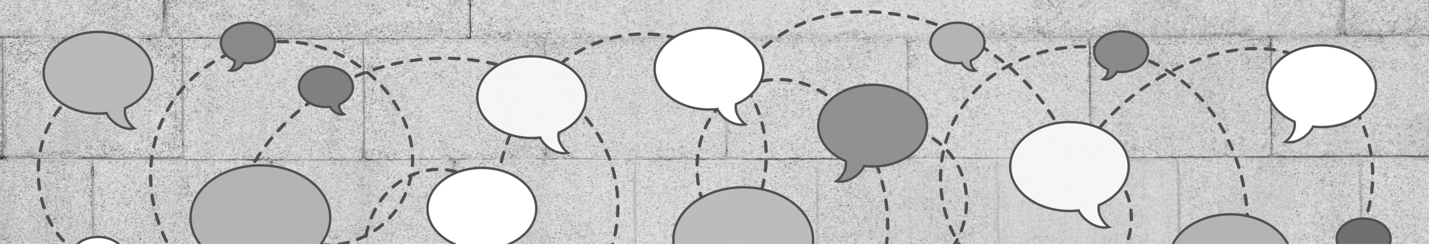

# 17 Computational Linguistics

Robert E. Mercer

**OVERVIEW**

In this chapter, you will be introduced to computational linguistics. Our objectives are to:

- **explore the effects of language ambiguity on computation;**
- **explain how computation is done;**
- **learn about aspects of language that are of interest to computational linguists;** and
- **introduce some applications.**

## 17.1 What Is Natural Language?

Humans communicate using **natural language**. The tokens used, phonemes in speech and words in written text, can be ambiguous. For instance, the word *walk* could be used as a verb or a noun, although their meanings are related. Other words, like *bear,* have senses that are not related at all. And *bear* and *bare* are different words but are homophones, so when heard are not distinguishable without extra information. Refer to Chapter 6 Semantics to read about word meaning and the relationship between words.

Typical computer software works with data that is structured to make it unambiguous. For instance, some numbers describing a person may be ordered so that the first number is their age in years, the second is their height in centimetres, the third, salary, and so on. Because of this ordering and because of the meaning of the numbers given in that ordering, the numbers are unambiguous. The software is designed knowing this structure.

Natural language does not come with knowledge of sentence structure like *the fourth word in a sentence is the verb*. So, if the fourth word is *walk*, deciding whether it is a noun or a verb must depend on other information, and this information may also be ambiguous. Computer software that deals with natural language as input must cope with the inherent ambiguity found in the input.

## 17.2 What Is Computational Linguistics?

**Computational linguistics** brings together aspects of computation (as computer software) and linguistics (as provided by the study of natural language), as the name suggests.

You might ask: "Why do We Want to do this?" One obvious answer is: *It's there to be explored!* It is our ocean depths or our Mars. We are curious. We want to discover things just beyond our current knowledge and abilities. We want to see whether we can provide computers with the capacity to deal with human language just like we do.

But there are other, more pragmatic reasons to bring together computers and language. Firstly, language contains information. Extracted information is more valuable than its non-extracted form, just like resources from the earth: Wood, metals, petroleum, once extracted, can be transformed and used in a variety of ways.

Similarly, people's views about consumer products, hotel accommodation, etc. can be extracted and made useful to the maker of the product or the hotel management. Computational linguistics is used for sentiment analysis. This would include questions such as: Did you enjoy your stay at a certain hotel or was your experience a negative one? Is the smart phone that you just bought easy to set up? Is it performing as advertised?

Secondly, language is used to make human artefacts: essays, letters, email messages, etc. If computers are given knowledge about language, they can become language assistants, providing help to correct spelling mistakes, grammar errors, etc. Microsoft Word uses knowledge about language to indicate spelling errors and bad grammar. Text messaging apps can correct spelling mistakes automatically, sometimes with embarrassing results.

However, extracting this information or being a language assistant is not straightforward. Remember, computers are simply machines that are provided with methods to process information. The methods that are supplied to the computer must be derived from linguistic knowledge.

Many challenges confront computational linguists. The first challenge is to capture all of the knowledge that we have about language and then utilize this knowledge in the computer algorithms that are developed to process language. The second problem is interesting from a computational point of view: Language is inherently ambiguous. We need some examples and an explanation of how this affects computation. These are the topics of the next two sections.

**LINGUISTICS TIDBITS: ASPECTS OF COMPUTATIONAL LINGUISTICS**

Did you know that computational linguistics utilizes the field of linguistics to then add something extra? Look at the following tidbits:

- Much of computational linguistics deals with topics that you have seen in previous chapters: phonology, morphology, syntax, semantics, and pragmatics.
- But, it transcends these common linguistic topics and is also concerned with aspects of language that are more pragmatic such as style, rhetoric, speech acts, and linking language to the world that it refers to.
- It is also concerned with applications of linguistics, such as translation between languages and summarization of texts.

## 17.3 Ambiguity of Language

As we just said, natural language is ambiguous. For example, read (1).

**(1)** Rice flies like sand.

The sentence in (1) has two meanings: a) grains of rice can be propelled in the air just as sand can be; and b) a certain species of fly that eats or lives in rice, hence called the rice fly, enjoys (maybe burrowing in) sand.

These two possible meanings are the result of ambiguity at the word level, which allows ambiguity at the syntax level, which results in the two ambiguous meanings just presented. At the word level, *rice* is a noun in both instances. In the first case *flies* is a verb, *like* is a preposition, and *sand* is a noun. *Rice* then is a noun phrase, *flies* is the head verb of a verb phrase, and *like sand* is a prepositional phrase that is the complement of the verb.

In the second case, *flies* is a noun and it is the head noun of a noun phrase with *rice* being a noun modifier of that head noun, *like* is a verb which takes *sand*, a noun phrase, as its complement. With these two syntactic structures, the semantics of each follows easily.

So, we see that the words, although they have the same surface form, may have different linguistic interpretations. Although there is a straightforward solution – take every interpretation of each word and make all possible combinations of these interpretations – this method soon becomes infeasible, as we will see in the next section.

Let’s look at another in (2).

**(2)** The horse raced past the barn fell.

When you reached the word *fell* in (2), you probably realized that the sentence doesn’t make sense. If you think back to Chapter 14 Psycholinguistics, we call this type of sentence a garden path sentence. Why? You most likely interpreted the word *raced* as a verb (in the sense of *The horse ran past the barn*) because it immediately follows the noun phrase *the horse*. Interpreting *raced* as a past participle gives the correct reading. But this interpretation of *raced* is probably a less likely choice for most readers. Linguistic clues could be added to the sentence to force the correct interpretation: Put the word *that* or the words *that was* between *horse* and *raced*, and possibly a slight pause after *barn* as in (3).

**(3)** The horse that (was) raced past the barn <pause> fell.

By adding the word(s) *that (was)*, the word *raced* is still a verb, but it is not processed as the main verb of the sentence, rather it is the verb of the subordinate clause modifying *horse*. So, the interpretation of the initial part of the sentence with the inclusion of the word *that* is *the horse which ran past the barn* and with the inclusion of the words *that was* is *the horse which was being run past the barn*. As we mentioned earlier, trying all possible interpretations of all words is not a feasible solution. However, endowing the computer with the most likely interpretation as the solution would also lead us down the garden path. We look at how to deal with this problem in the next section.

## 17.4 Computation

If language were not ambiguous, we could reliably prepare computer algorithms that resemble the computer programs which many of us are familiar with: a sequence of actions that produces an outcome (e.g., a sequence of words forming a sentence followed by the computation of the meaning of that sentence). But as we know, language is ambiguous and we must use a different computational paradigm.

One suggestion above was to try all possible combinations of the interpretations of the ambiguous items, but this was immediately dismissed as being infeasible. Let's see why this is the case: Each time some aspect (e.g., word category, grammar rule to use) of a sentence is ambiguous, this multiplies the total number of combinations of possibilities. These combinations grow extremely quickly. Let us consider the five-word sentence in (4).

**(4)** Orange fish cook green eggs.

Each word in (4) is two-way ambiguous with respect to word category. *Orange* can be a noun or an adjective, *fish* can be a noun or a verb, *cook* can be a noun or a verb, *green* can be a noun or an adjective, and *eggs* can be a noun or a verb.

So, we have five two-way ambiguities, which gives us 32 ways to combine the word categories for the sentence.

**PAUSE AND REFLECT 17.1**

Stop for a moment and ask yourself how the number 32 was obtained in the discussion of (4)?

Let's examine further the problem of ambiguity. Suppose now that we have a sentence with 40 words, and that each word is two-way ambiguous with respect to word category. Using a similar argument as above with the five-word sentence, we have 40 two-way ambiguities, which gives us almost 1,100,000,000,000 combinations.

**PAUSE AND REFLECT 17.2**

Using the method that was developed in Pause and Reflect 17.1, watch the number grow as the sentence increases to 40 words. We said that the number obtained with 40 two-way ambiguities was nearly 1,100,000,000,000 but can you figure the exact number? What needs to be changed in the computation, if each word is three-way ambiguous?

This number of possibilities is far too many to try; nevertheless, computational linguists use techniques that have been provided by the field of computer science to deal with this type of problem.

First of all, we need to model our language, that is, how might we capture which combinations make sense and which don't. One way to obtain this model is with human-crafted rules. A rule for word category tagging could be: A word following a determiner is an adjective. Another rule could be: A word following a determiner is a noun. Note that we have two competing tags for a word following a determiner. We also don't have a rule that specifies that a word following a determiner could be a verb, so having only these two rules would definitely rule out this possibility.

How might we use these competing rules? The goal is to choose a set of rules that when used together tag all the words in a sentence with their word categories. When a situation allows more than one rule to be used, a decision needs to be made regarding which rule to use. Going back to the example in (4), we might have two rules: a) A sentence can begin with a noun; and b) A sentence can begin with an adjective. We can choose either rule. Just as an example, let's choose *orange* to be a noun. We need to remember that at this decision point there remains one unused alternative.

We may reach a point where we have no rules to apply but we still have words to tag so we have chosen an unsuccessful set of rules. At one of the decision points, if we make another choice, this may lead to successfully tagging all of the words in the sentence. In our example (4), let's suppose that we have a rule that a noun cannot follow a noun. So, *fish* must be a verb. Let's also suppose that we have a rule that a verb cannot follow a verb. So, *cook* must be a noun. Let's continue: *green* can only be a noun or an adjective. Our rule stating that a noun cannot follow a noun means that it must be an adjective. But suppose we also have a rule that an adjective cannot follow a noun. We cannot go further because there are no alternative choices. We haven't given a word category tag to every word. We have reached a dead end.

If that decision point still has unused options, we take one of them, otherwise we backtrack to a decision point that has unused options. In (4), we had only one decision point in the maze at which we had a choice: *Orange* can now be chosen to be an adjective. We now move forward through the words in (4).

Our search for a set of rules that provides a word category tag for each word may take some time. If our set of rules is sufficient to model all possible sentences in our language, we are guaranteed to find a sequence of rule choices that correctly tags all of the words in the sentence. Let's continue. In (4), we probably have a rule that a verb cannot follow an adjective, so *fish* must be a noun. Since a noun cannot follow a noun, *cook* must be a verb. With the rule against a noun following a noun, and with a rule that a verb cannot follow an adjective, we would have two possible word category sequences. Only one of them is the one desired, so to remove the other, a rule not permitting two verbs in a sentence would be needed.

Sometimes you are lucky and make a lot of correct decisions, but sometimes this search methodology is no better than trying all possibilities.

## 17.5 Probabilities

Another method to capture linguistic information is by using probabilities. How to compute these probabilities could be as simple as taking a sufficiently large number of examples of sentences and counting the number of times that an adjective follows a determiner and a noun follows a determiner. These frequencies can then be used to compute probabilities.

**PAUSE AND REFLECT 17.3**

As we just stated, frequencies can be used to compute probabilities. Take two dice. Each one has six sides, and each side has a different number of one to six dots. When rolling the dice, how many combinations of dots on the two face-up sides are there? If we add the numbers on the dice we can have any of the numbers two to 12. How many ways can each number be achieved? What are the probabilities that each number will be rolled? What is the probability that when the dice are rolled, one of the faces is a six? Often, we are interested in the probability of something happening given another condition. For instance, what is the probability that one of the faces is a six when the roll of the dice gives a seven? What about when the roll of the dice is 11?

One way to use probabilities is with the search technique described earlier. To do this, let's return to our maze analogy. Let us say that we have experienced many mazes built by the same maze architect and that we have discovered that the builder of the maze favors one direction over the other, let's say taking the path to the right. So, normally we would take the path to the right, and most of the time we would find the exit more quickly. This can also be said for computational linguistics: our probabilities are obtained from many examples of sentences. For instance, in most sentences that we encounter, if we see a word that could be a verb immediately following a noun, it usually is a verb. This information often sends us down the correct path, but occasionally it sends us down a garden path.

Leaving our analogy aside, our new computational paradigm simply tries various ways (choices) to come up with the meaning of a sentence, some leading to failure, some to success. The information provided whenever a choice needs to be made could be an important piece of linguistic information, for example, if *flies* is a verb then *like* will not be a verb, so the only other alternative is that it is a preposition. Hence the wrong path (interpreting *like* as a verb) will not be taken.

## 17.6 Machine Learning

Another way to capture linguistic information is with a technique called **machine learning**. In the beginning, all computer programs were composed as sequences of instructions (or rules to be followed) crafted by humans. Most computer programs are still made in this way. However, since the 1990s, machine learning has been increasingly used to develop the decision-making processes that are executed by computers in applications such as self-driving automobiles, recommendation systems like Amazon and Netflix, and detection of fraudulent credit card usage. These decision-making processes are learned by the computer by analyzing millions of situations in order to act correctly in each situation. In the linguistic arena, IBM's Watson, a question-answering system that uses machine-learning techniques, famously defeated the two best human players in the game show *Jeopardy!* Google uses **deep learning**, one of the latest forms of machine learning that makes more human-like abstractions of what it is learning, in its Translate app.

To appreciate this computational method, let's look at another analogy. We want to predict the time that the sun will rise. In order to do this, we record each day, in a two-column table, the date and time that the sun rises. To be certain that our table of information is not simply a collection of random events, we will complete it for a second and third year. To predict when the sun will rise on a particular day of the year, we simply need to look up the date in the table. Because we have collected information for each day of the year for three different years, we may notice small variations, so our predictions will not be perfect, but they will be reasonably accurate.

Now we have learned the sunrise times. However, we will quickly realize that if our recording does not include a leap year, then we will have a problem in a leap year because we will not be able to predict the time of sunrise on 29 February. And if we move our location a small distance, the table may be unreliable, and if we move our location a significant distance, then the table will be completely useless. In order to solve the first problem, if a pattern is noticed, say the sunrise time changes by two minutes each day, then the table can be reformulated according to this simple mathematical regularity, and this regularity can be used rather than the explicit table.

Why is machine learning used to give computers the ability to gain the knowledge to perform sophisticated tasks? To program a computer to perform a task, all possible situations that can arise need to be known in advance so that the computer program can correctly perform the task in each of these situations. Some tasks are simply too complicated for human programmers to capture all of these situations in the computer program. With advances in machine learning, it is easier for a computer to analyze a large number of examples of situations (too many for a human to analyze) to capture patterns that can be found in these examples.

### LINGUISTICS TIDBITS: HOW *BIG* ARE BIG NUMBERS?

Can you imagine the size of the number 1,100,000,000,000 – the number of combinations of word categories of a 40-word sentence mentioned above?

- Many laptops today have a 1TB hard drive.
- The space required to simply put the numbers from 1 to 1,100,000,000,000 in computer memory would take between 5TB and 8TB.
- The space needed to store all of the multilingual Wikipedias (in 2015) is 10TB.
- To label the 40 words with their word category for all of the alternatives would require 40TB or 40 1TB hard drives – simply for one 40-word sentence!

## 17.7 Technological Changes

**Digital electronic computers** became available in the 1950s. Computational linguistics began in these early years because the world was in conflict: The Cold War provided a strong impetus to create computer programs that would automatically translate between English and Russian. The translation problem proved too difficult for these early computers, but much has changed since then. Let's examine some of these changes. In 1960, the most popular computer was the IBM 1401. It had typically 16,000 bytes of main memory and a clock speed of .087 MHz. This compares to today's personal computers with 16GB of main memory (one million times larger) and clock speeds of 3 GHz (40,000 times faster).

**LINGUISTICS TIDBITS: COMPUTER TECHNOLOGY CHANGES RAPIDLY**

How quickly does digital computer technology progress? Here is an example: The New Horizons interplanetary space probe was launched in 2006. It had onboard a specially adapted late 1980s style computer. This computer was able to perform 12 million operations per second. When the space probe reached Pluto ten years later (in 2016), computers in our smart phones were able to perform 1.4 billion operations per second, more than a 100-fold increase!

In addition to these changes, we now use computers in parallel, enhancing our computational forces beyond the single computer. Automatic translation between languages is still beyond our current capabilities, but because of the capacity and speed of current machines together with the use of technologies, such as machine learning and parallel computing, we are much closer to obtaining this goal.

What other computer-related techniques and artefacts have changed to allow us to generate significantly more sophisticated computational linguistics software?

In the 1950s and early 1960s, all information was represented using what is called 6-bit **binary-coded decimal** (BCD). This early coding system only allowed the encoding of the 26 uppercase alphabet letters, the digits 0–9, and some punctuation for use by computers. Computational linguistics focused on English. Because text existed only in printed form, texts had to be specially prepared for input into the computer.

Now we use **Unicode** Transformation Format (UTF), UTF-8, UTF-16, or UTF-32. This has expanded our ability to represent every common written language and all of these languages are of interest to computational linguists.

In the 1960s, the first large corpus of English text was compiled: the Brown corpus. It comprised one million words. In the 1990s the Switchboard corpus of spoken English was prepared. It had three million words.

Compare this to the situation now: **the Web**, which was created in 1989 and made useful for everyone in the 1990s, contains more than a trillion words of written text, doubling in quantity every few years. The Web now forms a previously unimaginable source of information for machine learning.

## 17.8 Linguistic Features and Applications

Let us now turn our attention to some of the aspects of language that are of interest to computational linguists and have become key components of computer applications.

### 17.8.1 Speech

Possibly the most natural way to communicate with a computer is through speech. The problem of ambiguity at the phoneme level is paramount. Great strides have been made through the use of machine learning. From some of the early speech-to-text translation applications that needed to undergo specialized training exercises to understand each user's voice, smart phones are now endowed with sophisticated language-comprehension systems that can understand almost anybody, albeit in narrow domains of language. These systems, such as Siri on Apple's iPhone, connect us to the information on the Web in a way that is far more effortless than typing with thumbs on a tiny keyboard. In addition to speech recognition, some applications are able to produce human-like speech.

#### EYES ON WORLD LANGUAGES: TRANSLATION

Europe is a place where language translation is important because of the close proximity of a number of languages used by business and holiday travellers and a multinational parliament that requires documents to be translated into all European languages. Much effort has gone into using computational linguistic techniques to assist with this translation.

By the year 2000, two highly reliable translation applications (German-English and German-Japanese) for spoken language used in typical conversations in hotels, restaurants, and grocery stores had been developed for mobile devices. By 2012, deep learning (a subarea of machine learning which is based on artificial neural networks with representation learning) was being used to build software that translates spoken English into spoken Mandarin reasonably well in real-time.

### 17.8.2 Text

#### The Word Level

Much can be accomplished at the word level. Computers can analyze words morphologically, and can also detect the word category of words. Because computers are able to perform **named entity recognition**, words can be analyzed in groups. With named entity recognition, the words in the following sequence *International*, *Business*, and *Machines* are not seen as isolated words but, when combined in sequence, they are understood to refer to the corporation IBM. Computers have achieved human-level abilities on these tasks.

**N-grams** are sequences of *n* linguistic items. The following discussion concerns sequences of *n* words. Examples of *n*-grams are: 1-grams are words; 2-grams (or bigrams) are sequences of two words, such as *by the*; 3-grams (or trigrams) are sequences of three words; 4-grams are sequences of four words; and 5-grams are sequences of five words, for example *to know how big the* (taken from the sentence *Astronomers want to know how big the universe is*). In 2006, Google produced a list of almost four billion 1-, 2-, 3-, 4-, and 5-grams that occur at least 40 times on the Web. Until Google did it, producing a list including 4- and 5-grams was a seemingly impossible task. One of the language models described earlier uses probabilities. Very accurate probabilities of word sequences can be calculated using this Google-gram list.

Word meaning and word association is being accomplished now by a technique called **word embedding**, which uses machine learning techniques to find clusters of similar words, such as *doctor-nurse, king-queen*, etc., for example.

### The Sentence Level

Ambiguity once again plays an important role in how we process language at the sentence level. Example (1) showed that grouping words into larger units is an important step in finding the meaning of a sentence. Much effort has gone into understanding how to reduce wrong decisions when combining different possible sentential grammatical constituents. Some of the ambiguity is reduced by using methods at word level, such as deciding the word category for each word. Over time, parsers based on grammar rules have been augmented with statistical information to guide the decision-making process. One of these statistical parsers has a further ranking of the possible parses which can take into account linguistic information not available at the grammar rule level, for instance, right branching parses, which are more common in English, can be preferred. The best statistical parsers come close to the ability that humans demonstrate. In May 2016, Google released Parsey McParseface, reported as the most accurate parser at that time. It has been developed using deep learning, a type of machine learning.

**LINGUISTICS TIDBITS: COMPUTERS ARE ANALYZING ALL TEXT ENTERED ONLINE**

Have you ever posted a comment or a review online about a consumer product that you have bought or a service, like a hotel, that you have used? Have you posted on Twitter?

All of these online media are being processed sentence-by-sentence by software that uses computational linguistic techniques. One aspect that is being monitored is sentiment: Did you like the product, did you comment on a particular fault, did you comment during an election about a policy statement? This information is valuable to the consumer product maker or the political party.

### The Discourse Level

The discourse level provides a different set of aspects of language of interest to computational linguists.

- Documents can be classified by the topic they are discussing.
- Documents can be summarized.
- Linguistic style can be analyzed.
- IBM's ***Watson***, a question answering system which beat the two best human ***Jeopardy!*** contestants, was able to understand aspects of language like puns.

Watson is now used in many software systems that require access to the information contained in unstructured texts.

**SUMMARY**

This has been a brief overview of computational linguistics. We have explored the effects of language ambiguity on computation, explained how computation is done, learned about aspects of language that are of interest to computational linguists, and introduced some applications.

Ambiguity comes in the form of word level ambiguity, syntactic ambiguity, and semantic ambiguity. Ambiguity causes a rapid increase in the number of possible interpretations of a natural language sentence. One method for dealing with this ambiguity was examined. Machine learning was mentioned as an alternative approach.

Applications of computational linguistics abound, a few of which have been mentioned in this overview. The rate at which new applications are developed and the increasing sophistication of the linguistic techniques that are incorporated into them is impressive. Computational linguistics is part of the technological revolution that is changing how we interact with our world.

## EXERCISES

**17.1** In linguistic morphology, word stemming is the process of reducing inflected and derived words to their word stem. Explore an online site that demonstrates word stemming (some examples include https://tartarus.org/martin/PorterStemmer/index.html and http://9ol.es/porter_js_demo.html) and perform word stemming on a number of examples. These two sites have a quotation by Oscar Wilde which you can stem. Other words that will provide some idea what stemming does include: *cats, catlike, catty, stems, stemmer, stemming, stemmed, fishing, fished, argue, argued, argues, arguing, argument, string, ringing, bring, living, loving, listening, ruling, swimming, lemming, lies, tries, is, was, were, will*. Try to reproduce the stemming algorithm by trying various words of your choice. Lemmatization attempts to remove inflectional affixes and to return the base form of a word, known as the lemma. Try an online lemmatization demo (for example, http://textanalysisonline.com/nltk-wordnet-lemmatizer) with the same words that you used on the stemming site. Note the differences in the two algorithms. A good resource to read is https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html. Try some of the examples given there on the stemming and lemmatization sites.

**17.2** Here is an exercise that you can do in a group. Take a page of text from any book and tabulate the frequency of each word. Your group members can take one paragraph each. Each group in the class should do a different page of text. Plot the frequency of each word with the x-axis being the words (ordered by decreasing frequency) and the y-axis being the frequency. The 100 most frequent words will probably resemble the following list in very close proximity to this order: *the, of, and, to, a, in, that, is, was, he, for, it, with, as, his, on, be, at, by, I, this, had, not, are, but, from, or, have, an, they, which, one, you, were, all, her, she, there, would, their, we, him, been, has, when, who, will, no, more, if, out, so, up, said, what, its, about, than, into, them, can, only, other, time, new, some, could, these, two, may, first, then, do, any, like, my, now, over, such, our, man, me, even, most, made, after, also, did, many, off, before, must, well, back, through, years, much, where, your, way.* What is the ratio of words in this list to all the words not in this list? If you now take all of the tabulations done by your class, how does this affect the numbers?

**17.3** Garden path sentences: *The old man the boat* and *The complex houses married and single soldiers and their families.* Can you parse them? Can you suggest why they are garden path sentences?

**17.4** *Time flies like an arrow* is a sentence that has a number of syntactic and semantic ambiguities. Give as many interpretations of this sentence as you can think of.

## FURTHER READING

For those of you who are interested in exploring computational linguistics more deeply, a number of books have been written. Here are a selected few:

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. Sebastopol, CA: O'Reilly Media.

Jurafsky, D., & Martin, J. (2008). *Speech and language processing* (2nd Edition). Upper Saddle River, NJ: Pearson.

Manning, C., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, MA: MIT Press.

Manning, C., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge, UK: Cambridge University Press.

Speech Recognition Breakthrough for the Spoken, Translated Word:

https://www.youtube.com/watch?v=Nu-nlQqFCKg