_{n_p}$; the path starts at state $i_p = s(e_1)$, where i_p is a start state; the path ends at state $f_p = s(e_{n_p})$, where f_p is a final state.

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

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

based on initial weights $\lambda(i_p)$ and final weights $\rho(f_p)$.

To assign a weight to a string, the weights of all paths which might accept or generate a string are added, as follows: for a string $x \in \Sigma^*$, let $P(x)$ be the set of paths which generate $x : x = i(e_1) \cdots i(e_{n_p})$. The cost assigned to the string x is

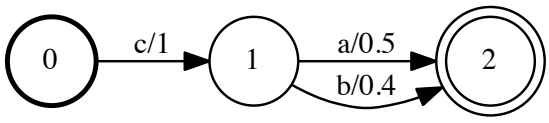
$$\bigoplus_{p \in P(x)} w(p) = \bigoplus_{p \in P(x)} \lambda(i_p) \otimes w(e_1) \otimes \cdots \otimes w(e_{n_p}) \otimes \rho(f_p)$$

[20%]

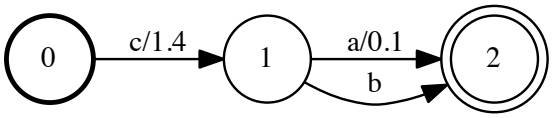
(b) For an arbitrary semiring the distance is: $d[0,3] = (1 \otimes 1 \otimes 1) \oplus (1 \otimes 2 \otimes 1)$
In the log semiring: $d[0,3] = (1+1+1) \oplus_{\log} (1+2+1) = -\log(e^{-3} + e^{-4})$
In the tropical semiring: $d[0,3] = (1+1+1) \min(1+2+1) = 3$ [20%]

(c) A determinised WFSA has the property that no two transitions leaving a state share the same input label. This is violated by the arcs leaving states 0, 1, and 2.

Determinisation preserves the weights assigned to strings under the semiring. A determinised verison is



Determinised machines need not be unique. Arc weights can be moved arbitrarily so long as the weights assigned to strings are preserved, as in this machine



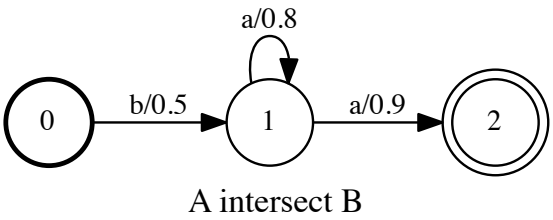
[30%]

(d)

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

[10%]

(e) A WFSA $C = A \cap B$ is



[20%]

Examiner's Comment: This was the least popular question but with some perfect solutions. Most answers to the straightforward questions on semiring and intersections definitions were done correctly. However the questions requiring the application of core algorithms, such as shortest path and determinisation, to WFSAs were not answered as well.

4. *Machine Translation*

(a) Words in the foreign sentence which don't have a direct correspondence in the source language are aligned to NULL. For example, the phrase 'Monsieur le Orateur' is a translation of the phrase 'Mr. Speaker' , but at the word level, 'le' is aligned to NULL in English. [10%]

(b) Making simplifying conditional independent assumptions:

$$\begin{aligned} P(f_1^J, a_1^J, J|e_0^I) &= P(f_1^J|a_1^J, J, e_0^I)\, P(a_1^J|J, e_0^I)\, P(J|I) \\ &= \prod_{j=1}^{J} P_T(f_j|e_{a_j})\, P_A(a_1^J|J, I)\, P_L(J|I) \end{aligned}$$

where $P_L(J|I)$ is the Sentence Length Distribution, $P_T(f|e)$ is the Word Translation Distribution, and $P_A(a_1^J|J, I)$ is the Word Alignment Distribution. [20%]

(c)

- Model 1:
$$P_A(a_1^J|J, I) = \frac{1}{I^J}$$
or $\frac{1}{(I+1)^J}$, to allow for alignment to NULL
- Model 2:
$$P_A(a_1^J|J, I) = \prod_{j=1}^{J} p_{M2}(a_j|j, J, I)$$
- HMM:
$$P_A(a_1^J|J, I) = \prod_{j=1}^{J} p_{HMM}(a_j|a_{j-1}, I)$$

Model 1 assumes that the link distribution is entirely flat, so that all positions are equally probable. Model 2 assumes that the link distribution depends on the foreign word location j. The HMM assumes that the link distribution depends on the link of the previous word, i.e. that the alignment sequence has a history of one. [20%]

(d)

$$
\begin{aligned}
P(f_1^J|e_0^I) &= \sum_{a_1^J} P(f_1^J, a_1^J|e_1^I) = \sum_{a_1=0}^{I} \cdots \sum_{a_J=0}^{I} P(f_1^J, a_1^J|e_0^I) \\
&= \sum_{a_1=0}^{I} \cdots \sum_{a_J=0}^{I} \prod_{j=1}^{J} p_{M2}(a_j|j, I, J)\, p_T(f_j|e_{a_j}) \\
&= \sum_{a_1=0}^{I} p_{M2}(a_1|1, I, J)\, p_T(f_1|e_{a_1}) \times \\
&\qquad \sum_{a_2=0}^{I} p_{M2}(a_2|2, I, J)\, p_T(f_2|e_{a_2}) \times \\
&\qquad\qquad \cdots \times \sum_{a_J=0}^{I} p_{M2}(a_J|J, I, J)\, p_T(f_J|e_{a_J}) \\
&= \prod_{j=1}^{J} \sum_{i=0}^{I} p_{M2}(i|j, I, J) p_T(f_j|e_i)
\end{aligned}
$$

[20%]

(e)(i) The word translation distribution in Model 1 is initialised to a uniform value. With iterations of the EM or Viterbi algorithm, the word translation distribution sharpens based on word pairs that tend to occur together in translated sentences. The word translation distribution from Model 1 can be used with Model 2, with the word alignment distribution set to a uniform value. As with Model 1, iterations of the EM or Viterbi algorithms refine word alignment distributions. [10%]

(e)(ii) In the context of SMT, 'phrases' are word sequences extracted from sentences. A phrase is sequence of words that can be translated. It need not correspond to a syntactic unit in either the source or target language.

Phrase pairs are extracted from word-level alignments, typically generated using a model estimated from a flat-start over the parallel text. Phrases pairs cover patterns of word alignments that are observed the automatically aligned bitext. Phrase pairs are extracted following heuristics, such as 'Two phrases are aligned if their words align only with each other'. [20%]

Examiner's Comment: Straightforward questions about the formulation of the word alignment model were answered correctly by many students. The most challenging question was the relatively straightforward derivation of IBM Model 2 posteriors, which followed fairly directly from the examples paper.