ARIS Volume 6
1999

Automatic speech recognition: a statistical affair

Helmer Strik
A²RT, Department of Language and Speech,
Nijmegen University, Netherlands

Most of us frequently use speech to communicate with other people. Most of us will also communicate regularly with a computer, but rarely by means of speech. The computer input usually comes from a keyboard or a mouse, and the output goes to the monitor or a printer. Still, in many cases the communication with a computer would be facilitated if speech could be used, if only because most people speak faster than they type. A necessary requirement for this is that the computer is able to recognise our speech: automatic speech recognition (ASR).

ASR is booming, and if you have not heard of it yet, this will certainly change in the near future. This is a prophecy I can easily make, since at this very moment it has been fulfilled. Why? Because in this article, I will explain how ASR works. And you will find out that ASR is 'a statistical affair'.

Speech recognition

In this article, I will often refer — by way of illustration — to a specific continuous speech recogniser (CSR), namely a Dutch CSR developed at our department (Strik *et al.*, 1997). This CSR is currently used in an operational spoken dialogue system — called OVIS — which automatically gives information about train journeys in the Netherlands via the telephone. For instance, in the third quarter of 1997 OVIS automatically answered some 200,000 calls. The CSR in OVIS is fairly typical for the CSR's currently being used elsewhere.

The first generation of speech recognisers stored a template for every word that had to be recognised. An incoming (unknown) word was compared to all templates, and the template with the smallest distance was chosen. This technique only works well if the number of words (that have to be recognised) is small. Furthermore, all words had to be uttered in isolation, with a pause before and after each word, because otherwise the recogniser often had difficulties in finding the beginning and end of the words. Many current applications should be able to recognise (tens of) thousands of words which have been uttered naturally (i.e. without pauses, and therefore such a recogniser is called a continuous speech recogniser, CSR). For instance, OVIS can recognise more than 1,400 different words (some are shown in *Table 2*). Since it is not feasible to store so many templates, phonemes are often used as basic recognition units.

Phonemes are the building blocks of speech. For every phoneme there is a symbol (see *Table 1*). For instance, in the Dutch word '*rekenwerk*' there are three different sounds for the letter 'e', which are thus all described by a different phoneme symbol ('e:', '@' and 'E', respectively: 're:k@nwErk'). Since most of you are probably not familiar with Dutch, I will also give English example. Ask a native English speaker to pronounce the word '*characterisation*', and if you listen carefully you will hear three different pronunciations of the letter 'a'. So, for the same letter (grapheme) 'a' there are three different phoneme symbols. The 37 basic units used in OVIS are given in *Table 1*. Note that there are two basic units for each of the phonemes 'l' and 'r'. The reason is that 'l' and 'r' are often pronounced differently in Dutch before and after a vowel.

Table 1 — The 37 basic units used in OVIS

For each phoneme are given:

[1] the symbol used in OVIS,
[2] an example of a Dutch word in which these phonemes occur (the target sound is shown in ***bold italics***).

Vowels			
[1]	[2]	[1]	[2]
i	l***ie***p	I	l***i***p
e:	l***ee***g	E	l***e***g
a:	l***aa***t	A	l***a***t
o:	b***oo***m	O	b***o***m
y	b***uu***t	Y	p***u***t
2:	d***eu***k	@	g***e***lijk
Ei	w***ij***s	u	b***oe***k
9y	h***ui***s	Au	k***ou***d

Consonants			
[1]	[2]	[1]	[2]
p	***p***ut	L	ba***l***
b	***b***ad	R	ba***r***
t	***t***ak	f	***f***iets
d	***d***ak	v	***v***at
k	***k***at	s	***s***ap
N	la***ng***	z	***z***at
m	***m***at	S	***sj***aal
n	***n***at	j	***j***as
l	***l***at	x	li***ch***t
r	***r***at	h	***h***ad
		w	***w***at

A CSR can only recognise words that are present in its lexicon. Part of the OVIS lexicon is given in *Table 2a*. There are two forms for each word. On the left is the orthographic form, i.e. the word as it is written. This is a sequence of letters (graphemes). On the right is a description of the word as it is usually pronounced. This is a sequence of phoneme symbols and is referred to as a phoneme transcription (of the word). A CSR has an acoustic model for every phoneme. By using phonemes as basic units, the total number of acoustic models (templates) that has to be stored is 37, which is much smaller than the total number of words. In order to make clear what the Dutch words in *Table 2a* mean, I have written in *Table 2b* the English translation and occasionally a remark between < and >, as follows: < *remark* >. (For obvious reasons, names of Dutch railway stations cannot be translated.)

Table 2

2a — Part of the OVIS lexicon		***2b — English translation*** and/or < *remark* >
Orthographic form	*Phoneme transcription*	
ravenstein	ra:v@nstEin	< *station name* >
rechtstreeks	rExstre:ks	directly
rechtstreekse	rExstre:ks@	direct
reis	rEis	travel < *noun* >
reisgelegenheid	rEisx@le:x@nhEit	travel opportunity
reizen	rEiz@	to travel < *verb* >
renesse	r@nEs@	< *station name* >
rest	rEst	rest, remainder
retour	r@tu~R	return ticket
retourreis	r@tu~rEis	two-way journey
retourtje	r@tu~Rc@	return ticket
reuver	r2:v@R	< *station name* >
rheden	re:d@	< *station name* >
rhenen	re:n@	< *station name* >
richting	rIxtIN	direction
ridderkerk	rId@RkERk	< *station name* >

Phoneme models

The most common type of model used in ASR is the ‘Hidden Markov Model’ (HMM). The structure of HMMs is based on the following premise about the production of speech: “An utterance is produced by the speech articulators passing through an ordered sequence of ‘stationary states’ of different duration.” The states may be viewed as corresponding roughly to acoustic events. This is a crude and drastically simplified model of the complexities of speech; speech is the realisation of a smooth and continuous process which does not jump from one articulatory position to another. However, the success of HMMs in speech recognition demonstrates that the approximation in the premise above is a useful one.

Markov models

A phoneme model should thus consist of an ordered sequence of states. The model given in *Figure 1* satisfies this demand. It consists of an ordered sequence of 6 states S$_i$. The two states A and B are only used to concatenate phoneme models in order to create word models.

Figure 1 — Structure of a phoneme model in OVIS

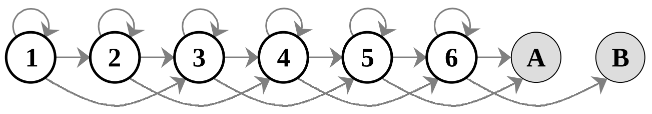

Figure 1 shows that not all transitions are possible. The possibility to make a transition from one state to another one is bounded by some rules. The allowed transitions and their probabilities are stored in a transition matrix. A fictitious example of a transition matrix is given in *Table 3*.

Table 3 — Transition matrix for the phoneme model in Figure 1

from state ↓	to state →	1	2	3	4	5	6	A	B
1	A =	0.4	0.2	0.4					
2			0.8	0.1	0.1				
3				0.8	0.1	0.1			
4					0.6	0.3	0.1		
5						0.2	0.5	0.3	
6							0.4	0.4	.02

The fact that speech is an ordered sequence of acoustic events is reflected in the model by the uni-directional flow of transitions. Because speech production is a process which evolves in time, and cannot be reversed in time, transitions from right to left are not possible. In the model of *Figure 1*, it is possible to remain in the same state, change to the next state, or skip a state. This is also reflected in the transition matrix in *Table 3*. Some transitions have a zero probability, i.e. they are not allowed. The fact that states can be repeated or skipped represents the variations in speaking rate. If the speaking rate is low, then a phoneme will have a longer duration and the state corresponding to that phoneme has to be repeated more often.

The statistical nature of the variation in speaking rate is accounted for by attaching a probability to each transition (see *Table 3*). For instance, if a phoneme often has a very long duration, then for the state corresponding to this phoneme the probability to re-enter the same state must be relatively high. The transition probability between S_i and S_j is the probability that the model is in state S_j at time instant t+1, given that it is in state S_i at time instant t. By definition, the sum of all probabilities to leave a state must be equal to 1. This includes the possibility to re-enter in the same state. Therefore, the sum of the probabilities in every row of A is equal to 1 (see *Table 3*).

An important characteristic of this model is that the state occupied at time instant t+1 is determined probabilistically and depends only on the state occupied at time t. This is called the Markov property, and hence the model is called a Markov model. Or, to be more precise, it is called a Markov model of order one, because the probability of occupying a state at time t+1 depends only on the previous state. If the probability of occupying a state at time t+1 depends on the previous two states, i.e. the state at time t and the state at time t–1, then the model is called a second-order Markov model.

Feature extraction

The approach used in ASR is that of statistical pattern classification. Generally, a pattern classifier consists of two parts: a feature extractor and a classifier. The classification process is described below. Here I will shortly explain feature extraction, which is often referred to as acoustic pre-processing. The values mentioned are those used in OVIS.

The microphone converts the speech pressure wave in an analogue, electrical signal. This analogue signal is then digitised (A/D conversion) by taking 16000 samples per second (i.e. the sampling frequency is 16 kHz). Next, every 10 msec. a frequency spectrum is calculated, and 28 feature values that describe the spectral contents of the signal are stored in a feature vector. Consequently, there are 100 feature vectors F_t per sec., and each one consists of M=28 values. How these feature vectors are used in ASR is described below.

Hidden Markov models

In speech both variations in speech rate and variations in pronunciation (articulation) are observed between different realisations of an utterance. The Markov model can account for variations in speech rate. In order to allow for variations in pronunciation, likelihoods are attached to each state. For a given feature vector F_t and state S_i, the likelihood is:

(1) $p(F_t|S_i) = N(F_t, \mu_i, \Sigma_i)$

The features in the feature matrix are generally chosen such that they are (almost) uncorrelated, in which case the covariance matrix Σ_i is (almost) diagonal and equation (1) reduces to

(2) $p(F_t|S_i) = N(F_t, \mu_i, \Sigma_i) = \prod N(F_{tm}, \mu_{im}, \sigma_{im})$
[product for m = 1,…,M]

This is the product over the M features in the feature matrix, and F_{tm}, μ_{im} and σ_{im} are the m-th component of F_t, μ_i and σ_i, respectively. In other words, F_{tm} is the m-th feature value and μ_{im} and σ_{im} are the mean and the standard deviation of the m-th feature for state S_i. They are estimated during training, as will be described below.

In the equations above a normal (or Gaussian) distribution is used. However, in practice not every distribution of the features can be described well by a single Gaussian. This is why Gaussian mixtures are usually used:

(3) $p(F_t|S_i) = \sum c_{ig} * N(F_t, \bullet_{ig}, \bullet_{ig})$
[summation for g = 1,...,G]

where G is the total number of Gaussians (32 in the OVIS system) and c_{ig} are the mixture gain coefficients. The c_{ig} are also determined during training, with the following constraints for all i = 1,...,I:

$cig \geq 0$ and $\Sigma\, cig = 1$ [summation for g = 1,• ,G]

Why are these HMMs called ‘hidden’? Well, in an HMM each state can emit more than one feature vector, and a single feature vector can be emitted by different states. Therefore, given a sequence of feature vectors, it is impossible to determine the underlying sequence of states that produced it. Put otherwise, it is impossible to determine for every time instant t in which state the model was, i.e. the state is hidden.

The starting point for ASR is a speech signal X of an unknown utterance. On the basis of this signal X the CSR has to decide which utterance W was spoken. In other words, the CSR has to decide what the corresponding string of words is: $W = \{w_1, w_2,\ldots, w_p,\ldots, w_P\}$. Most CSR’s use the maximum likelihood criterion: they search for the W for which P(W|X) is maximal. Unfortunately, P(W|X) cannot be calculated directly in practice. That is why Bayes’ rule is used:

(4) $P(W|X) = P(X|W) * P(W) / P(X)$

For a given utterance X, P(X) is constant and can be discarded. The goal now is to find the W that maximises $P(X|W) * P(W)$. Let me first describe how P(X|W) and P(W) are calculated.

P(X|W) — First, all words w_p of W are looked up in the lexicon and the orthographic form of the word is replaced by its phoneme transcription. In this way a phoneme transcription of the whole utterance is obtained. For a given X (= a sequence of feature vectors) and W (= a sequence of phonemes) an alignment is then calculated by means of an algorithm called the Viterbi algorithm. This alignment is also a segmentation, because in the speech signal the boundaries of every state in every phoneme are found. Consequently, the result is a sequence of states S_i with corresponding feature vectors F_t. For each pair (S_i, F_t) the likelihood can be calculated by means of equation (3). By combining the likelihoods of all pairs (S_i, F_t) in the utterance, the likelihood for the Viterbi alignment is obtained: P(X|W). It can be proven that the Viterbi alignment is optimal in the sense that it is the alignment for which P(X|W) is maximal.

P(W) — P(W) is the likelihood of a sequence of words: $P(W) = P(w_1, w_2, \ldots, w_p, \ldots, w_P)$. For calculation of P(W) the acoustic signal X is not needed. The models used to estimate P(W) are generally called language models, since they contain information about the likelihood of word sequences in a language. As language models the so called n-grams are often used:

(5) $(w_1, \ldots, w_P) = \prod_q P(w_q \mid w_{q-1}, w_{q-2}, \ldots, w_{q-n+1})$

In an n–gram the likelihood of a word thus depends on the previous (n–1) words.

Training

Large collections of utterances are used to train the phoneme models. Each example contains the acoustic signal and a description of the contents of that utterance (i.e. a sequence of words). Such a collection is called a speech corpus. During training the words in all utterances are replaced by their phoneme transcriptions (as described above). Next, the Viterbi algorithm is used to find the optimal alignment between the speech signals and the phoneme transcriptions. After all alignments (and thus segmentations) have been calculated it is possible to determine for every state S_i (of all phonemes) what the corresponding parts of the speech signals are. Put otherwise, it is possible for every state S_i to determine what the corresponding feature vectors are. Then, for each state S_i, the parameters c_{ig}, μ_{ig} and σ_{ig} can be calculated. These parameters are stored, and they make it possible to evaluate equation (3) during the recognition phase.

Besides the phoneme models, the language models also have to be trained. If there are L words present in the lexicon, the total number of possible n-grams is L^n. For instance, for OVIS (L = 1400) there are 1400 unigrams, $1400^2 = 1.96 * 10^6$ bigrams and $1400^3 = 2{,}744 * 10^9$ trigrams. It is obvious that for training these n-grams a large amount of data is needed, and the required amount of data grows rapidly with increasing n. That is one of the reasons why in OVIS only unigram and bigram language models are used. In order to train the n-grams, only text (and no acoustic signals) is needed. For this purpose, the texts present in the speech corpora could be used. However, since these speech corpora usually do not contain enough text for adequate training of the language models, generally much larger text corpora are used. Fortunately, large collections of written text are available, e.g. on WWW and at publishers (newspapers, books, etc.).

Recognition

Finally, we have now arrived at the crux of the matter: automatic speech recognition. As mentioned above, for a given unknown X the goal is the find the optimal word sequence W, which is the W_h that maximises $P(X|W_h) * P(W_h)$. For a given hypothesis W_h the phoneme models and the Viterbi algorithm can be used to calculate $P(X|W_h)$, while $P(W_h)$ can be calculated with the language model. After $P(X|W_h) * P(W_h)$ has been calculated for all possible hypotheses W_h, the W_h can be chosen for which $P(X|W_h) * P(W_h)$ is maximal. This maximum likelihood classification could be done for every separate word. However, previous research has shown that it is better to do it for the whole utterance. Let us do so.

Since at the beginning of recognition it is completely unknown which words are present in the utterance, the CSR starts generating all possible words at position 1, 2, 3, etc. Because we have just decided to do maximum likelihood classification for the whole utterance, the CSR should continue generating words until the end of the utterance is reached. However, the total number of possible word sequences is immense, especially if the CSR has a large lexicon and the utterance is long. Calculating the likelihood for all these hypotheses would cost a large amount of CPU time and memory — too much. That is why the following technique is applied, which is called beam search. At regular intervals, before the end of the utterance, the likelihood of each of the hypotheses is calculated with the Viterbi algorithm. All hypotheses for which the likelihood at that point is much less than that of the most likely hypothesis are discarded, because it is very unlikely that they are part of the optimal word sequence. In this way CPU time and memory are kept within reasonable bounds. This is important, as a CSR has to work ‘real-time’. In the end, the word sequence W_h with the largest likelihood is chosen, and this is the recognised utterance.

Applications

During the last 40 years the performance of the speech recognisers has gradually increased. Although CSR's still make mistakes, their performance is now good enough to use them in applications. Applications which are already frequently used are dictation systems (so that you can talk to your PC) and systems for giving various kinds of information via the telephone (similar to OVIS). You do not have to be a clairvoyant to predict that ASR will be used more and more in the near future. In this paper, I have tried to explain how ASR works, and you now know that ASR — Automatic Speech Recognition — is '**A STATISTICAL AFFAIR**.'

Acknowledgements

I would like to thank Gies Bouwman, Catia Cucchiarini, Judith Kessens, Mirjam Wester and Febe de Wet from the *A^{2}RT* group of the Dept. of Language and Speech for their comments on a previous version of this paper. Furthermore, I would like to thank Gies Bouwman for providing *Figure 1*.

Helmer Strik

Reference

H. STRIK, A. RUSSEL, H. VAN DEN HEUVEL, C. CUCCHIARINI, L. BOVES (1997), "A spoken dialog system for the Dutch public transport information service", *Int. Journal of Speech Technology*, Vol. **2**, No. 2, pp. 121-131.

Further reading

H. BOURLARD & N. MORGAN. *Connectionist Speech Recognition — A hybrid approach*. Kluwer Academic Publishers, 1994.
J.A. MARKOWITZ. *Using speech recognition*. New Jersey: Prentice-Hall Inc., 1996.
L.R. RABINER & B.-H. JUANG. *Fundamentals of speech recognition*. New Jersey: Prentice-Hall Inc., 1993.
C. SCHMANDT. *Voice communication with computers*. New York: Van Nostrand Reinhold, 1994.

Websites

- For links to interesting 'speech sites': http://lands.let.kun.nl/TSpublic/strik/speech-sites.html
- For links to information about OVIS: http://lands.let.kun.nl/TSpublic/strik/ovis.html