example, `/en/` for the English page and `/fr/` for the corresponding French page. The leading statistical translation systems train on hundreds of millions of words of parallel text and billions of words of monolingual text.

2. **Segment into sentences**: The unit of translation is a sentence, so we will have to break the corpus into sentences. Periods are strong indicators of the end of a sentence, but consider "Dr. J. R. Smith of Rodeo Dr. paid $29.99 on 9.9.09."; only the final period ends a sentence. One way to decide if a period ends a sentence is to train a model that takes as features the surrounding words and their parts of speech. This approach achieves about 98% accuracy.
3. **Align sentences**: For each sentence in the English version, determine what sentence(s) it corresponds to in the French version. Usually, the next sentence of English corresponds to the next sentence of French in a 1:1 match, but sometimes there is variation: one sentence in one language will be split into a 2:1 match, or the order of two sentences will be swapped, resulting in a 2:2 match. By looking at the sentence lengths alone (i.e. short sentences should align with short sentences), it is possible to align them (1:1, 1:2, or 2:2, etc.) with accuracy in the 90% to 99% range using a variation on the Viterbi algorithm. Even better alignment can be achieved by using landmarks that are common to both languages, such as numbers, dates, proper names, or words that we know from a bilingual dictionary have an unambiguous translation. For example, if the 3rd English and 4th French sentences contain the string "1989" and neighboring sentences do not, that is good evidence that the sentences should be aligned together.
4. **Align phrases**: Within a sentence, phrases can be aligned by a process that is similar to that used for sentence alignment, but requiring iterative improvement. When we start, we have no way of knowing that "qui dort" aligns with "sleeping," but we can arrive at that alignment by a process of aggregation of evidence. Over all the example sentences we have seen, we notice that "qui dort" and "sleeping" co-occur with high frequency, and that in the pair of aligned sentences, no phrase other than "qui dort" co-occurs so frequently in other sentences with "sleeping." A complete phrase alignment over our corpus gives us the phrasal probabilities (after appropriate smoothing).
5. **Extract distortions**: Once we have an alignment of phrases we can define distortion probabilities. Simply count how often distortion occurs in the corpus for each distance $d = 0, \pm 1, \pm 2, \ldots$, and apply smoothing.
6. **Improve estimates with EM**: Use expectation–maximization to improve the estimates of $P(f \mid e)$ and $P(d)$ values. We compute the best alignments with the current values of these parameters in the E step, then update the estimates in the M step and iterate the process until convergence.

23.5 Speech Recognition

SPEECH RECOGNITION

Speech recognition is the task of identifying a sequence of words uttered by a speaker, given the acoustic signal. It has become one of the mainstream applications of AI—millions of

people interact with speech recognition systems every day to navigate voice mail systems, search the Web from mobile phones, and other applications. Speech is an attractive option when hands-free operation is necessary, as when operating machinery.

Speech recognition is difficult because the sounds made by a speaker are ambiguous and, well, noisy. As a well-known example, the phrase "recognize speech" sounds almost the same as "wreck a nice beach" when spoken quickly. Even this short example shows several of the issues that make speech problematic. First, **segmentation**: written words in English have spaces between them, but in fast speech there are no pauses in "wreck a nice" that would distinguish it as a multiword phrase as opposed to the single word "recognize." Second, **coarticulation**: when speaking quickly the "s" sound at the end of "nice" merges with the "b" sound at the beginning of "beach," yielding something that is close to a "sp." Another problem that does not show up in this example is **homophones**—words like "to," "too," and "two" that sound the same but differ in meaning.

SEGMENTATION

COARTICULATION

HOMOPHONES

We can view speech recognition as a problem in most-likely-sequence explanation. As we saw in Section 15.2, this is the problem of computing the most likely sequence of state variables, $\mathbf{x}_{1:t}$, given a sequence of observations $\mathbf{e}_{1:t}$. In this case the state variables are the words, and the observations are sounds. More precisely, an observation is a vector of features extracted from the audio signal. As usual, the most likely sequence can be computed with the help of Bayes' rule to be:

$$\operatorname*{argmax}_{word_{1:t}} P(word_{1:t} \mid sound_{1:t}) = \operatorname*{argmax}_{word_{1:t}} P(sound_{1:t} \mid word_{1:t}) P(word_{1:t}) \ .$$

Here $P(sound_{1:t}|word_{1:t})$ is the **acoustic model**. It describes the sounds of words—that "ceiling" begins with a soft "c" and sounds the same as "sealing." $P(word_{1:t})$ is known as the **language model**. It specifies the prior probability of each utterance—for example, that "ceiling fan" is about 500 times more likely as a word sequence than "sealing fan."

ACOUSTIC MODEL

LANGUAGE MODEL

This approach was named the **noisy channel model** by Claude Shannon (1948). He described a situation in which an original message (the *words* in our example) is transmitted over a noisy channel (such as a telephone line) such that a corrupted message (the *sounds* in our example) are received at the other end. Shannon showed that no matter how noisy the channel, it is possible to recover the original message with arbitrarily small error, if we encode the original message in a redundant enough way. The noisy channel approach has been applied to speech recognition, machine translation, spelling correction, and other tasks.

NOISY CHANNEL MODEL

Once we define the acoustic and language models, we can solve for the most likely sequence of words using the Viterbi algorithm (Section 15.2.3 on page 576). Most speech recognition systems use a language model that makes the Markov assumption—that the current state $Word_t$ depends only on a fixed number n of previous states—and represent $Word_t$ as a single random variable taking on a finite set of values, which makes it a Hidden Markov Model (HMM). Thus, speech recognition becomes a simple application of the HMM methodology, as described in Section 15.3—simple that is, once we define the acoustic and language models. We cover them next.

Vowels		Consonants B–N		Consonants P–Z	
Phone	Example	Phone	Example	Phone	Example
[iy]	b**ea**t	[b]	**b**et	[p]	**p**et
[ih]	b**i**t	[ch]	**Ch**et	[r]	**r**at
[eh]	b**e**t	[d]	**d**ebt	[s]	**s**et
[æ]	b**a**t	[f]	**f**at	[sh]	**sh**oe
[ah]	b**u**t	[g]	**g**et	[t]	**t**en
[ao]	b**ough**t	[hh]	**h**at	[th]	**th**ick
[ow]	b**oa**t	[hv]	**h**igh	[dh]	**th**at
[uh]	b**oo**k	[jh]	**j**et	[dx]	bu**tt**er
[ey]	b**ai**t	[k]	**k**ick	[v]	**v**et
[er]	B**er**t	[l]	**l**et	[w]	**w**et
[ay]	b**uy**	[el]	bott**le**	[wh]	**wh**ich
[oy]	b**oy**	[m]	**m**et	[y]	**y**et
[axr]	din**er**	[em]	bott**om**	[z]	**z**oo
[aw]	d**ow**n	[n]	**n**et	[zh]	mea**s**ure
[ax]	**a**bout	[en]	butt**on**		
[ix]	ros**e**s	[ng]	si**ng**		
[aa]	c**o**t	[eng]	wash**ing**	[-]	*silence*

Figure 23.14 The ARPA phonetic alphabet, or **ARPAbet**, listing all the phones used in American English. There are several alternative notations, including an International Phonetic Alphabet (IPA), which contains the phones in all known languages.

23.5.1 Acoustic model

Sound waves are periodic changes in pressure that propagate through the air. When these waves strike the diaphragm of a microphone, the back-and-forth movement generates an electric current. An analog-to-digital converter measures the size of the current—which approximates the amplitude of the sound wave—at discrete intervals called the **sampling rate**. Speech sounds, which are mostly in the range of 100 Hz (100 cycles per second) to 1000 Hz, are typically sampled at a rate of 8 kHz. (CDs and mp3 files are sampled at 44.1 kHz.) The precision of each measurement is determined by the **quantization factor**; speech recognizers typically keep 8 to 12 bits. That means that a low-end system, sampling at 8 kHz with 8-bit quantization, would require nearly half a megabyte per minute of speech.

SAMPLING RATE

QUANTIZATION FACTOR

Since we only want to know what words were spoken, not exactly what they sounded like, we don't need to keep all that information. We only need to distinguish between different speech sounds. Linguists have identified about 100 speech sounds, or **phones**, that can be composed to form all the words in all known human languages. Roughly speaking, a phone is the sound that corresponds to a single vowel or consonant, but there are some complications: combinations of letters, such as "th" and "ng" produce single phones, and some letters produce different phones in different contexts (e.g., the "a" in *rat* and *rate*. Figure 23.14 lists

PHONE

PHONEME

all the phones that are used in English, with an example of each. A **phoneme** is the smallest unit of sound that has a distinct meaning to speakers of a particular language. For example, the "t" in "stick" sounds similar enough to the "t" in "tick" that speakers of English consider them the same phoneme. But the difference is significant in the Thai language, so there they are two phonemes. To represent spoken English we want a representation that can distinguish between different phonemes, but one that need not distinguish the nonphonemic variations in sound: loud or soft, fast or slow, male or female voice, etc.

First, we observe that although the sound frequencies in speech may be several kHz, the *changes* in the content of the signal occur much less often, perhaps at no more than 100 Hz. Therefore, speech systems summarize the properties of the signal over time slices called **frames**. A frame length of about 10 milliseconds (i.e., 80 samples at 8 kHz) is short enough to ensure that few short-duration phenomena will be missed. Overlapping frames are used to make sure that we don't miss a signal because it happens to fall on a frame boundary.

FRAME

FEATURE

Each frame is summarized by a vector of **features**. Picking out features from a speech signal is like listening to an orchestra and saying "here the French horns are playing loudly and the violins are playing softly." We'll give a brief overview of the features in a typical system. First, a Fourier transform is used to determine the amount of acoustic energy at about a dozen frequencies. Then we compute a measure called the **mel frequency cepstral coefficient (MFCC)** or MFCC for each frequency. We also compute the total energy in the frame. That gives thirteen features; for each one we compute the difference between this frame and the previous frame, and the difference between differences, for a total of 39 features. These are continuous-valued; the easiest way to fit them into the HMM framework is to discretize the values. (It is also possible to extend the HMM model to handle continuous mixtures of Gaussians.) Figure 23.15 shows the sequence of transformations from the raw sound to a sequence of frames with discrete features.

MEL FREQUENCY CEPSTRAL COEFFICIENT (MFCC)

We have seen how to go from the raw acoustic signal to a series of observations, $\mathbf{e}_t$. Now we have to describe the (unobservable) states of the HMM and define the transition model, $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$, and the sensor model, $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$. The transition model can be broken into two levels: word and phone. We'll start from the bottom: the **phone model** describes

PHONE MODEL

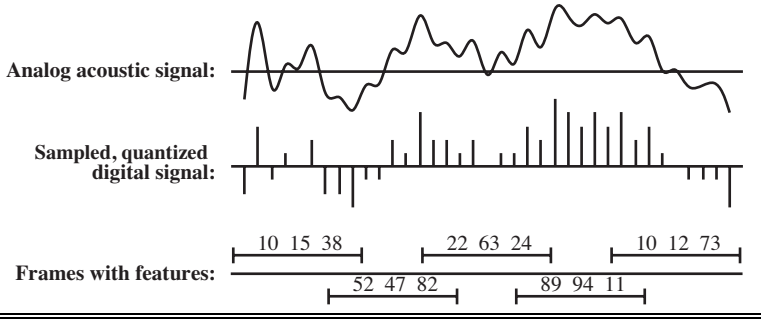
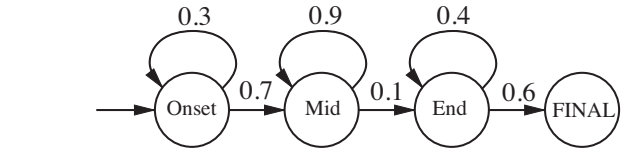


Figure 23.15 Translating the acoustic signal into a sequence of frames. In this diagram each frame is described by the discretized values of three acoustic features; a real system would have dozens of features.

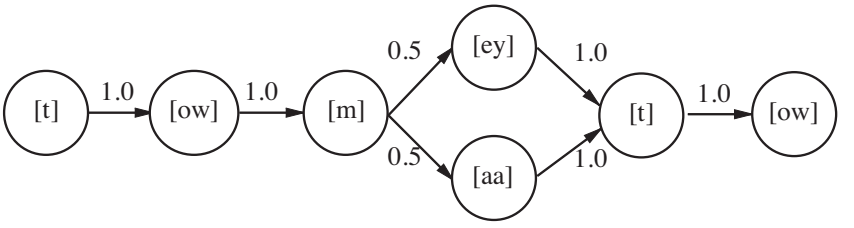


Output probabilities for the phone HMM:

Onset:	Mid:	End:
C_1: 0.5	C_3: 0.2	C_4: 0.1
C_2: 0.2	C_4: 0.7	C_6: 0.5
C_3: 0.3	C_5: 0.1	C_7: 0.4

Figure 23.16 An HMM for the three-state phone [m]. Each state has several possible outputs, each with its own probability. The MFCC feature labels C_1 through C_7 are arbitrary, standing for some combination of feature values.

(a) Word model with dialect variation:



(b) Word model with coarticulation and dialect variations

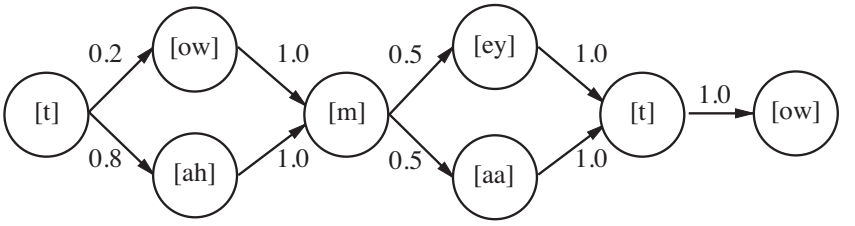


Figure 23.17 Two pronunciation models of the word "tomato." Each model is shown as a transition diagram with states as circles and arrows showing allowed transitions with their associated probabilities. (a) A model allowing for dialect differences. The 0.5 numbers are estimates based on the two authors' preferred pronunciations. (b) A model with a coarticulation effect on the first vowel, allowing either the [ow] or the [ah] phone.

a phone as three states, the onset, middle, and end. For example, the [t] phone has a silent beginning, a small explosive burst of sound in the middle, and (usually) a hissing at the end. Figure 23.16 shows an example for the phone [m]. Note that in normal speech, an average phone has a duration of 50–100 milliseconds, or 5–10 frames. The self-loops in each state allows for variation in this duration. By taking many self-loops (especially in the mid state), we can represent a long "mmmmmmmmmmm" sound. Bypassing the self-loops yields a short "m" sound.

PRONUNCIATION MODEL

In Figure 23.17 the phone models are strung together to form a **pronunciation model** for a word. According to Gershwin (1937), you say [t ow m ey t ow] and I say [t ow m aa t ow]. Figure 23.17(a) shows a transition model that provides for this dialect variation. Each of the circles in this diagram represents a phone model like the one in Figure 23.16.

In addition to dialect variation, words can have **coarticulation** variation. For example, the [t] phone is produced with the tongue at the top of the mouth, whereas the [ow] has the tongue near the bottom. When speaking quickly, the tongue doesn't have time to get into position for the [ow], and we end up with [t ah] rather than [t ow]. Figure 23.17(b) gives a model for "tomato" that takes this coarticulation effect into account. More sophisticated phone models take into account the context of the surrounding phones.

There can be substantial variation in pronunciation for a word. The most common pronunciation of "because" is [b iy k ah z], but that only accounts for about a quarter of uses. Another quarter (approximately) substitutes [ix], [ih] or [ax] for the first vowel, and the remainder substitute [ax] or [aa] for the second vowel, [zh] or [s] for the final [z], or drop "be" entirely, leaving "cuz."

23.5.2 Language model

For general-purpose speech recognition, the language model can be an n-gram model of text learned from a corpus of written sentences. However, spoken language has different characteristics than written language, so it is better to get a corpus of transcripts of spoken language. For task-specific speech recognition, the corpus should be task-specific: to build your airline reservation system, get transcripts of prior calls. It also helps to have task-specific vocabulary, such as a list of all the airports and cities served, and all the flight numbers.

Part of the design of a voice user interface is to coerce the user into saying things from a limited set of options, so that the speech recognizer will have a tighter probability distribution to deal with. For example, asking "What city do you want to go to?" elicits a response with a highly constrained language model, while asking "How can I help you?" does not.

23.5.3 Building a speech recognizer

The quality of a speech recognition system depends on the quality of all of its components—the language model, the word-pronunciation models, the phone models, and the signal-processing algorithms used to extract spectral features from the acoustic signal. We have discussed how the language model can be constructed from a corpus of written text, and we leave the details of signal processing to other textbooks. We are left with the pronunciation and phone models. The *structure* of the pronunciation models—such as the tomato models in

Figure 23.17—is usually developed by hand. Large pronunciation dictionaries are now available for English and other languages, although their accuracy varies greatly. The structure of the three-state phone models is the same for all phones, as shown in Figure 23.16. That leaves the probabilities themselves.

As usual, we will acquire the probabilities from a corpus, this time a corpus of speech. The most common type of corpus to obtain is one that includes the speech signal for each sentence paired with a transcript of the words. Building a model from this corpus is more difficult than building an n-gram model of text, because we have to build a hidden Markov model—the phone sequence for each word and the phone state for each time frame are hidden variables. In the early days of speech recognition, the hidden variables were provided by laborious hand-labeling of spectrograms. Recent systems use expectation–maximization to automatically supply the missing data. The idea is simple: given an HMM and an observation sequence, we can use the smoothing algorithms from Sections 15.2 and 15.3 to compute the probability of each state at each time step and, by a simple extension, the probability of each state–state pair at consecutive time steps. These probabilities can be viewed as *uncertain labels*. From the uncertain labels, we can estimate new transition and sensor probabilities, and the EM procedure repeats. The method is guaranteed to increase the fit between model and data on each iteration, and it generally converges to a much better set of parameter values than those provided by the initial, hand-labeled estimates.

The systems with the highest accuracy work by training a different model for each speaker, thereby capturing differences in dialect as well as male/female and other variations. This training can require several hours of interaction with the speaker, so the systems with the most widespread adoption do not create speaker-specific models.

The accuracy of a system depends on a number of factors. First, the quality of the signal matters: a high-quality directional microphone aimed at a stationary mouth in a padded room will do much better than a cheap microphone transmitting a signal over phone lines from a car in traffic with the radio playing. The vocabulary size matters: when recognizing digit strings with a vocabulary of 11 words (1-9 plus "oh" and "zero"), the word error rate will be below 0.5%, whereas it rises to about 10% on news stories with a 20,000-word vocabulary, and 20% on a corpus with a 64,000-word vocabulary. The task matters too: when the system is trying to accomplish a specific task—book a flight or give directions to a restaurant—the task can often be accomplished perfectly even with a word error rate of 10% or more.

23.6 SUMMARY

Natural language understanding is one of the most important subfields of AI. Unlike most other areas of AI, natural language understanding requires an empirical investigation of actual human behavior—which turns out to be complex and interesting.

- Formal language theory and **phrase structure** grammars (and in particular, **context-free** grammar) are useful tools for dealing with some aspects of natural language. The probabilistic context-free grammar (PCFG) formalism is widely used.

- Sentences in a context-free language can be parsed in $O(n^3)$ time by a **chart parser** such as the **CYK algorithm**, which requires grammar rules to be in **Chomsky Normal Form**.
- A **treebank** can be used to learn a grammar. It is also possible to learn a grammar from an unparsed corpus of sentences, but this is less successful.
- A **lexicalized PCFG** allows us to represent that some relationships between words are more common than others.
- It is convenient to **augment** a grammar to handle such problems as subject–verb agreement and pronoun case. **Definite clause grammar** (DCG) is a formalism that allows for augmentations. With DCG, parsing and semantic interpretation (and even generation) can be done using logical inference.
- **Semantic interpretation** can also be handled by an augmented grammar.
- **Ambiguity** is a very important problem in natural language understanding; most sentences have many possible interpretations, but usually only one is appropriate. Disambiguation relies on knowledge about the world, about the current situation, and about language use.
- **Machine translation** systems have been implemented using a range of techniques, from full syntactic and semantic analysis to statistical techniques based on phrase frequencies. Currently the statistical models are most popular and most successful.
- **Speech recognition** systems are also primarily based on statistical principles. Speech systems are popular and useful, albeit imperfect.
- Together, machine translation and speech recognition are two of the big successes of natural language technology. One reason that the models perform well is that large corpora are available—both translation and speech are tasks that are performed "in the wild" by people every day. In contrast, tasks like parsing sentences have been less successful, in part because no large corpora of parsed sentences are available "in the wild" and in part because parsing is not useful in and of itself.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

Like semantic networks, context-free grammars (also known as phrase structure grammars) are a reinvention of a technique first used by ancient Indian grammarians (especially Panini, ca. 350 B.C.) studying Shastric Sanskrit (Ingerman, 1967). They were reinvented by Noam Chomsky (1956) for the analysis of English syntax and independently by John Backus for the analysis of Algol-58 syntax. Peter Naur extended Backus's notation and is now credited (Backus, 1996) with the "N" in BNF, which originally stood for "Backus **N**ormal Form."

ATTRIBUTE GRAMMAR

Knuth (1968) defined a kind of augmented grammar called **attribute grammar** that is useful for programming languages. Definite clause grammars were introduced by Colmerauer (1975) and developed and popularized by Pereira and Shieber (1987).

Probabilistic context-free grammars were investigated by Booth (1969) and Salomaa (1969). Other algorithms for PCFGs are presented in the excellent short monograph by

Charniak (1993) and the excellent long textbooks by Manning and Schütze (1999) and Jurafsky and Martin (2008). Baker (1979) introduces the inside–outside algorithm for learning a PCFG, and Lari and Young (1990) describe its uses and limitations. Stolcke and Omohundro (1994) show how to learn grammar rules with Bayesian model merging; Haghighi and Klein (2006) describe a learning system based on prototypes.

Lexicalized PCFGs (Charniak, 1997; Hwa, 1998) combine the best aspects of PCFGs and n-gram models. Collins (1999) describes PCFG parsing that is lexicalized with head features. Petrov and Klein (2007a) show how to get the advantages of lexicalization without actual lexical augmentations by learning specific syntactic categories from a treebank that has general categories; for example, the treebank has the category NP, from which more specific categories such as NP_O and NP_S can be learned.

There have been many attempts to write formal grammars of natural languages, both in "pure" linguistics and in computational linguistics. There are several comprehensive but informal grammars of English (Quirk *et al.*, 1985; McCawley, 1988; Huddleston and Pullum, 2002). Since the mid-1980s, there has been a trend toward putting more information in the lexicon and less in the grammar. Lexical-functional grammar, or LFG (Bresnan, 1982) was the first major grammar formalism to be highly lexicalized. If we carry lexicalization to an extreme, we end up with **categorial grammar** (Clark and Curran, 2004), in which there can be as few as two grammar rules, or with **dependency grammar** (Smith and Eisner, 2008; Kübler *et al.*, 2009) in which there are no syntactic categories, only relations between words. Sleator and Temperley (1993) describe a dependency parser. Paskin (2001) shows that a version of dependency grammar is easier to learn than PCFGs.

The first computerized parsing algorithms were demonstrated by Yngve (1955). Efficient algorithms were developed in the late 1960s, with a few twists since then (Kasami, 1965; Younger, 1967; Earley, 1970; Graham *et al.*, 1980). Maxwell and Kaplan (1993) show how chart parsing with augmentations can be made efficient in the average case. Church and Patil (1982) address the resolution of syntactic ambiguity. Klein and Manning (2003) describe A* parsing, and Pauls and Klein (2009) extend that to *K*-best A* parsing, in which the result is not a single parse but the *K* best.

Leading parsers today include those by Petrov and Klein (2007b), which achieved 90.6% accuracy on the Wall Street Journal corpus, Charniak and Johnson (2005), which achieved 92.0%, and Koo *et al.* (2008), which achieved 93.2% on the Penn treebank. These numbers are not directly comparable, and there is some criticism of the field that it is focusing too narrowly on a few select corpora, and perhaps overfitting on them.

Formal semantic interpretation of natural languages originates within philosophy and formal logic, particularly Alfred Tarski's (1935) work on the semantics of formal languages. Bar-Hillel (1954) was the first to consider the problems of pragmatics and propose that they could be handled by formal logic. For example, he introduced C. S. Peirce's (1902) term *indexical* into linguistics. Richard Montague's essay "English as a formal language" (1970) is a kind of manifesto for the logical analysis of language, but the books by Dowty *et al.* (1991) and Portner and Partee (2002) are more readable.

The first NLP system to solve an actual task was probably the BASEBALL question answering system (Green *et al.*, 1961), which handled questions about a database of baseball

statistics. Close after that was Woods's (1973) LUNAR, which answered questions about the rocks brought back from the moon by the Apollo program. Roger Schank and his students built a series of programs (Schank and Abelson, 1977; Schank and Riesbeck, 1981) that all had the task of understanding language. Modern approaches to semantic interpretation usually assume that the mapping from syntax to semantics will be learned from examples (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005).

Hobbs *et al.* (1993) describes a quantitative nonprobabilistic framework for interpretation. More recent work follows an explicitly probabilistic framework (Charniak and Goldman, 1992; Wu, 1993; Franz, 1996). In linguistics, optimality theory (Kager, 1999) is based on the idea of building soft constraints into the grammar, giving a natural ranking to interpretations (similar to a probability distribution), rather than having the grammar generate all possibilities with equal rank. Norvig (1988) discusses the problems of considering multiple simultaneous interpretations, rather than settling for a single maximum-likelihood interpretation. Literary critics (Empson, 1953; Hobbs, 1990) have been ambiguous about whether ambiguity is something to be resolved or cherished.

Nunberg (1979) outlines a formal model of metonymy. Lakoff and Johnson (1980) give an engaging analysis and catalog of common metaphors in English. Martin (1990) and Gibbs (2006) offer computational models of metaphor interpretation.

The first important result on **grammar induction** was a negative one: Gold (1967) showed that it is not possible to reliably learn a correct context-free grammar, given a set of strings from that grammar. Prominent linguists, such as Chomsky (1957) and Pinker (2003), have used Gold's result to argue that there must be an innate **universal grammar** that all children have from birth. The so-called *Poverty of the Stimulus* argument says that children aren't given enough input to learn a CFG, so they must already "know" the grammar and be merely tuning some of its parameters. While this argument continues to hold sway throughout much of Chomskyan linguistics, it has been dismissed by some other linguists (Pullum, 1996; Elman *et al.*, 1997) and most computer scientists. As early as 1969, Horning showed that it *is* possible to learn, in the sense of PAC learning, a *probabilistic* context-free grammar. Since then, there have been many convincing empirical demonstrations of learning from positive examples alone, such as the ILP work of Mooney (1999) and Muggleton and De Raedt (1994), the sequence learning of Nevill-Manning and Witten (1997), and the remarkable Ph.D. theses of Schütze (1995) and de Marcken (1996). There is an annual International Conference on Grammatical Inference (ICGI). It is possible to learn other grammar formalisms, such as regular languages (Denis, 2001) and finite state automata (Parekh and Honavar, 2001). Abney (2007) is a textbook introduction to semi-supervised learning for language models.

UNIVERSAL GRAMMAR

Wordnet (Fellbaum, 2001) is a publicly available dictionary of about 100,000 words and phrases, categorized into parts of speech and linked by semantic relations such as synonym, antonym, and part-of. The Penn Treebank (Marcus *et al.*, 1993) provides parse trees for a 3-million-word corpus of English. Charniak (1996) and Klein and Manning (2001) discuss parsing with treebank grammars. The British National Corpus (Leech *et al.*, 2001) contains 100 million words, and the World Wide Web contains several trillion words; (Brants *et al.*, 2007) describe n-gram models over a 2-trillion-word Web corpus.

In the 1930s Petr Troyanskii applied for a patent for a "translating machine," but there were no computers available to implement his ideas. In March 1947, the Rockefeller Foundation's Warren Weaver wrote to Norbert Wiener, suggesting that machine translation might be possible. Drawing on work in cryptography and information theory, Weaver wrote, "When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in strange symbols. I will now proceed to decode.'" For the next decade, the community tried to decode in this way. IBM exhibited a rudimentary system in 1954. Bar-Hillel (1960) describes the enthusiasm of this period. However, the U.S. government subsequently reported (ALPAC, 1966) that "there is no immediate or predictable prospect of useful machine translation." However, limited work continued, and starting in the 1980s, computer power had increased to the point where the ALPAC findings were no longer correct.

The basic statistical approach we describe in the chapter is based on early work by the IBM group (Brown *et al.*, 1988, 1993) and the recent work by the ISI and Google research groups (Och and Ney, 2004; Zollmann *et al.*, 2008). A textbook introduction on statistical machine translation is given by Koehn (2009), and a short tutorial by Kevin Knight (1999) has been influential. Early work on sentence segmentation was done by Palmer and Hearst (1994). Och and Ney (2003) and Moore (2005) cover bilingual sentence alignment.

The prehistory of speech recognition began in the 1920s with Radio Rex, a voice-activated toy dog. Rex jumped out of his doghouse in response to the word "Rex!" (or actually almost any sufficiently loud word). Somewhat more serious work began after World War II. At AT&T Bell Labs, a system was built for recognizing isolated digits (Davis *et al.*, 1952) by means of simple pattern matching of acoustic features. Starting in 1971, the Defense Advanced Research Projects Agency (DARPA) of the United States Department of Defense funded four competing five-year projects to develop high-performance speech recognition systems. The winner, and the only system to meet the goal of 90% accuracy with a 1000-word vocabulary, was the HARPY system at CMU (Lowerre and Reddy, 1980). The final version of HARPY was derived from a system called DRAGON built by CMU graduate student James Baker (1975); DRAGON was the first to use HMMs for speech. Almost simultaneously, Jelinek (1976) at IBM had developed another HMM-based system. Recent years have been characterized by steady incremental progress, larger data sets and models, and more rigorous competitions on more realistic speech tasks. In 1997, Bill Gates predicted, "The PC five years from now—you won't recognize it, because speech will come into the interface." That didn't quite happen, but in 2008 he predicted "In five years, Microsoft expects more Internet searches to be done through speech than through typing on a keyboard." History will tell if he is right this time around.

Several good textbooks on speech recognition are available (Rabiner and Juang, 1993; Jelinek, 1997; Gold and Morgan, 2000; Huang *et al.*, 2001). The presentation in this chapter drew on the survey by Kay, Gawron, and Norvig (1994) and on the textbook by Jurafsky and Martin (2008). Speech recognition research is published in *Computer Speech and Language*, *Speech Communications*, and the IEEE *Transactions on Acoustics, Speech, and Signal Processing* and at the DARPA Workshops on Speech and Natural Language Processing and the Eurospeech, ICSLP, and ASRU conferences.

Ken Church (2004) shows that natural language research has cycled between concentrating on the data (empiricism) and concentrating on theories (rationalism). The linguist John Firth (1957) proclaimed "You shall know a word by the company it keeps," and linguistics of the 1940s and early 1950s was based largely on word frequencies, although without the computational power we have available today. Then Noam (Chomsky, 1956) showed the limitations of finite-state models, and sparked an interest in theoretical studies of syntax, disregarding frequency counts. This approach dominated for twenty years, until empiricism made a comeback based on the success of work in statistical speech recognition (Jelinek, 1976). Today, most work accepts the statistical framework, but there is great interest in building statistical models that consider higher-level models, such as syntactic trees and semantic relations, not just sequences of words.

Work on applications of language processing is presented at the biennial Applied Natural Language Processing conference (ANLP), the conference on Empirical Methods in Natural Language Processing (EMNLP), and the journal *Natural Language Engineering*. A broad range of NLP work appears in the journal *Computational Linguistics* and its conference, ACL, and in the Computational Linguistics (COLING) conference.

EXERCISES

23.1 Read the following text once for understanding, and remember as much of it as you can. There will be a test later.

> The procedure is actually quite simple. First you arrange things into different groups. Of course, one pile may be sufficient depending on how much there is to do. If you have to go somewhere else due to lack of facilities that is the next step, otherwise you are pretty well set. It is important not to overdo things. That is, it is better to do too few things at once than too many. In the short run this may not seem important but complications can easily arise. A mistake is expensive as well. At first the whole procedure will seem complicated. Soon, however, it will become just another facet of life. It is difficult to foresee any end to the necessity for this task in the immediate future, but then one can never tell. After the procedure is completed one arranges the material into different groups again. Then they can be put into their appropriate places. Eventually they will be used once more and the whole cycle will have to be repeated. However, this is part of life.

23.2 An *HMM grammar* is essentially a standard HMM whose state variable is N (nonterminal, with values such as Det, $Adjective$, $Noun$ and so on) and whose evidence variable is W (word, with values such as is, $duck$, and so on). The HMM model includes a prior $\mathbf{P}(N_0)$, a transition model $\mathbf{P}(N_{t+1}|N_t)$, and a sensor model $\mathbf{P}(W_t|N_t)$. Show that every HMM grammar can be written as a PCFG. [Hint: start by thinking about how the HMM prior can be represented by PCFG rules for the sentence symbol. You may find it helpful to illustrate for the particular HMM with values A, B for N and values x, y for W.]

23.3 Consider the following PCFG for simple verb phrases:

$0.1 : VP \rightarrow Verb$
$0.2 : VP \rightarrow Copula\ Adjective$
$0.5 : VP \rightarrow Verb\ the\ Noun$
$0.2 : VP \rightarrow VP\ Adverb$
$0.5 : Verb \rightarrow is$
$0.5 : Verb \rightarrow shoots$
$0.8 : Copula \rightarrow is$
$0.2 : Copula \rightarrow seems$
$0.5 : Adjective \rightarrow$ **unwell**
$0.5 : Adjective \rightarrow$ **well**
$0.5 : Adverb \rightarrow$ **well**
$0.5 : Adverb \rightarrow$ **badly**
$0.6 : Noun \rightarrow$ **duck**
$0.4 : Noun \rightarrow$ **well**

a. Which of the following have a nonzero probability as a VP? (i) shoots the duck well well well (ii) seems the well well (iii) shoots the unwell well badly

b. What is the probability of generating "is well well"?

c. What types of ambiguity are exhibited by the phrase in (b)?

d. Given any PCFG, is it possible to calculate the probability that the PCFG generates a string of exactly 10 words?

23.4 Outline the major differences between Java (or any other computer language with which you are familiar) and English, commenting on the "understanding" problem in each case. Think about such things as grammar, syntax, semantics, pragmatics, compositionality, context-dependence, lexical ambiguity, syntactic ambiguity, reference finding (including pronouns), background knowledge, and what it means to "understand" in the first place.

23.5 This exercise concerns grammars for very simple languages.

a. Write a context-free grammar for the language $a^n b^n$.

b. Write a context-free grammar for the palindrome language: the set of all strings whose second half is the reverse of the first half.

c. Write a context-sensitive grammar for the duplicate language: the set of all strings whose second half is the same as the first half.

23.6 Consider the sentence "Someone walked slowly to the supermarket" and a lexicon consisting of the following words:

$Pronoun \rightarrow$ **someone**	$Verb \rightarrow$ **walked**
$Adv \rightarrow$ **slowly**	$Prep \rightarrow$ **to**
$Article \rightarrow$ **the**	$Noun \rightarrow$ **supermarket**

Which of the following three grammars, combined with the lexicon, generates the given sentence? Show the corresponding parse tree(s).

(A):	(B):	(C):
$S \rightarrow NP\ \ VP$	$S \rightarrow NP\ \ VP$	$S \rightarrow NP\ \ VP$
$NP \rightarrow Pronoun$	$NP \rightarrow Pronoun$	$NP \rightarrow Pronoun$
$NP \rightarrow Article\ \ Noun$	$NP \rightarrow Noun$	$NP \rightarrow Article\ \ NP$
$VP \rightarrow VP\ \ PP$	$NP \rightarrow Article\ \ NP$	$VP \rightarrow Verb\ \ Adv$
$VP \rightarrow VP\ \ Adv\ \ Adv$	$VP \rightarrow Verb\ \ Vmod$	$Adv \rightarrow Adv\ \ Adv$
$VP \rightarrow Verb$	$Vmod \rightarrow Adv\ \ Vmod$	$Adv \rightarrow PP$
$PP \rightarrow Prep\ \ NP$	$Vmod \rightarrow Adv$	$PP \rightarrow Prep\ \ NP$
$NP \rightarrow Noun$	$Adv \rightarrow PP$	$NP \rightarrow Noun$
	$PP \rightarrow Prep\ \ NP$	

For each of the preceding three grammars, write down three sentences of English and three sentences of non-English generated by the grammar. Each sentence should be significantly different, should be at least six words long, and should include some new lexical entries (which you should define). Suggest ways to improve each grammar to avoid generating the non-English sentences.

23.7 Collect some examples of time expressions, such as "two o'clock," "midnight," and "12:46." Also think up some examples that are ungrammatical, such as "thirteen o'clock" or "half past two fifteen." Write a grammar for the time language.

23.8 In this exercise you will transform $\mathcal{E}_0$ into Chomsky Normal Form (CNF). There are five steps: (a) Add a new start symbol, (b) Eliminate ϵ rules, (c) Eliminate multiple words on right-hand sides, (d) Eliminate rules of the form ($X \rightarrow Y$), (e) Convert long right-hand sides into binary rules.

a. The start symbol, S, can occur only on the left-hand side in CNF. Add a new rule of the form $S' \rightarrow S$, using a new symbol S'.

b. The empty string, ϵ cannot appear on the right-hand side in CNF. $\mathcal{E}_0$ does not have any rules with ϵ, so this is not an issue.

c. A word can appear on the right-hand side in a rule only of the form ($X \rightarrow$ *word*). Replace each rule of the form ($X \rightarrow \ldots$ *word* $\ldots$) with ($X \rightarrow \ldots W' \ldots$) and ($W' \rightarrow$ *word*), using a new symbol W'.

d. A rule ($X \rightarrow Y$) is not allowed in CNF; it must be ($X \rightarrow Y\ Z$) or ($X \rightarrow$ *word*). Replace each rule of the form ($X \rightarrow Y$) with a set of rules of the form ($X \rightarrow \ldots$), one for each rule ($Y \rightarrow \ldots$), where ($\ldots$) indicates one or more symbols.

e. Replace each rule of the form ($X \rightarrow Y\ Z \ldots$) with two rules, ($X \rightarrow Y\ Z'$) and ($Z' \rightarrow Z \ldots$), where Z' is a new symbol.

Show each step of the process and the final set of rules.

23.9 Using DCG notation, write a grammar for a language that is just like $\mathcal{E}_1$, except that it enforces agreement between the subject and verb of a sentence and thus does not generate ungrammatical sentences such as "I smells the wumpus."

23.10 Consider the following PCFG:

$S \rightarrow NP\ VP$ [1.0]
$NP \rightarrow Noun$ [0.6] | $Pronoun$ [0.4]
$VP \rightarrow Verb\ NP$ [0.8] | $Modal\ Verb$ [0.2]

$Noun \rightarrow$ **can** [0.1] | **fish** [0.3] | ...
$Pronoun \rightarrow$ **I** [0.4] | ...
$Verb \rightarrow$ **can** [0.01] | **fish** [0.1] | ...
$Modal \rightarrow$ **can** [0.3] | ...

The sentence "I can fish" has two parse trees with this grammar. Show the two trees, their prior probabilities, and their conditional probabilities, given the sentence.

23.11 An augmented context-free grammar can represent languages that a regular context-free grammar cannot. Show an augmented context-free grammar for the language $a^n b^n c^n$. The allowable values for augmentation variables are 1 and SUCCESSOR(n), where n is a value. The rule for a sentence in this language is

$$S(n) \rightarrow A(n)\ B(n)\ C(n)\ .$$

Show the rule(s) for each of A, B, and C.

23.12 Augment the $\mathcal{E}_1$ grammar so that it handles article–noun agreement. That is, make sure that "agents" and "an agent" are NPs, but "agent" and "an agents" are not.

23.13 Consider the following sentence (from *The New York Times,* July 28, 2008):

> Banks struggling to recover from multibillion-dollar loans on real estate are curtailing loans to American businesses, depriving even healthy companies of money for expansion and hiring.

a. Which of the words in this sentence are lexically ambiguous?
b. Find two cases of syntactic ambiguity in this sentence (there are more than two.)
c. Give an instance of metaphor in this sentence.
d. Can you find semantic ambiguity?

23.14 Without looking back at Exercise 23.1, answer the following questions:

a. What are the four steps that are mentioned?
b. What step is left out?
c. What is "the material" that is mentioned in the text?
d. What kind of mistake would be expensive?
e. Is it better to do too few things or too many? Why?

23.15 Select five sentences and submit them to an online translation service. Translate them from English to another language and back to English. Rate the resulting sentences for grammaticality and preservation of meaning. Repeat the process; does the second round of

iteration give worse results or the same results? Does the choice of intermediate language make a difference to the quality of the results? If you know a foreign language, look at the translation of one paragraph into that language. Count and describe the errors made, and conjecture why these errors were made.

23.16 The D_i values for the sentence in Figure 23.13 sum to 0. Will that be true of every translation pair? Prove it or give a counterexample.

23.17 (Adapted from Knight (1999).) Our translation model assumes that, after the phrase translation model selects phrases and the distortion model permutes them, the language model can unscramble the permutation. This exercise investigates how sensible that assumption is. Try to unscramble these proposed lists of phrases into the correct order:

a. have, programming, a, seen, never, I, language, better

b. loves, john, mary

c. is the, communication, exchange of, intentional, information brought, by, about, the production, perception of, and signs, from, drawn, a, of, system, signs, conventional, shared

d. created, that, we hold these, to be, all men, truths, are, equal, self-evident

Which ones could you do? What type of knowledge did you draw upon? Train a bigram model from a training corpus, and use it to find the highest-probability permutation of some sentences from a test corpus. Report on the accuracy of this model.

23.18 Calculate the most probable path through the HMM in Figure 23.16 for the output sequence $[C_1, C_2, C_3, C_4, C_4, C_6, C_7]$. Also give its probability.

23.19 We forgot to mention that the text in Exercise 23.1 is entitled "Washing Clothes." Reread the text and answer the questions in Exercise 23.14. Did you do better this time? Bransford and Johnson (1973) used this text in a controlled experiment and found that the title helped significantly. What does this tell you about how language and memory works?