Chapter 1

Introduction to Natural Language Processing

1.1 Why Language Processing is Useful

How do we write programs to manipulate natural language? What questions about language could we answer? How would the programs work, and what data would they need? These are just some of the topics we will cover in this book. Before we tackle the subject systematically, we will take a quick look at some simple tasks in which computational tools manipulate language data in a variety of interesting and non-trivial ways.

Our first example involves word stress. The CMU Pronunciation Dictionary is a machine-readable dictionary that gives the pronunciation of over 125,000 words in North American English. Each entry consists of a word in standard orthography followed by a phonological transcription. For example, the entry for *language* is the following:

(1) language / L AE1 NG G W AH0 JH .

Each character or group of characters following the slash represents an English phoneme (e.g., 'AE' corresponds to the IPA symbol æ), and the final numbers indicate word stress. That is, 'AE1' is the nucleus of a stressed syllable, while 'AH0' is the nucleus of an unstressed one. Let's suppose that we want to find every word in the dictionary which exhibits a particular stress pattern; say, words whose primary stress is on their fifth-last syllable (this is called **pre-preantepenultimate** stress). Searching through the dictionary by hand would be tedious, and we would probably miss some of the cases. We can write a simple program that will extract the numerals 0, 1 and 2 from transcription, and then creates a new field `stress_pattern` for each word which is just a sequence of these stress numbers. After this has been done, it is easy to scan the extracted stress patterns for any sequence which ends with `1 0 0 0 0`. Here are some of the words that we can find using this method:

(2)
```
ACCUMULATIVELY / AH0 K Y UW1 M Y AH0 L AH0 T IH0 V L IY0
AGONIZINGLY / AE1 G AH0 N AY0 Z IH0 NG L IY0
CARICATURIST / K EH1 R AH0 K AH0 CH ER0 AH0 S T
CUMULATIVELY / K Y UW1 M Y AH0 L AH0 T IH0 V L IY0
FORMALIZATION / F AO1 R M AH0 L AH0 Z EY0 SH AH0 N
HYPERSENSITIVITY / HH AY2 P ER0 S EH1 N S AH0 T IH0 V AH0 T IY0
IMAGINATIVELY / IH2 M AE1 JH AH0 N AH0 T IH0 V L IY0
INSTITUTIONALIZES / IH2 N S T AH0 T UW1 SH AH0 N AH0 L AY0 Z AH0 Z
```

```
SPIRITUALIST / S P IH1 R IH0 CH AH0 W AH0 L AH0 S T
UNALIENABLE / AH0 N EY1 L IY0 EH0 N AH0 B AH0 L
```

Our second example also involves phonology. When we construct an inventory of sounds for a language, we are usually interested in just those sounds which can make a difference in word meaning. To do this, we look for **minimal pairs**; that is, distinct words which differ in only one sound. For example, we might argue that the sounds [p] and [b] in English are distinctive because if we replace one with the other, we often end up with a different word:

(3) **p**at vs. **b**at
ni**p** vs. ni**b**

Suppose we want to do this more systematically for a language where we have a list of words, but are still trying to determine the sound inventory. As a case in point, NLTK includes a lexicon for Rotokas, an East Papuan language spoken on Bougainville Island, near Papua New Guinea. Let's suppose we are interested in how many vowels there are in Rotokas. We can write a program to find all four-letter words which differ only by their first vowel, and tabulate the results to illustrate vowel contrasts:

(4)

```
kasi -    kesi kusi kosi
kava -    -    kuva kova
karu kiru keru kuru koru
kapu kipu -    -    kopu
karo kiro -    -    koro
kari kiri keri kuri kori
kapa -    kepa -    kopa
kara kira kera -    kora
kaku -    -    kuku koku
kaki kiki -    -    koki
```

The two preceding examples have used lexical resources. We can also write programs to analyze texts in various ways. In this example, we try to build a model of the patterns of adjacent words in the book of Genesis. Each pair of adjacent words is known as a **bigrams**, and we can build a very simple model of biblical language just be counting bigrams. There are many useful things we can do with such information, such as identifying genres of literature or even identifying the author of a piece of text. Here we use it for a more whimsical purpose: to generate random text in the style of Genesis. As you will see, we have managed to capture something about the flow of text from one word to the next, but beyond this it is simply nonsense:

(5)

```
lo, it came to the land of his father and he said, i will not be a
wife unto him, saying, if thou shalt take our money in their kind,
cattle, in thy seed after these are my son from off any more than all
that is this day with him into egypt, he, hath taken away unawares to
pass, when she bare jacob said one night, because they were born two
hundred years old, as for an altar there, he had made me out at her
pitcher upon every living creature after thee shall come near her:
yea,
```

For our last example, let's suppose we are engaged in research to study semantic contrast in English verbs. We hypothesize that one useful source of data for exploring such contrasts might be a list of verb phrases which are conjoined with the word *but*. So we need to carry out some grammatical analysis to find conjoined verb phrases, and also need to be able to specify *but* as the conjunction. Rather than trying to do the grammatical analysis ourselves, we can make use of a resource in which the syntactic trees have already been added to lots of sentences. The best known of such resources is the University of Pennsylvania Treebank corpus (or Penn Treebank for short), and we can write a program to read trees from this corpus, find instances of verb phrase conjunctions involving the word *but*, and display parsed text corresponding to the two verb phrases.

(6)
```
(VBZ has) (VP opened its market to foreign cigarettes)
  BUT (VBZ restricts) (NP advertising) (PP-CLR to designated places)
(VBZ admits) (SBAR 0 she made a big mistake)
  BUT (VBD did) (RB n't) (VP elaborate)
(VBD confirmed) (SBAR 0 he had consented to the sanctions)
  BUT (VBD declined) (S *-1 to comment further)
(VBP are) (NP-PRD a guide to general levels)
  BUT (VBP do) (RB n't) (ADVP-TMP always) (VP represent actual transactions)
(VBN flirted) (PP with a conversion to tabloid format) (PP-TMP for years)
  BUT (ADVP-TMP never) (VBN executed) (NP the plan)
(VBD ended) (ADVP-CLR slightly higher)
  BUT (VBD trailed) (NP gains in the Treasury market)
(VBD confirmed) (NP the filing)
  BUT (MD would) (RB n't) (VP elaborate)
```

In presenting these examples, we have tried to give you a flavour of the range of things that can be done with natural language using computational tools. All the above examples were generated using simple programming techniques and a small amount of Python code. After working through the first few chapters of this book, you will be able write such programs yourself. In the process, you will come to understand the basics of **natural language processing** (henceforth abbreviated as NLP) as a subject. In the remainder of this chapter, we will give you more reasons to think that NLP is both important and fun.

1.2 The Language Challenge

1.2.1 Language is rich and complex

Language is the chief manifestation of human intelligence. Through language we express basic needs and lofty aspirations, technical know-how and flights of fantasy. Ideas are shared over great separations of distance and time. The following samples from English illustrate the richness of language:

1. a. Overhead the day drives level and grey, hiding the sun by a flight of grey spears. (William Faulkner, *As I Lay Dying*, 1935)
   b. When using the toaster please ensure that the exhaust fan is turned on. (sign in dormitory kitchen)
   c. Amiodarone weakly inhibited CYP2C9, CYP2D6, and CYP3A4-mediated activities with Ki values of 45.1-271.6 µM (Medline, PMID: 10718780)

d. Iraqi Head Seeks Arms (spoof news headline)

e. The earnest prayer of a righteous man has great power and wonderful results. (James 5:16b)

f. Twas brillig, and the slithy toves did gyre and gimble in the wabe (Lewis Carroll, *Jabberwocky*, 1872)

g. There are two ways to do this, AFAIK :smile: (internet discussion archive)

Thanks to this richness, the study of language is part of many disciplines outside of linguistics, including translation, literary criticism, philosophy, anthropology and psychology. Many less obvious disciplines investigate language use, such as law, hermeneutics, forensics, telephony, pedagogy, archaeology, cryptanalysis and speech pathology. Each applies distinct methodologies to gather observations, develop theories and test hypotheses. Yet all serve to deepen our understanding of language and of the intellect which is manifested in language.

The importance of language to science and the arts is matched in significance by the cultural treasure embodied in language. Each of the world's ~7,000 human languages is rich in unique respects, in its oral histories and creation legends, down to its grammatical constructions and its very words and their nuances of meaning. Threatened remnant cultures have words to distinguish plant subspecies according to therapeutic uses which are unknown to science. Languages evolve over time as they come into contact with each other and they provide a unique window onto human pre-history. Technological change gives rise to new words like *blog* and new morphemes like *e-* and *cyber-*. In many parts of the world, small linguistic variations from one town to the next add up to a completely different language in the space of a half-hour drive. For its breathtaking complexity and diversity, human language is as a colourful tapestry stretching through time and space.

1.2.2 Language and the Internet

Today, both professionals and ordinary people are confronted by unprecedented volumes of information, the vast bulk of which is stored as unstructured text. In 2003, it was estimated that the annual production of books amounted to 8 Terabytes. (A Terabyte is 1,000 Gigabytes, i.e., equivalent to 1,000 pickup trucks filled with books.) It would take a human being about five years to read the new scientific material that is produced every 24 hours. Although these estimates are based on printed materials, increasingly the information is also available electronically. Indeed, there has been an explosion of text and multimedia content on the World Wide Web. For many people, a large and growing fraction of work and leisure time is spent navigating and accessing this universe of information.

The presence of so much text in electronic form is a huge challenge to NLP. Arguably, the only way for humans to cope with the information explosion is to exploit computational techniques which can sift through huge bodies of text.

Although existing search engines have been crucial to the growth and popularity of the Web, humans require skill, knowledge, and some luck, to extract answers to such questions as *What tourist sites can I visit between Philadelphia and Pittsburgh on a limited budget? What do expert critics say about Canon digital cameras? What predictions about the steel market were made by credible commentators in the past week?* Getting a computer to answer them automatically is a realistic long-term goal, but would involve a range of language processing tasks, including information extraction, inference, and summarization, and would need to be carried out on a scale and with a level of robustness that is still beyond our current capabilities.

1.2.3 The Promise of NLP

As we have seen, NLP is important for scientific, economic, social, and cultural reasons. NLP is experiencing rapid growth as its theories and methods are deployed in a variety of new language technologies. For this reason it is important for a wide range of people to have a working knowledge of NLP. Within academia, this includes people in areas from humanities computing and corpus linguistics through to computer science and artificial intelligence. Within industry, it includes people in human-computer interaction, business information analysis, and Web software development. We hope that you, a member of this diverse audience reading these materials, will come to appreciate the workings of this rapidly growing field of NLP and will apply its techniques in the solution of real-world problems.

The following chapters present a carefully-balanced selection of theoretical foundations and practical application, and equips readers to work with large datasets, to create robust models of linguistic phenomena, and to deploy them in working language technologies. By integrating all of this into the Natural Language Toolkit (NLTK), we hope this book opens up the exciting endeavour of practical natural language processing to a broader audience than ever before.

1.3 Language and Computtion

1.3.1 NLP and Intelligence

A long-standing challenge within computer science has been to build intelligent machines. The chief measure of machine intelligence has been a linguistic one, namely the Turing Test: can a dialogue system, responding to a user's typed input with its own textual output, perform so naturally that users cannot distinguish it from a human interlocutor using the same interface? Today, there is substantial ongoing research and development in such areas as machine translation and spoken dialogue, and significant commercial systems are in widespread use. The following dialogue illustrates a typical application:

S: How may I help you?
U: When is Saving Private Ryan playing?
S: For what theater?
U: The Paramount theater.
S: Saving Private Ryan is not playing at the Paramount theater, but
 it's playing at the Madison theater at 3:00, 5:30, 8:00, and 10:30.

Today's commercial dialogue systems are strictly limited to narrowly-defined domains. We could not ask the above system to provide driving instructions or details of nearby restaurants unless the requisite information had already been stored and suitable question and answer sentences had been incorporated into the language processing system. Observe that the above system appears to understand the user's goals: the user asks when a movie is showing and the system correctly determines from this that the user wants to see the movie. This inference seems so obvious to humans that we usually do not even notice it has been made, yet a natural language system needs to be endowed with this capability in order to interact naturally. Without it, when asked *Do you know when Saving Private Ryan is playing*, a system might simply — and unhelpfully — respond with a cold *Yes*. While it appears that this dialogue system can perform simple inferences, such sophistication is only found in cutting edge research prototypes. Instead, the developers of commercial dialogue systems use contextual assumptions and simple business logic to ensure that the different ways in which a user might express requests or provide information are handled in a way that makes sense for the particular application. Thus, whether the user says *When is ...*, or *I want to know when ...*, or *Can you tell me when ...*, simple rules will always yield screening times. This is sufficient for the system to provide a useful service.

Despite some recent advances, it is generally true that those natural language systems which have been fully deployed still cannot perform common-sense reasoning or draw on world knowledge. We can wait for these difficult artificial intelligence problems to be solved, but in the meantime it is necessary to live with some severe limitations on the reasoning and knowledge capabilities of natural language systems. Accordingly, right from the beginning, an important goal of NLP research has been to make progress on the holy grail of natural linguistic interaction *without* recourse to this unrestricted knowledge and reasoning capability. This is an old challenge, and so it is instructive to review the history of the field.

1.3.2 Language and Symbol Processing

The very notion that natural language could be treated in a computational manner grew out of a research program, dating back to the early 1900s, to reconstruct mathematical reasoning using logic, most clearly manifested in work by Frege, Russell, Wittgenstein, Tarski, Lambek and Carnap. This work led to the notion of language as a formal system amenable to automatic processing. Three later developments laid the foundation for natural language processing. The first was **formal language theory**. This defined a language as a set of strings accepted by a class of automata, such as context-free languages and pushdown automata, and provided the underpinnings for computational syntax.

The second development was **symbolic logic**. This provided a formal method for capturing selected aspects of natural language that are relevant for expressing logical proofs. A formal calculus in symbolic logic provides the syntax of a language, together with rules of inference and, possibly, rules of interpretation in a set-theoretic model; examples are propositional logic and First Order Logic. Given such a calculus, with a well-defined syntax and semantics, it becomes possible to associate meanings with expressions of natural language by translating them into expressions of the formal calculus. For example, if we translate *John saw Mary* into a formula **`saw(j,m)`**, we (implicitly or explicitly) intepret the English verb *saw* as a binary relation, and *John* and *Mary* as denoting individuals. More general statements like *All birds fly* require quantifiers, in this case $\forall$, meaning *for all*: $\forall x(bird(x) \rightarrow fly(x))$. This use of logic provided the technical machinery to perform inferences that are an important part of language understanding.

A closely related development was the **principle of compositionality**, namely that the meaning of a complex expression is comprised of the meaning of its parts and their mode of combination. This principle provided a useful correspondence between syntax and semantics, namely that the meaning of a complex expression could be computed recursively. Consider the sentence *It is not true that*:;x: *p*, where *p* is a proposition. We can represent the meaning of this sentence as $not(p)$. Similarly, we can represent the meaning of *John saw Mary* as $saw(j,m)$. Now we can compute the interpretation of *It is not true that John saw Mary* recursively, using the above information, to get $not(saw(j,m))$.

The approaches just outlined share the premise that computing with natural language crucially relies on rules for manipulating symbolic representations. For a certain period in the development of NLP, particularly during the 1980s, this premise provided a common starting point for both linguists and practioners of NLP, leading to a family of grammar formalisms known as unification-based (or feature-based) grammar, and to NLP applications implemented in the Prolog programming language. Although grammar-based NLP is still a significant area of research, it has become somewhat eclipsed in the last 15–20 years dues to a variety of factors. One significant influence came from automatic speech recognition. Although early work in speech processing adopted a model which emulated the kind of rule-based phonological processing typified by Chomsky and Halle's *SPE*, this turned out to be hopelessly inadequate in dealing with the hard problem of recognizing actual speech in anything like real time. By contrast, systems which involved learning patterns from large bodies of speech data were significantly more accurate, efficient and robust. In addition, the speech community found that progress in building better systems was hugely assisted by the construction of shared resources for quantitatively measuring performance against common test data. Eventually, much of the NLP community embraced a **data intensive** orientation to language processing, coupled with a growing use of machine-learning techniques and evaluation-led methodology.

1.3.3 Philosophical Divides

The contrasting approaches to NLP described in the preceding section relates back to early metaphysical debates about **rationalism** versus **empiricism** and **realism** versus **idealism** that occurred in the Enlightenment period of Western philosophy. These debates took place against a backdrop of orthodox thinking in which the source of all knowledge was believed to be divine revelation. During this period of the seventeenth and eighteenth centuries, philosophers argued that human reason or sensory experience has priority over revelation. Descartes and Leibniz, amongst others, took the rationalist position, asserting that all truth has its origins in human thought, and in the existence of 'innate ideas' implanted in our minds from birth. For example, they argued that the principles of Euclidean geometry were developed using human reason, and were not the result of supernatural revelation or sensory experience. In contrast, Locke and others took the empiricist view, that our primary source of knowledge is the experience of our faculties, and that human reason plays a secondary role in reflecting on that experience. Prototypical evidence for this position was Galileo's discovery — based on careful observation of the motion of the planets — that the solar system is heliocentric and not geocentric. In the context of linguistics, this debate leads to the following question: to what extent does human linguistic experience, versus our innate 'language faculty', provide the basis for our knowledge of language? In NLP this matter surfaces as differences in the priority of corpus data versus linguistic introspection in the construction of computational models. We will return to this issue later in the book.

A further concern, enshrined in the debate between *realism* and *idealism*, was the metaphysical status of the constructs of a theory. Kant argued for a distinction between phenomena, the manifestations we can experience, and "things in themselves" which can never been known directly. A linguistic realist would take a theoretical construct like **noun phrase** to be real world entity that exists independently of human perception and reason, and which actually *causes* the observed linguistic phenomena. A linguistic idealist, on the other hand, would argue that noun phrases, along with more abstract constructs like semantic representations, are intrinsically unobservable, and simply play the role of useful fictions. The way linguists write about theories often betrays a realist position, while NLP practitioners occupy neutral territory or else lean towards the idealist position. Thus, in NLP, it is often enough if a theoretical abstraction leads to a useful result; it does not matter whether this result sheds any light on human linguistic processing.

These issues are still alive today, and show up in the distinctions between symbolic vs statistical methods, deep vs shallow processing, binary vs gradient classifications, and scientific vs engineering goals. However, such contrasts are now highly nuanced, and the debate is no longer as polarized as it once was. In fact, most of the discussions — and most of the advances even — involve a 'balancing act'. For example, one intermediate position is to assume that humans are innately endowed with analogical and memory-based learning methods (weak rationalism), and use these methods to identify meaningful patterns in their sensory language experience (empiricism). For a more concrete illustration, consider the way in which statistics from large corpora may serve as evidence for binary choices in a symbolic grammar. For instance, dictionaries describe the words *absolutely* and *definitely* as nearly synonymous, yet their patterns of usage are quite distinct when combined with a following verb, as shown below:

Absolutely vs *Definitely* (Liberman 2005, LanguageLog.org)				
Google hits	*adore*	*love*	*like*	*prefer*
absolutely	289,000	905,00	16,200	644
definitely	1,460	51,000	158,000	62,600
ratio	198:1	18:1	1:10	1:97

Table 1.1:

As you will see, *absolutely adore* is about 200 times as popular as *definitely adore*, while *absolutely prefer* is about 100 times rarer then *definitely prefer*. This information is used by statistical language models, but it also counts as evidence for a symbolic account of word combination in which *absolutely* can only modify extreme actions or attributes, a property that could be represented as a binary-valued feature of certain lexical items. Thus, we see statistical data informing symbolic models. Once this information has been codified symbolically, it is available to be exploited as a contextual feature for statistical language modelling, alongside many other rich sources of symbolic information, like hand-constructed parse trees and semantic representations. Now the circle is closed, and we see symbolic information informing statistical models.

This new rapprochement is giving rise to many exciting new developments. We will touch on some of these in the ensuing pages. We too will perform this balancing act, employing approaches to NLP that integrate these historically-opposed philosophies and methodologies.

1.4 The Architecture of linguistic and NLP systems

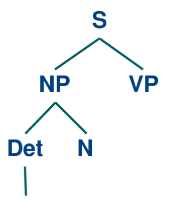
1.4.1 Generative Grammar and Modularity

One of the intellectual descendants of formal language theory was the linguistic framework known as **generative grammar**. Such a grammar contains a set of rules that recursively specify (or *generate*) the set of well-formed strings in a language. While there is a wide spectrum of models which owe some allegiance to this core, Chomsky's transformational grammar, in its various incarnations, is probably the best known. In the Chomskyan tradition, it is claimed that humans have distinct kinds of linguistic knowledge, organized into different modules: for example, knowledge of a language's sound structure (**phonology**), knowledge of word structure (**morphology**), knowledge of phrase structure (**syntax**), and knowledge of meaning (**semantics**). In a formal linguistic theory, each kind of linguistic knowledge is made explicit as different **module** of the theory, consisting of a collection of basic elements together with a way of combining them into complex structures. For example, a phonological module might provide a set of phonemes together with an operation for concatenating phonemes into phonological strings. Similarly, a syntactic module might provide labelled nodes as primitives together wih a mechanism for assembling them into trees. A set of linguistic primitives, together with some operators for defining complex elements, is often called a **level of representation**.

As well as defining modules, a generative grammar will prescribe how the modules interact. For example, well-formed phonological strings will provide the phonological content of words, and words will provide the terminal elements of syntax trees. Well-formed syntactic trees will be mapped to semantic representations, and contextual or pragmatic information will ground these semantic representations in some real-world situation.

As we indicated above, an important aspect of theories of generative grammar is that they are intended to model the linguistic knowledge of speakers and hearers; they are not intended to explain how humans actually process linguistic information. This is, in part, reflected in the claim that a generative grammer encodes the **competence** of an idealized native speaker, rather than the speaker's **performance**. A closely related distinction is to say that a generative grammar encodes **declarative** rather than **procedural** knowledge. Declarative knowledge can be glossed as 'knowing what', whereas procedural knowledge is 'knowing how'. As you might expect, computational linguistics has the crucial role of proposing procedural models of language. A central example is parsing, where we have to develop computational mechanisms which convert strings of words into structural representations such as syntax trees. Nevertheless, it is widely accepted that well-engineered computational models of language contain both declarative and procedural aspects. Thus, a full account of parsing will say how declarative knowledge in the form of a grammar and lexicon combines with procedural knowledge which determines how a syntactic analysis should be assigned to a given string of words. This procedural knowledge will be expressed as an **algorithm**: that is, an explicit recipe for mapping some input into an appropriate output in a finite number of steps.

A simple parsing algorithm for context-free grammars, for instance, looks first for a rule of the form $\mathtt{S} \longrightarrow \mathtt{X}_1 \cdots \mathtt{X}_n$, and builds a partial tree structure. It then steps through the grammar rules one-by-one, looking for a rule of the form $\mathtt{X}_1 \longrightarrow \mathtt{Y}_1 \ldots \mathtt{Y}_j$ which will expand the leftmost daughter introduced by the $\mathtt{S}$ rule, and further extends the partial tree. This process continues, for example by looking for a rule of the form $\mathtt{Y}_1 \longrightarrow \mathtt{Z}_1 \ldots \mathtt{Z}_k$ and expanding the partial tree appropriately, until the leftmost node label in the partial tree is a lexical category; the parser then checks to see if the first word of the input can belong to the category. To illustrate, let's suppose that the first grammer rule chosen by the parser is $\mathtt{S} \longrightarrow \mathtt{NP}$ $\mathtt{VP}$ and the second rule chosen is $\mathtt{NP} \longrightarrow \mathtt{Det}\ \mathtt{N}$; then the partial tree will be as follows:



If we assume that the input string we are trying to parse is *the cat slept*, we will succeed in identifying *the* as a word which can belong to the category DET. In this case, the parser goes on to the next node of the tree, N, and next input word, *cat*. However, if we had built the same partial tree with an input string *did the cat sleep*, the parse would fail at this point, since *did* is not of category DET. The parser would throw away the structure built so far and look for an alternative way of going from the S node down to a leftmost lexical category (e.g., using a rule S → V NP VP). The important point for now is not the details of this or other parsing algorithms; we discuss this topic much more fully in the chapter on parsing. Rather, we just want to illustrate the idea that an algorithm can be broken down into a fixed number of steps which produce a definite result at the end.

In figure 1.1 we further illustrate some of these points in the context of a spoken dialogue system, such as our earlier example of an application that offers the user information about movies currently on show.

Down the lefthand side of the diagram is a 'pipeline' of some representative speech understanding **components**. These map from speech input *via* syntactic parsing to some kind of meaning representation. Up the righthand side is an inverse pipeline of components for concept-to-speech generation.

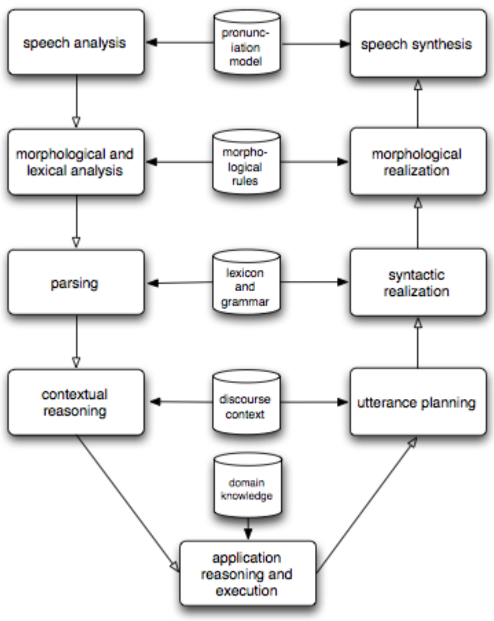


Figure 1.1: Architecture of Spoken Dialogue System

These components constitute the dynamic or procedural aspect of the system's natural language processing. In the central column of the diagram are some representative bodies of static information: the repositories of language-related data which are called upon by the processing components.

The diagram illustrates that linguistically-motivated ways of modularizing linguistic knowledge are often reflected in computational systems. That is, the various components are organized so that the data which they exchange corresponds roughly to different levels of representation. For example, the output of the speech analysis component will contain sequences of phonological representations of words, and the output of the parser will be a semantic representation. Of course the parallel is not precise, in part because it is often a matter of practical expedience where to place the boundaries between different processing components. For example, we can assume that within the parsing component there is a level of syntactic representation, although we have chosen not to expose this at the level of the system diagram. Despite such idiosyncracies, most NLP systems break down their work into a series of discrete steps. In the process of natural language understanding, these steps go from more concrete levels to more abstract ones, while in natural language production, the direction is reversed.

1.5 Before Proceeding Further...

An important aspect of learning NLP using these materials is to experience both the challenge and — we hope — the satisfaction of creating software to process natural language. The accompanying software, NLTK, is available for free and runs on most operating systems including Linux/Unix, Mac OSX and Microsoft Windows. You can download NLTK from <http://nltk.sourceforge.net/>, along with extensive documentation. We encourage you to install NLTK on your machine before reading beyond the end of this chapter.

1.6 Further Reading

Several NLP systems have online interfaces that you might like to experiment with, e.g.:

- WordNet: <http://wordnet.princeton.edu/>
- Translation: <http://world.altavista.com/>
- ChatterBots: <http://www.loebner.net/Prizef/loebner-prize.html>
- Question Answering: <http://www.answerbus.com/>
- Summarisation: <http://newsblaster.cs.columbia.edu/>

Useful websites with substantial information about NLP:

- <http://www.lt-world.org/>
- <http://www.aclweb.org/>
- <http://www.elsnet.org/>

The ACL website contains an overview of computational linguistics, including copies of introductory chapters from recent textbooks, at <http://www.aclweb.org/archive/what.html>.

Recent field-wide surveys: Mitkov, Dale et al, HLT Survey.

Acknowledgements: The dialogue example is taken from Bob Carpenter and Jennifer Chu-Carroll's ACL-99 Tutorial on Spoken Dialogue Systems.