Chapter 1

Introduction

1.1 Applications of NLP

Natural language processing (NLP) is about making computers do all kinds of things with natural language. I can think of three broad areas where this would be useful.

First, we'd like to be able to interact with computers using natural language. This idea has captured imaginations for a long time, at least since *Star Trek* and *2001: A Space Odyssey*'s HAL 9000, and became a major goal of NLP research and development – for example, Bill Gates was a major advocate, saying things like "Most of [our research] now is focused on what we call the natural interface – the computer being able to listen and talk and recognize handwriting. . . .Now we're betting the company on these natural interface technologies." [1] Today, the technology for this is good enough for systems like Apple Siri to become commercial products, but it still has a long way to go.

There are also situations where we'd like to understand or communicate with other humans, but face a limitation that we'd like NLP to overcome. One limitation is when the other person doesn't speak the same language, and we'd like to use NLP to translate between the two languages. Historically, this was the oldest application of NLP, and indeed one of the very oldest applications of computers. The most well-known early system was developed by Georgetown and IBM in the early 1950s for translating Russian into English. Now, you can use Google Translate to get translations that are very high quality under the right conditions, but still need work under other conditions.

Another limitation is when there is too much language: I can read a book, but I can't read a million books. I'd like to use NLP to read them for me and then answer questions about them, summarize them, extract relevant pieces of information from them, etc. As more and more data comes into existence, and much of it in the form of natural language, this application of NLP becomes more and more important. The highest-profile recent demonstration of this use of NLP was IBM's Watson, which defeated Ken Jennings at Jeopardy in 2011.

1.2 Stages of NLP

How can we make computers do these things? We can break a typical NLP system into several stages of processing, which correspond to the units of this course.

[1] Remarks at Gartner Symposium, 1997/10/06, Orlando, FL.

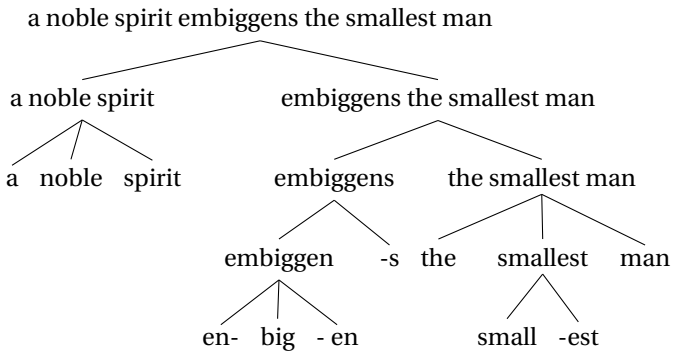

Figure 1.1: Levels of structure.

1.2.1 Text

Raw language input exists in many forms: primarily, speech (for spoken languages) and signing (for sign languages), and secondarily, all the ways that people have come up with over the centuries for encoding language, like handwriting, printing, and keyboard input. (There are other forms of language like whistling and drums that are not the focus of any serious NLP research that I'm aware of.)

The first stage of language processing involves ingesting language in one or more of the above forms and getting it into a representation that computers can do useful things with. Nearly always, that representation is plain text.

Converting each of the above forms of language into text is a research field in its own right: speech recognition, sign language recognition, handwriting recognition, optical character recognition. Even converting typing into text is not trivial (think about mobile devices, or users with disabilities), and research on text input methods sits at the border between human-computer interaction and NLP.

In this first part of the course, we'll take a brief look at these topics and how they interface with natural language processing.

1.2.2 Structure

In the second part of the course, we'll study how computers can automatically discern the *structure* of natural language text (Figure 1.1): words combine to form phrases, phrases combine to form sentences; in the other direction, words can often be broken down into smaller units called *morphemes*.

The reason that we're interested in structure is that we believe that structure is the key to understanding language, as well as other understanding-like tasks. For example, suppose you want to translate this sentence into Latin:

(1.1) spiritus nobile minimum virum auget
spirit noble smallest man embiggens

In order to do this right, your system has to learn that in Latin, verbs (auget = embiggen) usually come after their objects (minimum virum = the smallest man). These elements belong to syntactic structure, which is not explicit in our data (punctuation in text and intonation in speech give hints, but not very much).

The big problem at this stage is *ambiguity*: a given expression can have more than one structure. In fact, most expressions have many, many structures. So the computer's job is to figure out which structure out of all the possible structures is the right one.

1.2.3 Meaning

Each of the units discussed in the previous section (1.2.2) has a meaning: a letter/sound *r* doesn't have meaning, but a morpheme *re-* has meaning; likewise, words, phrases, and sentences have meanings.

The principle of *compositionality*, which originates in the philosophy of language, says that the meaning of an expression is a function of the meanings of its subexpressions. A word's meaning is a function of its morphemes' meanings; a phrase's meaning is a function of its words' meanings, and so on. So, having determined the structure of a piece of text, computing its meaning is thought to be a bottom-up process, from the morphemes at the bottom all the way up to sentences and beyond.

1.2.4 Generation

Finally, in many (though not all) applications, we need the computer to go in the reverse direction, from internal representations of meaning to spoken or written language. In the last unit of the course, we'll look at the problem of generation, though I'll primarily focus on something I understand better, natural language translation.

1.3 Approaches to NLP

1.3.1 Language

The above division of natural language processing into stages is informed by the science of linguistics. Very early NLP was more or less *ad hoc*, but in the 1960s, a committee of scientists appointed by the government prescribed more basic research into *computational linguistics*, the use of computational methods for the scientific study of language. The hope was, and is, that by understanding better how human language works, we will do a better job programming computers to imitate it.

For some (myself included), computational linguistics is interesting even if it doesn't lead to NLP applications. Although human languages seem so different from the formal languages and computer languages invented by people, they, too, are governed by rules, rules that you were never explicitly taught by your parents or in school. To take one example, if you are a native speaker of English, then you know that the sentence

(1.2) Who did Bill ask when arrived?

is not English. You have to say "Bill asked when *who* arrived?" instead. A theory of language should explain why, and one particularly simple explanation comes from computational linguistics, using a grammar slightly more powerful than a context-free grammar, called a tree-adjoining grammar.

1.3.2 Learning

In the 1990s, there was a second major shift in the way natural language processing was done. Instead of just building systems that simulate human use of language, we began trying to simulate a second human behavior: *learning* language. In other words, we used to program the rules of language directly into

the computer, but now we program computers to learn the rules, and their weights, automatically from *data*. So the goal of modern, statistical NLP is to build computer systems that learn from data how to use human language.

Initially, people who used linguistics and the people who used statistics were at odds with each other. The reason was simple: linguistics is primarily interested in structures and representations that exist in the mind and cannot be directly observed, whereas statistics are based on observable quantities. So for a time, it was assumed that if you were using linguistics, you did not believe in statistics, and if you were using statistics, you did not believe in linguistics.

1.3.3 The synthesis

Over time, however, a synthesis emerged, as people started to build datasets annotated with linguistic structures (for example, the Penn Treebank) and to experiment with models that can learn unobserved things. The result is that nearly all work that appears at NLP conferences today is statistical, and a great deal of it also operates on linguistic structures – not necessarily tied to a particular linguistic theory, but very much drawing on the collective insights of linguistics about how language works.

In my view, the key to this synthesis was the formal models of language – finite automata and context-free grammars – proposed by and quickly discarded by Chomsky. These models make it possible to represent linguistically-reasonable structures in a way that machine learning algorithms can be very effectively adapted to.