Chapter 4

Speech and Writing

4.1 Speech

Speech recognition has been an intense area of research for decades, and is now a commonplace feature in PCs and mobile devices. For a long time, there was a fairly standardized approach using hidden Markov models (§4.1.2) that was modified to use neural networks (§4.1.2) and remains (I believe) the state of the art. However, a lot of progress is being made on *end-to-end* neural models (§4.1.3) that are much simpler.

4.1.1 Preprocessing

Given a sample of speech, we can convert it into a sequence of real-valued 39-dimensional vectors, called the *mel-frequency cepstral coefficients*:

1. Slice the signal into overlapping frames, typically 25 ms wide and starting every 10 ms.
2. Perform a Fourier transform on each frame, which is the amount of power at each frequency.
3. Compute how much power there is in each of several frequency bands arranged according to the *mel scale*, which is supposed to match human perception of pitch.
4. Take the log of each power, which matches human perception of loudness.

Sometimes, two additional steps are taken:

5. Take a *discrete cosine transform*, which removes correlations between the components.
6. Also compute the change of the components over time ($\Delta c_t = c_{t+2} - c_{t-2}$), and the change of the change ($\Delta^2 c_t = \Delta c_{t+1} - \Delta c_{t-1}$), for a total of 39 components.

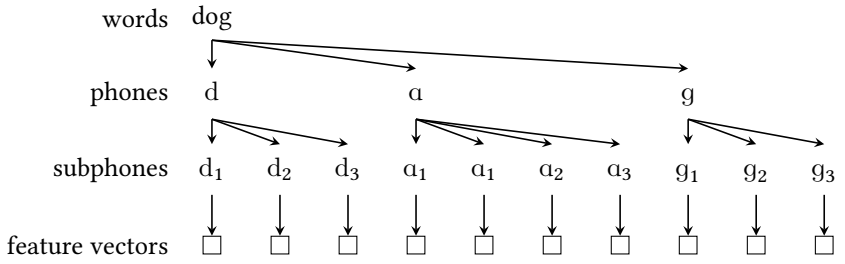


Figure 4.1: Overview of HMM-GMM.

4.1.2 Hidden Markov Models

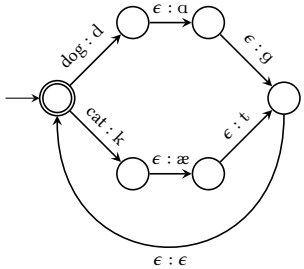
A speech recognizer using HMMs is a noisy-channel model, which can be formulated in terms of finite automata. We give only a brief overview here. For a more detailed treatment, see the survey by Mohri, Pereira, and Riley (2002).

Given a sample of speech, we want to find the most likely sequence of words that this speech came from. To do this, we build a cascade of transducers that maps from text to speech (feature vectors):

1. Generate words using an *n*-gram language model.
2. Map words to their pronunciations, which are strings of *phones*.
3. Divide each phone into *subphones*, and stretch out each subphone to last for one or more frames.
4. For each frame, map the subphone to a feature vector.

Words to phones

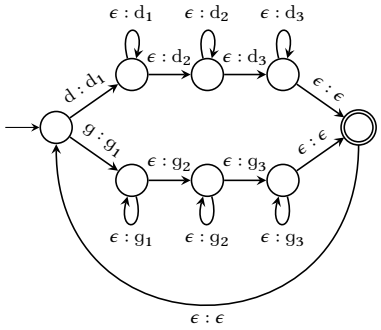
We've seen automata for *n*-gram models already. To maps from words to their pronunciations (shown just for two words), we can use a *finite transducer*. In a finite transducer, each transition has both an input and an output symbol, so the machine describes a mapping from input strings to output strings.



Hand-built dictionaries are available (e.g., the CMU pronunciation dictionary), or a grapheme-to-phoneme model could be used or even learned automatically.

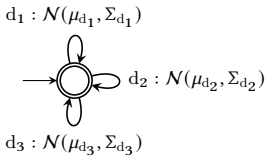
Phones to subphones

The next stage divides each phone into *subphones* and decides the duration of each subphone. Typically, each phone is divided into three subphones; for example, phone d would be divided into d_1, d_2, and d_3. The duration of each subphone is one or more time slices. So the transducer looks like this (shown just for two phones):



Subphones to feature vectors: GMMs

Finally, we map from the sequence of subphone symbols to the observed sequence of feature vectors. The problem is that the vectors have real-valued components, but we only know how to define transducers with a finite output alphabet. So we have to generalize our notation. We need something like this (shown just for one phone, d):



This means that the transducer can read symbol d_1, and outputs a 39-dimensional vector drawn from the multivariate normal distribution, $\mathcal{N}(\mu_{\mathrm{d}_1}, \Sigma_{\mathrm{d}_1})$. The μ's and Σ's are parameters to be learned.

Because the sound of a subphone might not be modeled well by a multivariate normal distribution, we can include multiple transitions for each subphone, each with a multivariate normal distribution with different parameters. This is called a *Gaussian mixture model* (GMM).

Subphones to feature vectors: neural networks

A so-called *hybrid* approach (between statistical and neural approaches) is (I believe) the current state of the art in speech recognition (Dahl et al., 2012).

In this approach, the last step (subphones to feature vectors) is replaced with a neural network. Since the overall model is a noisy-channel model, the last step is a probability distribution $P(\text{vectors} \mid \text{subphones})$. When replacing it with a neural network, however, we "unreverse" the direction of this model, making it a

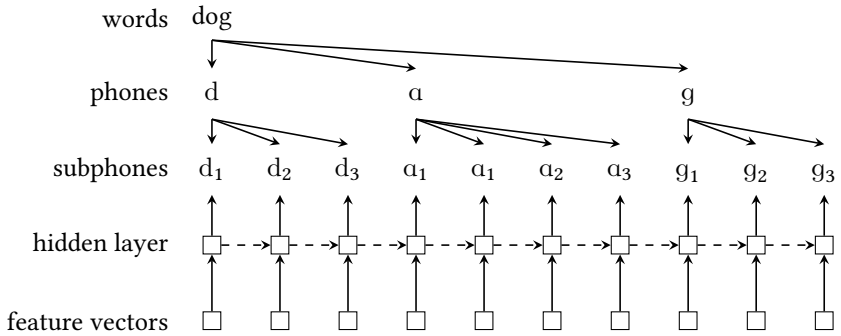


Figure 4.2: Overview of hybrid HMM-DNN. Some models have the dashed connections and some don't.

probability distribution $P(\text{subphones} \mid \text{vectors})$. This means that the recognizer wants to find

$$\arg\max_{\text{words}} P(\text{words} \mid \text{vectors}) \approx \arg\max_{\text{words}} P(\text{words}) \cdot P(\text{phones} \mid \text{words}) \cdot P(\text{subphones} \mid \text{phones}) \cdot P(\text{subphones} \mid \text{vectors}).$$

Mathematically, this doesn't make very much sense, but it works (Figure 4.2).

The neural network can be a feedforward neural network (tanh layers followed by a softmax layer) or a recurrent neural network (Graves, Jaitly, and Mohamed, 2013). Additionally, the language model could be replaced with an RNN language model or a combination of an RNN language model and an n-gram language model.

4.1.3 End-to-End Neural Models

End-to-end neural models take feature vectors as input and predict words as output. They dispense with the discrete cosine transform and Δ's in preprocessing, and they dispense with the multiple, separately-trained stages of the HMM approach. In particular, they dispense with the word-to-phone model, which often had to be constructed by hand.

Attention-based models

One family of end-to-end approaches essentially applies models from machine translation to speech, either the RNN plus attention model (§3.6.3) or the Transformer (§3.6.4). Various modifications can be made to account for differences between the speech and translation problems.

First, the sequence of input vectors is typically much longer than the sequence of output characters. Although this isn't a fatal problem, it can be helpful to reduce the input length, for example, by using a "pyramidal" structure in the encoder like this (Chan et al., 2016):

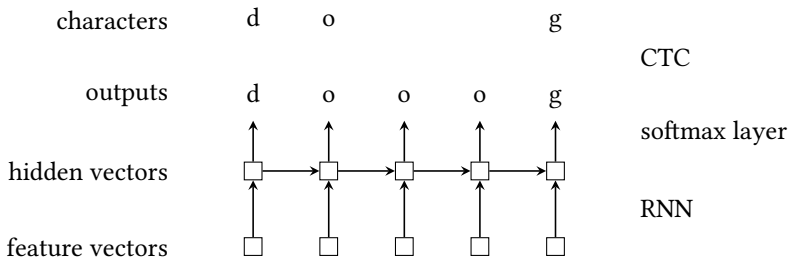
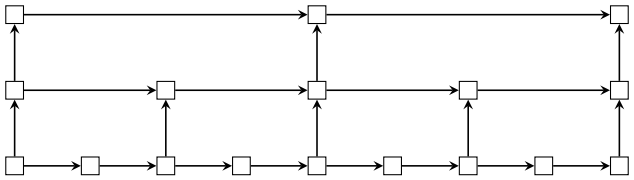


Figure 4.3: RNN with connectionist temporal classification (Graves, Fernández, and Gomez, 2006).

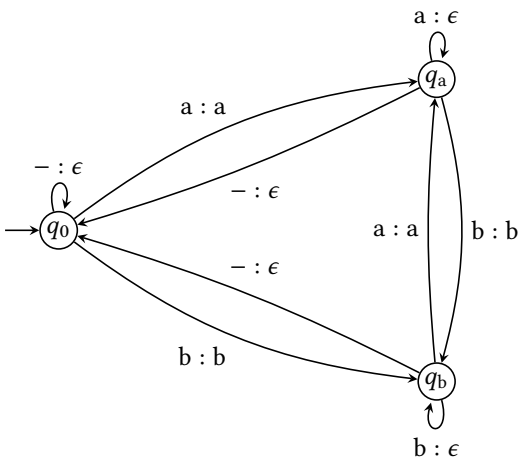


Second, while reordering is extremely common in translation, it's nearly impossible in speech recognition. ("Nearly" because maybe one could argue that some words, like *comfortable*, are spelled and pronounced in different orders.) We turn to this problem next.

RNN and CTC

We can avoid reordering using models that use RNNs without attention. The simplest and oldest such approach uses a RNN to map the sequence of input vectors into another sequence of hidden vectors, of the same length, and then a softmax layer to make a sequence of predicted outputs (Figure 4.3). The predicted outputs are a sequence of outputs, but repeated characters are allowed (to account for the length difference between the input and output sequences). To represent genuinely repeated characters, as in *bot* versus *boot*, the model can also predict the special character −. That is, bbooott represents *bot*, but bboo-ott represents *boot*.

To remove the repeated characters (and −), we feed the output sequence through a finite transducer more commonly known as *connectionist temporal classification* or CTC (Graves, Fernández, and Gomez, 2006). Here's what this transducer looks like for just the alphabet $\{a, b\}$:



Another approach known as a *RNN transducer* or RNN-T (Graves, 2012) uses two RNNs that are synchronized using a finite transducer. What these approaches (as well as the HMM approaches of §4.1.2) have in common is that there are multiple possible alignments between the input vectors and output characters, which are represented by multiple paths through a finite transducer.

4.2 Character and Handwriting Recognition

Optical character recognition and handwriting recognition are the task of converting images containing printed and handwritten text (respectively) into text. For cleanly printed text in high-resource languages, this is a fairly mature technology, but in other settings, this continues to be an active area of research.

4.2.1 Preprocessing

As in speech, we want to turn image data into a sequence of vectors:

1. De-skewing: The image is rotated so that the lines are horizontal
2. Line finding: The image is sliced up into horizontal lines of text. This can be done using a hidden Markov model, but we don't discuss this, since this is more of a vision problem than a language problem.
3. Slice each line in vertical frames (see Figure 4.4).

Newer neural models can learn to perform these steps using attention, making preprocessing unnecessary (Bluche, 2016).

4.2.2 Hidden Markov models

In the previous decade, many statistical writing recognition systems were based on speech recognition systems. Here's one example, selected for its similarity to the HMM-GMM speech model presented in the last chapter, BBN's BYBLOS system (Bazzi, Schwartz, and Makhoul, 1999).

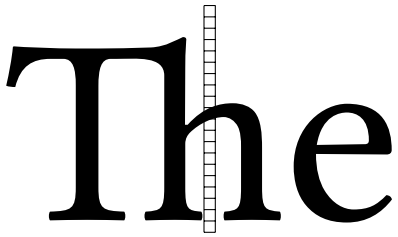


Figure 4.4: A frame and cells.

First, just as older speech systems did some transformations (discrete cosine transform and deltas) to make the vectors easier to model, the frames are transformed as follows:

- For each vertical position, how dark the frame is at that position.
- Deltas of the above, in both horizontal and vertical directions.
- Local slope and correlation across a 2×2 window. (I'm not really sure how these are computed.)

In all, there are 80 features for each frame.

Now the image has been converted into a stream of vectors, just like speech, and we can train and use a speech-recognition system on it. BYBLOS was an HMM-GMM system (§4.1.2), with a few differences:

- The language model is the same, a n-gram model over words.
- The "pronunciation" model just maps words to their spellings (trivial).
- Instead of dividing each phone into three subphones, we divide each character into 14 subcharacters. The transducer that models subcharacters can skip subcharacters, but it can't skip two in a row.
- Finally, each frame is mapped to a feature vector using a Gaussian mixture model.

4.2.3 End-to-end neural models

Just as end-to-end neural speech models are increasingly based on machine translation, it's not surprising that the same trend is occurring in writing recognition. The dominant approach is RNNs with CTC (§4.1.3), with various enhancements. RNNs with attention, as well as transformers, have also been tried, with good results. Below is an outline of a typical RNN+CTC system:

1. For preprocessing, the image is divided up into small (e.g., 4×3 pixel) blocks.
2. The blocks are fed through various two-dimensional layers, for example:

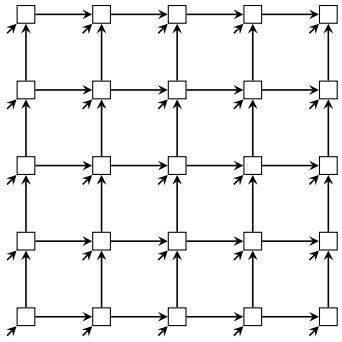


Figure 4.5: Multi-dimensional RNN (Graves, Fernández, and Schmidhuber, 2007).

- One variant that has been important in handwriting recognition is multi-dimensional RNNs (Graves, Fernández, and Schmidhuber, 2007), in which each hidden vector depends on its predecessors in two directions (Figure 4.5).
- Other variants use techniques from computer vision, like convolution and pooling layers.

3. The two-dimensional array of vectors is collapsed into a one-dimensional array of vectors by summing or taking the maximum over the vertical dimension.
4. Then for each column, we can predict a character.
5. Finally, we apply CTC to remove duplicate characters.

Bibliography

Bazzi, Issam, Richard Schwartz, and John Makhoul (1999). "An Omnifont Open-Vocabulary OCR System for English and Arabic". In: *Trans. Pattern Analysis and Machine Intelligence* 21.6.

Bluche, Theodore (2016). "Joint Line Segmentation and Transcription for End-to-End Handwritten Paragraph Recognition". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. URL: https://proceedings.neurips.cc/paper/2016/file/2bb232c0b13c774965ef8558f0fbd615-Paper.pdf.

Chan, William et al. (2016). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Proc. ICASSP*, pp. 4960–4964. URL: https://arxiv.org/pdf/1508.01211.pdf.

Dahl, George E. et al. (2012). "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition". In: *IEEE Trans. Audio, Speech, and Language Processing* 20.1, pp. 30–42. DOI: 10.1109/TASL.2011.2134090.

Graves, Alex (2012). In: *Proc. ICML Workshop on Representation Learning*. URL: https://arxiv.org/pdf/1211.3711.pdf.

Graves, Alex, Santiago Fernández, and Faustino Gomez (2006). "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks". In: *Proc. ICML*, pp. 369–376.

Graves, Alex, Santiago Fernández, and Jürgen Schmidhuber (2007). "Multi-dimensional Recurrent Neural Networks". In: *Artificial Neural Networks – ICANN 2007*, pp. 549–558. URL: https://arxiv.org/abs/0705.2011.

Graves, Alex, Navdeep Jaitly, and Abdel-rahman Mohamed (2013). "Hybrid Speech Recognition with Deep Bidirectional LSTM". In: *Proc. ASRU*, pp. 273–278. DOI: 10.1109/ASRU.2013.6707742.

Mohri, Mehryar, Fernando Pereira, and Michael Riley (2002). "Weighted finite-state transducers in speech recognition". In: *Computer Speech and Language* 16, pp. 69–88.