Chapter 11

Mathematical Methods

11.1 Overview

Hans Uszkoreit

Deutsches Forschungzentrum für Künstliche Intelligenz, Saarbrücken, Germany
and Universität des Saarlandes, Saarbrücken, Germany

Processing of written and spoken human language is computation. However, language processing is not a *single* type of computation. The different levels of information encoding in our language as well as the spectrum of applications for language technology pose a range of very distinct computational tasks. Therefore a wide variety of mathematical methods have been employed in human language technology. Some of them have led to specialized tools for a very restricted task, while others are part of the mathematical foundations of the technology. In this overview a very general map is drawn that groups the different approaches. Some particularly relevant classes of methods are highlighted in the remaining sections of the chapter.

In the early days of language processing, most if not all researchers underestimated the complexity of the problem. Many of them tried to bypass a mathematical characterization of their tasks and solve the problem simply by looking at the envisaged inputs and outputs of their systems. Purely procedural early approaches to machine translation fall in this category. These attempts failed very badly. However, there is one major difference between language processing and most other areas with highly complex calculation tasks, e.g., computational meteorology. One system exists that can handle human language quite decently, i.e., the human cognitive system. Moreover, there is a scientific discipline that strives for a formal description of the human language faculty. Very soon the great majority of researchers became convinced that one needed to utilize insights from linguistics—including phonetics and psycholinguistics—in order to make progress in modelling the human language user.

Modern linguistics tries to characterize the mapping between a spoken or written utterance and its meaning. Linguists do this in roughly the following way. They break up the complex mapping into suitable levels of representation, specify representations in dedicated formalisms, and they employ the same formalisms for specifying the implicit linguistic knowledge of a human language user. Traditionally their main data are collected or invented example sentences, judged and interpreted by introspection. Almost exclusively, discrete symbolic methods have been employed for representing different types of information in linguistics. (The only exception was phonetic signal processing, where Fourier transformations converted the two-dimensional acoustic signal into a three-dimensional spectrogram.)

11.1.1 High-level Linguistic Methods

The mathematical methods of syntax, morphology and phonology are suited for describing sets of strings, especially hierarchically structured strings. Most of these methods came from formal language theory. Most notable is the formal theory of languages and grammars emerging from the Chomsky hierarchy, an inclusion hierarchy of classes of languages. Each class of languages corresponds to a certain level of computational complexity. For these classes of languages, classes of grammars were found that generate the languages. Each class of languages also corresponds to a type of automaton that accepts strings of a language. Typical for this research was the close interaction between theoretical computer science and formal syntax with strong influences in both directions. Much investigation has gone into the question of the proper characterization of human language with respect to the Chomsky hierarchy.

The grammars and automata of formal language theory can rarely be applied to natural language processing without certain modifications. The grammar models developed in linguistics do not directly correspond to the ones from formal language theory. A variety of grammar models have been designed in linguistics and language technology. Some of these models are mentioned in section 3.3. For a comprehensive description you will have to resort to handbooks such as Jacobs, v. Stechow, et al. (1993). A long tradition of work was devoted to efficient parsing algorithms for many grammar models. This work is summarized in section 11.4.

The grammars of formal language theory are rewrite systems with atomic nonterminal symbols that stand for lexical and syntactic categories. However, in human language such categories have complex properties that influence their syntactic distribution. Therefore mathematical tools were developed for expressing linguistic entities as sets of complex features. A new class of logic, so called feature logic evolved. This branch of research, often subsumed under the term unification grammar, had close links with similar developments in knowledge representation and programming languages. The specialized processing methods that were developed for unification grammars had strong connections with constraint logic programming. In this book, unification grammars are described in section 3.1. For a more detailed introduction, please refer to Shieber (1988).

The situation is somewhat different in semantics. Here representation languages are needed in which we can represent the meaning—or better informational content—of an utterance. In order to provide unambiguous representations of meaning that can serve as the basis for inferences, logic is employed. Many varieties of higher order predicate logic have been developed for this purpose. Special representation languages such as frame and script languages came from artificial intelligence (AI). In the last few years, many of them have received a logical foundation. General purpose and specialized inference techniques have been employed for interpreting the meaning representation in connection with knowledge about the linguistic context, situational context, and the world. Logical deduction is the inference technique mainly used, but there are also approaches that utilize abduction methods. For the use of some semantic formalisms in language technology, refer to section 3.5.

The last two decades witnessed a convergence of theoretical linguistics and language processing with respect to their mathematical methods. On the one hand, this movement proved very fruitful in many areas of language processing. On the other hand, it also lead to some disillusionment concerning the potentials of formal linguistic tools among practitioners of language technology.

Although the specification of linguistic knowledge improved quite a bit through the use of advanced representation techniques, the resulting systems still lacked coverage, robustness, and efficiency, the properties required for realistic applications. It even seemed that every increase in linguistic coverage was accompanied by a loss of efficiency since efficient processing methods for linguistic representation formalisms are still missing.

11.1.2 Statistical and Low-level Processing Methods

Encouraged by a breakthrough in the recognition of spoken words, many researchers turned to statistical data-driven methods for designing language technology applications. For most of them, the line of reasoning went as follows. Linguistic investigation of linguistic competence and cognitive modelling of human language processing have not yet achieved a sufficient understanding and formalization of the mapping from the language signal to the informational contents of the utterance or vice versa. However, only very few applications need the complete mapping anyway. Even if we had a formal model of the complete mapping, we do not have a model of the cognitive system that could support it, since AI has not come close to modelling human knowledge processing.

If one cannot get access to the human linguistic competence through standard methods of linguistic research, it may be possible to induce the knowledge necessary for a specific application indirectly by correlating linguistic data with the desired outputs of the machine. In case of a dictation system, this output is the appropriate written words. For a machine translation system, the output consists of the translated sentences. For an information access system the output will be a query to a data base. After decades of expecting technological progress mainly from investigating the cognitive structures and processes of the human language user, attention moved back to the linguistic data produced by humans and to be processed by language technology.

The predominant approach is based on an information theoretic view of language processing as a noisy-channel information transmission. In this metaphor, it is assumed that a message is transmitted which we have to recover from observing the output of the noisy channel. It is described as the source-channel model in section 1.6. The approach requires a model that characterizes the transmission by giving for every message the probability of the observed output. The other component is the language model which gives the so-called a-priori distribution, the probability of a message in its context to be sent.

A special type of stochastic finite-state automata, hidden Markov models (HMMs) have been utilized for the recognition of spoken words, syllables or phonemes (section 1.5). Probabilistic derivatives of many grammar models have been proposed. Statistical methods are employed today for substituting or supporting discrete symbolic methods in almost every area of language processing. Examples of promising approaches are statistical part-of-speech tagging (section 3.2.2), probabilistic parsing (section 3.7), ambiguity resolution (section 3.7), lexical knowledge acquisition (Pustejovsky, 1992), and statistical machine translation (Brown, Cocke, et al., 1990).

A special area that has developed rapidly during the last few years, mainly in conjunction with the statistical methods, is the utilization of optimization techniques for spoken and written language processing. Optimization methods are used to find the best solution or solutions among a number of possible solutions applying some evaluation criterion. Since the number of possible solutions, e.g., word hypotheses for a whole utterance in speech recognition, can be rather large, the search needs to be highly efficient. Optimization techniques, especially from dynamic programming, are presented in section 11.7.

Connectionist methods constitute a different paradigm for statistical learning and probabilistic processing on the basis of an acquired language model. Neural nets have proven very useful in pattern recognition. In language technology, both self-trained and prestructured nets are explored for a variety of tasks. A major problem for syntactic and semantic processing is the limitations of connectionist methods concerning the modelling of recursion. A major problem for applying neural nets to speech processing that stems from the temporal nature of speech is time sequence matching. In section 11.5 connectionist methods for speech processing are summarized. A combination of connectionist methods with hidden Markov models is described.

Another major area of very promising new technology is the development of specialized low-level processing methods for natural language. Especially noteworthy is the renaissance of finite-state processing techniques. Finite state transducers were applied with great success to morphological processing. These approaches are described in section 11.6. Recently finite state parsers were constructed that out-perform their competition in coverage and performance. The finite-state technology for syntax is presented in section 3.2.2. Finite-state methods are also applied in semantics and in discourse modelling.

11.1.3 Future Directions

Challenges for future research concerning the individual mathematical methods are presented in the sections that describe them. We will conclude this section by addressing key research problems that extend over the multitude of approaches.

One major challenge is posed by the lack of good formal methods for concurrent symbolic processing. Although there have been various attempts to employ methods and programming languages for concurrent processing in language technology, the results are not yet convincing. The appropriate hardware and well-suited problems for parallel processing are there. What are missing are better formal concepts of concurrency in computation.

Badly needed for progress in language technology is a better general view linking the diverse formal approaches and characterizing their respective virtues and shortcomings. With respect to the employed mathematical methods we currently witness the coexistence of three major research paradigms, shown in Figure 11.1. However, when we look at individual research systems and new applications, we rarely see a system that does not combine formal tools from more than one of the paradigms. In most cases the observed combinations of methods are rather ad-hoc. There is no general methodology yet that tells us which mix of methods is most appropriate for a certain type of application.

A few examples from recent research may illustrate the relevant direction for future

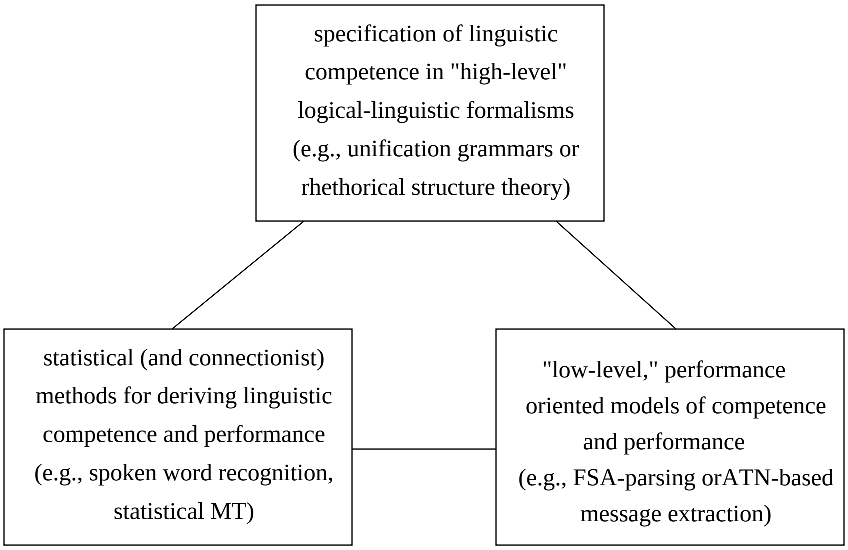

Figure 11.1: Major research paradigms.

investigation:

Compilation methods may be used to relate high-level competence grammars and low-level performance methods (Alshawi, 1992; Kasper, Kiefer, et al., 1995). Alternatively, learning methods such as explanation-based learning can also be applied in order to derive low-level performance grammars from high-level linguistic specifications (Samuelsson, 1994).

The connections between statistical methods and general automata theory are addressed in Pereira, Riley, et al. (1994), where it is proposed that the concept of weighted finite-state automata (acceptors and transducers) may serve as the common formal foundation for language processing.

Statistical methods can be used for extending knowledge specified in high-level formalisms. An example is the learning of lexical valency information from corpora (Manning, 1993). Statistical methods can also be used for deriving control information that may speed up processing with high-level grammars. Specific linguistic generalizations could be merged or intersected with statistical language models in order to improve their robustness. Intersecting linguistic and statistical models, for instance, can improve precision in part-of-speech tagging.

We expect that extensive research on the theoretical and practical connections among the diverse methods will lead to a more unified mathematical foundation of human language processing.

11.2 Statistical Modeling and Classification

Steve Levinson

AT&T Bell Laboratories, Murray Hill, New Jersey, USA

The speech communication process is a very complicated one for which we have only an incomplete understanding. It has thus far proven to be best treated as a stochastic process which is well characterized by its statistical properties.

There are two fundamental assumptions which underlie the statistical model of speech. The first is that speech is literate. That is, it is well represented by a small set of abstract symbols which correspond to basic acoustic patterns. Second, the acoustic patterns are differentiated by their short duration amplitude spectra. Measurement of these spectra shows that they possess a high degree of variability even for the same symbol and thus are most accurately classified by means of statistical decision theory.

There is another property of speech that makes the application of statistical methods to it more difficult. Speech comprises several overlapping levels of linguistic structure including phonetic, phonological, phonotactic, prosodic, syntactic, semantic and pragmatic information. Thus, to be useful, a statistical characterization of the acoustic speech signal must include a principled means by which the statistics can be combined to satisfy all of the constraints composed by the aforementioned linguistic structure. This section describes briefly the most effective statistical methods currently in use to model the speech signal.

Although it is beyond the scope of this section to describe other uses of the statistical methodologies outlined below, it is worth noting that other important applications of the technology include machine translation (Alshawi et al., 1994), language identification (Kadambe & Hieronymus, 1994), and handwriting recognition (Wilfong, 1995).

11.2.1 Primitive Acoustic Features

We call the voltage analog of the sound pressure wave the speech signal. The signal is usually sampled at 0.1 msec. intervals to form a sequence of 16-bit integers. Because the signal is nonstationary, we divide it into short frames of, say, 10 to 30 msec. duration at 5 to 10 msec. intervals. Thus frames of around a hundred samples are considered to be stationary and a short duration spectrum is computed for each frame. Although many methods of spectral estimation have been advocated in the past, the method of linear prediction (Atal & Hanauer, 1971; Baker, 1979) is often used. Linear prediction provides an n-dimensional (where n is often twelve) parameterization of the spectrum usually in the form of cepstral coefficients (Juang et al., 1987). Thus the information bearing features of the speech signal are usually taken to be a sequence of twelve-dimensional cepstral vectors with one vector computed every 10 msec.

Due to the intrinsic variability of the speech signal and specific assumptions inherent in the linear prediction method, the sequence of cepstral vectors is a random process whose statistical properties can be estimated. The representation of speech signals is discussed in more detail in section 1.3 and section 11.3.

11.2.2 Quantization

Since we assume that speech is literate, the signal could also be represented as a sequence of symbols where each symbol corresponds to a phonetic unit of varying duration. Each phonetic unit corresponds to a region of the twelve-dimensional acoustic feature space. The regions are defined statistically by estimating the probability of each class conditioned on the vectors belonging to that class and then computing the pairwise decision boundaries between the classes as the locus of points in the feature space where both classes are equally probable.

If the decision boundaries are explicitly computed as described above, then an arbitrary feature vector can be classified as resulting from the utterance of one phonetic class simply by finding which region of the space it lies in. As a matter of practice, however, this computation is not performed explicitly. Rather, a statistical decision rule is used. The rule simply states that a feature vector belongs to that class whose probability is largest conditioned on the vector. The effect of this decision rule is to quantize the entire twelve-dimensional feature space into a small number of regions corresponding to the phonetic classes.

11.2.3 Maximum Likelihood and Related Rules

Although the above-described rule is very intuitively appearing, it is not easily implemented because there is no direct method for computing the probability of a phonetic unit given its acoustic features. If, however, a large set of acoustic vectors which have been phonetically labeled is available, then an indirect method, based on Bayes rule, can be devised. Bayes rule allows us to estimate the probability of a phonetic class given its features from the likelihood of the features given the class. This method leads to the maximum likelihood classifier which assigns an unknown vector to that class whose probability density function conditioned on the class has the maximum value. It is most important to understand that ALL statistical methods of speech recognition are based on this and related rules. The philosophy of maximum likelihood is now so much a part of speech recognition that citations of it in the literature are no longer given. However, the interested reader will find its origins recounted in any standard text on pattern recognition such as Duda and Hart (1973); Meisel (1972); Patrick (1972). Statistical methods in speech recognition are often called by different names such as minimum prediction residual method, minimum risk, minimum probability of error or nearest neighbor. These rules are all closely related as they derive from the same Bayesian argument.

11.2.4 Class Conditional Density Functions

From the previous section, it is clear that ALL statistical methods in speech recognition rest on the estimation of class conditional density functions for phonetic units or, as we shall see later, other linguistic constituents. Thus, the performance of a speech processing algorithm depends critically on the accuracy of the estimates of the class conditional density functions. These, in turn, depend on the existence of a sufficiently large, correctly labeled training set and well understood statistical estimation techniques. Regarding the former, there is little to be said of a practical nature except that the more data available, the better. There are some theoretical results, such as the Cramer-Rao (Patrick, 1972) bound relating variance of estimators to sample size. Obviously the larger the size, the lower the variance and hence the lower the error rate. However, it is quite difficult to relate estimator variance to error rate precisely, so the various rules of thumb which are often invoked to determine sample size needed for a specified performance level are unreliable.

There is one serious flaw to the above-described decision theory. It is predicated on the principle that if the class conditional density functions are known exactly, then no other decision rule based on the same training data can yield asymptotically better performance. Unfortunately, the assumption of exact knowledge of the class conditional density functions is never met in reality. The error may simply be in the parameter values of the densities or, worse, their form may be incorrect.

An elegant solution to this problem is to directly minimize the classification error. This may be done by Juang's method of Generalized Probabilistic Descent (GPD) (Juang, 1992) which has proven to be effective in very difficult speech recognition problems (Wilpon, 1994).

Another variant of the maximum likelihood methodology is clustering. In many classification problems, the items in a class differ amongst themselves in a systematic way. A simple example is the pronunciation of a word by speakers of different national or regional accents or dialects. In such cases the class conditional densities will be multi-modal with the modes and their shapes unknown. Such densities can be estimated by clustering techniques. The most effective such techniques are based on the Lloyd-Max optimum quantizer (Lloyd, 1982) which has come to be known as the k-means algorithm. These techniques have been applied to speech recognition by Levinson et al. (1979). As we shall see in the next section, clustering methods are implicitly used in the state-of-the-art recognition methods.

11.2.5 Hidden Markov Model Methodology

If speech were composed solely of isolated acoustic patterns then the classical statistical decision theory outlined above would be sufficient to perform speech recognition. Unfortunately, that is not the case. That is, the putative fundamental units of speech are combined according to a rich set of linguistic rules which are, themselves, so complicated that they are best captured statistically. The problem is that in order to accomplish speech recognition, one must capture all the subtlety of linguistic structure with a computationally tractable model. The lack of such a representation held back progress in speech recognition for many years.

In the early 1970's both Baker (1975) and Jelinek (1976) independently applied an existing mathematical technique based on the hidden Markov model (HMM) to speech recognition. As the name of the method implies, the original concepts were proposed by A. A. Markov himself (Markov, 1913). The modern form of the mathematics was developed by Baum and his colleagues (Baum & Eagon, 1967; Baum & Sell, 1968; Baum, 1972; Baum, Petrie, et al., 1970); the application of these methods to speech recognition is described in more detail in section 1.5.

11.2.6 Syntax

While the HMM has proven itself to be highly effective in representing several aspects of linguistic structure, other techniques are presently preferred for dealing with the cognitive aspects of language, syntax and semantics. Let us first consider syntax which refers to the relationship that words bear to each other in a sentence. Several aspects of this grammatical structure are well-captured using statistical methods.

The simplest useful way of thinking about syntax is to define it as word order constraint. That is, only certain words can follow certain other words. One way to quantify this is to make a list of allowable n-grams, sequences of n words, in a language. We can augment this list with n-gram probabilities, the probability of a word given its $n-1$ predecessors. This reduces syntax to a Markov n-chain, not to be confused with an HMM, and the desired probabilities can be estimated by counting relative frequencies in large text corpora. Once these numbers are available, an error-ridden lexical transcription of a spoken sentence can be corrected by finding the sequence of n-grams of maximum probability conditioned on the lexical transcription. A number of optimal search strategies are available to find the best sequence (Nilsson, 1971; Viterbi, 1967). For such practical cases, trigrams are typically used. A more detailed discussion of n-gram syntactic models is provided in section 1.6.

While the n-gram methods are useful, they do not constitute full syntactic analysis since syntax is more than word order constraint. In fact, the reason why syntax is considered to be part of cognition is that grammatical structure is a prerequisite to meaning. This aspect of syntax can also be exploited by statistical methods.

There are many ways to use full syntactic analysis, but the most intuitively appealing method is to use the linguist's notion of parts of speech. These syntactic constituents are categories which define the function of a word in a sentence. Associated with the parts of speech are rules of grammar which specify how parts of speech can be combined to form phrases which, in turn, can be combined to form sentences. Finally, using relative frequencies derived from text corpora, probabilities can be assigned to the grammatical rules. Using techniques from the theory of stochastic grammars (Fu, 1974), it is possible to find a sentence the joint probability of whose lexical transcription and syntactic structure, or parse, is maximized for a given corrupted transcription from a speech recognizer. In addition to these statistical parsing techniques, methods similar in spirit to HMM techniques have been studied by Baker (1979) and Jelinek (1990). In either case, syntax analysis both increases the accuracy of speech recognition and, as we shall see in the next section, provides information necessary for the extraction of meaning from a spoken utterance.

11.2.7 Semantics

The ultimate goal of speech recognition is to enable computers to understand the meaning of ordinary spoken discourse. Semantics is that aspect of linguistic structure relating words to meaning. Thus, the ultimate speech recognition machine will necessarily include a semantic analyzer. At present, there exists no general theory of the semantics of natural language. There are many proposed theories some of which are abstract and others of which are worked out for specific limited domains of discourse. All such theories rest on the idea that formal logical operations acting on lexical tokens and syntactic structures yield a formal symbolic representation of meaning. These theories have not yet been made statistical in any coherent way, although a new approach (Pieraccini, Levin, et al., 1993) based on the HMM seems promising.

There is, however, a statistical methodology which captures useful semantic information. It is called word sense disambiguation. A simple example is found in the word *bank* which has two meanings or senses. One sense is that of a financial institution and another refers to the shores of a river. Clearly, the words commonly associated with the two senses are quite different. If we know the words that are appropriate to each word sense, then we can use search techniques to maximize the joint probability of word sequences and word sense. This will result in higher lexical transcription accuracy.

The key to this line of reasoning is the precise measurement of the closeness of word associations. Church and Hanks (1990) proposed using the information theoretic measure of mutual information and has analyzed large text corpora to show that words clustered by large mutual information contents are indicative of a single word sense. It is thus possible to compile word sense statistics for large lexicons and apply them in statistical parsing techniques as described earlier.

11.2.8 Performance

It would be impossible in a short paper such as this is to completely and quantitatively characterize the performance of statistical speech recognizers. Instead, I will briefly mention three benchmarks established by systems based on the methodologies described above. They are moderate vocabulary speech understanding, large vocabulary speech recognition and small vocabulary recognition of telephony. Detailed summaries may be found in ARPA (1994); Wilpon (1994).

The most ambitious speech understanding experiment is presently ongoing under the sponsorship of ARPA. Several laboratories have built (ATIS) systems that provide airline travel information from spoken input. With a nominal vocabulary of 2000 words and spontaneous discourse from undesignated but cooperative speakers, approximately 95% of all queries are correctly answered.

Another highly ambitious project in speech recognition is also sponsored by ARPA. In this large vocabulary recognition task, the goal is lexical transcription only; so unlike the ATIS task, no semantic processing is used. The material is text read from North American business publications by undesignated speakers. The nominal vocabulary is 60,000 words. For this task, several laboratories have achieved word error rates of 11% or less. Unfortunately, such results are obtained by computer programs requiring hundreds of times real time.

Finally, the largest commercial use of speech recognition is in the AT&T telephone network for the placement of calls. In this case, customers are allowed to ask for one of five categories of service using any words they like so long as their utterance contains one of five key words. This system is currently processing about 1 billion calls per year. Calls are correctly processed more than 95% of the time without operator intervention.

11.2.9 Future Directions

Incremental improvements can be made to statistical models and classification methods in two distinct ways. First, existing models can be made more faithful. Second, existing models can be expanded to capture more linguistic structure. Making existing models more faithful reduces to the mathematical problem of lowering the variance of the statistical estimates of parameter values in the models. There are two ways to accomplish this. First, collect more data, more diverse data, more well-classified data, more data representing specific phenomena. The data needed is both text and speech. Second, improve estimation techniques by deriving estimators that have inherently lower variances. The statistical literature is replete with estimation techniques very few of which have been applied to large speech or text corpora. A related but different idea is to improve classification rules. One possibility would be to include a loss function reflecting the fact that some classification errors are more detrimental to transcription than others. The loss function could be estimated empirically and employed in a minimum risk decision rule rather than a maximum likelihood or minimum error probability rule. Existing models can also be made more general by making them represent known, well-understood linguistic structure. Two prime candidates are prosody and syntax. Speech synthesizers make extensive use of prosodic models yet none of that knowledge has found its way into speech recognition. Syntactic models tend to be of the n-gram variety and could capture much more structure if association statistics were collected on the basis of syntactic role rather than simple adjacency. Although semantics is much less well understood than either prosody or syntax, it is still amenable to more detailed statistical modeling than is presently done and the use of integrated syntactico-semantic models also seems worthy of further exploration. The above suggestions are indicative of the myriad possibilities for improvement of the speech technologies by building directly upon existing methods. However, the speech research community would do well to consider the possibility that no amount of incremental improvement will lead to a technology which displays human-like proficiency with language. The obvious and prudent policy for avoiding such an impasse is to encourage completely new concepts and models of speech processing and new generations of researchers to invent them.

11.3 DSP Techniques

John Makhoul

BBN Systems and Technologies, Cambridge, Massachusetts, USA

Digital signal processing (DSP) techniques have been at the heart of progress in speech processing during the last 25 years (Rabiner & Schafer, 1978). Simultaneously, speech processing has been an important catalyst for the development of DSP theory and practice. Today, DSP methods are used in speech analysis, synthesis, coding, recognition, and enhancement, as well as voice modification, speaker recognition, and language identification.

DSP techniques have also been very useful in written language recognition in all its forms (on-line, off-line, printed, handwritten). Some of the methods include preprocessing techniques for noise removal, normalizing transformations for line width and slant removal, global transforms (e.g., Fourier transform, correlation), and various feature extraction methods. Local features include the computation of slopes, local densities, variable masks, etc., while others deal with various geometrical characteristics of letters (e.g., strokes, loops). For summaries of various DSP techniques employed in written language recognition, the reader is referred to Impedovo, Ottaviano, et al. (1991); Tappert, Suen, et al. (1990), as well as the following edited special issues: Impedovo (1994); Pavlidis and Mori (1992); Impedovo and Simon (1992).

This section is a brief summary of DSP techniques that are in use today, or that may be useful in the future, especially in the speech recognition area. Many of these techniques are also useful in other areas of speech processing.

11.3.1 Feature Extraction

In theory, it should be possible to recognize speech directly from the digitized waveform. However, because of the large variability of the speech signal, it is a good idea to perform some form of feature extraction that would reduce that variability. In particular, computing the envelope of the short-term spectrum reduces the variability significantly by smoothing the detailed spectrum, thus eliminating various source information, such as whether the sound is voiced or fricated and, if voiced, it eliminates the effect of the periodicity or pitch. For nontonal languages, such as English, the loss of source information does not appear to affect recognition performance much because it turns out that the spectral envelope is highly correlated with the source information. However, for tonal languages, such as Mandarin Chinese, it is important to include an estimate of the fundamental frequency as an additional feature to aid in the recognition of tones (Hon, Yuan, et al., 1994).

To capture the dynamics of the vocal tract movements, the short-term spectrum is typically computed every 10–20 ms using a window of 20–30 ms. The spectrum can be represented directly in terms of the signal's Fourier coefficients or as the set of power values at the outputs from a bank of filters. The envelope of the spectrum can be represented indirectly in terms of the parameters of an all-pole model, using linear predictive coding (LPC), or in terms of the first dozen or so coefficients of the cepstrum—the inverse Fourier transform of the logarithm of the spectrum.

One reason for computing the short-term spectrum is that the cochlea of the human ear performs a quasi-frequency analysis. The analysis in the cochlea takes place on a nonlinear frequency scale (known as the Bark scale or the mel scale). This scale is approximately linear up to about 1000 Hz and is approximately logarithmic thereafter. So, in the feature extraction, it is very common to perform a frequency warping of the frequency axis after the spectral computation.

Researchers have experimented with many different types of features for use in speech recognition (Rabiner & Juang, 1993). Variations on the basic spectral computation, such as the inclusion of time and frequency masking, have been shown to provide some benefit in certain cases (Aikawa, Singer, et al., 1993; Bacchiani & Aikawa, 1994; Hermansky, 1990). The use of auditory models as the basis of feature extraction has been useful in some systems (Cohen, 1989), especially in noisy environments (Hunt, Richardson, et al., 1991).

Perhaps the most popular features used for speech recognition today are what are known as mel-frequency cepstral coefficients (MFCCs) (Davis & Mermelstein, 1980). These coefficients are obtained by taking the inverse Fourier transform of the log spectrum after it is warped according to the mel scale. Additional discussion of feature extraction issues can be found in section 1.3 and section 11.2.

11.3.2 Dealing with Channel Effects

Spectral distortions due to various channels, such as a different microphone or telephone, can have enormous effects on the performance of speech recognition systems. To render recognition systems more robust to such distortions, many researchers perform some form of removal of the average spectrum. In the cepstral domain, spectral removal amounts to subtracting out the average cepstrum. Typically, the average cepstrum is estimated over a period of time equal to about one sentence (a few seconds), and that average is updated on an ongoing basis to track any changes in the channel. Other similarly simple methods of filtering the cepstral coefficients have been proposed for removing channel effects (Hermansky, Morgan, et al., 1993). All these methods have been very effective in combating recognition problems due to channel effects. Further discussion of issues related to robust speech recognition can be found in section 1.4.

11.3.3 Vector Quantization

For recognition systems that use hidden Markov models, it is important to be able to estimate probability distributions of the computed feature vectors. Because these distributions are defined over a high-dimensional space, it is often easier to start by quantizing each feature vector to one of a relatively small number of template vectors, which together comprise what is called a codebook. A typical codebook would contain about 256 or 512 template vectors. Estimating probability distributions over this finite set of templates then becomes a much simpler task. The process of quantizing a feature vector into a finite number of template vectors is known as vector quantization (Makhoul, Roucos, et al., 1985). The process takes a feature vector as input and finds the template vector in the codebook that is closest in distance. The identity of that template is then used in the recognition system.

11.3.4 Future Directions

Historically, there has been an ongoing search for features that are resistant to speaker, noise, and channel variations. In spite of the relative success of MFCCs as basic features for recognition, there is a general belief that there must be more that can be done. One challenge is to develop ways in which our knowledge of the speech signal, and of speech production and perception, can be incorporated more effectively into recognition methods. For example, the fact that speakers have different vocal tract lengths could be used to develop more compact models for improved speaker-independent recognition. Another challenge is somehow to integrate speech analysis into the training optimization process. For the near term, such integration will no doubt result in massive increases in computation that may not be affordable.

There have been recent developments in DSP that point to potential future use of new nonlinear signal processing techniques for speech recognition purposes. Artificial neural networks, which are capable of computing arbitrary nonlinear functions, have been explored extensively for purposes of speech recognition, usually as an adjunct or substitute for hidden Markov models. However, it is possible that neural networks may be best utilized for the computation of new feature vectors that would rival today's best features.

Work by Maragos, Kaiser, et al. (1992) with instantaneous energy operators, which have been shown to separate amplitude and frequency modulations, may be useful in discovering such modulations in the speech signal and, therefore, may be the source of new features for speech recognition. The more general quadratic operators proposed by Atlas and Fang (1992) offer a rich family of possible operators that can be used to compute a large number of features that exhibit new properties which should have some utility for speech processing in general and speech recognition in particular.

11.4 Parsing Techniques

Aravind Joshi

University of Pennsylvania, Philadelphia, Pennsylvania, USA

Parsing a sentence means to compute the structural description (descriptions) of the sentence assigned by a grammar, assuming, of course, that the sentence is well-formed. Mathematical work on parsing consists of at least the following activities.

1. Mathematical characterization of derivations in a grammar and the associated parsing algorithms.
2. Computing the time and space complexities of these algorithms in terms of the length of the sentence and the size of the grammar, primarily,
3. Comparing different grammar formalisms and showing equivalences among them wherever possible, thereby developing uniform parsing algorithms for a class of grammars.
4. Characterizing parsing as deduction and a uniform specification of parsing algorithms for a wide class of grammars.
5. Combining grammatical and statistical information for improving the efficiency of parsers and ranking of multiple parses for a sentence.

The structural descriptions provided by a grammar depend on the grammar formalism to which the grammar belongs. For the well-known context-free grammar (CFG) the structural description is, of course, the conventional phrase structure tree (tress) associated with the sentence. The parse tree describes the structure of the sentence. It is also the record of the history of derivation of the sentence. Thus, in this case the structural description and the history of the derivation are the same objects. Later, we will comment on other grammar formalisms and the structural descriptions and histories of derivation associated with them.

11.4.1 Parsing Complexity

For CFGs we have the well-known algorithms by Cocke, Kasami, and Younger (CKY) (Kasami, 1965; Younger, 1967) and the Earley algorithm (Earley, 1970). All CFG algorithms are related to these two in one way or another. As regards the complexity of these algorithms it is well-known that the worst case complexity is $O(n^3)$ where n is the length of the sentence. There is a multiplicative factor which depends on the size of the grammar and it is $O(G^2)$ where G is the size of the grammar (expressed appropriately in terms of the number of rules and the number of non-terminals). There are results which show improvements in the exponents of both n and G but these are not significant for our purpose. Of course, these complexity results mostly are worst case results (upper bounds) and therefore, they are not directly useful. They do however establish polynomial parsing of these grammars. There are no mathematical average case results. All average case results reported are empirical. In practice, most algorithms for CFGs run much better than the worst case and the real limiting factor in practice is the size of the grammar. For a general discussion of parsing strategies, see Leermakers (1993); Nedderhoff (1994); Sikkel (1994).

During the past decade or so a number of new grammar formalisms have been introduced for a variety of reasons, for example, eliminating transformations in a grammar, accounting linguistic structures beyond the reach of context-free grammars, integrating syntax and semantics directly, etc.. Among these new formalisms, there is one class of grammars called *mildly context-sensitive grammars* that has been mathematically investigated very actively. In particular, it has been shown that a number of grammar formalisms belonging to this class are weakly equivalent, i.e., they generate the same set of string languages. Specifically, tree-adjoining grammars (TAG), combinatory categorial grammars (CCG), linear indexed grammars (LIG), and head grammars (HG) are weakly equivalent. From the perspective of parsing, weak equivalence by itself is not very interesting because weak equivalence alone cannot guarantee that a parsing technique developed for one class of grammars can be extended to other classes, or a uniform parsing procedure can be developed for all these equivalent grammars. Fortunately, it has been shown that indeed it is possible to extend a recognition algorithm for CFGs (the CKY algorithm) for parsing linear indexed grammars (LIG). Then this parser can be adapted for parsing TAGs, HGs, as well as CCGs. This new algorithm is polynomial, the complexity being $O(n^6)$. The key mathematical notion behind the development of this general algorithms is that in all these grammars what can happen in a derivation depends only on which of the finite set of *states* the derivation is in. For CFG these states can be nonterminal symbols. This property called the *context-freeness* property is crucial because it allows one to keep only a limited amount of context during the parsing process, thus resulting in a polynomial time algorithm. For CFGs this property holds trivially. The significant result here is that this property also extends to the grammars more powerful than CFGs, mentioned above. An Earley type algorithm has also been developed for the tree-adjoining grammars and its complexity has been shown to be also $O(n^6)$. For further information on these results, see Joshi, Vijay-Shanker, et al. (1991); Schabes and Joshi (1988); Vijay-Shanker and Weir (1993).

11.4.2 Derivation Trees

Although the grammars mentioned above are weakly equivalent and uniform parsing strategies have been developed for them, it should be noted that the notions of what constitutes a parse are quite different for each one of these grammars. Thus in a TAG the real parse of a sentence is the so-called derivation tree, which is a record of how the elementary trees of a TAG are put together by the operations of substitution and adjoining in order to obtain the derived tree whose yield is the string being parsed. The nodes of the derivation tree are labeled by the names of the elementary trees and the edges are labeled by the addresses of the tree labeling the parent node in which the trees labeling the daughter nodes are either substituted or adjoined. This derivation tree is unlike the derivation tree for a CFG for which the notions of the derivation tree and the derived tree are the same. For TAG these are distinct notions. For HG which deals with headed strings and operations of concatenation and wrapping (both are string operations) there is no notion of a derived tree as such. There is only the notion of a derivation tree which is a record of how the elementary strings are put together and what operations were used in this process. The terminal nodes are labeled by elementary strings (headed strings) and the other nodes are labeled by the operations used for combining the strings labeling the daughter nodes and also by the string resulting by performing this operation. Thus this derivation tree is quite unlike the standard phrase structure tree, especially when the combining operation labeling a node is wrapping (wrapping one string around another to the right or left of its head) as a non-standard constituent structure can be defined for the resultant tree.

For a CCG the parse of a sentence is the proof tree of the derivation. It is like the phrase structure tree in the sense that the nodes are labeled by categories in CCG, however, for each node the name of the operation used in making the reduction (for example, function application or function composition) has to be stated at the node also. Thus, in this sense they are like the derivation trees of HG. The derivation trees of LIG are like the phrase structure trees except that with each node the contents of the stack associated with that node are stated. Given this wide divergence of what constitutes a structural description, the significance of the equivalence result and the existence of a general polynomial parsing strategy can be better appreciated.

11.4.3 Unification-based Grammars

Almost all computational grammars incorporate feature structures (attribute value structures), the category label being a special attribute singled out for linguistic convenience and not for any formal reasons. These feature structures are manipulated by the operation of unification, hence the term *unification-based grammars*. CFGs or any of the grammars mentioned above can serve as the backbones for the unification-based grammars, CFGs being the most common (Shieber, 1988). As soon as feature structures and unification are added to a CFG the resulting grammars are Turing equivalent and they are no longer polynomially parsable. In practice, conditions are often placed on the possible feature structures which allow the polynomial probability to be restored. The main reason for the excessive power of the unification-based grammars is that recursion can be encoded in the feature structures. For certain grammars in the class of mildly context-sensitive grammars, in particular for TAGs, recursion is factored away from the statement of so-called long-distance dependencies. The feature structures can be without any recursion in them, thus preserving the polynomial parsability of the TAG backbone.

Some unification-based grammar formalisms, for example, the lexical-functional grammar (LFG) are very explicit in assigning both a phrase structure and a feature structure based functional structure to the sentence being parsed. Formal properties of the interface between these two components have been studied recently. In particular, computational complexity of this interface has been studied independently of the complexity of the phrase structure component and the feature structure component. A number of properties of different interface strategies have been studied that can be exploited for computational advantage. A surprising result here is that under certain circumstances an interface strategy that does no pruning in the interface performs significantly better than one that does. For an interesting discussion of these results, see Maxwell and Kaplan (1993).

11.4.4 Parsing as Deduction

Parsing can be viewed as a deductive process as is the case in CCG mentioned above. The Lambek Calculus (LC) is a very early formulation of parsing as deduction (Lambek, 1958). The relationship between LC and CFG was an open question for over thirty years. Very recently it has been shown that LC and CFG are weakly equivalent (Pentus, 1993). However, the proof of this equivalence does not seem to suggest a construction of a polynomial parsing algorithm for LC and this is an important open question. The framework of parsing as deduction allows modular separation of the logical aspects of the grammar and the proof search procedure, thus providing a framework for investigating a wide range of parsing algorithms. Such theoretical investigations have led to the development of a program for rapid prototyping and experimentation with new parsing algorithms and has been also used in the development of algorithms for CCGs, TAGs, and lexicalized CFGs. For further details, see Shieber, Schabes, et al. (1994).

11.4.5 LR Parsing

Left-to-right, rightmost derivation (LR) parsing was introduced initially for efficient parsing of languages recognized by deterministic pushdown automata and have proven useful for compilers. For natural language parsing LR parsers are not powerful enough, however conflicts between multiple choices are solved by pseudo-parallelism (Lang, 1974; Tomita, 1987). Johnson and Kipps independently noted that the Tomita method is not bounded by any polynomial in the length of the input string and the size of the grammar. Kipps also shows how to repair this problem. These results are presented in Tomita (1991). LR parsing has been applied to non-context-free languages also in the context of natural language parsing (Schabes & Vijayshanker, 1990).

11.4.6 Parsing by Finite State Transducers

Finite state devices have always played a key role in natural language processing. There is renewed interest in these devices because of their successful use in morphological analysis by representing very large dictionaries by finite state automata (FSA) and by representing two-level rules and lexical information with finite state transducers (FST). FSAs have been used for parsing also, as well as for approximating CFGs. A main drawback of using FSA for parsing is the difficulty of representing hierarchical structure, thus giving incomplete parses in some sense. Recently, there has been theoretical work on FSTs for their use in parsing, one of the approaches being the use of an FST and computing the parse as a fixed point of the transduction. The parsers can be very efficient and are well suited for large highly lexicalized grammars. For further details on these issues, see (Karttunen, Kaplan, et al., 1992; Pereira, Rebecca, et al., 1991; Roche, 1994).

11.4.7 Remarks

We have not discussed various related mathematical topics, for example, mathematical properties of grammars, unless they are directly relevant to parsing, and the mathematical results in the application of statistical techniques to parsing. This latter topic is the subject of another contribution in this chapter. It is worth pointing out that recent mathematical investigations on lexicalized grammars have great significance to the use of statistical techniques for parsing as these grammars allow a very direct representation of the appropriate lexical dependencies.

11.4.8 Future Directions

Some of the key research problems are (1) techniques for improving the efficiency of the parsing systems by exploiting lexical dependencies, (2) techniques for exploiting certain regularities in specific domains, e.g., particular sentence patterns tend to appear more often in specific domains, (3) systematic techniques for computing partial parses, less than complete parses in general, (4) applying finite state technology for parsing, in particular for partial parses, (5) systematic techniques for integrating parsing with semantic interpretation and translation, (6) investigating parallel processing techniques for parsing and experimenting with large grammars.

Small workshops held in a periodic fashion where both theoretical and experimental results can be informally discussed will be the best way to exchange information in this area. Specific initiatives for parsing technology, for example for parallel processing technology for parsing are needed for rapid progress in this area.

11.5 Connectionist Techniques

Hervé Bourlard[a] & Nelson Morgan[b]

[a] Faculté Polytechnique de Mons, Mons, Belgium
[b] International Computer Science Institute, Berkeley, California, USA

There are several motivations for the use of connectionist systems in human language technology. Some of these are:

- Artificial Neural Networks (ANN) can learn in either a supervised or unsupervised way from training examples. This property is certainly not specific to ANNs, as many kinds of pattern recognition incorporate learning. In the case of ANNs, however, it sometimes is easier to eliminate some system heuristics, or to partially supplant the arbitrary or semi-informed selection of key parameters. Of course this is not often done completely or in an entirely blind way, but usually requires the application of some task-dependent knowledge on the part of the system designer.

- When ANNs are trained for classification, they provide discriminant learning. In particular, ANN outputs can estimate posterior probabilities of output classes conditioned on the input pattern. This can be proved for the case when the network is trained for classification to minimize one of several common cost functions (e.g., least mean square error or relative entropy). It can be easily shown that a system that computes these posterior probabilities minimizes the error rate while maximizing discrimination between the correct output class and rival ones; the latter property is described by the term *discriminant*. In practice, it has been shown experimentally that real ANN-based systems could be trained to generate good estimates of these ideal probabilities, resulting in useful pattern recognizers.

- When used for classification, prediction or parsing (or any other input/output mapping), ANNs with one hidden layer and *enough* hidden nodes can approximate any continuous function.

- Because ANNs can incorporate multiple constraints and find optimal combinations of constraints, there is no need for strong assumptions about the statistical distributions of the input features or about high order correlation of the input data. In theory, this can be discovered automatically by the ANNs during training.

- ANN architecture is flexible, accommodating contextual inputs and feedback. Also, ANNs are typically highly parallel and regular structures, permitting efficient hardware implementations.

In the following, we briefly review some of the typical functional building blocks for HLT, and show how connectionist techniques could be used to improve them. In subsection 11.5.3, we discuss a particular instance that we are experienced in using. Finally, in the last subsection, we discuss some key research problems.

11.5.1 ANNs and Feature Extraction

Feature extraction consists of transforming the raw input data into a concise representation that contains the relevant information and is robust to variations. For speech, for instance, the waveform is typically translated into some kind of a function of a short term spectrum. For handwriting recognition, pixels are sometimes complemented by dynamic information before they are translated into task-relevant features.

It would be desirable to automatically determine the parameters or features for a particular HLT task. In some limited cases, it appears to be possible to automatically derive features from raw data, given significant application-specific constraints. This is the case for the AT&T handwritten zip code recognizer (le Cun, Boser, et al., 1990), in which a simple convolutional method was used to extract important features such as lines and edges that are used for classification by an ANN.

Connectionist networks have also been used to investigate a number of other approaches to unsupervised data analysis, including linear dimension reduction [including Bourlard and Kamp (1988), in which it was shown that feedforward networks used in auto-associative mode are actually performing principal component analysis (PCA)], non-linear dimension reduction (Oja, 1991; Kambhatla & Leen, 1994), Reference-point based classifiers, such as vector quantization vector quantization and topological map (Kohonen, 1988).

It is however often better to make use of any task-specific knowledge whenever it is possible to reduce the amount of information to be processed by the network and to make its task easier. As a consequence, we note that, while automatic feature extraction is a desirable goal, most ASR systems use neural networks to classify speech sounds using standard signal processing tools (like Fourier transform) or features that are selected by the experimenter (e.g., Cole, Fanty, et al. (1991)).

11.5.2 ANNs and Pattern Sequence Matching

Although ANNs have been shown to be quite powerful in static pattern classification, their formalism is not very well suited to address most issues in HLT. Indeed, in most of these cases, patterns are primarily sequential and dynamical. For example, in both ASR and handwriting recognition, there is a time dimension or a sequential dimension which is highly variable and difficult to handle directly in ANNs. We note however that ANNs have been successfully applied to time series prediction in several task domains (Weigend & Gershenfeld, 1994). HLT presents several challenges. In fact, many HLT problems can be formulated as follows: how can an input sequence (e.g., a sequence of spectra in the case of speech and a sequence of pixel vectors in the case of handwriting recognition) be properly explained in terms of an output sequence (e.g., sequence of phonemes, words or sentences in the case of ASR or a sequence of written letters, words or phrases in the case of handwriting recognition) when the two sequences are not synchronous (since there usually are multiple inputs associated with each pronounced or written word)?

Several neural network architectures have been developed for (time) sequence classification, including:[1]

- Static networks with an input buffer to transform a temporal pattern into a spatial pattern (Bourlard & Morgan, 1993; Lippmann, 1989).
- Recurrent networks that accept input vectors sequentially and use a recurrent internal state that is a function of the current input and the previous internal state (Jordan, 1989; Kuhn, Watrous, et al., 1990; Robinson & Fallside, 1991).
- Time-delay neural networks, approximating recurrent networks by feedforward networks (Lang, Waibel, et al., 1990).

In the case of ASR, all of these models have been shown to yield good performance (sometimes better than HMMs) on short isolated speech units. By their recurrent aspect and their implicit or explicit temporal memory they can perform some kind of integration over time. This conclusion remains valid for related HLT problems. However, neural networks by themselves have not been shown to be effective for large scale recognition of continuous speech or cursive handwriting. The next section describes a new approach that combines ANNs and HMMs for large vocabulary continuous speech recognition.

11.5.3 Hybrid HMM/ANN Approach

Most commonly, the basic technological approach for automatic speech recognition (ASR) is statistical pattern recognition using hidden Markov models (HMMs) as

[1] For a good earlier review of the different approaches using neural networks for speech recognition, see Lippmann (1989). For a good overview of ANNs for speech processing in general, see Morgan and Scofield (1991).

presented in sections 1.5 and 11.2. The HMM formalism has also been applied to other HLT problems such as handwriting recognition (Chen, Kundu, et al., 1994). [2]

Recently, a new formalism of classifiers particularly well suited to sequential patterns (like speech and handwritten text) and which combines the respective properties of ANNs and HMMs was proposed and successfully used for difficult ASR (continuous speech recognition) tasks (Bourlard & Morgan, 1993). This system, usually referred to as the hybrid HMM/ANN combines HMM sequential modeling structure with ANN pattern classification (Bourlard & Morgan, 1993). Although this approach is quite general and recently was also used for handwriting recognition (Schenkel, Guyon, et al., 1994; Schenkel, Guyon, et al., 1995) and speaker verification (Naik & Lubensky, 1994), the following description will mainly apply to ASR problems.

As in standard HMMs, hybrid HMM/ANN systems applied to ASR use a Markov process to temporally model the speech signal. The connectionist structure is used to model the local feature vector conditioned on the Markov process. For the case of speech this feature vector is local in time, while in the case of handwritten text it is local in space. This hybrid is based on the theoretical result that ANNs satisfying certain regularity conditions can estimate class (posterior) probabilities for input patterns (Bourlard & Morgan, 1993); i.e., if each output unit of an ANN is associated with each possible HMM state, it is possible to train ANNs to generate posterior probabilities of the state conditioned on the input. This probability can then be used, after some modifications (Bourlard & Morgan, 1993), as local probabilities in HMMs.

Advantages of the HMM/ANN hybrid for speech recognition include:

- a natural structure for discriminative training,
- no strong assumptions about the statistical distribution of the acoustic space,
- parsimonious use of parameters,
- better robustness to insufficient training data,
- an ability to model acoustic correlation (using contextual inputs or recurrence).

In recent years these hybrid approaches have been compared with the best classical HMM approaches on a number of HLT tasks. In cases where the comparison was controlled (e.g., where the same system was used in both cases except for the means of estimating emission probabilities), the hybrid approach performed better when the number of parameters were comparable, and about the same for some cases in which the

[2] Neural network equivalents of standard HMMs have been studied, but essentially are different implementations of the same formalism (Bridle, 1990; Lippmann, 1989) and are not discussed further here.

classical system used many more parameters. Also, the hybrid system was quite efficient in terms of CPU and memory run-time requirements. Evidence for this can be found in a number of sources, including:

- Renals, Morgan, et al. (1994) in which results on Resource Management (a standard reference database for testing ASR systems) are presented, and
- Lubensky, Asadi, et al. (1994) in which high recognition accuracy on a connected digit recognition task is achieved using a fairly straightforward HMM/ANN hybrid (and is compared to state-of-the-art multi-Gaussian HMMs).
- More recently, such a system has been evaluated under both the North American ARPA program and the European LRE SQALE project (20,000 word vocabulary, speaker independent continuous speech recognition). In the preliminary results of the SQALE evaluation (reported in Steeneken and Van Leeuwen (1995)) the system was found to perform slightly better than any other leading European system and required an order of magnitude less CPU resources to complete the test.

More generally, though, complete systems achieve their performance through detailed design, and comparisons are not predictable on the basis of the choice of the emission probability estimation algorithm alone.

ANNs can also be incorporated in a hybrid HMM system by training the former to do nonlinear prediction (Levin, 1993), leading to a nonlinear version of what is usually referred to as autoregressive HMMs (Juang & Rabiner, 1985).

11.5.4 Language Modeling and Natural Language Processing

Connectionist approaches have also been applied to natural language processing. Like the acoustic case, NLP requires the sequential processing of symbol sequences (word sequences). For example, HMMs are a particular case of a FSM, and the techniques used to simulate or improve acoustic HMMs are also valid for language models. As a consequence, much of the work on connectionist NLP has used ANNs to simulate standard language models like FSMs.

In 1969, Minsky and Papert (1969), showed that ANNs can be used to simulate a FSM. More recently, several works showed that recurrent networks simulate or validate regular and context-free grammars. For instance, in Liu, Sun, et al. (1990), a recurrent network feeding back output activations to the previous (hidden) layer was used to validate a string of symbols generated by a regular grammar. In Sun, Chen, et al. (1990), this was extended to CFGs. Structured connectionist parsers were developed by a number of researchers, including Fanty (1985) and Jain (1992). The latter parser was incorporated in a speech-to-speech translation system (for a highly constrained conference-registration task) that was described in Waibel, Jain, et al. (1992).

Neural networks have also been used to model semantic relations. There have been many experiments of this kind over the years. For example, Elman (1988) showed that neural networks can be trained to learn pronoun reference. He used a partially recurrent network for this purpose, consisting of a feedforward MLP with feedback from the hidden layer back into the input.

The work reported so far has focused on simulating standard approaches with neural networks, and it is not yet known whether this can be helpful in the integration of different knowledge sources into a complete HLT system. Generally speaking, connectionist language modeling and NLP has thus far played a relatively small role in large or difficult HLT tasks.

11.5.5 Future Directions

There are many open problems in applying connectionist approaches to HLT, and in particular for ASR, including:

- Better modeling of nonstationarity—speech and cursive handwriting are in fact not piecewise stationary. Two promising directions for this in the case of speech are the use of articulatory or other segment-based models, and perceptually-based models that may reduce the modeling space based on what is significant to the auditory system.
- Avoiding conditional independence assumptions—to some extent the HMM/ANN hybrid approach already does this, but there are still a number of assumptions regarding temporal independence that are not justified. Perceptual approaches may also help here, as may the further development of dynamic or predictive models.
- Developing better signal representations—the above approaches may need new features in order to work well in realistic conditions.
- Better incorporation of language models and world knowledge—while we have briefly mentioned the use of connectionist language models and NLP, all of the schemes currently employed provide only a rudimentary application of knowledge about the world, word meanings, and sentence structure. These factors must

ultimately be incorporated in the statistical theory, and connectionist approaches are a reasonable candidate for the underlying model.

- Learning to solve these problems through a better use of modular approaches—this consists of both the design of better solutions to subtasks, and more work on their integration.

These are all long term research issues. Many intermediate problems will have to be solved before anything like an optimal solution can be found.

11.6 Finite State Technology

Ronald M. Kaplan

Xerox Palo Alto Research Center, Palo Alto, California, USA

A formal language is a set of strings (sometimes called sentences) made up by concatenating together symbols (characters or words) drawn from a finite alphabet or vocabulary. If a language has only a finite number of sentences, then a complete characterization of the set can be given simply by presenting a finite list of all the sentences. But if the language contains an infinite number of sentences (as all interesting languages do), then some sort of recursive or iterative description must be provided to characterize the sentences. This description is sometimes given in the form of a grammar, a set of pattern-matching rules that can be applied either to produce the sentences in the language one after another or else to recognize whether a given string belongs to the set. The description may also be provided by specifying an automaton, a mechanistic device that also operates either to produce or recognize the sentences of the language.

Languages have been categorized according to the complexity of the patterns that their sentences must satisfy, and the basic classifications are presented in all the standard textbooks on formal language theory, e.g., Hopcroft and Ullman (1979). The sentences of a *regular language*, for example, have the property that what appears at one position in a string can depend only on a bounded amount of information about the symbols at earlier positions. Consider the language over the alphabet a, b, c whose sentences end with a c and contain an a at a given position only if there is an earlier b. The strings *cccc* and *bac* belong to this language but *abc* does not. This is a regular language since it only requires a single bit to record whether or not a b has previous appeared. On the other hand, the language whose sentences consist of some number of a's followed by exactly the same number of b's is not a regular language since there is no upper bound on the number of a's that the allowable number of b's depends on. This set of strings belongs instead to the mathematically and computationally more complex class of *context-free* languages.

A regular language can be described by grammars in various notations that are known to be equivalent in their expressive power—they each can be used to describe all and only the regular languages. The most common way of specifying a regular language is by means of a *regular expression*, a formula that indicates the order in which symbols can be concatenated, whether there are alternative possibilities at each position, and whether substrings can be arbitrarily repeated. The regular expression $\{c^*b\{a|b|c\}^*|\epsilon\}c$ denotes the regular language described above. In the notation used here, concatenation is represented by sequence in the formula, alternatives are enclosed in braces and separated by vertical bars, asterisks (often called the *Kleene closure* operator) indicate that strings satisfying the previous subexpressions can be freely repeated, and ϵ denotes the empty string, the string containing no elements.

The regular languages are also exactly those languages that can be accepted by a particular kind of automaton, a *finite-state machine*. A finite-state machine (fsm) consists of a finite number of states and a function that determines transitions from one state to another as symbols are read from an input tape. The machine starts at a distinguished initial state with the tape positioned at the first symbol of a particular string. The machine transitions from state to state as it reads the tape, eventually coming to the end of the string. At that point, if the machine is in one of a designated set of *final* states, we say that the machine has *accepted* the string or that the string belongs to the language that the machine characterizes. An fsm is often depicted in a state-transition diagram where circles representing the states are connected by arcs that denote the transitions. An arrow points to the initial state and final states are marked with a double circle. The fsm, shown in Figure 11.2, accepts the language $\{c^*b\{a|b|c\}^*|\epsilon\}c$:

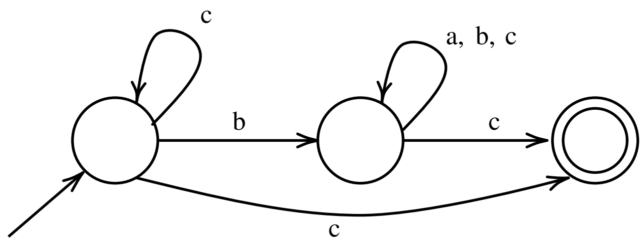

Figure 11.2: Finite-state machine diagram.

Because of their mathematical and computational simplicity, regular languages and finite-state machines have been applied in many information processing tasks. Regular expressions are often used to specify global search patterns in word-processors and in operating-system utilities such as Unix' Grep. The lexical analysis component of most modern programming language compilers is defined as a finite-state machine that recognizes identifier classes, punctuation, numbers, etc. (Aho & Ullman, 1978). But until relatively recently, finite-state techniques have not been widely used in natural language processing. This is in large measure the result of Chomsky's argument (Chomsky, 1957) that natural language sentences are so rich and complicated that they cannot be accurately characterized by means of regular or finite-state descriptions. One consequence of this was that a generation of linguists and computational linguists came to believe that finite-state techniques were of little or no interest.

It may well be true that complete and accurate natural language syntactic and semantic dependencies lie beyond the power of finite-state description, but work in the last twenty years (and particularly in the last five years) has identified a number of important problems for which very efficient and effective finite-state solutions can be found.

One set of solutions relies on the observation that finite-state descriptions can provide an approximation to the proper grammatical description of a language, an approximation that is often good enough for practical purposes. The *information extraction* problem, for example, requires that documents and passages be identified that are likely to contain information relevant to a given user's needs. Full-blown syntactic and semantic analyses of documents would certainly help to solve this problem. But such analyses may provide much more information than this task actually requires. Indeed, Appelt, Hobbs, et al. (1993) have constructed a finite-state solution that is extremely efficient compared to more powerful but more complex extraction systems and also has very favorable recall and precision scores. Their finite-state pattern specifications can only approximate a complete analysis of a text, but the approximation seems close enough for this particular purpose.

There has also been growing interest in using finite-state machines for storing and accessing natural language dictionaries. Appel and Jacobson (1988) observed that the words in a large lexicon can be very compactly represented in a state-transition graph. This is because the graph can be transformed using determinization and minimization techniques that are well-known from finite-state theory, with the result that prefixes and suffixes common to many words are collapsed into a single set of transitions. Lucchesi and Kowaltowski (1993) discuss access methods for finite-state dictionary representations that permit efficient retrieval of translation and synonymy information associated with particular words.

A third set of problems require that strings be systematically transformed into other strings. For example, the negative prefix *in* in the abstract phonological representation such as *in+practical* must be realized with an assimilated nasal as in *impractical*, or the inflected form *stopping* must be mapped either to its stem *stop* (for use in information retrieval indexing) or to its morphological analysis *stop*+PresentParticiple as an initial step in further processing. One formalism for describing these kinds of string transformations are context-sensitive rewriting rules as discussed for example by Chomsky and Halle (1968). Johnson (1972) and later Kaplan and Kay (1981); Kaplan and Kay (1994) showed that for each rule of this type there is a finite-state *transducer* that maps input strings to exactly the same outputs that the rule prescribes. A transducer is a simple extension of a basic finite-state machine: as it reads its input tape and makes a transition from one state to another, it can also write a corresponding symbol on a second tape. When it reaches a final state and the end of the input, the string produced on the second tape is taken to be the output that the input string is mapped to.

Mathematically, the collection of input-output string-pairs for a given transducer (corresponding perhaps to a particular rule) constitutes a *regular relation*. Regular relations share many (but not all) of the formal properties of the regular languages (for example, closure under union and concatenation) and also enjoy certain other properties. In particular, the regular relations are closed under *composition*, and Kaplan and Kay use this fact to show that the effect of an entire collection of rewriting rules making up a phonological or morphological grammar can be implemented as a single finite-state transducer. This device is much more efficient than any scheme using the rules directly for performing the string transformations that the grammar describes. Koskenniemi (1983) proposed a different rule notation, called two-level rules, for characterizing phonological and morphological variations. These rules also denote only regular relations and can be transformed to equivalent transducers.

Future Directions

In the years since Chomsky's original criticism of finite-state techniques used for syntactic description, our understanding of their mathematical and computational properties has increased substantially. Of most importance, however, is our increased awareness of the wide range of natural language processing problems that they can fruitfully be applied to. Problems such as dictionary access and morphological analysis seem to be inherently finite-state in character, and finite-state solutions for these problems are complete as well as efficient. For applications such as information extraction, finite-state approaches appear to provide extremely useful approximate solutions. An exciting body of current research is exploiting more sophisticated finite-state techniques to improve the accuracy of approximate syntactic analyses without sacrificing their processing efficiency (e.g., Koskenniemi, 1990; Voutilainen & Tapanainen, 1993). Given these improvements, we expect to see finite-state techniques incorporated into a growing number of practical language processing systems.

11.7 Optimization and Search in Speech and Language Processing

John Bridle

Dragon Systems UK Ltd., Cheltenham, UK

An *optimal search method* is one that always finds the best solution (or *a* best solution, if there is more than one). For our purposes *best* is in terms of the value of a criterion function, which defines a score for any possible solution. *Optimization* can be defined as the process of finding a best solution, but it is also used, more loosely, meaning to find a sequence of better and better solutions.

Optimization and *search* are vital to modern speech and natural language processing systems. Although there are optimization techniques which are not normally thought of as search, and search methods which are not optimal or not defined as optimizing anything, most often we are dealing with search methods which seek to optimize a well-defined criterion, and usually we understand the search method in terms of the way it approximates to an optimal search.

Three well-known problems for which optimal search methods are important are: training a set of models for speech recognition, decoding an acoustic pattern in terms of a sequence of word models, and parsing an errorful symbol string with the least number of assumptions of errors. Speech recognition and parsing are combinatorial optimization problems: a solution is a sequence (or more complicated data structure) of symbols, which in a speech recognition system might represent words. Training the parameters of a set of models is a problem in (non-linear) continuous optimization: a solution is a vector of real numbers (e.g., probabilities and parameters of probability density functions). It is common, in speech and language processing, to reserve the term *search* for combinatorial problems, particularly speech recognition, and to use *optimization* for continuous problems such as training.

We must never forget that an optimal solution is not necessarily correct: it is only optimal given the way the problem is posed. Speech recognition based on optimal search has the important and useful property that we can usually categorize recognition errors as *model errors* (the recognition criterion was inappropriate) or *search errors* (we failed to find the mathematically optimal solution), by checking the value of the criterion for the correct answer as well as the incorrect answer. In the case of model errors we must improve the structure of the models, or the method of training or the training data itself.

A search method for ASR is usually expected to be an optimal (finds the best-scoring answer) to good (and preferably controllable) approximation. The main properties of search methods, apart from optimality, are speed and use of computing resources. Other attributes include delay (not the same as speed) and providing extra information, such as alternative answers or details like scores and positions.

For some problems the optimal solution can be computed directly, using standard numerical algorithms. A feed-forward neural network (see section 11.5) can be used as a classifier without performing a search, and matrix inversion can be used as the basis for some training methods. However, the most successful methods for dealing with timescales and sequences in speech recognition are based on searches.

The most desirable search methods are those that are provably optimal. This is the case with the standard speech recognition algorithms, which are based on dynamic programming and the Markov property. The standard training methods use an efficient re-estimation approach, which usually converges to at least a local optimum in a few iterations through the training data. The standard model structures and training criteria are chosen partly because they are compatible with such efficient search methods. Less efficient methods, such as gradient descent or optimization by simulated annealing can be used for more difficult (but possibly more appropriate) models and training criteria.

11.7.1 Dynamic Programming-based Search for Speech Recognition

The most important search techniques for combinatorial problems in speech recognition are based on dynamic programming (DP) principles (Bellman, 1957). DP is often the key to efficient searches for the optimum path through a graph structure, when paths are evaluated in terms of an accumulation of scores along a path. The Viterbi algorithm (Forney, 1973) is an application of DP, and the forward-backward algorithm (Jelinek, 1976) is very closely related. Three speech recognition problems usually solved by DP are those of *unknown timescales*, *unknown word sequence* and *unknown word boundary position*. Our first example is a simple whole-word speech recognizer.

Consider a simple *isolated word* discrimination system with one word-model per vocabulary word. Each word model consists of a sequence of *states*, each state corresponding to a position along the word. Associated with each state is information (the *output distribution*) about the range of acoustic data (spectrum shapes, etc.) to be expected at that point in the word. Most modern speech recognition systems use word models composed of shared *sub-word models* (see section 1.5), (Poritz, 1988).

When given an unknown sound pattern (a sequence of *frames* of spectrum measurements) we shall decide which word of the vocabulary has been said by picking the word model which fits best.

Different utterances of the same word can have very different timescales: both the overall duration and the details of timing can vary greatly. The degree of fit is usually in two parts: the fit to the spectrum and the fit to the timescale. To keep our example simple, we assume that all we know (or want to use) about the timescales of words is that states are used in strict sequence, each one being used one or more times before moving on to the next one. There are many more elaborate schemes (section 1.5), most of which penalize time-scale distortion. We shall also assume that best means minimum sum of individual degree-of-fit numbers, which might be distances, or strain energy, or negative log probabilities.

If we knew how to align the states of the model with the frames of the unknown pattern we could score the model (and hence the word hypothesis) by combining the spectrum fit scores. We define the model score as the result of choosing the *best* alignment. We find that score (and the alignment if we want it) using a DP algorithm, as follows: we introduce an optimal partial score, F, where $F_{i,t}$ is the score for optimally aligning the first i states of the model to the first t frames of the input pattern. We are interested in $F_{N,T}$, the score for optimally aligning all N states of the model to all T frames of the input pattern.

For our simple example F can be computed from the DP step:

$$F_{i,t} = Min[F_{i,t-1}, F_{i-1,t-1}] + d_{i,t}$$

where $d_{i,t}$ is the degree of fit of the i^{th} state of the word model to the measured spectrum at time t. (If we are going to align state i with frame t we must align frame $t-1$ with state i or with state $i-1$.) $F_{N,T}$ can be computed by starting with $F_{1,1} = d_{i,t}$ and working through to frame T.

Simple *connected word* recognition is done by processing all the word models together, and allowing the scores to propagate from the end of one word to the start of another. In practice we choose between words that could end at the current input frame, and propagate just the best one. A pair of arrays, indexed by frame number, can keep track of the identity of the best word ending at each input frame, and the best time for it to start (Vintsyuk, 1971). The start-time for the current word is propagated with F within the words. There are several alternatives and elaborations, e.g., Ney (1984).

It is possible to operate such a connected word recognizer continuously using *partial trace-back*: when all active optimal partial paths agree about the interpretation of some past stretch of input data then nothing in the future data can change our interpretation of that stretch: we can output that result and recover the associated storage.

In a large-vocabulary system we can save a lot of storage and computation by keeping only the relatively good-scoring hypotheses. This pruning or *beam search* (Lowerre, 1976) can also reduce the delay in the partial trace-back.

The connected-word search method outlined above processes each input frame as it arrives, and considers all word endings at that time. There is another important class of methods, called stack-decoders, in which words which *start* at the same time are processed together (Paul, 1992).

We have considered tasks of finding optimal time alignments and sequence of words. It can also be very useful to find close-scoring alternative sequences (Schwartz & Austin, 1991), which can be analyzed subsequently using sources of knowledge which it is inconvenient to incorporate into the main search: for instance alternative acoustic scoring methods or application specific systems.

Most real-time large-vocabulary systems rely for their speed on a *multi-resolution* search: an initial *fast match*, using a relatively crude set of models, eliminates many possibilities, and the fine match can then be done with much less work. Sometimes the searches are performed in alternating forward and backward directions, with a combination of scores used for pruning (Austin, Schwartz, et al., 1991). Additional discussion of search in HMM-based systems can be found in section 1.5.

11.7.2 Training/Learning as Optimization

Training is the process by which useful information in training data is incorporated into models or other forms used in recognition. The term *learning* is used of the complete system which performs the training. We normally talk about training hidden Markov models, but many artificial neural networks are defined so they include the training algorithms, so we can refer to the neural network as learning from the data.

Most early artificial intelligence research on automatic learning focussed on problems of learning structure. A classic problem is learning a grammar from example sentences. Learning structure directly is a combinatorial problem, and is very difficult. Among the techniques available are genetic algorithms (Goldberg, 1989) and optimization by simulated annealing (Kirkpatrick, Gelatt, et al., 1983). Most established learning methods used in speech recognition avoid the difficulties by replacing discrete optimization problems with continuous problems. As an example, the rules of a regular grammar can be replaced by a set of probabilities, and the probabilities define a continuous (but bounded) space within which continuous optimization methods can be used. Many types of neural networks can be seen as the result of generalizing discrete, logical systems of rules so that continuous optimization methods can be applied.

In training, we are usually trying to optimize the value of a function E of the training data and of unknown parameters θ, and E is made up of a sum over the training instances $\{x_t\}$: $E = \Sigma_t E_t$, where $E_t = F(x_t, \theta)$. Evaluation of E for a given θ needs a pass over all the training data. An example from HMMs is the probability of the model generating the training data (usually a *very* small number!)

Available methods depend on the type of response surface (form of E as a function of θ) and amount of extra information available, such as derivatives.

One of the simplest cases is when E is quadratic in θ. There are standard methods for finding the optimum θ in a number of evaluations not much more than the dimensionality of θ (Press, Flannery, et al., 1988).

When $E(\theta)$ is smooth and we can also compute the partial derivative of E with respect to the components of θ, $\frac{\partial E}{\partial \theta}$, then there are many *gradient based* methods. In the simplest case (gradient descent or ascent) we adjust θ in proportion to $\frac{\partial E}{\partial \theta}$. Variations of gradient descent, such as momentum smoothing, adaptive step size, conjugate gradients and quasi-Newton methods, are available in the literature, e.g., Press, Flannery, et al. (1988), and many have been applied for training *neural networks* of the multi-layer logistic perceptron type (Rumelhart, Hinton, et al., 1986). The motivation and application of these methods in speech recognition is discussed in section 11.5.

However, the most important methods in use in speech recognition are of the expectation-maximization (E-M) type, which are particularly applicable to maximum likelihood training of stochastic models such as HMMs. The basic idea is not difficult. As an example, consider the problem of constructing the type of word model described in section 11.7.1. Assume we have a crude version of the word model, and wish to improve it. We first align the states of the model to each of several examples of the word using the DP method outlined above, then re-estimate the output distributions associated with each state, using statistics (such as the mean) of the data frames aligned to that state. This cycle can be repeated, but usually converges well in a few iterations. In the Baum-Welch procedure the alignment is a more sophisticated, probabilistic, assignment of frames to states, computed using the forward-backward algorithm, and convergence is provable for the most common forms of model. For details see section 1.5, or e.g., Poritz (1988).

11.7.3 Future Directions

It is difficult to separate progress in search and optimization algorithms from the way that the problem is posed (usually this means the form of the models). Dynamic programming style algorithms rely on a key property of the structure of the models: the pattern of immediate dependencies between random variables must form a singly connected graph. More general (but efficient) search and optimization methods would allow the exploration of models in which, for instance, multiple parallel factors explain the data more naturally and more parsimoniously.

11.8 Chapter References

Aho, A. and Ullman, J. (1978). *Principles of compiler design.* Addison-Wesley, Reading, Massachusetts.

Aikawa, K., Singer, H., Kawahara, H., and Tohkura, Y. (1993). A dynamic cepstrum incorporating time-frequency masking and its application to continuous speech recognition. In *Proceedings of the 1993 International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 668–671, Minneapolis, Minnesota. Institute of Electrical and Electronic Engineers.

Alshawi, H., editor (1992). *The Core Language Engine.* MIT Press, Cambridge, Massachusetts.

Alshawi, H. et al. (1994). Overview of the Bell Laboratories automatic speech translator. Bell Laboratories technical memorandum.

Appel, A. W. and Jacobson, G. J. (1988). The world's fastest scrabble program. *Communications of the ACM*, 31(5):572–578.

Appelt, D., Hobbs, J., Bear, J., Israel, D., Kameyama, M., and Tyson, M. (1993). The SRI MUC-5 JV-FASTUS information extraction system. In *Proceedings of the Fifth Message Understanding Conference*, Baltimore, Maryland. Morgan Kaufmann.

ARPA (1994). *Proceedings of the 1994 ARPA Human Language Technology Workshop*, Princeton, New Jersey. Advanced Research Projects Agency, Morgan Kaufmann.

Atal, B. S. and Hanauer, S. L. (1971). Speech analysis and synthesis by linear prediction of the speech wave. *Journal of the Acoustical Society of America*, 50(2):637–655.

Atlas, L. and Fang, J. (1992). Quadratic detectors for general nonlinear analysis of speech. In *Proceedings of the 1992 International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 9–12, San Francisco. Institute of Electrical and Electronic Engineers.

Austin, S., Schwartz, R., and Placeway, P. (1991). The forward-backward search algorithm. In *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 697–700, Toronto. Institute of Electrical and Electronic Engineers.

Bacchiani, M. and Aikawa, K. (1994). Optimization of time-frequency masking filters using the minimum classification error criterion. In *Proceedings of the 1994 International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 197–200, Adelaide, Australia. Institute of Electrical and Electronic Engineers.

Baker, J. K. (1975). Stochastic modeling for automatic speech understanding. In Reddy, D. R., editor, *Speech Recognition*, pages 521–542. Academic Press, New York.

Baker, J. K. (1979). Trainable grammars for speech recognition. In Wolf, J. J. and Klatt, D. H., editors, *Speech communication papers presented at the 97th Meeting of the Acoustical Society of America*, pages 547–550. Acoustical Society of America, MIT Press.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. *Inequalities*, 3:1–8.

Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology. *Bulletin of the American Medical Society*, 73:360–363.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171.

Baum, L. E. and Sell, G. R. (1968). Growth transformations for functions on manifolds. *Pac. J. Math.*, 27:211–227.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294.

Bourlard, H. and Morgan, N. (1993). *Connectionist Speech Recognition—A Hybrid Approach*. Kluwer Academic.

Bridle, J. (1990). Alpha-nets: A recurrent neural network architecture with a hidden Markov model interpretation. *Speech Communication*, 9:83–92.

Brown, P., Cocke, J., Pietra, S. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Chen, M. Y., Kundu, A., and Zhou, J. (1994). Off-line handwritten word recognition using a hidden Markov model type stochastic network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):481–496.

Chomsky, N. (1957). *Syntactic structures*. Mouton, The Hague.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English*. Harper and Row.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29.

Cohen, J. R. (1989). Application of an auditory model to speech recognition. *Journal of the Acoustical Society of America*, 85(6):2623–2629.

Cole, R. A., Fanty, M., Gopalakrishnan, M., and Janssen, R. D. T. (1991). Speaker-independent name retrieval from spellings using a database of 50,000 names. In *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 325–328, Toronto. Institute of Electrical and Electronic Engineers.

Davis, S. B. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-28:357–366.

Duda, R. O. and Hart, P. E. (1973). *Pattern Recognition and Scene Analysis*. John Wiley, New York.

Earley, J. C. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Elman, J. (1988). Finding structure in time. Technical Report CRL 8903, University of California, San Diego, Center for Research in Language, University of California, San Diego.

Fanty, M. (1985). Context free parsing in connectionist networks. Technical Report TR174, University of Rochester, Computer Science Department, University of Rochester, New York.

Forney, Jr., G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61:266–278.

Fu, K. S. (1974). *Syntactic methods in pattern recognition*. Academic Press, New York.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis for speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752.

Hermansky, H., Morgan, N., and Hirsch, H. G. (1993). Recognition of speech in additive and convolutional noise based on RASTA spectral processing. In *Proceedings of the 1993 International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 83–86, Minneapolis, Minnesota. Institute of Electrical and Electronic Engineers.

Hon, H.-W., Yuan, B., Chow, Y.-L., Narayan, S., and Lee, K.-F. (1994). Towards large vocabulary Mandarin Chinese speech recognition. In *Proceedings of the 1994 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 545–548, Adelaide, Australia. Institute of Electrical and Electronic Engineers.

Hopcroft, J. and Ullman, J. (1979). *Introduction to automata theory, languages, and computation*. Addison-Wesley.

Hunt, M. J., Richardson, S. M., Bateman, D. C., and Piau, A. (1991). An investigation of PLP and IMELDA acoustic representations and of their potential for combination. In *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 881–884, Toronto. Institute of Electrical and Electronic Engineers.

ICASSP (1991). *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, Toronto. Institute of Electrical and Electronic Engineers.

ICASSP (1992). *Proceedings of the 1992 International Conference on Acoustics, Speech, and Signal Processing*, San Francisco. Institute of Electrical and Electronic Engineers.

ICASSP (1993). *Proceedings of the 1993 International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, Minnesota. Institute of Electrical and Electronic Engineers.

ICASSP (1994). *Proceedings of the 1994 International Conference on Acoustics, Speech, and Signal Processing*, Adelaide, Australia. Institute of Electrical and Electronic Engineers.

ICSLP (1994). *Proceedings of the 1994 International Conference on Spoken Language Processing*, Yokohama, Japan.

IJCNN (1990). *Proceedings of the 1990 International Joint Conference on Neural Networks*, Washington, DC.

Impedovo, S., editor (1994). *Fundamentals in Handwriting Recognition.* NATO-Advanced Study Institute Series F. Springer-Verlag.

Impedovo, S., Ottaviano, L., and Occhinegro, S. (1991). Optical Character Recognition—A Survey. *International Journal on Pattern Recognition and Artificial Intelligence*, 5(1-2):1–24.

Impedovo, S. and Simon, J. C., editors (1992). *From Pixels to Features III.* Elsevier Science, Amsterdam.

Jacobs, J., v. Stechow, A., Sternefeld, W., and Vennemann, T., editors (1993). *Syntax, An International Handbook of Contemporary Research.* de Gruyter, Berlin, New York.

Jain, A. (1992). Generalization performance in PARSEC - a structured connectionist parsing architecture. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 4*, pages 209–216. Morgan Kaufmann.

Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64:532–556.

Jelinek, F. (1990). Computation of the probability of initial substring generation by stochastic context free grammars. Technical report, IBM T. J. Watson Research Center, Yorktown Heights, New York.

Johnson, C. D. (1972). *Formal Aspects of Phonological Description.* Mouton, The Hague.

Jordan, M. (1989). Serial order: A parallel distributed processing approach. In Elman, J. L. and Rumelhart, D. E., editors, *Advances in Connectionist Theory: Speech.* Hillsade.

Joshi, A. K., Vijay-Shanker, K., and Weir, D. J. (1991). The convergence of mildly context-sensitive grammatical formalisms. In Sells, P., Shieber, S., and Wasow, T., editors, *Foundational Issues in Natural Language Processing.* MIT Press.

Juang, B. H. (1992). Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40(12).

Juang, B. H. et al. (1987). On the use of bandpass littering in speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35:947–954.

Juang, B. H. and Rabiner, L. R. (1985). Mixture autoregressive hidden Markov models for speech signals. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(6):1404–1413.

Kadambe, S. and Hieronymus, J. L. (1994). Spontaneous speech language identification with a knowledge of linguistics. In *Proceedings of the 1994 International Conference on Spoken Language Processing*, volume 4, pages 1879–1882, Yokohama, Japan.

Kambhatla, N. and Leen, T. K. (1994). Fast non-linear dimension reduction. In Cowan, J. D. et al., editors, *Advances in Neural Information Processing Systems VI*, pages 152–159. Morgan Kaufmann.

Kaplan, R. and Kay, M. (1981). Phonological rules and finite-state transducers. In *Proceedings of the Winter meeting of the Linguistic Society of America*, New York.

Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378. written in 1980.

Karttunen, L., Kaplan, R. M., and Zaenen, A. (1992). Two-level morphology with composition. In *Proceedings of the 14th International Conference on Computational Linguistics*, volume 1, pages 141–148, Nantes, France. ACL.

Kasami, T. (1965). An efficient recognition and syntax algorithm for context-free languages. Technical Report AF-CRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

Kasper, R., Kiefer, B., Netter, K., and Vijay-Shanker, K. (1995). Compilation of HPSG to TAG. In *Proceeds of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 92–99, Cambridge, Massachusetts.

Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P. (1983). Optimisation by simulated annealing. *Science*, 220(4598):671–680.

Kohonen, T. (1988). The 'neural' phonetic typewriter. *IEEE Computer*, pages 11–22.

Koskenniemi, K. (1983). *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production.* PhD thesis, University of Helsinki. Publications of the Department of General Linguistics,University of Helsinki, No. 11. Helsinki.

Koskenniemi, K. (1990). Finite-state parsing and disambiguation. In Karlgren, H., editor, *Proceedings of the 13th International Conference on Computational Linguistics*, volume 2, pages 229–232, Helsinki. ACL.

Kuhn, G., Watrous, R. L., and Ladendorf, D. (1990). Connected recognition with a recurrent network. *Speech Communication*, 9(1):41–48.

Lambek, J. (1958). The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170.

Lang, B. (1974). Deterministic techniques for efficient non-deterministic parsers. In Loeckx, J., editor, *Proceedings of the 2nd Colloquium on Automata, Languages and Programming*, pages 255–269, Saarbrücken, Germany. Springer-Verlag.

Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43.

le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan Kaufmann.

Leermakers, R. (1993). *The Functional Treatment of Parsing.* Kluwer.

Levin, E. (1993). Hidden control neural architecture modeling of nonlinear time varying systems and its applications. *IEEE Transactions on Neural Networks*, 4(1):109–116.

Levinson, S. E. et al. (1979). Interactive clustering techniques for selecting speaker independent reference templates for isolated word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-27:134–141.

Lippmann, R. P. (1989). Review of neural networks for speech recognition. *Neural Computation*, 1(1):1–38.

Liu, Y. D., Sun, G. Z., Chen, H. H., Lee, Y. C., and Giles, C. L. (1990). Grammatical inference and neural network state machines. In *Proceedings of the 1990 International Joint Conference on Neural Networks*, pages 285–288, Washington, DC.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, IT-28:129–136.

Lowerre, B. (1976). *The HARPY speech recognition system.* PhD thesis, Carnegie-Mellon University Dept of Computer Science.

Lubensky, D. M., Asadi, A. O., and Naik, J. M. (1994). Connected digit recognition using connectionist probability estimators and mixture-gaussian densities. In *Proceedings of the 1994 International Conference on Spoken Language Processing*, volume 1, pages 295–298, Yokohama, Japan.

Lucchesi, C. L. and Kowaltowski, T. (1993). Applications of finite automata representing large vocabularies. *Software-Practice and Experience*, 23(1):15–30.

Makhoul, J., Roucos, S., and Gish, H. (1985). Vector quantization in speech coding. *Proceedings of the IEEE*, 73(11):1551–1588.

Manning, C. D. (1993). Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 235–242, Ohio State University. Association for Computational Linguistics.

Maragos, P., Kaiser, J., and Quatieri, T. (1992). On separating amplitude from frequency modulations using energy operators. In *Proceedings of the 1992*

International Conference on Acoustics, Speech, and Signal Processing, volume 2, pages 1–4, San Francisco. Institute of Electrical and Electronic Engineers.

Markov, A. A. (1913). An example of statistical investigation in the text of 'Eugene Onyegin' illustrating coupling of 'tests' in chains. *Proceedings of the Academy of Science, St. Petersburg*, 7:153–162.

Maxwell, John T., I. and Kaplan, R. M. (1993). The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590.

Meisel, W. S. (1972). *Computer Oriented Approaches to Pattern Recognition.* Academic Press, New York.

Minsky, M. and Papert, S. (1969). *Perceptrons.* MIT Press, Cambridge, Massachusetts.

Morgan, D. P. and Scofield, C. L. (1991). *Neural Networks and Speech Processing.* Kluwer Academic.

Naik, J. M. and Lubensky, D. M. (1994). A hybrid HMM-MLP speaker verification algorithm for telephonespeech. In *Proceedings of the 1994 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 153–156, Adelaide, Australia. Institute of Electrical and Electronic Engineers.

Nedderhoff, M.-J. (1994). An optimal tabular parsing algorithm. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico. Association for Computational Linguistics.

Ney, H. (1984). The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 32:263–271.

Nilsson, N. J. (1971). *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, New York.

Oja, E. (1991). Data compression, feature extraction, and auto-association in feedforward neural networks. In *Artificial Neural Networks*, pages 737–745. Elsevier Science, Amsterdam.

Patrick, E. A. (1972). *Fundamentals of Pattern Recognition.* Prentice-Hall, Englewood Cliffs, New Jersey.

Paul, D. B. (1992). An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model. In *Proceedings of the 1992 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 25–28, San Francisco. Institute of Electrical and Electronic Engineers.

Pavlidis, T. and Mori, S. (1992). Special issue on optical character recognition. *Proceedings of the IEEE*, 80(7).

Pentus, M. (1993). Lambek grammars are context-free. In *Proceedings of the Eighth Annual IEEE Symposium on Logic and Computation Science*, pages 429–433, Montreal, Canada. IEEE Computer Society Press.

Pereira, F., Rebecca, C. N., and Wright, N. (1991). Finite state approximation of phrase structure grammars. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, California. Association for Computational Linguistics.

Pereira, F., Riley, M., and Sproat, R. (1994). Weighted rational transductions and their applications to human language processing. In *Proceedings of the Human Language Technology Workshop*, Plainsboro, New Jersey. Morgan Kaufmann.

Pieraccini, R., Levin, E., and Vidal, E. (1993). Learning how to understanding language. In *Eurospeech '93, Proceedings of the Third European Conference on Speech Communication and Technology*, volume 2, pages 1407–1414, Berlin. European Speech Communication Association. Keynote address.

Poritz, A. B. (1988). Hidden Markov models: a guided tour. In *Proceedings of the 1988 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 7–13, New York. Institute of Electrical and Electronic Engineers.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1988). *Numerical Recipes in C: the art of scientific computing.* Cambridge University Press.

Pustejovsky, J. (1992). The acquisition of lexical semantic knowledge from large corpora. In *Proceedings of the Fifth DARPA Speech and Natural Language Workshop.* Morgan Kaufmann.

Rabiner, L. R. and Juang, B.-H. (1993). *Fundamentals of Speech Recognition.* Prentice-Hall, Englewood Cliffs, New Jersey.

Rabiner, L. R. and Schafer, R. W. (1978). *Digital Processing of Speech Signals.* Signal Processing. Prentice-Hall, Englewood Cliffs, New Jersey.

Renals, S., Morgan, N., Bourlard, H., Cohen, M., and Franco, H. (1994). Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing*, 12(1):161–174.

Robinson, T. and Fallside, F. (1991). A recurrent error propagation network speech recognition system. *Computer Speech and Language*, 5:259–274.

Roche, E. (1994). Two parsing algorithms by means of finite-state transducers. In *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstucture of Cognition*. MIT Press.

Samuelsson, C. (1994). *Fast natural-language parsing using explanation-based learning.* PhD thesis, Royal Institute of Technology, Akademitryck, Edsbruk, Sweden.

Schabes, Y. and Joshi, A. K. (1988). An Earley-type parsing algorithm for tree-adjoining grammars. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, SUNY, Buffalo, New York. Association for Computational Linguistics.

Schabes, Y. and Vijayshanker, K. (1990). Deterministic left to right parsing of tree-adjoining languages. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Pittsburgh, Pennsylvania. Association for Computational Linguistics.

Schenkel, M., Guyon, I., and Henderson, D. (1994). On-line cursive script recognition using neural networks and hidden Markov models. In *Proceedings of the 1994 International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 637–640, Adelaide, Australia. Institute of Electrical and Electronic Engineers.

Schenkel, M., Guyon, I., and Henderson, D. (1995). On-line cursive script recognition using time delay neural networks and hidden Markov models. *Machine Vision and Applications*. Special issue on Cursive Script Recognition.

Schwartz, R. and Austin, S. (1991). A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses. In *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 701–704, Toronto. Institute of Electrical and Electronic Engineers.

Shieber, S. M. (1988). *An Introduction to Unification-based Approaches to Grammar.* CSLI, Stanford.

Shieber, S. M., Schabes, Y., and Pereira, F. C. N. (1994). Principles and implementation of deductive parsing. *Journal of Logic Programming*. In press.

Sikkel, K. (1994). *Parsing Schemata*. Springer-Verlag.

Steeneken, J. and Van Leeuwen, D. (1995). Multi-lingual assessment of speaker independent large vocabulary speech-recognition systems: the SQALE project (speech recognition quality assessment for language engineering). In *Eurospeech '95, Proceedings of the Fourth European Conference on Speech Communication and Technology*, Madrid, Spain.

Sun, G. Z., Chen, H. H., Giles, C. L., Lee, Y. C., and Chen, D. (1990). Connectionist pushdown automata that learn context-free grammars. In *Proceedings of the 1990 International Joint Conference on Neural Networks*, pages 577–580, Washington, DC.

Tappert, C. C., Suen, C. Y., and Wakahara, T. (1990). The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808.

Tomita, M. (1987). An efficient augmented context-free parsing algorithm. *Computational Linguistics*, 13(1):31–46.

Tomita, M., editor (1991). *Current Issues in Parsing Technology.* Kluwer Academic Press, Dordrecht.

Touretzky, D. S., editor (1992). *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann.

Vijay-Shanker, K. and Weir, D. J. (1993). Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

Vintsyuk, T. K. (1971). Element-wise recognition of continuous speech consisting of words from a specified vocabulary. *Cybernetics (Kibernetica)*, pages 133–143.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269.

Voutilainen, A. and Tapanainen, P. (1993). Ambiguity resolution in a reductionist parser. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, pages 394–403, Utrecht University, The Netherlands. European Chapter of the Association for Computational Linguistics.

Waibel, A., Jain, A., McNair, A., Tebelskis, J., Osterholtz, L., Saito, H., Schmidbauer, O., Sloboda, T., and Woszczyna, M. (1992). JANUS: Speech-to-speech translation using connectionist and non-connectionist techniques. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 4*, pages 183–190. Morgan Kaufmann.

Weigend, A. S. and Gershenfeld, N. A., editors (1994). *Time Series Prediction: Forecasting the Future and Understanding the Past.* Santa Fe. Addison-Wesley.

Wilfong, G. (1995). On-line recognition of handwritten symbols. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* In press.

Wilpon, J. G. (1994). Applications of voice processing technology in telecommunications. In Roe, D. B. and Wilpon, J. G., editors, *Voice Communication between Humans and Machines*, pages 280–310. National Academy Press, Washington, DC.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.