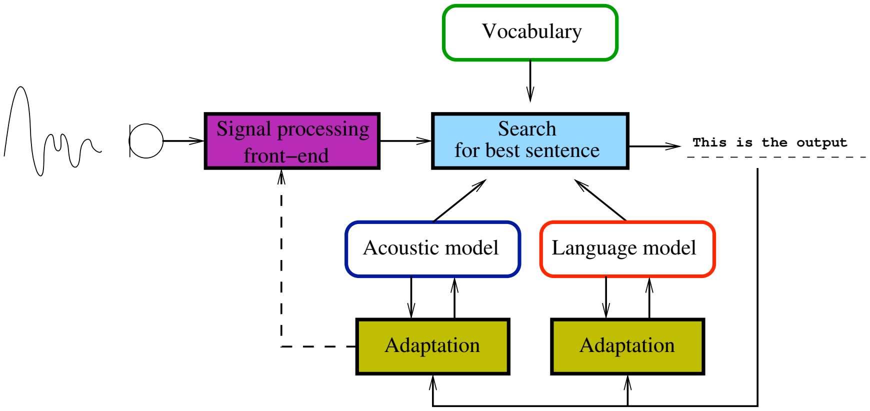
Solutions: 4F11 Speech and Language Processing, 2015

1. *Hidden Markov Models for Speech Recognition*

(a) The generic speech recogniser is



The primary models are the *acoustic* model and the *language* model. The language model assigns a probability $P(W)$ to sentence hypotheses W constructed from words in the *vocabulary* (which also defines the pronunciation in terms of the acoustic model units). The acoustic model assigns likelihood $p(O|W)$ to the acoustic observation sequence which is produced by the *front-end* (e.g. MFCCs). The search process finds the most likely string. *Adaptation*, if used can be performed to tune the parameters of the acoustic and language models. [15%]

(b) 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%]

(c)(i) The forward probability is defined as

$$\alpha_j(t) = p(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t, x(t) = j|\lambda)$$

where $x(t)$ denotes the state occupied at time t, and λ represents the HMM parameter set.

The backward probability, $\beta_j(t)$, is defined as

$$\beta_j(t) = p(\boldsymbol{o}_{t+1}, \ldots, \boldsymbol{o}_T | x(t) = j, \lambda)$$

$\alpha_j(t)$ can be computed efficiently

$$\alpha_j(t) = \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} b_j(\boldsymbol{o}_t)$$

where $\alpha_i(0) = 1$ for $i = 1$ and zero otherwise. Recursion moves forwards in time.

$\beta_j(t)$ can be computed efficiently

$$\beta_j(t) = \sum_{i=1}^{N} a_{ji} b_i(\boldsymbol{o}_{t+1}) \beta_i(t+1)$$

where $\beta_j(T+1) = a_{jN}$. Recursion moves backward in time. Note $p(\mathbf{O} = \beta_1(0)$. [25%]

(c)(ii)

$$\begin{aligned}
\alpha_j(t)\beta_j(t) &= p(\boldsymbol{o}_1 \ldots \boldsymbol{o}_t, x(t) = j\,|\lambda) p(\boldsymbol{o}_{t+1} \ldots \boldsymbol{o}_T\,|x(t) = j, \lambda) \\
&= p(\mathbf{O}, x(t) = j\,|\mathcal{M}) \\
&= p(\mathbf{O}\,|\lambda) P(x(t) = j\,|), \lambda) \\
&= p(\mathbf{O}\,|\lambda) L_j(t)
\end{aligned}$$

Hence,

$$L_j(t) = \frac{1}{p(\mathbf{O}\,|\lambda)} \alpha_j(t)\beta_j(t)$$

(d) A composite HMM is constructed for each sentence in the training corpus according to the word level transcription (allowing for optional silences), and the pronunciation dictionary. If there isn't a single pronunciation per word then the composite HMM has branches and either the *Viterbi algorithm* can be used to select pronunciation variants (given some initial models) or run forward-backward on the network. Note that this is also an issue for optional inter-word silences. Then the forward-backward algorithm is run on the complete sentence-level composite HMMs and suitable statistics for HMM training found. The HMMs can be initialised typically either by a flat start (all parameters equal, just define left-to-right topology) or by using some estimates of HMM parameters from a phone-level labelled set of data. [20%]

(e)(i)/(ii) Forward probability pruning. Calculate the maximum value of $\log \alpha_j(t)$ at each time and only consider models which have a maximum value $\log \alpha_j(t)$ above a threshold. The same method can be applied on the backwards pass to the $\log \beta_j(t)$ values. The state posterior is found from $\alpha_j(t)\beta_j(t)$ and a threshold applied to this from the maximum value gives a very tight beamwidth. In practice the $\alpha_j(t)$ pruning results in a beam a few words wide and the posterior pruning is then one or two phones wide. Note that this is very important for long utterances since only a few percent of the models can be active and greatly speeds training. [20%]

2. *Large vocabulary recognition*

(a) MFCCs - take cosine transform of log filterbank energies (assuming that the filterbank is on a mel scale). Might take a vector of 24 energies and compute 12 MFCCs. If P channels and m_j is the energy of the jth channel then the MFCC component

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

This will reduce computation (smaller feature vectors) and storage. It will also increase accuracy as diagonal covariance matrices are a better approximation. [15%]

(b) It is proposed to add MFCC time differentials to the feature vector. Normally these are smoothed over several frames. They take account of the first-order local dynamics to account (in part) for the poor HMM assumptions. The change will increase accuracy but make the feature vector two or three times as large and will increase computation (unless pruning is more effective due to increase in accuracy). [15%]

(c) The state distribution is not completely uncorrelated and may be non-Gaussian. Better to use a Gaussian mixture with

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

c_{jm} is the component weight, or prior. For this to be a pdf it is necessary that

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

This increases computation and storage roughly proportional to the number of components but also reduces word error rate (if there is enough training data). [20%]

(d) 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 is greatly increased and some type of smoothing or parameter sharing is required. Here it is suggested that state-tying via phonetic decision trees are used. These group contexts into equivalence classes so that models can be robustly estimated for the grouped contexts and importantly triphones unseen in the training data can also 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 based on single Gaussian statistics using a greedy node splitting criterion. The questions for splitting contexts are chosen from a pool of linguistic questions which yield generalisation ability. Normally a threshold on the number of frames associated with each leaf node is also set.

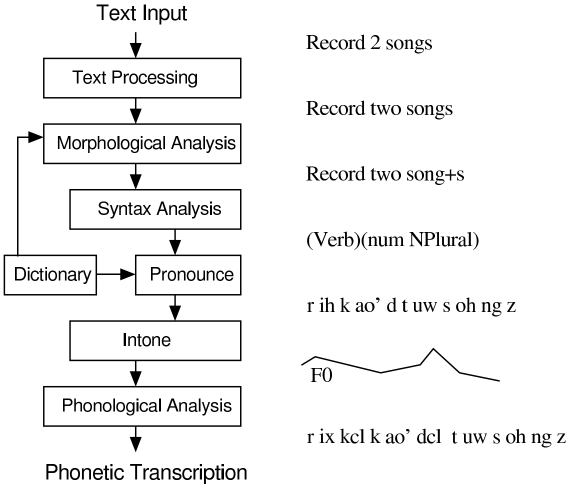
For cross-word triphones (from monophones) decoding is greatly complicated and computational cost increases (and storage since there will usually be far more parameters even with tying). It can have a large decrease in word error rate (factor of two or more). [25%]

(e) Use of a trigram language model. This uses a 2 word history to predict the next word (the prior probability in recognition). Typically trigram language models use back-off or interpolation for robust estimation. Furthermore they will be reduced in size to avoid having to store too many trigrams. During recognition the two-word (rather than single word) context needs to be made explicit in the network. There are a number of ways that this can be approached to efficiently decode. First is to generate word lattices with a bigram model and then rescore these with a trigram model. This only requires expansion of the network structure that is present in the lattice and is hence much more efficient. The second is to use a Weighted Finite State transducer model to optimise the overall structure which can greatly compress the structure. In either case (or a combination) the memory use for the language model goes up significantly but pruning is more effective and the run-time may be similar. However the word error rate is often significantly reduced. [25%]

3. *Text-to-speech*

(a) The input text can come from arbitrary sources and must be normalised prior to synthesis. The text must be transformed into a high level description of speech sounds that can be used to synthesis signals. The output of the analysis stage is a sequence of phonetic symbols accompanied by intonation information. The output of the second stage is the synthesized speech signal. [10%]

(b)(i) Linguistic analysis stages block diagram (drawn from Lecture 14): [15%]



(b)(ii) The text preprocessing stage is needed for text normalisation; the morphological analysis prepares for pronunciation and for syntactic analysis; syntactic analysis is done to aid part-of-speech tagging, to help determine pronunciation of homographs; pronunciation generation produces pronunciation for words in isolation. The syntactic analysis and pronunciation is used to generate intonation patterns (energy, F0, duration). The final stage accounts for cross-word articulatory effects. [15%]

(c) A synthesis unit is the basic entity for which parameters are stored in the speech synthesis stage. Longer units lead to few artefacts due to concatenation effects, but can be difficult to generalise; shorter units can generalise better, but care must be taken in synthesizing larger units from smaller units, due to the artefacts mentioned above. [15%]

(d) The question asks for two ways in which ASR and TTS differ in their use of HMMs. Some possible reasons include the following: [20%]

- For an acoustic sequence O_1^T, in ASR we find the most likely sequence as

  $$\underset{W}{\text{argmax}}\ P(W|O_1^T) = \underset{W}{\text{argmax}}\ P_{\text{HMM}}(O_1^T|W)\ P_{\text{LM}}(W)$$

  whereas in TTS, for a word sequence W , we find the most likely acoustic sequence $\widehat{O_1^T}$ as

  $$\underset{O_1^T}{\text{argmax}}\ P(O_1^T|W) = \underset{O_1^T}{\text{argmax}}\ P_{\text{HMM}}(O_1^T|W)$$

note that there is no language model, and that the HMM is used directly as a generative model.

- In HMM-based TTS, the state sequence Q through the composite HMM is defined similarly as in ASR. However, state durations are determined by models for phone/state duration:

$$P(Q|\lambda) = \prod_{i=1}^{N} p_i(d_i)$$

- HMMs used in TTS tend not to have mixture components, i.e. the state observation distribution is a single Gaussian
- F_0 is modelled as a separate stream in TTS; in ASR, it is often ignored
- state clustering questions used in TTS can address much longer context, e.g. at the sentence, phrase, syllable or even corpus level.
- In ASR, the cepstral sequence and its delta coefficients are often treated as conditionally independent given the state sequence, whereas they are often modelled jointly in TTS (see next question)

(e) Cepstral sequences are generated given the HMM state sequence $\widehat{Q}$ as

$$\underset{O_1^T}{\operatorname{argmax}}\ P(O_1^T|W) = \underset{O_1^T}{\operatorname{argmax}}\ P_{\text{HMM}}(O_1^T|W)$$

Recall that $P_{\text{HMM}}(O_1^T|Q_!^T) = \prod_{t=1}^{T} P(O_t|Q_t)$, it becomes clear that the sequence is generated as $\operatorname{argmax}_{O_t} P(O_t|Q_t)$, which will be piece-wise constant, changing only if $Q_t \neq Q_{t+1}$. For single Gaussian observaiton distributions, the output is simply a sequence of the mean-value vectors of the Gaussians associated with each state.

If a matrix W is used to enforce the relationship $O_1^T = Wc_1^T$, which c_t is a cepstral feature vector, and O_t is the feature vector at time t containing the dynamic features, then the observation sequence can be produced as $O_1^T = W\widehat{c_1^T}$ where

$$\widehat{c_1^T} = \underset{c_1^T}{\operatorname{argmax}}\, P(Wc_1^T|Q_1^T)$$

This will produce a smoother cepstral sequence, as the differentials must also have high likelihood under the HMM. [20%]

(f) In ASR, the search algorithm must apply the HMM clustering questions during search as the word hypotheses are constructed. This can be done efficiently with e.g. triphones, but cannot be easily extended to e.g. phrase or sentence level features. In TTS, the model is conditioned on a single given word sequence, and so HMM state clustering can make use of much broader context. [10%]

4. *Machine Translation*

(a) A Semiring is defined by a sum $\oplus$ operation, a product $\otimes$ operation, and two identity elements $\bar{0}$ and $\bar{1}$ such that:
- for a weight $k \in \mathbb{K} : \bar{0} \oplus k = k$; $\bar{1} \otimes k = k$; $\bar{0} \otimes k = \bar{0}$
- $\oplus$ and $\otimes$ distribute and commute in the familiar way. [10%]

(b) The completed version of Table 1 is as follows:

Semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, \infty\}$	$\oplus_{\log}$	$+$	∞	0
Tropical	$\mathbb{R} \cup \{-\infty, \infty\}$	min	$+$	∞	0

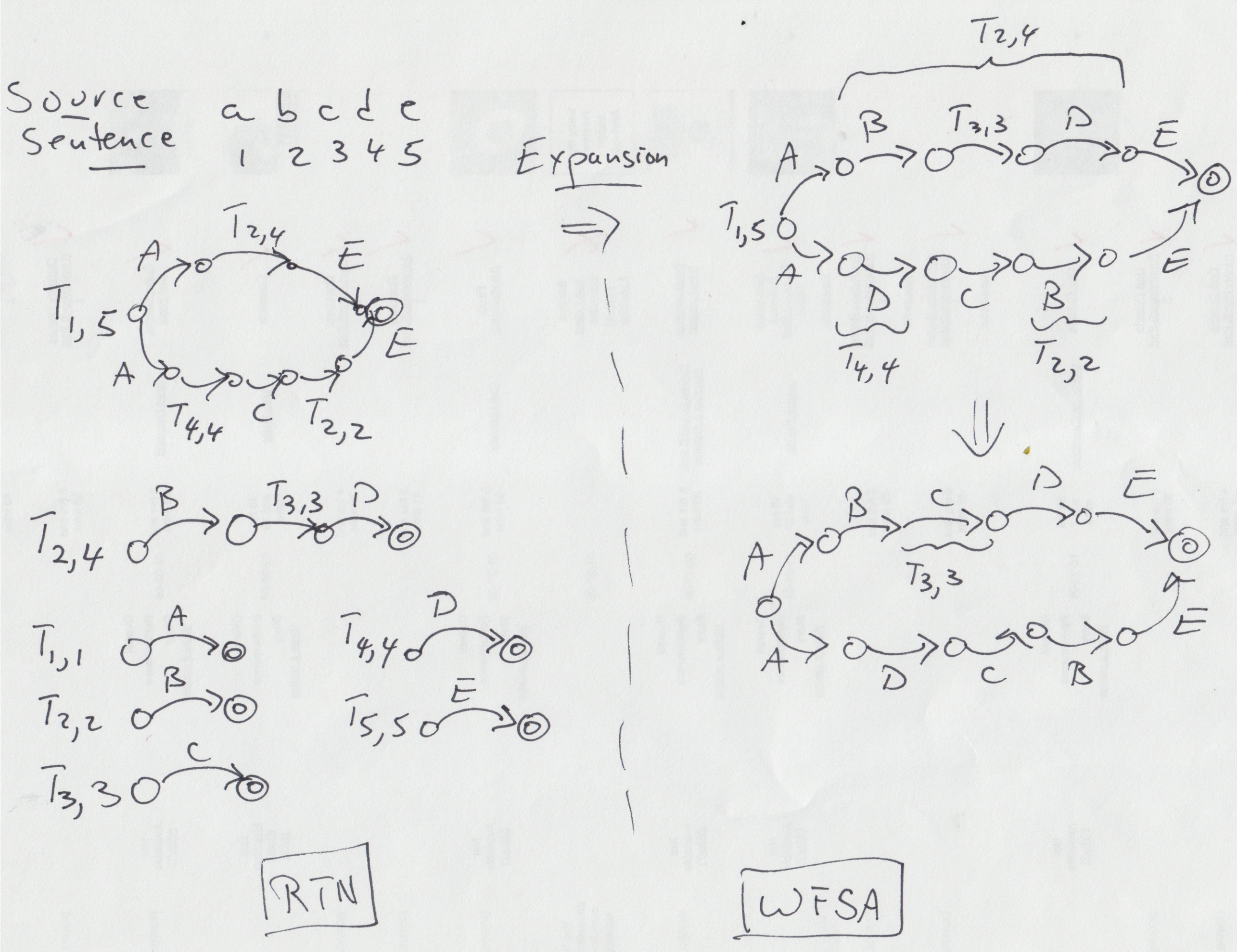
$\oplus_{\log} : k_1 \oplus_{\log} k_2 = -\log(e^{-k_1} + e^{-k_2})$

[20%]

(c) To convert the WFSA, it is enough to replace each arc weight in the probability semiring, which take values in $[0, 1]$, by its negative log ($-\log$). These weights are appropriate for both the Log and Tropical semiring.

The shortest path operation should yield the same result under the Probability and the Log semirings. This cost assigned to a string is accumulated over all paths that might accept the string. Under the tropical semiring, the shortest path can yield a different result, corresponding to the cost of the best single path accepting a string. This corresponds to the difference between the marginal probability and the Viterbi score, which agree if the accepting path is unique. [20%]

(d) (i)/(ii) A sketch of the RTN and the conversion to WFSA are given below.



[20% for each part]

(e) Advantages of using translation grammars are:

- Syntax can capture some regular structure in translation between languages, e.g. `ne X1 pa` → `not X1`, and `le X2 X1` → `X1 of the X2`.
- The use of non-terminals allows control over very long-distance movement of words and phrases.

[10%]

4F11 Assessor’s comments

Q1: Hidden Markov Models
The question covered the basics of the use of hidden Markov models (HMMs) in speech recognition, including maximum likelihood training. The final parts discussed (i) the use of sub-word HMMs on a corpus labelled at the word level and (ii) possible efficiency improvements using pruning (not covered in lectures).

Q2: Large Vocabulary Speech Recognition
This question covered the use of mel frequency cepstra and differentials, Gaussian mixture distributions, cross-word triphones with decision tree state tying and trigram language models. This question was attempted by all candidates with some very good answers.

Q3 Speech Synthesis
A rather unpopular question. This question covered the architecture of text-to-speech synthesis systems, linguistic analysis and the use of hidden Markov models (HMMs) in speech synthesis. The material on HMM synthesis was taught for the first time this year and this probably explains
why there were few attempts.

Q4 Weighted Finite State Acceptors
This question covered semirings, and the use of translation grammars.