1 INTRODUCTION

Dave Bowman: Open the pod bay doors, HAL.
HAL: I'm sorry Dave, I'm afraid I can't do that.
Stanley Kubrick and Arthur C. Clarke,
screenplay of *2001: A Space Odyssey*

This book is about a new interdisciplinary field variously called **computer speech and language processing** or **human language technology** or **natural language processing** or **computational linguistics**. The goal of this new field is to get computers to perform useful tasks involving human language, tasks like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech.

CONVERSATIONAL AGENT

One example of a useful such task is a **conversational agent**. The HAL 9000 computer in Stanley Kubrick's film *2001: A Space Odyssey* is one of the most recognizable characters in twentieth-century cinema. HAL is an artificial agent capable of such advanced language-processing behavior as speaking and understanding English, and at a crucial moment in the plot, even reading lips. It is now clear that HAL's creator Arthur C. Clarke was a little optimistic in predicting when an artificial agent such as HAL would be available. But just how far off was he? What would it take to create at least the language-related parts of HAL? We call programs like HAL that converse with humans via natural language **conversational agents** or **dialogue systems**. In this text we study the various components that make up modern conversational agents, including language input (**automatic speech recognition** and **natural language understanding**) and language output (**natural language generation** and **speech synthesis**).

CONVERSATIONAL AGENTS
DIALOGUE SYSTEMS

Let's turn to another useful language-related task, that of making available to non-English-speaking readers the vast amount of scientific information on the Web in English. Or translating for English speakers the hundreds of millions of Web pages written in other languages like Chinese. The goal of **machine translation** is to automatically translate a document from one language to another. Machine translation is far from a solved problem; we will cover the algorithms currently used in the field, as well as important component tasks.

MACHINE TRANSLATION

Many other language processing tasks are also related to the Web. Another such task is **Web-based question answering**. This is a generalization of simple web search, where instead of just typing keywords a user might ask complete questions, ranging from easy to hard, like the following:

QUESTION ANSWERING

- What does "divergent" mean?
- What year was Abraham Lincoln born?
- How many states were in the United States that year?
- How much Chinese silk was exported to England by the end of the 18th century?
- What do scientists think about the ethics of human cloning?

Some of these, such as **definition** questions, or simple **factoid** questions like dates and locations, can already be answered by search engines. But answering more complicated questions might require extracting information that is embedded in other text on a Web page, or doing **inference** (drawing conclusions based on known facts), or synthesizing and summarizing information from multiple sources or web pages. In this text we study the various components that make up modern understanding systems of this kind, including **information extraction**, **word sense disambiguation**, and so on.

Although the subfields and problems we've described above are all very far from completely solved, these are all very active research areas and many technologies are already available commercially. In the rest of this chapter we briefly summarize the kinds of knowledge that is necessary for these tasks (and others like **spell correction**, **grammar checking**, and so on), as well as the mathematical models that will be introduced throughout the book.

1.1 Knowledge in Speech and Language Processing

What distinguishes language processing applications from other data processing systems is their use of *knowledge of language*. Consider the Unix `wc` program, which is used to count the total number of bytes, words, and lines in a text file. When used to count bytes and lines, `wc` is an ordinary data processing application. However, when it is used to count the words in a file it requires *knowledge about what it means to be a word*, and thus becomes a language processing system.

Of course, `wc` is an extremely simple system with an extremely limited and impoverished knowledge of language. Sophisticated conversational agents like HAL, or machine translation systems, or robust question-answering systems, require much broader and deeper knowledge of language. To get a feeling for the scope and kind of required knowledge, consider some of what HAL would need to know to engage in the dialogue that begins this chapter, or for a question answering system to answer one of the questions above.

HAL must be able to recognize words from an audio signal and to generate an audio signal from a sequence of words. These tasks of **speech recognition** and **speech synthesis** tasks require knowledge about **phonetics and phonology**; how words are pronounced in terms of sequences of sounds, and how each of these sounds is realized acoustically.

Note also that unlike Star Trek's Commander Data, HAL is capable of producing contractions like *I'm* and *can't*. Producing and recognizing these and other variations of individual words (e.g., recognizing that *doors* is plural) requires knowledge about **morphology**, the way words break down into component parts that carry meanings like *singular* versus *plural*.

Moving beyond individual words, HAL must use structural knowledge to properly string together the words that constitute its response. For example, HAL must know that the following sequence of words will not make sense to Dave, despite the fact that it contains precisely the same set of words as the original.

I'm I do, sorry that afraid Dave I'm can't.

The knowledge needed to order and group words together comes under the heading of **syntax**.

Now consider a question answering system dealing with the following question:

- How much Chinese silk was exported to Western Europe by the end of the 18th century?

In order to answer this question we need to know something about **lexical semantics**, the meaning of all the words (*export*, or *silk*) as well as **compositional semantics** (what exactly constitutes *Western Europe* as opposed to Eastern or Southern Europe, what does *end* mean when combined with *the 18th century*. We also need to know something about the relationship of the words to the syntactic structure. For example we need to know that *by the end of the 18th century* is a temporal end-point, and not a description of the agent, as the by-phrase is in the following sentence:

- How much Chinese silk was exported to Western Europe by southern merchants?

We also need the kind of knowledge that lets HAL determine that Dave's utterance is a request for action, as opposed to a simple statement about the world or a question about the door, as in the following variations of his original statement.

REQUEST:	*HAL, open the pod bay door.*
STATEMENT:	*HAL, the pod bay door is open.*
INFORMATION QUESTION:	*HAL, is the pod bay door open?*

Next, despite its bad behavior, HAL knows enough to be polite to Dave. It could, for example, have simply replied *No* or *No, I won't open the door*. Instead, it first embellishes its response with the phrases *I'm sorry* and *I'm afraid*, and then only indirectly signals its refusal by saying *I can't*, rather than the more direct (and truthful) *I won't*.[1] This knowledge about the kind of actions that speakers intend by their use of sentences is **pragmatic** or **dialogue** knowledge.

Another kind of pragmatic or **discourse** knowledge is required to answer the question

- How many states were in the United States *that year*?

What year is *that year*? In order to interpret words like *that year* a question answering system need to examine the the earlier questions that were asked; in this case the previous question talked about the year that Lincoln was born. Thus this task of **coreference resolution** makes use of knowledge about how words like *that* or pronouns like *it* or *she* refer to previous parts of the **discourse**.

To summarize, engaging in complex language behavior requires various kinds of knowledge of language:

[1] For those unfamiliar with HAL, it is neither sorry nor afraid, nor is it incapable of opening the door. It has simply decided in a fit of paranoia to kill its crew.

- Phonetics and Phonology — knowledge about linguistic sounds
- Morphology — knowledge of the meaningful components of words
- Syntax — knowledge of the structural relationships between words
- Semantics — knowledge of meaning
- Pragmatics — knowledge of the relationship of meaning to the goals and intentions of the speaker.
- Discourse — knowledge about linguistic units larger than a single utterance

1.2 AMBIGUITY

AMBIGUITY
AMBIGUOUS

A perhaps surprising fact about these categories of linguistic knowledge is that most tasks in speech and language processing can be viewed as resolving **ambiguity** at one of these levels. We say some input is **ambiguous** if there are multiple alternative linguistic structures that can be built for it. Consider the spoken sentence *I made her duck.* Here's five different meanings this sentence could have (see if you can think of some more), each of which exemplifies an ambiguity at some level:

(1.1) I cooked waterfowl for her.

(1.2) I cooked waterfowl belonging to her.

(1.3) I created the (plaster?) duck she owns.

(1.4) I caused her to quickly lower her head or body.

(1.5) I waved my magic wand and turned her into undifferentiated waterfowl.

These different meanings are caused by a number of ambiguities. First, the words *duck* and *her* are morphologically or syntactically ambiguous in their part-of-speech. *Duck* can be a verb or a noun, while *her* can be a dative pronoun or a possessive pronoun. Second, the word *make* is semantically ambiguous; it can mean *create* or *cook*. Finally, the verb *make* is syntactically ambiguous in a different way. *Make* can be transitive, that is, taking a single direct object (1.2), or it can be ditransitive, that is, taking two objects (1.5), meaning that the first object (*her*) got made into the second object (*duck*). Finally, *make* can take a direct object and a verb (1.4), meaning that the object (*her*) got caused to perform the verbal action (*duck*). Furthermore, in a spoken sentence, there is an even deeper kind of ambiguity; the first word could have been *eye* or the second word *maid*.

We will often introduce the models and algorithms we present throughout the book as ways to **resolve** or **disambiguate** these ambiguities. For example deciding whether *duck* is a verb or a noun can be solved by **part-of-speech tagging**. Deciding whether *make* means "create" or "cook" can be solved by **word sense disambiguation**. Resolution of part-of-speech and word sense ambiguities are two important kinds of **lexical disambiguation**. A wide variety of tasks can be framed as lexical disambiguation problems. For example, a text-to-speech synthesis system reading the word *lead* needs to decide whether it should be pronounced as in *lead pipe* or as in *lead me on*. By contrast, deciding whether *her* and *duck* are part of the same entity (as in (1.1) or (1.4)) or are different entity (as in (1.2)) is an example of **syntactic disambiguation** and can

be addressed by **probabilistic parsing**. Ambiguities that don't arise in this particular example (like whether a given sentence is a statement or a question) will also be resolved, for example by **speech act interpretation**.

1.3 MODELS AND ALGORITHMS

One of the key insights of the last 50 years of research in language processing is that the various kinds of knowledge described in the last sections can be captured through the use of a small number of formal models, or theories. Fortunately, these models and theories are all drawn from the standard toolkits of computer science, mathematics, and linguistics and should be generally familiar to those trained in those fields. Among the most important models are **state machines**, **rule systems**, **logic**, **probabilistic models**, and **vector-space models**. These models, in turn, lend themselves to a small number of algorithms, among the most important of which are **state space search** algorithms such as **dynamic programming**, and machine learning algorithms such as **classifiers** and **EM** and other learning algorithms.

In their simplest formulation, state machines are formal models that consist of states, transitions among states, and an input representation. Some of the variations of this basic model that we will consider are **deterministic** and **non-deterministic finite-state automata** and **finite-state transducers**.

Closely related to these models are their declarative counterparts: formal rule systems. Among the more important ones we will consider are **regular grammars** and **regular relations**, **context-free grammars**, **feature-augmented grammars**, as well as probabilistic variants of them all. State machines and formal rule systems are the main tools used when dealing with knowledge of phonology, morphology, and syntax.

The third model that plays a critical role in capturing knowledge of language is logic. We will discuss **first order logic**, also known as the **predicate calculus**, as well as such related formalisms as lambda-calculus, feature-structures, and semantic primitives. These logical representations have traditionally been used for modeling semantics and pragmatics, although more recent work has focused on more robust techniques drawn from non-logical lexical semantics.

Probabilistic models are crucial for capturing every kind of linguistic knowledge. Each of the other models (state machines, formal rule systems, and logic) can be augmented with probabilities. For example the state machine can be augmented with probabilities to become the **weighted automaton** or **Markov model**. We will spend a significant amount of time on **hidden Markov models** or **HMMs**, which are used everywhere in the field, in part-of-speech tagging, speech recognition, dialogue understanding, text-to-speech, and machine translation. The key advantage of probabilistic models is their ability to to solve the many kinds of ambiguity problems that we discussed earlier; almost any speech and language processing problem can be recast as: "given N choices for some ambiguous input, choose the most probable one".

Finally, vector-space models, based on linear algebra, underlie information retrieval and many treatments of word meanings.

Processing language using any of these models typically involves a search through

a space of states representing hypotheses about an input. In speech recognition, we search through a space of phone sequences for the correct word. In parsing, we search through a space of trees for the syntactic parse of an input sentence. In machine translation, we search through a space of translation hypotheses for the correct translation of a sentence into another language. For non-probabilistic tasks, such as state machines, we use well-known graph algorithms such as **depth-first search**. For probabilistic tasks, we use heuristic variants such as **best-first** and **A* search**, and rely on dynamic programming algorithms for computational tractability.

For many language tasks, we rely on machine learning tools like **classifiers** and **sequence models**. Classifiers like **decision trees**, **support vector machines**, **Gaussian Mixture Models** and **logistic regression** are very commonly used. A hidden Markov model is one kind of sequence model; other are **Maximum Entropy Markov Models** or **Conditional Random Fields**.

Another tool that is related to machine learning is methodological; the use of distinct training and test sets, statistical techniques like **cross-validation**, and careful evaluation of our trained systems.

1.4 Language, Thought, and Understanding

To many, the ability of computers to process language as skillfully as we humans do will signal the arrival of truly intelligent machines. The basis of this belief is the fact that the effective use of language is intertwined with our general cognitive abilities. Among the first to consider the computational implications of this intimate connection was Alan Turing (1950). In this famous paper, Turing introduced what has come to be known as the **Turing Test**. Turing began with the thesis that the question of what it would mean for a machine to think was essentially unanswerable due to the inherent imprecision in the terms *machine* and *think*. Instead, he suggested an empirical test, a game, in which a computer's use of language would form the basis for determining if it could think. If the machine could win the game it would be judged intelligent.

TURING TEST

In Turing's game, there are three participants: two people and a computer. One of the people is a contestant and plays the role of an interrogator. To win, the interrogator must determine which of the other two participants is the machine by asking a series of questions via a teletype. The task of the machine is to fool the interrogator into believing it is a person by responding as a person would to the interrogator's questions. The task of the second human participant is to convince the interrogator that the other participant is the machine, and that they are human.

The following interaction from Turing's paper illustrates the kind of interactions he had in mind. Clearly a convincing simulation of humanity doesn't require being an expert in all domains:

Q: Please write me a sonnet on the topic of the Forth Bridge.
A: Count me out on this one. I never could write poetry.
Q: Add 34957 to 70764.
A: (Pause about 30 seconds and then give answer as) 105621.

Turing predicted that roughly by the end of the twentieth century a machine with 10

gigabytes of memory would have around a 30% chance of fooling a human interrogator after 5 minutes of questioning. Given the fact that you can fool some of the people all the time, it is not clear how rigorous this particular standard is. Nevertheless, the critical issue for Turing was that using language as humans do is sufficient, by itself, as an operational test for intelligence.

Surprisingly, many of the issues raised by Turing became relevant as early as 1966 in the context of the ELIZA program (Weizenbaum, 1966). ELIZA was an early natural language processing system capable of carrying on a limited form of conversation with a user. Consider the following session with a version of ELIZA that imitated the responses of a Rogerian psychotherapist.

User_1:	You are like my father in some ways.
ELIZA_1:	WHAT RESEMBLANCE DO YOU SEE
User_2:	You are not very aggressive but I think you don't want me to notice that.
ELIZA_2:	WHAT MAKES YOU THINK I AM NOT AGGRESSIVE
User_3:	You don't argue with me.
ELIZA_3:	WHY DO YOU THINK I DON'T ARGUE WITH YOU
User_4:	You are afraid of me.
ELIZA_4:	DOES IT PLEASE YOU TO BELIEVE I AM AFRAID OF YOU

As we will discuss in more detail in Ch. 2, ELIZA is a remarkably simple program that makes use of pattern-matching to process the input and translate it into suitable outputs. The success of this simple technique in this domain is due to the fact that ELIZA doesn't actually need to *know* anything to mimic a Rogerian psychotherapist. As Weizenbaum notes, this is one of the few dialogue genres where the listener can act as if they know nothing of the world.

ELIZA's deep relevance to Turing's ideas is that many people who interacted with ELIZA came to believe that it really *understood* them and their problems. Indeed, Weizenbaum (1976) notes that many of these people continued to believe in ELIZA's abilities even after the program's operation was explained to them. In more recent years, Weizenbaum's informal reports have been repeated in a somewhat more controlled setting. Since 1991, an event known as the Loebner Prize competition has attempted to put various computer programs to the Turing test. Although these contests seem to have little scientific interest, a consistent result over the years has been that even the crudest programs can fool some of the judges some of the time (Shieber, 1994). Not surprisingly, these results have done nothing to quell the ongoing debate over the suitability of the Turing test as a test for intelligence among philosophers and AI researchers (Searle, 1980).

Fortunately, for the purposes of this book, the relevance of these results does not hinge on whether or not computers will ever be intelligent, or understand natural language. Far more important is recent related research in the social sciences that has confirmed another of Turing's predictions from the same paper.

> Nevertheless I believe that at the end of the century the use of words and educated opinion will have altered so much that we will be able to speak of machines thinking without expecting to be contradicted.

It is now clear that regardless of what people believe or know about the inner workings of computers, they talk about them and interact with them as social entities. People act

toward computers as if they were people; they are polite to them, treat them as team members, and expect among other things that computers should be able to understand their needs, and be capable of interacting with them naturally. For example, Reeves and Nass (1996) found that when a computer asked a human to evaluate how well the computer had been doing, the human gives more positive responses than when a different computer asks the same questions. People seemed to be afraid of being impolite. In a different experiment, Reeves and Nass found that people also give computers higher performance ratings if the computer has recently said something flattering to the human. Given these predispositions, speech and language-based systems may provide many users with the most natural interface for many applications. This fact has led to a long-term focus in the field on the design of **conversational agents**, artificial entities that communicate conversationally.

1.5 THE STATE OF THE ART

> We can only see a short distance ahead, but we can see plenty there that needs to be done.
>
> Alan Turing.

This is an exciting time for the field of speech and language processing. The startling increase in computing resources available to the average computer user, the rise of the Web as a massive source of information and the increasing availability of wireless mobile access have all placed speech and language processing applications in the technology spotlight. The following are examples of some currently deployed systems that reflect this trend:

- Travelers calling Amtrak, United Airlines and other travel-providers interact with conversational agents that guide them through the process of making reservations and getting arrival and departure information.
- Luxury car makers such as Mercedes-Benz models provide automatic speech recognition and text-to-speech systems that allow drivers to control their environmental, entertainment and navigational systems by voice. A similar spoken dialogue system has been deployed by astronauts on the International Space Station .
- Blinkx, and other video search companies, provide search services for million of hours of video on the Web by using speech recognition technology to capture the words in the sound track.
- Google provides cross-language information retrieval and translation services where a user can supply queries in their native language to search collections in another language. Google translates the query, finds the most relevant pages and then automatically translates them back to the user's native language.
- Large educational publishers such as Pearson, as well as testing services like ETS, use automated systems to analyze thousands of student essays, grading and assessing them in a manner that is indistinguishable from human graders.

- Interactive tutors, based on lifelike animated characters, serve as tutors for children learning to read, and as therapists for people dealing with aphasia and Parkinsons disease. (?, ?)
- Text analysis companies such as Nielsen Buzzmetrics, Umbria, and Collective Intellect, provide marketing intelligence based on automated measurements of user opinions, preferences, attitudes as expressed in weblogs, discussion forums and and user groups.

1.6 Some Brief History

Historically, speech and language processing has been treated very differently in computer science, electrical engineering, linguistics, and psychology/cognitive science. Because of this diversity, speech and language processing encompasses a number of different but overlapping fields in these different departments: **computational linguistics** in linguistics, **natural language processing** in computer science, **speech recognition** in electrical engineering, **computational psycholinguistics** in psychology. This section summarizes the different historical threads which have given rise to the field of speech and language processing. This section will provide only a sketch; see the individual chapters for more detail on each area and its terminology.

1.6.1 Foundational Insights: 1940s and 1950s

The earliest roots of the field date to the intellectually fertile period just after World War II that gave rise to the computer itself. This period from the 1940s through the end of the 1950s saw intense work on two foundational paradigms: the **automaton** and **probabilistic** or **information-theoretic models**.

The automaton arose in the 1950s out of Turing's (1936) model of algorithmic computation, considered by many to be the foundation of modern computer science. Turing's work led first to the **McCulloch-Pitts neuron** (McCulloch and Pitts, 1943), a simplified model of the neuron as a kind of computing element that could be described in terms of propositional logic, and then to the work of Kleene (1951) and (1956) on finite automata and regular expressions. Shannon (1948) applied probabilistic models of discrete Markov processes to automata for language. Drawing the idea of a finite-state Markov process from Shannon's work, Chomsky (1956) first considered finite-state machines as a way to characterize a grammar, and defined a finite-state language as a language generated by a finite-state grammar. These early models led to the field of **formal language theory**, which used algebra and set theory to define formal languages as sequences of symbols. This includes the context-free grammar, first defined by Chomsky (1956) for natural languages but independently discovered by Backus (1959) and Naur et al. (1960) in their descriptions of the ALGOL programming language.

The second foundational insight of this period was the development of probabilistic algorithms for speech and language processing, which dates to Shannon's other contribution: the metaphor of the **noisy channel** and **decoding** for the transmission of language through media like communication channels and speech acoustics. Shannon

also borrowed the concept of **entropy** from thermodynamics as a way of measuring the information capacity of a channel, or the information content of a language, and performed the first measure of the entropy of English using probabilistic techniques.

It was also during this early period that the sound spectrograph was developed (Koenig et al., 1946), and foundational research was done in instrumental phonetics that laid the groundwork for later work in speech recognition. This led to the first machine speech recognizers in the early 1950s. In 1952, researchers at Bell Labs built a statistical system that could recognize any of the 10 digits from a single speaker (Davis et al., 1952). The system had 10 speaker-dependent stored patterns roughly representing the first two vowel formants in the digits. They achieved 97–99% accuracy by choosing the pattern which had the highest relative correlation coefficient with the input.

1.6.2 The Two Camps: 1957–1970

By the end of the 1950s and the early 1960s, speech and language processing had split very cleanly into two paradigms: symbolic and stochastic.

The symbolic paradigm took off from two lines of research. The first was the work of Chomsky and others on formal language theory and generative syntax throughout the late 1950s and early to mid 1960s, and the work of many linguistics and computer scientists on parsing algorithms, initially top-down and bottom-up and then via dynamic programming. One of the earliest complete parsing systems was Zelig Harris's Transformations and Discourse Analysis Project (TDAP), which was implemented between June 1958 and July 1959 at the University of Pennsylvania (Harris, 1962).[2] The second line of research was the new field of artificial intelligence. In the summer of 1956 John McCarthy, Marvin Minsky, Claude Shannon, and Nathaniel Rochester brought together a group of researchers for a two-month workshop on what they decided to call artificial intelligence (AI). Although AI always included a minority of researchers focusing on stochastic and statistical algorithms (include probabilistic models and neural nets), the major focus of the new field was the work on reasoning and logic typified by Newell and Simon's work on the Logic Theorist and the General Problem Solver. At this point early natural language understanding systems were built, These were simple systems that worked in single domains mainly by a combination of pattern matching and keyword search with simple heuristics for reasoning and question-answering. By the late 1960s more formal logical systems were developed.

The stochastic paradigm took hold mainly in departments of statistics and of electrical engineering. By the late 1950s the Bayesian method was beginning to be applied to the problem of optical character recognition. Bledsoe and Browning (1959) built a Bayesian system for text-recognition that used a large dictionary and computed the likelihood of each observed letter sequence given each word in the dictionary by multiplying the likelihoods for each letter. Mosteller and Wallace (1964) applied Bayesian methods to the problem of authorship attribution on *The Federalist* papers.

The 1960s also saw the rise of the first serious testable psychological models of

[2] This system was reimplemented recently and is described by Joshi and Hopely (1999) and Karttunen (1999), who note that the parser was essentially implemented as a cascade of finite-state transducers.

human language processing based on transformational grammar, as well as the first on-line corpora: the Brown corpus of American English, a 1 million word collection of samples from 500 written texts from different genres (newspaper, novels, non-fiction, academic, etc.), which was assembled at Brown University in 1963–64 (Kučera and Francis, 1967; Francis, 1979; Francis and Kučera, 1982), and William S. Y. Wang's 1967 DOC (Dictionary on Computer), an on-line Chinese dialect dictionary.

1.6.3 Four Paradigms: 1970–1983

The next period saw an explosion in research in speech and language processing and the development of a number of research paradigms that still dominate the field.

The **stochastic** paradigm played a huge role in the development of speech recognition algorithms in this period, particularly the use of the Hidden Markov Model and the metaphors of the noisy channel and decoding, developed independently by Jelinek, Bahl, Mercer, and colleagues at IBM's Thomas J. Watson Research Center, and by Baker at Carnegie Mellon University, who was influenced by the work of Baum and colleagues at the Institute for Defense Analyses in Princeton. AT&T's Bell Laboratories was also a center for work on speech recognition and synthesis; see Rabiner and Juang (1993) for descriptions of the wide range of this work.

The **logic-based** paradigm was begun by the work of Colmerauer and his colleagues on Q-systems and metamorphosis grammars (Colmerauer, 1970, 1975), the forerunners of Prolog, and Definite Clause Grammars (Pereira and Warren, 1980). Independently, Kay's (1979) work on functional grammar, and shortly later, Bresnan and Kaplan's (1982) work on LFG, established the importance of feature structure unification.

The **natural language understanding** field took off during this period, beginning with Terry Winograd's SHRDLU system, which simulated a robot embedded in a world of toy blocks (Winograd, 1972). The program was able to accept natural language text commands *(Move the red block on top of the smaller green one)* of a hitherto unseen complexity and sophistication. His system was also the first to attempt to build an extensive (for the time) grammar of English, based on Halliday's systemic grammar. Winograd's model made it clear that the problem of parsing was well-enough understood to begin to focus on semantics and discourse models. Roger Schank and his colleagues and students (in what was often referred to as the *Yale School*) built a series of language understanding programs that focused on human conceptual knowledge such as scripts, plans and goals, and human memory organization (Schank and Albelson, 1977; Schank and Riesbeck, 1981; Cullingford, 1981; Wilensky, 1983; Lehnert, 1977). This work often used network-based semantics (Quillian, 1968; Norman and Rumelhart, 1975; Schank, 1972; Wilks, 1975b, 1975a; Kintsch, 1974) and began to incorporate Fillmore's notion of case roles (Fillmore, 1968) into their representations (Simmons, 1973).

The logic-based and natural-language understanding paradigms were unified on systems that used predicate logic as a semantic representation, such as the LUNAR question-answering system (Woods, 1967, 1973).

The **discourse modeling** paradigm focused on four key areas in discourse. Grosz and her colleagues introduced the study of substructure in discourse, and of discourse

focus (Grosz, 1977; Sidner, 1983), a number of researchers began to work on automatic reference resolution (Hobbs, 1978), and the **BDI** (Belief-Desire-Intention) framework for logic-based work on speech acts was developed (Perrault and Allen, 1980; Cohen and Perrault, 1979).

1.6.4 Empiricism and Finite State Models Redux: 1983–1993

This next decade saw the return of two classes of models which had lost popularity in the late 1950s and early 1960s, partially due to theoretical arguments against them such as Chomsky's influential review of Skinner's *Verbal Behavior* (Chomsky, 1959). The first class was finite-state models, which began to receive attention again after work on finite-state phonology and morphology by Kaplan and Kay (1981) and finite-state models of syntax by Church (1980). A large body of work on finite-state models will be described throughout the book.

The second trend in this period was what has been called the "return of empiricism"; most notably here was the rise of probabilistic models throughout speech and language processing, influenced strongly by the work at the IBM Thomas J. Watson Research Center on probabilistic models of speech recognition. These probabilistic methods and other such data-driven approaches spread from speech into part-of-speech tagging, parsing and attachment ambiguities, and semantics. This empirical direction was also accompanied by a new focus on model evaluation, based on using held-out data, developing quantitative metrics for evaluation, and emphasizing the comparison of performance on these metrics with previous published research.

This period also saw considerable work on natural language generation.

1.6.5 The Field Comes Together: 1994–1999

By the last five years of the millennium it was clear that the field was vastly changing. First, probabilistic and data-driven models had become quite standard throughout natural language processing. Algorithms for parsing, part-of-speech tagging, reference resolution, and discourse processing all began to incorporate probabilities, and employ evaluation methodologies borrowed from speech recognition and information retrieval. Second, the increases in the speed and memory of computers had allowed commercial exploitation of a number of subareas of speech and language processing, in particular speech recognition and spelling and grammar checking. Speech and language processing algorithms began to be applied to Augmentative and Alternative Communication (AAC). Finally, the rise of the Web emphasized the need for language-based information retrieval and information extraction.

,

1.6.6 The Rise of Machine Learning: 2000–2007

The empiricist trends begun in the latter part of the 1990s accelerated at an astounding pace in the new century. This acceleration was largely driven by three synergistic trends. First, large amounts of spoken and written material became widely available through the auspices of the Linguistic Data Consortium (LDC), and other similar or-

ganizations. Importantly, included among these materials were annotated collections such as the Penn Treebank(Marcus et al., 1993), Prague Dependency Treebank(Hajič, 1998), PropBank(Palmer et al., 2005), Penn Discourse Treebank(Miltsakaki et al., 2004), RSTBank(Carlson et al., 2001) and TimeBank(?), all of which layered standard text sources with various forms of syntactic, semantic and pragmatic annotations. The existence of these resources promoted the trend of casting more complex traditional problems, such as parsing and semantic analysis, as problems in supervised machine learning. These resources also promoted the establishment of additional competitive evaluations for parsing (Dejean and Tjong Kim Sang, 2001), information extraction(?, ?), word sense disambiguation(Palmer et al., 2001; Kilgarriff and Palmer, 2000) and question answering(Voorhees and Tice, 1999).

Second, this increased focus on learning led to a more serious interplay with the statistical machine learning community. Techniques such as support vector machines (?; Vapnik, 1995), multinomial logistic regression (MaxEnt) (Berger et al., 1996), and graphical Bayesian models (Pearl, 1988) became standard practice in computational linguistics. Third, the widespread availability of high-performance computing systems facilitated the training and deployment of systems that could not have been imagined a decade earlier.

Finally, near the end of this period, largely unsupervised statistical approaches began to receive renewed attention. Progress on statistical approaches to machine translation(Brown et al., 1990; Och and Ney, 2003) and topic modeling (?) demonstrated that effective applications could be constructed from systems trained on unannotated data alone. In addition, the widespread cost and difficulty of producing reliably annotated corpora became a limiting factor in the use of supervised approaches for many problems. This trend towards the use unsupervised techniques will likely increase.

1.6.7 On Multiple Discoveries

Even in this brief historical overview, we have mentioned a number of cases of multiple independent discoveries of the same idea. Just a few of the "multiples" to be discussed in this book include the application of dynamic programming to sequence comparison by Viterbi, Vintsyuk, Needleman and Wunsch, Sakoe and Chiba, Sankoff, Reichert *et al.*, and Wagner and Fischer (Chapters 3, 5 and 6) the HMM/noisy channel model of speech recognition by Baker and by Jelinek, Bahl, and Mercer (Chapters 6, 9, and 10); the development of context-free grammars by Chomsky and by Backus and Naur (Chapter 12); the proof that Swiss-German has a non-context-free syntax by Huybregts and by Shieber (Chapter 15); the application of unification to language processing by Colmerauer *et al.* and by Kay in (Chapter 16).

Are these multiples to be considered astonishing coincidences? A well-known hypothesis by sociologist of science Robert K. Merton (1961) argues, quite the contrary, that

> all scientific discoveries are in principle multiples, including those that on the surface appear to be singletons.

Of course there are many well-known cases of multiple discovery or invention; just a few examples from an extensive list in Ogburn and Thomas (1922) include the multiple

invention of the calculus by Leibnitz and by Newton, the multiple development of the theory of natural selection by Wallace and by Darwin, and the multiple invention of the telephone by Gray and Bell.[3] But Merton gives a further array of evidence for the hypothesis that multiple discovery is the rule rather than the exception, including many cases of putative singletons that turn out be a rediscovery of previously unpublished or perhaps inaccessible work. An even stronger piece of evidence is his ethnomethodological point that scientists themselves act under the assumption that multiple invention is the norm. Thus many aspects of scientific life are designed to help scientists avoid being "scooped"; submission dates on journal articles; careful dates in research records; circulation of preliminary or technical reports.

1.6.8 A Final Brief Note on Psychology

Many of the chapters in this book include short summaries of psychological research on human processing. Of course, understanding human language processing is an important scientific goal in its own right and is part of the general field of cognitive science. However, an understanding of human language processing can often be helpful in building better machine models of language. This seems contrary to the popular wisdom, which holds that direct mimicry of nature's algorithms is rarely useful in engineering applications. For example, the argument is often made that if we copied nature exactly, airplanes would flap their wings; yet airplanes with fixed wings are a more successful engineering solution. But language is not aeronautics. Cribbing from nature is sometimes useful for aeronautics (after all, airplanes do have wings), but it is particularly useful when we are trying to solve human-centered tasks. Airplane flight has different goals than bird flight; but the goal of speech recognition systems, for example, is to perform exactly the task that human court reporters perform every day: transcribe spoken dialog. Since people already do this well, we can learn from nature's previous solution. Since an important application of speech and language processing systems is for human-computer interaction, it makes sense to copy a solution that behaves the way people are accustomed to.

1.7 SUMMARY

This chapter introduces the field of speech and language processing. The following are some of the highlights of this chapter.

- A good way to understand the concerns of speech and language processing research is to consider what it would take to create an intelligent agent like HAL from 2001: A Space Odyssey, or build a web-based question answerer, or a machine translation engine.
- Speech and language technology relies on formal models, or representations, of

[3] Ogburn and Thomas are generally credited with noticing that the prevalence of multiple inventions suggests that the cultural milieu and not individual genius is the deciding causal factor in scientific discovery. In an amusing bit of recursion, however, Merton notes that even this idea has been multiply discovered, citing sources from the 19th century and earlier!

knowledge of language at the levels of phonology and phonetics, morphology, syntax, semantics, pragmatics and discourse. A small number of formal models including state machines, formal rule systems, logic, and probabilistic models are used to capture this knowledge.

- The foundations of speech and language technology lie in computer science, linguistics, mathematics, electrical engineering and psychology. A small number of algorithms from standard frameworks are used throughout speech and language processing,
- The critical connection between language and thought has placed speech and language processing technology at the center of debate over intelligent machines. Furthermore, research on how people interact with complex media indicates that speech and language processing technology will be critical in the development of future technologies.
- Revolutionary applications of speech and language processing are currently in use around the world. The creation of the web, as well as significant recent improvements in speech recognition and synthesis, will lead to many more applications.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

Research in the various subareas of speech and language processing is spread across a wide number of conference proceedings and journals. The conferences and journals most centrally concerned with natural language processing and computational linguistics are associated with the Association for Computational Linguistics (ACL), its European counterpart (EACL), and the International Conference on Computational Linguistics (COLING). The annual proceedings of ACL, NAACL, and EACL, and the biennial COLING conference are the primary forums for work in this area. Related conferences include various proceedings of ACL Special Interest Groups (SIGs) such as the Conference on Natural Language Learning (CoNLL), as well as the conference on Empirical Methods in Natural Language Processing (EMNLP).

Research on speech recognition, understanding, and synthesis is presented at the annual INTERSPEECH conference, which is called the International Conference on Spoken Language Processing (ICSLP) and the European Conference on Speech Communication and Technology (EUROSPEECH) in alternating years, or the annual IEEE International Conference on Acoustics, Speech, and Signal Processing (IEEE ICASSP). Spoken language dialogue research is presented at these or at workshops like SIGDial.

Journals include *Computational Linguistics*, *Natural Language Engineering*, *Speech Communication*, *Computer Speech and Language*, the *IEEE Transactions on Audio, Speech & Language Processing* and the *ACM Transactions on Speech and Language Processing*.

Work on language processing from an Artificial Intelligence perspective can be found in the annual meetings of the American Association for Artificial Intelligence (AAAI), as well as the biennial International Joint Conference on Artificial Intelli-

gence (IJCAI) meetings. Artificial intelligence journals that periodically feature work on speech and language processing include *Machine Learning*, *Journal of Machine Learning Research*, and the *Journal of Artificial Intelligence Research*.

There are a fair number of textbooks available covering various aspects of speech and language processing. Manning and Schütze (1999) (*Foundations of Statistical Language Processing*) focuses on statistical models of tagging, parsing, disambiguation, collocations, and other areas. Charniak (1993) (*Statistical Language Learning*) is an accessible, though older and less-extensive, introduction to similar material. Manning et al. (2008) focuses on information retrieval, text classification, and clustering. NLTK, the Natural Language Toolkit (Bird and Loper, 2004), is a suite of Python modules and data for natural language processing, together with a Natural Language Processing book based on the NLTK suite. Allen (1995) (*Natural Language Understanding*) provides extensive coverage of language processing from the AI perspective. Gazdar and Mellish (1989) (*Natural Language Processing in Lisp/Prolog*) covers especially automata, parsing, features, and unification and is available free online. Pereira and Shieber (1987) gives a Prolog-based introduction to parsing and interpretation. Russell and Norvig (2002) is an introduction to artificial intelligence that includes chapters on natural language processing. Partee et al. (1990) has a very broad coverage of mathematical linguistics. A historically significant collection of foundational papers can be found in Grosz et al. (1986) (*Readings in Natural Language Processing*).

Of course, a wide-variety of speech and language processing resources are now available on the Web. Pointers to these resources are maintained on the home-page for this book at:

`http://www.cs.colorado.edu/~martin/slp.html`.

Allen, J. (1995). *Natural Language Understanding*. Benjamin Cummings, Menlo Park, CA.

Backus, J. W. (1959). The syntax and semantics of the proposed international algebraic language of the Zurch ACM-GAMM Conference. In *Information Processing: Proceedings of the International Conference on Information Processing, Paris*, pp. 125–132. UNESCO.

Berger, A., Della Pietra, S. A., and Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*(1), 39–71.

Bird, S. and Loper, E. (2004). NLTK: The Natural Language Toolkit. In *Proceedings of the ACL 2004 demonstration session*, Barcelona, Spain, pp. 214–217.

Bledsoe, W. W. and Browning, I. (1959). Pattern recognition and reading by machine. In *1959 Proceedings of the Eastern Joint Computer Conference*, pp. 225–232. Academic, New York.

Bresnan, J. and Kaplan, R. M. (1982). Introduction: Grammars as mental representations of language. In Bresnan, J. (Ed.), *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

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

Carlson, L., Marcu, D., and Okurowski, M. E. (2001). Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of SIGDIAL*.

Charniak, E. (1993). *Statistical Language Learning*. MIT Press.

Chomsky, N. (1956). Three models for the description of language. *IRI Transactions on Information Theory*, *2*(3), 113–124.

Chomsky, N. (1959). A review of B. F. Skinner's "Verbal Behavior". *Language*, *35*, 26–58.

Church, K. W. (1980). On memory limitations in natural language processing. Master's thesis, MIT. Distributed by the Indiana University Linguistics Club.

Cohen, P. R. and Perrault, C. R. (1979). Elements of a plan-based theory of speech acts. *Cognitive Science*, *3*(3), 177–212.

Colmerauer, A. (1970). Les systèmes-q ou un formalisme pour analyser et synthétiser des phrase sur ordinateur. Internal publication 43, Département d'informatique de l'Université de Montréal†.

Colmerauer, A. (1975). Les grammaires de métamorphose GIA. Internal publication, Groupe Intelligence artificielle, Faculté des Sciences de Luminy, Université Aix-Marseille II, France, Nov 1975. English version, Metamorphosis grammars. In L. Bolc, (Ed.), *Natural Language Communication with Computers*, *Lecture Notes in Computer Science 63*, Springer Verlag, Berlin, 1978, pp. 133–189.

Cullingford, R. E. (1981). SAM. In Schank, R. C. and Riesbeck, C. K. (Eds.), *Inside Computer Understanding: Five Programs plus Miniatures*, pp. 75–119. Lawrence Erlbaum, Hillsdale, NJ.

Davis, K. H., Biddulph, R., and Balashek, S. (1952). Automatic recognition of spoken digits. *Journal of the Acoustical Society of America*, *24*(6), 637–642.

Dejean, H. and Tjong Kim Sang, E. F. (2001). Introduction to the CoNLL-2001 shared task: Clause identification. In *Proceedings of CoNLL-2001*.

Fillmore, C. J. (1968). The case for case. In Bach, E. W. and Harms, R. T. (Eds.), *Universals in Linguistic Theory*, pp. 1–88. Holt, Rinehart & Winston, New York.

Francis, W. N. (1979). A tagged corpus – problems and prospects. In Greenbaum, S., Leech, G., and Svartvik, J. (Eds.), *Studies in English linguistics for Randolph Quirk*, pp. 192–209. Longman, London and New York.

Francis, W. N. and Kučera, H. (1982). *Frequency Analysis of English Usage*. Houghton Mifflin, Boston.

Gazdar, G. and Mellish, C. (1989). *Natural Language Processing in LISP*. Addison Wesley.

Grosz, B. J. (1977). The representation and use of focus in a system for understanding dialogs. In *IJCAI-77*, Cambridge, MA, pp. 67–76. Morgan Kaufmann. Reprinted in Grosz et al. (1986).

Grosz, B. J., Jones, K. S., and Webber, B. L. (Eds.). (1986). *Readings in Natural Language Processing*. Morgan Kaufmann, Los Altos, Calif.

Hajič, J. (1998). *Building a Syntactically Annotated Corpus: The Prague Dependency Treebank*, pp. 106–132. Karolinum, Prague/Praha.

Harris, Z. S. (1962). *String Analysis of Sentence Structure*. Mouton, The Hague.

Hobbs, J. R. (1978). Resolving pronoun references. *Lingua*, *44*, 311–338. Reprinted in Grosz et al. (1986).

Joshi, A. K. and Hopely, P. (1999). A parser from antiquity. In Kornai, A. (Ed.), *Extended Finite State Models of Language*, pp. 6–15. Cambridge University Press, Cambridge.

Kaplan, R. M. and Kay, M. (1981). Phonological rules and finite-state transducers. Paper presented at the Annual meeting of the Linguistics Society of America. New York.

Karttunen, L. (1999). Comments on Joshi. In Kornai, A. (Ed.), *Extended Finite State Models of Language*, pp. 16–18. Cambridge University Press, Cambridge.

Kay, M. (1979). Functional grammar. In *BLS-79*, Berkeley, CA, pp. 142–158.

Kilgarriff, A. and Palmer, M. (Eds.). (2000). *Computing and the Humanities: Special Issue on SENSEVAL*, Vol. 34. Kluwer.

Kintsch, W. (1974). *The Representation of Meaning in Memory*. Wiley, New York.

Kleene, S. C. (1951). Representation of events in nerve nets and finite automata. Tech. rep. RM-704, RAND Corporation. RAND Research Memorandum†.

Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In Shannon, C. and McCarthy, J. (Eds.), *Automata Studies*, pp. 3–41. Princeton University Press, Princeton, NJ.

Koenig, W., Dunn, H. K., Y., L., and Lacy (1946). The sound spectrograph. *Journal of the Acoustical Society of America*, *18*, 19–49.

Kučera, H. and Francis, W. N. (1967). *Computational analysis of present-day American English*. Brown University Press, Providence, RI.

Lehnert, W. G. (1977). A conceptual theory of question answering. In *IJCAI-77*, Cambridge, MA, pp. 158–164. Morgan Kaufmann.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, *19*(2), 313–330.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133. Reprinted in *Neurocomputing: Foundations of Research, ed. by J. A. Anderson and E Rosenfeld. MIT Press 1988.*

Merton, R. K. (1961). Singletons and multiples in scientific discovery. *American Philosophical Society Proceedings*, *105*(5), 470–486.

Miltsakaki, E., Prasad, R., Joshi, A. K., and Webber, B. L. (2004). The Penn Discourse Treebank. In *LREC-04*.

Mosteller, F. and Wallace, D. L. (1964). *Inference and Disputed Authorship: The Federalist*. Springer-Verlag, New York. 2nd Edition appeared in 1984 and was called *Applied Bayesian and Classical Inference.*

Naur, P., Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijnagaarden, A., and Woodger, M. (1960). Report on the algorithmic language ALGOL 60. *Communications of the ACM*, *3*(5), 299–314. Revised in CACM 6:1, 1-17, 1963.

Norman, D. A. and Rumelhart, D. E. (1975). *Explorations in Cognition*. Freeman, San Francisco, CA.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, *29*(1), 19–51.

Ogburn, W. F. and Thomas, D. S. (1922). Are inventions inevitable? A note on social evolution. *Political Science Quarterly*, *37*, 83–98.

Palmer, M., Fellbaum, C., Cotton, S., Delfs, L., and Dang, H. T. (2001). English tasks: All-words and verb lexical sample. In *Proceedings of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*, Toulouse, France.

Palmer, M., Kingsbury, P., and Gildea, D. (2005). The proposition bank: An annotated corpus of semantic roles.. *Computational Linguistics*, *31*(1), 71–106.

Partee, B. H., ter Meulen, A., and Wall, R. E. (1990). *Mathematical Methods in Linguistics*. Kluwer, Dordrecht.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, Ca.

Pereira, F. C. N. and Shieber, S. M. (1987). *Prolog and Natural-Language Analysis*, Vol. 10 of *CSLI Lecture Notes*. Chicago University Press, Chicago.

Pereira, F. C. N. and Warren, D. H. D. (1980). Definite clause grammars for language analysis— a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, *13*(3), 231–278.

Perrault, C. R. and Allen, J. (1980). A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, *6*(3-4), 167–182.

Quillian, M. R. (1968). Semantic memory. In Minsky, M. (Ed.), *Semantic Information Processing*, pp. 227–270. MIT Press, Cambridge, MA.

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

Reeves, B. and Nass, C. (1996). *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*. Cambridge University Press, Cambridge.

Russell, S. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ. Second edition.

Schank, R. C. (1972). Conceptual dependency: A theory of natural language processing. *Cognitive Psychology*, *3*, 552–631.

Schank, R. C. and Albelson, R. P. (1977). *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum, Hillsdale, NJ.

Schank, R. C. and Riesbeck, C. K. (Eds.). (1981). *Inside Computer Understanding: Five Programs plus Miniatures*. Lawrence Erlbaum, Hillsdale, NJ.

Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, *3*, 417–457.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379–423. Continued in following volume.

Shieber, S. M. (1994). Lessons from a restricted Turing test. *Communications of the ACM*, *37*(6), 70–78.

Sidner, C. L. (1983). Focusing in the comprehension of definite anaphora. In Brady, M. and Berwick, R. C. (Eds.), *Computational Models of Discourse*, pp. 267–330. MIT Press, Cambridge, MA.

Simmons, R. F. (1973). Semantic networks: Their computation and use for understanding English sentences. In Schank, R. C. and Colby, K. M. (Eds.), *Computer Models of Thought and Language*, pp. 61–113. W.H. Freeman and Co., San Francisco.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *42*, 230–265. Read to the Society in 1936, but published in 1937. Correction in volume 43, 544–546.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*, 433–460.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Voorhees, E. M. and Tice, D. M. (1999). The TREC-8 question answering track evaluation. Proceedings of the TREC-8 Workshop.

Weizenbaum, J. (1966). ELIZA – A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Weizenbaum, J. (1976). *Computer Power and Human Reason: From Judgement to Calculation*. W.H. Freeman and Company, San Francisco.

Wilensky, R. (1983). *Planning and Understanding*. Addison-Wesley, Reading, MA.

Wilks, Y. (1975a). Preference semantics. In Keenan, E. L. (Ed.), *The Formal Semantics of Natural Language*, pp. 329–350. Cambridge Univ. Press, Cambridge.

Wilks, Y. (1975b). A preferential, pattern-seeking, semantics for natural language inference. *Artificial Intelligence*, *6*(1), 53–74.

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, *3*(1), 1–191. Reprinted as a book by Academic Press, 1972.

Woods, W. A. (1967). *Semantics for a Question-Answering System*. Ph.D. thesis, Harvard University.

Woods, W. A. (1973). Progress in natural language understanding. In *Proceedings of AFIPS National Conference*, pp. 441–450.

CHAPTER

17 Sequence Labeling for Parts of Speech and Named Entities

To each word a warbling note
A Midsummer Night's Dream, V.I

Dionysius Thrax of Alexandria (*c.* 100 B.C.), or perhaps someone else (it was a long time ago), wrote a grammatical sketch of Greek (a "*technē*") that summarized the linguistic knowledge of his day. This work is the source of an astonishing proportion of modern linguistic vocabulary, including the words *syntax*, *diphthong*, *clitic*, and *analogy*. Also included are a description of eight **parts of speech**: noun, verb, pronoun, preposition, adverb, conjunction, participle, and article. Although earlier scholars (including Aristotle as well as the Stoics) had their own lists of parts of speech, it was Thrax's set of eight that became the basis for descriptions of European languages for the next 2000 years. (All the way to the *Schoolhouse Rock* educational television shows of our childhood, which had songs about 8 parts of speech, like the late great Bob Dorough's *Conjunction Junction*.) The durability of parts of speech through two millennia speaks to their centrality in models of human language.

parts of speech

Proper names are another important and anciently studied linguistic category. While parts of speech are generally assigned to individual words or morphemes, a proper name is often an entire multiword phrase, like the name "Marie Curie", the location "New York City", or the organization "Stanford University". We'll use the term **named entity** for, roughly speaking, anything that can be referred to with a proper name: a person, a location, an organization, although as we'll see the term is commonly extended to include things that aren't entities per se.

named entity

Parts of speech (also known as **POS**) and named entities are useful clues to sentence structure and meaning. Knowing whether a word is a noun or a verb tells us about likely neighboring words (nouns in English are preceded by determiners and adjectives, verbs by nouns) and syntactic structure (verbs have dependency links to nouns), making part-of-speech tagging a key aspect of parsing. Knowing if a named entity like *Washington* is a name of a person, a place, or a university is important to many natural language processing tasks like question answering, stance detection, or information extraction.

POS

In this chapter we'll introduce the task of **part-of-speech tagging**, taking a sequence of words and assigning each word a part of speech like NOUN or VERB, and the task of **named entity recognition** (**NER**), assigning words or phrases tags like PERSON, LOCATION, or ORGANIZATION.

Such tasks in which we assign, to each word x_i in an input word sequence, a label y_i, so that the output sequence Y has the same length as the input sequence X are called **sequence labeling** tasks. We'll introduce classic sequence labeling algorithms, one generative— the Hidden Markov Model (HMM)—and one discriminative—the Conditional Random Field (CRF). In following chapters we'll introduce modern sequence labelers based on RNNs and Transformers.

sequence labeling

17.1 (Mostly) English Word Classes

Until now we have been using part-of-speech terms like **noun** and **verb** rather freely. In this section we give more complete definitions. While word classes do have semantic tendencies—adjectives, for example, often describe *properties* and nouns *people*— parts of speech are defined instead based on their grammatical relationship with neighboring words or the morphological properties about their affixes.

	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red*, *young*, *awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very*, *slowly*, *home*, *yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm*, *cat*, *mango*, *beauty*
	VERB	words for actions and processes	*draw*, *provide*, *go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina*, *IBM*, *Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh*, *um*, *yes*, *hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by, under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and*, *or*, *but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, 2026, 11:00, hundred*
	PART	Particle: a function word that must be associated with another word	*'s, not, (infinitive) to*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*whether*, *because*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Figure 17.1 The 17 parts of speech in the Universal Dependencies tagset (de Marneffe et al., 2021). Features can be added to make finer-grained distinctions (with properties like number, case, definiteness, and so on).

closed class
open class

Parts of speech fall into two broad categories: **closed class** and **open class**. Closed classes are those with relatively fixed membership, such as prepositions—new prepositions are rarely coined. By contrast, nouns and verbs are open classes—new nouns and verbs like *iPhone* or *to fax* are continually being created or borrowed.

function word

Closed class words are generally **function words** like *of*, *it*, *and*, or *you*, which tend to be very short, occur frequently, and often have structuring uses in grammar.

Four major open classes occur in the languages of the world: **nouns** (including proper nouns), **verbs**, **adjectives**, and **adverbs**, as well as the smaller open class of **interjections**. English has all five, although not every language does.

noun
common noun

Nouns are words for people, places, or things, but include others as well. **Common nouns** include concrete terms like *cat* and *mango*, abstractions like *algorithm* and *beauty*, and verb-like terms like *pacing* as in *His pacing to and fro became quite annoying*. Nouns in English can occur with determiners (*a goat, this bandwidth*) take possessives (*IBM's annual revenue*), and may occur in the plural (*goats, abaci*).

count noun
mass noun

Many languages, including English, divide common nouns into **count nouns** and **mass nouns**. Count nouns can occur in the singular and plural (*goat/goats, relationship/relationships*) and can be counted (*one goat, two goats*). Mass nouns are used when something is conceptualized as a homogeneous group. So *snow, salt*, and *communism* are not counted (i.e., **two snows* or **two communisms*).

proper noun

Proper nouns, like *Regina*, *Colorado*, and *IBM*, are names of specific persons or entities.

verb **Verbs** refer to actions and processes, including main verbs like *draw*, *provide*, and *go*. English verbs have inflections (non-third-person-singular (*eat*), third-person-singular (*eats*), progressive (*eating*), past participle (*eaten*)). While many scholars believe that all human languages have the categories of noun and verb, others have argued that some languages, such as Riau Indonesian and Tongan, don't even make this distinction (Broschart 1997; Evans 2000; Gil 2000) .

adjective **Adjectives** often describe properties or qualities of nouns, like color (*white*, *black*), age (*old*, *young*), and value (*good*, *bad*), but there are languages without adjectives. In Korean, for example, the words corresponding to English adjectives act as a subclass of verbs, so what is in English an adjective "beautiful" acts in Korean like a verb meaning "to be beautiful".

adverb **Adverbs** are a hodge-podge. All the italicized words in this example are adverbs:

Actually, I ran *home extremely quickly yesterday*

Adverbs generally modify something (often verbs, hence the name "adverb", but also other adverbs and entire verb phrases). **Directional adverbs** or **locative adverbs** (*home*, *here*, *downhill*) specify the direction or location of some action; **degree adverbs** (*extremely*, *very*, *somewhat*) specify the extent of some action, process, or property; **manner adverbs** (*slowly*, *slinkily*, *delicately*) describe the manner of some action or process; and **temporal adverbs** describe the time that some action or event took place (*yesterday*, *Monday*).

locative degree manner temporal

interjection **Interjections** (*oh, hey, alas, uh, um*) are a smaller open class that also includes greetings (*hello, goodbye*) and question responses (*yes, no, uh-huh*).

preposition English adpositions occur before nouns, hence are called **prepositions**. They can indicate spatial or temporal relations, whether literal (*on it*, *before then*, *by the house*) or metaphorical (*on time*, *with gusto*, *beside herself*), and relations like marking the agent in *Hamlet was written by Shakespeare*.

particle A **particle** resembles a preposition or an adverb and is used in combination with a verb. Particles often have extended meanings that aren't quite the same as the prepositions they resemble, as in the particle *over* in *she turned the paper over*. A verb and a particle acting as a single unit is called a **phrasal verb**. The meaning of phrasal verbs is often **non-compositional**—not predictable from the individual meanings of the verb and the particle. Thus, *turn down* means 'reject', *rule out* 'eliminate', and *go on* 'continue'.

phrasal verb

determiner article **Determiners** like *this* and *that* (*this chapter*, *that page*) can mark the start of an English noun phrase. **Articles** like *a*, *an*, and *the*, are a type of determiner that mark discourse properties of the noun and are quite frequent; *the* is the most common word in written English, with *a* and *an* right behind.

conjunction **Conjunctions** join two phrases, clauses, or sentences. Coordinating conjunctions like *and*, *or*, and *but* join two elements of equal status. Subordinating conjunctions are used when one of the elements has some embedded status. For example, the subordinating conjunction *that* in *"I thought that you might like some milk"* links the main clause *I thought* with the subordinate clause *you might like some milk*. This clause is called subordinate because this entire clause is the "content" of the main verb *thought*. Subordinating conjunctions like *that* which link a verb to its argument in this way are also called **complementizers**.

complementizer

pronoun **Pronouns** act as a shorthand for referring to an entity or event. Personal pronouns refer to persons or entities (*you*, *she*, *I*, *it*, *me*, etc.). Possessive pronouns are forms of personal pronouns that indicate either actual possession or more often just an abstract relation between the person and some object (*my, your, his, her, its, one's, our, their*). **Wh-pronouns** (*what, who, whom, whoever*) are used in certain question

wh

forms, or act as complementizers (*Frida, who married Diego...*).

auxiliary

Auxiliary verbs mark semantic features of a main verb such as its tense, whether it is completed (aspect), whether it is negated (polarity), and whether an action is necessary, possible, suggested, or desired (mood). English auxiliaries include the **copula** verb *be*, the two verbs *do* and *have*, forms, as well as **modal verbs** used to mark the mood associated with the event depicted by the main verb: *can* indicates ability or possibility, *may* permission or possibility, *must* necessity.

copula

modal

An English-specific tagset, the Penn Treebank tagset (Marcus et al., 1993), shown in Fig. 17.2, has been used to label many syntactically annotated corpora like the Penn Treebank corpora, so it is worth knowing about.

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	infinitive to	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	+, %, &	WRB	wh-adverb	*how, where*

Figure 17.2 Penn Treebank core 36 part-of-speech tags.

Below we show some examples with each word tagged according to both the UD (in blue) and Penn (in red) tagsets. Notice that the Penn tagset distinguishes tense and participles on verbs, and has a special tag for the existential *there* construction in English. Note that since *London Journal of Medicine* is a proper noun, both tagsets mark its component nouns as PROPN/NNP, including *journal* and *medicine*, which might otherwise be labeled as common nouns (NOUN/NN).

(17.1) There/PRON/EX are/VERB/VBP 70/NUM/CD children/NOUN/NNS there/ADV/RB ./PUNC/.

(17.2) Preliminary/ADJ/JJ findings/NOUN/NNS were/AUX/VBD reported/VERB/VBN in/ADP/IN today/NOUN/NN 's/PART/POS London/PROPN/NNP Journal/PROPN/NNP of/ADP/IN Medicine/PROPN/NNP

17.2 Part-of-Speech Tagging

part-of-speech tagging

Part-of-speech tagging is the process of assigning a part-of-speech to each word in a text. The input is a sequence $x_1, x_2, ..., x_n$ of (tokenized) words and a tagset, and the output is a sequence $y_1, y_2, ..., y_n$ of tags, each output y_i corresponding exactly to one input x_i, as shown in the intuition in Fig. 17.3.

ambiguous

Tagging is a **disambiguation** task; words are **ambiguous** —have more than one possible part-of-speech—and the goal is to find the correct tag for the situation. For example, *book* can be a verb (*book that flight*) or a noun (*hand me that book*). *That* can be a determiner (*Does that flight serve dinner*) or a complementizer (*I*

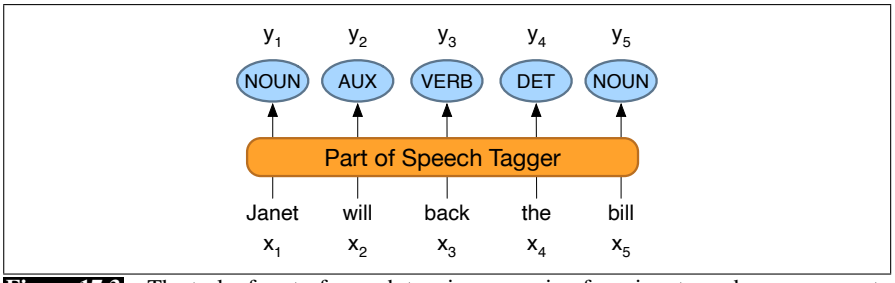


Figure 17.3 The task of part-of-speech tagging: mapping from input words $x_1, x_2, ..., x_n$ to output POS tags $y_1, y_2, ..., y_n$.

ambiguity resolution

thought that your flight was earlier). The goal of POS-tagging is to **resolve** these ambiguities, choosing the proper tag for the context.

accuracy

The **accuracy** of part-of-speech tagging algorithms (the percentage of test set tags that match human gold labels) is extremely high. One study found accuracies over 97% across 15 languages from the Universal Dependency (UD) treebank (Wu and Dredze, 2019). Accuracies on various English treebanks are also 97% (no matter the algorithm; HMMs, CRFs, BERT perform similarly). This 97% number is also about the human performance on this task, at least for English (Manning, 2011).

		WSJ	Brown
Types:			
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)
Tokens:			
Unambiguous	(1 tag)	577,421 (**45%**)	384,349 (**33%**)
Ambiguous	(2+ tags)	711,780 (**55%**)	786,646 (**67%**)

Figure 17.4 Tag ambiguity in the Brown and WSJ corpora (Treebank-3 45-tag tagset).

We'll introduce algorithms for the task in the next few sections, but first let's explore the task. Exactly how hard is it? Fig. 17.4 shows that most word types (85-86%) are unambiguous (*Janet* is always NNP, *hesitantly* is always RB). But the ambiguous words, though accounting for only 14-15% of the vocabulary, are very common, and 55-67% of word tokens in running text are ambiguous. Particularly ambiguous common words include *that*, *back*, *down*, *put* and *set*; here are some examples of the 6 different parts of speech for the word *back*:

earnings growth took a **back/JJ** seat
a small building in the **back/NN**
a clear majority of senators **back/VBP** the bill
Dave began to **back/VB** toward the door
enable the country to buy **back/RP** debt
I was twenty-one **back/RB** then

Nonetheless, many words are easy to disambiguate, because their different tags aren't equally likely. For example, *a* can be a determiner or the letter *a*, but the determiner sense is much more likely.

This idea suggests a useful **baseline**: given an ambiguous word, choose the tag which is **most frequent** in the training corpus. This is a key concept:

Most Frequent Class Baseline: Always compare a classifier against a baseline at least as good as the most frequent class baseline (assigning each token to the class it occurred in most often in the training set).

The most-frequent-tag baseline has an accuracy of about 92%[1]. The baseline thus differs from the state-of-the-art and human ceiling (97%) by only 5%.

17.3 Named Entities and Named Entity Tagging

Part of speech tagging can tell us that words like *Janet*, *Stanford University*, and *Colorado* are all proper nouns; being a proper noun is a grammatical property of these words. But viewed from a semantic perspective, these proper nouns refer to different kinds of entities: Janet is a person, Stanford University is an organization, and Colorado is a location.

named entity

Here we re-introduce the concept of a **named entity**, which was also introduced in Section **??** for readers who haven't yet read Chapter 10.

named entity

named entity recognition

NER

A **named entity** is, roughly speaking, anything that can be referred to with a proper name: a person, a location, an organization. The task of **named entity recognition** (**NER**) is to find spans of text that constitute proper names and tag the type of the entity. Four entity tags are most common: **PER** (person), **LOC** (location), **ORG** (organization), or **GPE** (geo-political entity). However, the term **named entity** is commonly extended to include things that aren't entities per se, including dates, times, and other kinds of temporal expressions, and even numerical expressions like prices. Here's an example of the output of an NER tagger:

> Citing high fuel prices, [ORG **United Airlines**] said [TIME **Friday**] it has increased fares by [MONEY **$6**] per round trip on flights to some cities also served by lower-cost carriers. [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said. [ORG **United**], a unit of [ORG **UAL Corp.**], said the increase took effect [TIME **Thursday**] and applies to most routes where it competes against discount carriers, such as [LOC **Chicago**] to [LOC **Dallas**] and [LOC **Denver**] to [LOC **San Francisco**].

The text contains 13 mentions of named entities including 5 organizations, 4 locations, 2 times, 1 person, and 1 mention of money. Figure 17.5 shows typical generic named entity types. Many applications will also need to use specific entity types like proteins, genes, commercial products, or works of art.

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	**Mt. Sanitas** is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states	**Palo Alto** is raising the fees for parking.

Figure 17.5 A list of generic named entity types with the kinds of entities they refer to.

Named entity tagging is a useful first step in lots of natural language processing tasks. In sentiment analysis we might want to know a consumer's sentiment toward a particular entity. Entities are a useful first stage in question answering, or for linking text to information in structured knowledge sources like Wikipedia. And named entity tagging is also central to tasks involving building semantic representations, like extracting events and the relationship between participants.

[1] In English, on the WSJ corpus, tested on sections 22-24.

Unlike part-of-speech tagging, where there is no segmentation problem since each word gets one tag, the task of named entity recognition is to find and label *spans* of text, and is difficult partly because of the ambiguity of segmentation; we need to decide what's an entity and what isn't, and where the boundaries are. Indeed, most words in a text will not be named entities. Another difficulty is caused by type ambiguity. The mention `JFK` can refer to a person, the airport in New York, or any number of schools, bridges, and streets around the United States. Some examples of this kind of cross-type confusion are given in Figure 17.6.

[PER Washington] was born into slavery on the farm of James Burroughs.
[ORG Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [LOC Washington] for what may well be his last state visit.
In June, [GPE Washington] passed a primary seatbelt law.

Figure 17.6 Examples of type ambiguities in the use of the name *Washington*.

The standard approach to sequence labeling for a span-recognition problem like NER is **BIO** tagging (Ramshaw and Marcus, 1995). This is a method that allows us to treat NER like a word-by-word sequence labeling task, via tags that capture both the boundary and the named entity type. Consider the following sentence:

[PER **Jane Villanueva**] of [ORG **United**] , a unit of [ORG **United Airlines Holding**] , said the fare applies to the [LOC **Chicago**] route.

BIO

Figure 17.7 shows the same excerpt represented with **BIO** tagging, as well as variants called **IO** tagging and **BIOES** tagging. In BIO tagging we label any token that *begins* a span of interest with the label B, tokens that occur *inside* a span are tagged with an I, and any tokens *outside* of any span of interest are labeled O. While there is only one O tag, we'll have distinct B and I tags for each named entity class. The number of tags is thus $2n+1$ tags, where n is the number of entity types. BIO tagging can represent exactly the same information as the bracketed notation, but has the advantage that we can represent the task in the same simple sequence modeling way as part-of-speech tagging: assigning a single label y_i to each input word x_i:

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Figure 17.7 NER as a sequence model, showing IO, BIO, and BIOES taggings.

We've also shown two variant tagging schemes: IO tagging, which loses some information by eliminating the B tag, and BIOES tagging, which adds an end tag E for the end of a span, and a span tag S for a span consisting of only one word. A sequence labeler (HMM, CRF, RNN, Transformer, etc.) is trained to label each token in a text with tags that indicate the presence (or absence) of particular kinds of named entities.

Abney, S. P., R. E. Schapire, and Y. Singer. 1999. Boosting applied to tagging and PP attachment. *EMNLP/VLC*.

Bada, M., M. Eckert, D. Evans, K. Garcia, K. Shipley, D. Sitnikov, W. A. Baumgartner, K. B. Cohen, K. Verspoor, J. A. Blake, and L. E. Hunter. 2012. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):161.

Bahl, L. R. and R. L. Mercer. 1976. Part of speech assignment by a statistical decision algorithm. *Proceedings IEEE International Symposium on Information Theory*.

Bamman, D., S. Popat, and S. Shen. 2019. An annotated dataset of literary entities. *NAACL HLT*.

Bikel, D. M., S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: A high-performance learning name-finder. *ANLP*.

Brants, T. 2000. TnT: A statistical part-of-speech tagger. *ANLP*.

Broschart, J. 1997. Why Tongan does it differently. *Linguistic Typology*, 1:123–165.

Charniak, E., C. Hendrickson, N. Jacobson, and M. Perkowitz. 1993. Equations for part-of-speech tagging. *AAAI*.

Chiticariu, L., M. Danilevsky, Y. Li, F. Reiss, and H. Zhu. 2018. SystemT: Declarative text understanding for enterprise. *NAACL HLT*, volume 3.

Chiticariu, L., Y. Li, and F. R. Reiss. 2013. Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems! *EMNLP*.

Christodoulopoulos, C., S. Goldwater, and M. Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? *EMNLP*.

Church, K. W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. *ANLP*.

Church, K. W. 1989. A stochastic parts program and noun phrase parser for unrestricted text. *ICASSP*.

Clark, S., J. R. Curran, and M. Osborne. 2003. Bootstrapping POS-taggers using unlabelled data. *CoNLL*.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

DeRose, S. J. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39.

Evans, N. 2000. Word classes in the world's languages. In G. Booij, C. Lehmann, and J. Mugdan, eds, *Morphology: A Handbook on Inflection and Word Formation*, 708–732. Mouton.

Francis, W. N. and H. Kučera. 1982. *Frequency Analysis of English Usage*. Houghton Mifflin, Boston.

Garside, R. 1987. The CLAWS word-tagging system. In R. Garside, G. Leech, and G. Sampson, eds, *The Computational Analysis of English*, 30–41. Longman.

Garside, R., G. Leech, and A. McEnery. 1997. *Corpus Annotation*. Longman.

Gil, D. 2000. Syntactic categories, cross-linguistic variation and universal grammar. In P. M. Vogel and B. Comrie, eds, *Approaches to the Typology of Word Classes*, 173–216. Mouton.

Greene, B. B. and G. M. Rubin. 1971. Automatic grammatical tagging of English. Department of Linguistics, Brown University, Providence, Rhode Island.

Hajič, J. 2000. Morphological tagging: Data vs. dictionaries. *NAACL*.

Hakkani-Tür, D., K. Oflazer, and G. Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Journal of Computers and Humanities*, 36(4):381–410.

Harris, Z. S. 1962. *String Analysis of Sentence Structure*. Mouton, The Hague.

Householder, F. W. 1995. Dionysius Thrax, the *technai*, and Sextus Empiricus. In E. F. K. Koerner and R. E. Asher, eds, *Concise History of the Language Sciences*, 99–103. Elsevier Science.

Hovy, E. H., M. P. Marcus, M. Palmer, L. A. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90% solution. *HLT-NAACL*.

Huang, Z., W. Xu, and K. Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Joshi, A. K. and P. Hopely. 1999. A parser from antiquity. In A. Kornai, ed., *Extended Finite State Models of Language*, 6–15. Cambridge University Press.

Karlsson, F., A. Voutilainen, J. Heikkilä, and A. Anttila, eds. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Karttunen, L. 1999. Comments on Joshi. In A. Kornai, ed., *Extended Finite State Models of Language*, 16–18. Cambridge University Press.

Klein, S. and R. F. Simmons. 1963. A computational approach to grammatical coding of English words. *Journal of the ACM*, 10(3):334–347.

Kupiec, J. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242.

Lafferty, J. D., A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Lample, G., M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. 2016. Neural architectures for named entity recognition. *NAACL HLT*.

Lee, H., M. Surdeanu, and D. Jurafsky. 2017. A scaffolding approach to coreference resolution integrating statistical and rule-based models. *Natural Language Engineering*, 23(5):733–762.

Ma, X. and E. H. Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *ACL*.

Manning, C. D. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? *CICLing 2011*.

Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

de Marneffe, M.-C., C. D. Manning, J. Nivre, and D. Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Marshall, I. 1983. Choice of grammatical word-class without global syntactic analysis: Tagging words in the LOB corpus. *Computers and the Humanities*, 17:139–150.

Marshall, I. 1987. Tag selection using probabilistic methods. In R. Garside, G. Leech, and G. Sampson, eds, *The Computational Analysis of English*, 42–56. Longman.

McCallum, A., D. Freitag, and F. C. N. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. *ICML*.

McCallum, A. and W. Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. *CoNLL*.

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

Mikheev, A., M. Moens, and C. Grover. 1999. Named entity recognition without gazetteers. *EACL*.

Oravecz, C. and P. Dienes. 2002. Efficient stochastic part-of-speech tagging for Hungarian. *LREC*.

Ramshaw, L. A. and M. P. Marcus. 1995. Text chunking using transformation-based learning. *Proceedings of the 3rd Annual Workshop on Very Large Corpora*.

Ratnaparkhi, A. 1996. A maximum entropy part-of-speech tagger. *EMNLP*.

Sampson, G. 1987. Alternative grammatical coding systems. In R. Garside, G. Leech, and G. Sampson, eds, *The Computational Analysis of English*, 165–183. Longman.

Schütze, H. and Y. Singer. 1994. Part-of-speech tagging using a variable memory Markov model. *ACL*.

Søgaard, A. 2010. Simple semi-supervised training of part-of-speech taggers. *ACL*.

Stolz, W. S., P. H. Tannenbaum, and F. V. Carstensen. 1965. A stochastic approach to the grammatical coding of English. *CACM*, 8(6):399–405.

Thede, S. M. and M. P. Harper. 1999. A second-order hidden Markov model for part-of-speech tagging. *ACL*.

Toutanova, K., D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. *HLT-NAACL*.

Voutilainen, A. 1999. Handcrafted rules. In H. van Halteren, ed., *Syntactic Wordclass Tagging*, 217–246. Kluwer.

Weischedel, R., M. Meteer, R. Schwartz, L. A. Ramshaw, and J. Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382.

Wu, S. and M. Dredze. 2019. Beto, Bentz, Becas: The surprising cross-lingual effectiveness of BERT. *EMNLP*.

Zhou, G., J. Su, J. Zhang, and M. Zhang. 2005. Exploring various knowledge in relation extraction. *ACL*.

CHAPTER

18 Context-Free Grammars and Constituency Parsing

Because the Night by Bruce Springsteen and Patti Smith
The Fire Next Time by James Baldwin
If on a winter's night a traveler by Italo Calvino
Love Actually by Richard Curtis
Suddenly Last Summer by Tennessee Williams
A Scanner Darkly by Philip K. Dick

Six titles that are not constituents, from Geoffrey K. Pullum on Language Log (who was pointing out their incredible rarity).

One morning I shot an elephant in my pajamas.
How he got into my pajamas I don't know.

Groucho Marx, *Animal Crackers*, 1930

The study of grammar has an ancient pedigree. The grammar of Sanskrit was described by the Indian grammarian Pāṇini sometime between the 7th and 4th centuries BCE, in his famous treatise the Aṣṭādhyāyī ('8 books'). And our word **syntax** comes from the Greek *sýntaxis*, meaning "setting out together or arrangement", and refers to the way words are arranged together. We have seen syntactic notions in previous chapters like the use of part-of-speech categories (Chapter 17). In this chapter and the next one we introduce formal models for capturing more sophisticated notions of grammatical structure and algorithms for parsing these structures.

syntax

Our focus in this chapter is **context-free grammars** and the **CKY algorithm** for parsing them. Context-free grammars are the backbone of many formal models of the syntax of natural language (and, for that matter, of computer languages). Syntactic parsing is the task of assigning a syntactic structure to a sentence. Parse trees (whether for context-free grammars or for the dependency or CCG formalisms we introduce in following chapters) can be used in applications such as **grammar checking**: sentence that cannot be parsed may have grammatical errors (or at least be hard to read). Parse trees can be an intermediate stage of representation for **formal semantic analysis**. And parsers and the grammatical structure they assign a sentence are a useful text analysis tool for text data science applications that require modeling the relationship of elements in sentences.

In this chapter we introduce context-free grammars, give a small sample grammar of English, introduce more formal definitions of context-free grammars and grammar normal form, and talk about **treebanks**: corpora that have been annotated with syntactic structure. We then discuss parse ambiguity and the problems it presents, and turn to parsing itself, giving the famous Cocke-Kasami-Younger (CKY) algorithm (Kasami 1965, Younger 1967), the standard dynamic programming approach to syntactic parsing. The CKY algorithm returns an efficient representation of the set of parse trees for a sentence, but doesn't tell us **which** parse tree is the right one. For that, we need to augment CKY with scores for each possible constituent. We'll see how to do this with neural span-based parsers. Finally, we'll introduce the standard set of metrics for evaluating parser accuracy.

18.1 Constituency

Syntactic constituency is the idea that groups of words can behave as single units, or constituents. Part of developing a grammar involves building an inventory of the constituents in the language. How do words group together in English? Consider the **noun phrase**, a sequence of words surrounding at least one noun. Here are some examples of noun phrases (thanks to Damon Runyon):

noun phrase

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

What evidence do we have that these words group together (or "form constituents")? One piece of evidence is that they can all appear in similar syntactic environments, for example, before a verb.

three parties from Brooklyn *arrive...*
a high-class spot such as Mindy's *attracts...*
the Broadway coppers *love...*
they *sit*

But while the whole noun phrase can occur before a verb, this is not true of each of the individual words that make up a noun phrase. The following are not grammatical sentences of English (recall that we use an asterisk (*) to mark fragments that are not grammatical English sentences):

*from *arrive...*	*as *attracts...*
*the *is...*	*spot *sat...*

Thus, to correctly describe facts about the ordering of these words in English, we must be able to say things like "*Noun Phrases can occur before verbs*". Let's now see how to do this in a more formal way!

18.2 Context-Free Grammars

A widely used formal system for modeling constituent structure in natural language is the **context-free grammar**, or **CFG**. Context-free grammars are also called **phrase-structure grammars**, and the formalism is equivalent to **Backus-Naur form**, or **BNF**. The idea of basing a grammar on constituent structure dates back to the psychologist Wilhelm Wundt (1900) but was not formalized until Chomsky (1956) and, independently, Backus (1959).

CFG

A context-free grammar consists of a set of **rules** or **productions**, each of which expresses the ways that symbols of the language can be grouped and ordered together, and a **lexicon** of words and symbols. For example, the following productions express that an **NP** (or **noun phrase**) can be composed of either a *ProperNoun* or a determiner (*Det*) followed by a *Nominal*; a *Nominal* in turn can consist of one or

rules
lexicon
NP

more *Nouns*.[1]

$$\begin{aligned} NP &\rightarrow Det\ Nominal \\ NP &\rightarrow ProperNoun \\ Nominal &\rightarrow Noun \mid Nominal\ Noun \end{aligned}$$

Context-free rules can be hierarchically embedded, so we can combine the previous rules with others, like the following, that express facts about the lexicon:

$$\begin{aligned} Det &\rightarrow a \\ Det &\rightarrow the \\ Noun &\rightarrow flight \end{aligned}$$

terminal

non-terminal

The symbols that are used in a CFG are divided into two classes. The symbols that correspond to words in the language ("the", "nightclub") are called **terminal** symbols; the lexicon is the set of rules that introduce these terminal symbols. The symbols that express abstractions over these terminals are called **non-terminals**. In each context-free rule, the item to the right of the arrow ($\rightarrow$) is an ordered list of one or more terminals and non-terminals; to the left of the arrow is a single non-terminal symbol expressing some cluster or generalization. The non-terminal associated with each word in the lexicon is its lexical category, or part of speech.

A CFG can be thought of in two ways: as a device for generating sentences and as a device for assigning a structure to a given sentence. Viewing a CFG as a generator, we can read the $\rightarrow$ arrow as "rewrite the symbol on the left with the string of symbols on the right".

So starting from the symbol:	*NP*
we can use our first rule to rewrite *NP* as:	*Det Nominal*
and then rewrite *Nominal* as:	*Noun*
and finally rewrite these parts-of-speech as:	*a flight*

We say the string *a flight* can be derived from the non-terminal *NP*. Thus, a CFG can be used to generate a set of strings. This sequence of rule expansions is called a **derivation** of the string of words. It is common to represent a derivation by a **parse tree** (commonly shown inverted with the root at the top). Figure 18.1 shows the tree representation of this derivation.

derivation

parse tree

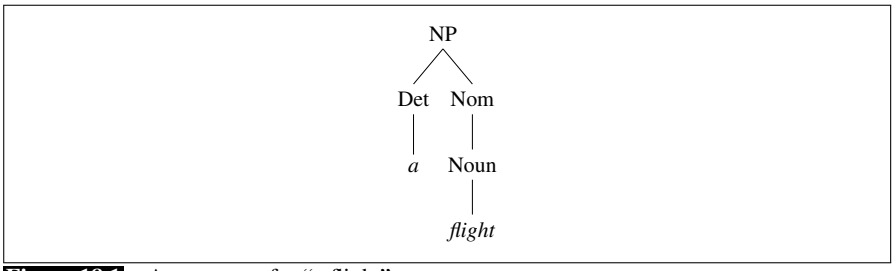


Figure 18.1 A parse tree for "a flight".

dominates

In the parse tree shown in Fig. 18.1, we can say that the node *NP* **dominates** all the nodes in the tree (*Det*, *Nom*, *Noun*, *a*, *flight*). We can say further that it immediately dominates the nodes *Det* and *Nom*.

start symbol

The formal language defined by a CFG is the set of strings that are derivable from the designated **start symbol**. Each grammar must have one designated start

[1] When talking about these rules we can pronounce the rightarrow $\rightarrow$ as "goes to", and so we might read the first rule above as "NP goes to Det Nominal".

symbol, which is often called *S*. Since context-free grammars are often used to define sentences, *S* is usually interpreted as the "sentence" node, and the set of strings that are derivable from *S* is the set of sentences in some simplified version of English.

verb phrase

Let's add a few additional rules to our inventory. The following rule expresses the fact that a sentence can consist of a noun phrase followed by a **verb phrase**:

$$S \rightarrow NP\ VP \quad \text{I prefer a morning flight}$$

A verb phrase in English consists of a verb followed by assorted other things; for example, one kind of verb phrase consists of a verb followed by a noun phrase:

$$VP \rightarrow Verb\ NP \quad \text{prefer a morning flight}$$

Or the verb may be followed by a noun phrase and a prepositional phrase:

$$VP \rightarrow Verb\ NP\ PP \quad \text{leave Boston in the morning}$$

Or the verb phrase may have a verb followed by a prepositional phrase alone:

$$VP \rightarrow Verb\ PP \quad \text{leaving on Thursday}$$

A prepositional phrase generally has a preposition followed by a noun phrase. For example, a common type of prepositional phrase in the ATIS corpus is used to indicate location or direction:

$$PP \rightarrow Preposition\ NP \quad \text{from Los Angeles}$$

The *NP* inside a *PP* need not be a location; *PPs* are often used with times and dates, and with other nouns as well; they can be arbitrarily complex. Here are ten examples from the ATIS corpus:

to Seattle	on these flights
in Minneapolis	about the ground transportation in Chicago
on Wednesday	of the round trip flight on United Airlines
in the evening	of the AP fifty seven flight
on the ninth of July	with a stopover in Nashville

Figure 18.2 gives a sample lexicon, and Fig. 18.3 summarizes the grammar rules we've seen so far, which we'll call $\mathscr{L}_0$. Note that we can use the or-symbol | to indicate that a non-terminal has alternate possible expansions.

Noun →	*flights* | *flight* | *breeze* | *trip* | *morning*
Verb →	*is* | *prefer* | *like* | *need* | *want* | *fly* | *do*
Adjective →	*cheapest* | *non-stop* | *first* | *latest* | *other* | *direct*
Pronoun →	*me* | *I* | *you* | *it*
Proper-Noun →	*Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American*
Determiner →	*the* | *a* | *an* | *this* | *these* | *that*
Preposition →	*from* | *to* | *on* | *near* | *in*
Conjunction →	*and* | *or* | *but*

Figure 18.2 The lexicon for $\mathscr{L}_0$.

We can use this grammar to generate sentences of this "ATIS-language". We start with *S*, expand it to *NP VP*, then choose a random expansion of *NP* (let's say, to

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

Figure 18.3 The grammar for $\mathscr{L}_0$, with example phrases for each rule.

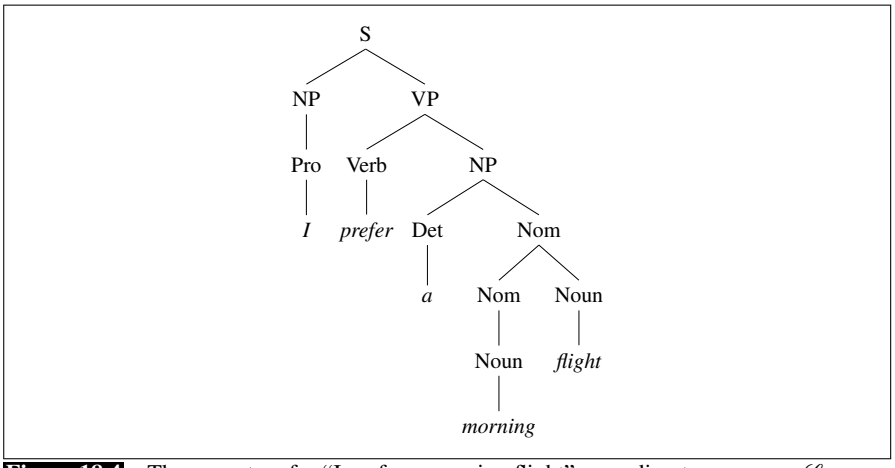


Figure 18.4 The parse tree for "I prefer a morning flight" according to grammar $\mathscr{L}_0$.

I), and a random expansion of *VP* (let's say, to *Verb NP*), and so on until we generate the string *I prefer a morning flight*. Figure 18.4 shows a parse tree that represents a complete derivation of *I prefer a morning flight*.

bracketed notation

We can also represent a parse tree in a more compact format called **bracketed notation**; here is the bracketed representation of the parse tree of Fig. 18.4:

(18.1) [$_S$ [$_{NP}$ [$_{Pro}$ I]] [$_{VP}$ [$_V$ prefer] [$_{NP}$ [$_{Det}$ a] [$_{Nom}$ [$_N$ morning] [$_{Nom}$ [$_N$ flight]]]]]]

grammatical

ungrammatical

A CFG like that of $\mathscr{L}_0$ defines a formal language. Sentences (strings of words) that can be derived by a grammar are in the formal language defined by that grammar, and are called **grammatical** sentences. Sentences that cannot be derived by a given formal grammar are not in the language defined by that grammar and are referred to as **ungrammatical**. This hard line between "in" and "out" characterizes all formal languages but is only a very simplified model of how natural languages really work. This is because determining whether a given sentence is part of a given natural language (say, English) often depends on the context. In linguistics, the use of formal languages to model natural languages is called **generative grammar** since the language is defined by the set of possible sentences "generated" by the grammar. (Note that this is a different sense of the word 'generate' than when we talk about

generative grammar

Abney, S. P., R. E. Schapire, and Y. Singer. 1999. Boosting applied to tagging and PP attachment. *EMNLP/VLC*.

Bada, M., M. Eckert, D. Evans, K. Garcia, K. Shipley, D. Sitnikov, W. A. Baumgartner, K. B. Cohen, K. Verspoor, J. A. Blake, and L. E. Hunter. 2012. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):161.

Bahl, L. R. and R. L. Mercer. 1976. Part of speech assignment by a statistical decision algorithm. *Proceedings IEEE International Symposium on Information Theory*.

Bamman, D., S. Popat, and S. Shen. 2019. An annotated dataset of literary entities. *NAACL HLT*.

Bikel, D. M., S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: A high-performance learning name-finder. *ANLP*.

Brants, T. 2000. TnT: A statistical part-of-speech tagger. *ANLP*.

Broschart, J. 1997. Why Tongan does it differently. *Linguistic Typology*, 1:123–165.

Charniak, E., C. Hendrickson, N. Jacobson, and M. Perkowitz. 1993. Equations for part-of-speech tagging. *AAAI*.

Chiticariu, L., M. Danilevsky, Y. Li, F. Reiss, and H. Zhu. 2018. SystemT: Declarative text understanding for enterprise. *NAACL HLT*, volume 3.

Chiticariu, L., Y. Li, and F. R. Reiss. 2013. Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems! *EMNLP*.

Christodoulopoulos, C., S. Goldwater, and M. Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? *EMNLP*.

Church, K. W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. *ANLP*.

Church, K. W. 1989. A stochastic parts program and noun phrase parser for unrestricted text. *ICASSP*.

Clark, S., J. R. Curran, and M. Osborne. 2003. Bootstrapping POS-taggers using unlabelled data. *CoNLL*.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

DeRose, S. J. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39.

Evans, N. 2000. Word classes in the world's languages. In G. Booij, C. Lehmann, and J. Mugdan, eds, *Morphology: A Handbook on Inflection and Word Formation*, 708–732. Mouton.

Francis, W. N. and H. Kučera. 1982. *Frequency Analysis of English Usage*. Houghton Mifflin, Boston.

Garside, R. 1987. The CLAWS word-tagging system. In R. Garside, G. Leech, and G. Sampson, eds, *The Computational Analysis of English*, 30–41. Longman.

Garside, R., G. Leech, and A. McEnery. 1997. *Corpus Annotation*. Longman.

Gil, D. 2000. Syntactic categories, cross-linguistic variation and universal grammar. In P. M. Vogel and B. Comrie, eds, *Approaches to the Typology of Word Classes*, 173–216. Mouton.

Greene, B. B. and G. M. Rubin. 1971. Automatic grammatical tagging of English. Department of Linguistics, Brown University, Providence, Rhode Island.

Hajič, J. 2000. Morphological tagging: Data vs. dictionaries. *NAACL*.

Hakkani-Tür, D., K. Oflazer, and G. Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Journal of Computers and Humanities*, 36(4):381–410.

Harris, Z. S. 1962. *String Analysis of Sentence Structure*. Mouton, The Hague.

Householder, F. W. 1995. Dionysius Thrax, the *technai*, and Sextus Empiricus. In E. F. K. Koerner and R. E. Asher, eds, *Concise History of the Language Sciences*, 99–103. Elsevier Science.

Hovy, E. H., M. P. Marcus, M. Palmer, L. A. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90% solution. *HLT-NAACL*.

Huang, Z., W. Xu, and K. Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Joshi, A. K. and P. Hopely. 1999. A parser from antiquity. In A. Kornai, ed., *Extended Finite State Models of Language*, 6–15. Cambridge University Press.

Karlsson, F., A. Voutilainen, J. Heikkilä, and A. Anttila, eds. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Karttunen, L. 1999. Comments on Joshi. In A. Kornai, ed., *Extended Finite State Models of Language*, 16–18. Cambridge University Press.

Klein, S. and R. F. Simmons. 1963. A computational approach to grammatical coding of English words. *Journal of the ACM*, 10(3):334–347.

Kupiec, J. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242.

Lafferty, J. D., A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Lample, G., M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. 2016. Neural architectures for named entity recognition. *NAACL HLT*.

Lee, H., M. Surdeanu, and D. Jurafsky. 2017. A scaffolding approach to coreference resolution integrating statistical and rule-based models. *Natural Language Engineering*, 23(5):733–762.

Ma, X. and E. H. Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *ACL*.

Manning, C. D. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? *CICLing 2011*.

Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

de Marneffe, M.-C., C. D. Manning, J. Nivre, and D. Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Marshall, I. 1983. Choice of grammatical word-class without global syntactic analysis: Tagging words in the LOB corpus. *Computers and the Humanities*, 17:139–150.

Marshall, I. 1987. Tag selection using probabilistic methods. In R. Garside, G. Leech, and G. Sampson, eds, *The Computational Analysis of English*, 42–56. Longman.

McCallum, A., D. Freitag, and F. C. N. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. *ICML*.

McCallum, A. and W. Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. *CoNLL*.

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

Mikheev, A., M. Moens, and C. Grover. 1999. Named entity recognition without gazetteers. *EACL*.

Oravecz, C. and P. Dienes. 2002. Efficient stochastic part-of-speech tagging for Hungarian. *LREC*.

Ramshaw, L. A. and M. P. Marcus. 1995. Text chunking using transformation-based learning. *Proceedings of the 3rd Annual Workshop on Very Large Corpora*.

Ratnaparkhi, A. 1996. A maximum entropy part-of-speech tagger. *EMNLP*.

Sampson, G. 1987. Alternative grammatical coding systems. In R. Garside, G. Leech, and G. Sampson, eds, *The Computational Analysis of English*, 165–183. Longman.

Schütze, H. and Y. Singer. 1994. Part-of-speech tagging using a variable memory Markov model. *ACL*.

Søgaard, A. 2010. Simple semi-supervised training of part-of-speech taggers. *ACL*.

Stolz, W. S., P. H. Tannenbaum, and F. V. Carstensen. 1965. A stochastic approach to the grammatical coding of English. *CACM*, 8(6):399–405.

Thede, S. M. and M. P. Harper. 1999. A second-order hidden Markov model for part-of-speech tagging. *ACL*.

Toutanova, K., D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. *HLT-NAACL*.

Voutilainen, A. 1999. Handcrafted rules. In H. van Halteren, ed., *Syntactic Wordclass Tagging*, 217–246. Kluwer.

Weischedel, R., M. Meteer, R. Schwartz, L. A. Ramshaw, and J. Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382.

Wu, S. and M. Dredze. 2019. Beto, Bentz, Becas: The surprising cross-lingual effectiveness of BERT. *EMNLP*.

Zhou, G., J. Su, J. Zhang, and M. Zhang. 2005. Exploring various knowledge in relation extraction. *ACL*.

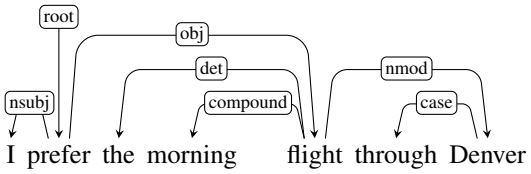
CHAPTER

19 Dependency Parsing

> Tout mot qui fait partie d'une phrase... Entre lui et ses voisins, l'esprit aperçoit des connexions, dont l'ensemble forme la charpente de la phrase.
>
> [Between each word in a sentence and its neighbors, the mind perceives **connections**. These connections together form the scaffolding of the sentence.]
>
> Lucien Tesnière. 1959. Éléments de syntaxe structurale, A.1.§4

dependency grammars

The focus of the last chapter was on context-free grammars and constituent-based representations. Here we present another important family of grammar formalisms called **dependency grammars**. In dependency formalisms, phrasal constituents and phrase-structure rules do not play a direct role. Instead, the syntactic structure of a sentence is described solely in terms of directed binary grammatical relations between the *words*, as in the following dependency parse:



(19.1)

typed dependency

Relations among the words are illustrated above the sentence with directed, labeled arcs from **heads** to **dependents**. We call this a **typed dependency structure** because the labels are drawn from a fixed inventory of grammatical relations. A *root* node explicitly marks the root of the tree, the head of the entire structure.

Figure 19.1 on the next page shows the dependency analysis from Eq. 19.1 but visualized as a tree, alongside its corresponding phrase-structure analysis of the kind given in the prior chapter. Note the absence of nodes corresponding to phrasal constituents or lexical categories in the dependency parse; the internal structure of the dependency parse consists solely of directed relations between words. These head-dependent relationships directly encode important information that is often buried in the more complex phrase-structure parses. For example, the arguments to the verb *prefer* are directly linked to it in the dependency structure, while their connection to the main verb is more distant in the phrase-structure tree. Similarly, *morning* and *Denver*, modifiers of *flight*, are linked to it directly in the dependency structure. This fact that the head-dependent relations are a good proxy for the semantic relationship between predicates and their arguments is an important reason why dependency grammars are currently more common than constituency grammars in natural language processing.

free word order

Another major advantage of dependency grammars is their ability to deal with languages that have a relatively **free word order**. For example, word order in Czech can be much more flexible than in English; a grammatical *object* might occur before or after a *location adverbial*. A phrase-structure grammar would need a separate rule

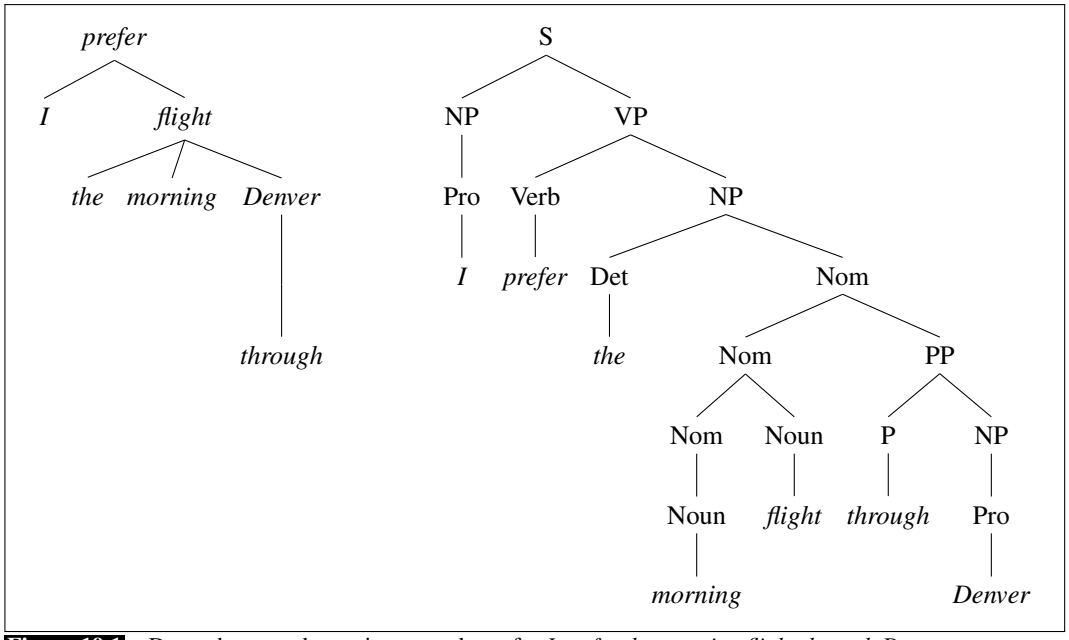


Figure 19.1 Dependency and constituent analyses for *I prefer the morning flight through Denver.*

for each possible place in the parse tree where such an adverbial phrase could occur. A dependency-based approach can have just one link type representing this particular adverbial relation; dependency grammar approaches can thus abstract away a bit more from word order information.

In the following sections, we'll give an inventory of relations used in dependency parsing, discuss two families of parsing algorithms (transition-based, and graph-based), and discuss evaluation.

19.1 Dependency Relations

grammatical relation

head

dependent

The traditional linguistic notion of **grammatical relation** provides the basis for the binary relations that comprise these dependency structures. The arguments to these relations consist of a **head** and a **dependent**. The head plays the role of the central organizing word, and the dependent as a kind of modifier. The head-dependent relationship is made explicit by directly linking heads to the words that are immediately dependent on them.

grammatical function

In addition to specifying the head-dependent pairs, dependency grammars allow us to classify the kinds of grammatical relations, or **grammatical function** that the dependent plays with respect to its head. These include familiar notions such as *subject*, *direct object* and *indirect object*. In English these notions strongly correlate with, but by no means determine, both position in a sentence and constituent type and are therefore somewhat redundant with the kind of information found in phrase-structure trees. However, in languages with more flexible word order, the information encoded directly in these grammatical relations is critical since phrase-based constituent syntax provides little help.

Linguists have developed taxonomies of relations that go well beyond the familiar notions of subject and object. While there is considerable variation from theory

Clausal Argument Relations	Description
NSUBJ	Nominal subject
OBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

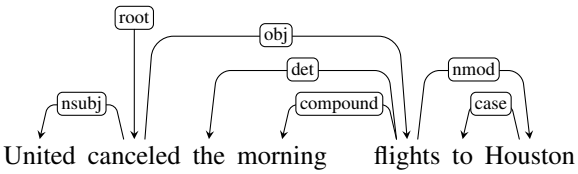
Figure 19.2 Some of the Universal Dependency relations (de Marneffe et al., 2021).

Universal Dependencies

to theory, there is enough commonality that cross-linguistic standards have been developed. The **Universal Dependencies** (UD) project (de Marneffe et al., 2021), an open community effort to annotate dependencies and other aspects of grammar across more than 100 languages, provides an inventory of 37 dependency relations. Fig. 19.2 shows a subset of the UD relations and Fig. 19.3 provides some examples.

The motivation for all of the relations in the Universal Dependency scheme is beyond the scope of this chapter, but the core set of frequently used relations can be broken into two sets: clausal relations that describe syntactic roles with respect to a predicate (often a verb), and modifier relations that categorize the ways that words can modify their heads.

Consider, for example, the following sentence:



(19.2)

Here the clausal relations NSUBJ and OBJ identify the subject and direct object of the predicate *cancel*, while the NMOD, DET, and CASE relations denote modifiers of the nouns *flights* and *Houston*.

19.1.1 Dependency Formalisms

A dependency structure can be represented as a directed graph $G = (V, A)$, consisting of a set of vertices V, and a set of ordered pairs of vertices A, which we'll call arcs.

For the most part we will assume that the set of vertices, V, corresponds exactly to the set of words in a given sentence. However, they might also correspond to punctuation, or when dealing with morphologically complex languages the set of vertices might consist of stems and affixes. The set of arcs, A, captures the head-dependent and grammatical function relationships between the elements in V.

Different grammatical theories or formalisms may place further constraints on these dependency structures. Among the more frequent restrictions are that the structures must be connected, have a designated root node, and be acyclic or planar. Of most relevance to the parsing approaches discussed in this chapter is the common,

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
OBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
COMPOUND	We took the **morning** *flight*.
NMOD	*flight* to **Houston**.
AMOD	Book the **cheapest** *flight*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Figure 19.3 Examples of some Universal Dependency relations.

dependency tree

computationally-motivated, restriction to rooted trees. That is, a **dependency tree** is a directed graph that satisfies the following constraints:

1. There is a single designated root node that has no incoming arcs.
2. With the exception of the root node, each vertex has exactly one incoming arc.
3. There is a unique path from the root node to each vertex in V.

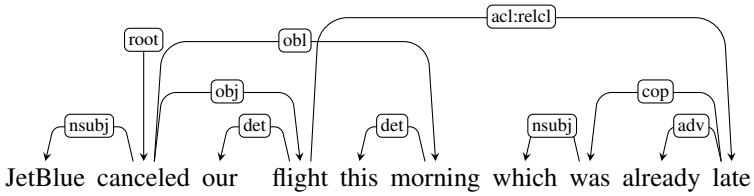
Taken together, these constraints ensure that each word has a single head, that the dependency structure is connected, and that there is a single root node from which one can follow a unique directed path to each of the words in the sentence.

19.1.2 Projectivity

projective

The notion of projectivity imposes an additional constraint that is derived from the order of the words in the input. An arc from a head to a dependent is said to be **projective** if there is a path from the head to every word that lies between the head and the dependent in the sentence. A dependency tree is then said to be projective if all the arcs that make it up are projective. All the dependency trees we've seen thus far have been projective. There are, however, many valid constructions which lead to non-projective trees, particularly in languages with relatively flexible word order.

Consider the following example.



(19.3)

In this example, the arc from *flight* to its modifier *late* is non-projective since there is no path from *flight* to the intervening words *this* and *morning*. As we can see from this diagram, projectivity (and non-projectivity) can be detected in the way we've been drawing our trees. A dependency tree is projective if it can be drawn with no crossing edges. Here there is no way to link *flight* to its dependent *late* without crossing the arc that links *morning* to its head.

Our concern with projectivity arises from two related issues. First, the most widely used English dependency treebanks were automatically derived from phrase-structure treebanks through the use of head-finding rules. The trees generated in such a fashion will always be projective, and hence will be incorrect when non-projective examples like this one are encountered.

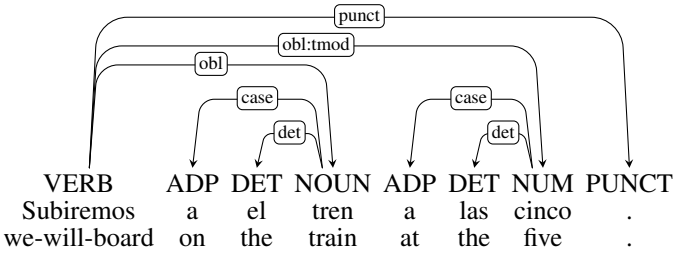
Second, there are computational limitations to the most widely used families of parsing algorithms. The transition-based approaches discussed in Section 19.2 can only produce projective trees, hence any sentences with non-projective structures will necessarily contain some errors. This limitation is one of the motivations for the more flexible graph-based parsing approach described in Section 19.3.

19.1.3 Dependency Treebanks

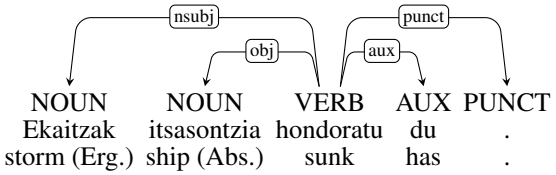
Treebanks play a critical role in the development and evaluation of dependency parsers. They are used for training parsers, they act as the gold labels for evaluating parsers, and they also provide useful information for corpus linguistics studies.

Dependency treebanks are created by having human annotators directly generate dependency structures for a given corpus, or by hand-correcting the output of an automatic parser. A few early treebanks were also based on using a deterministic process to translate existing constituent-based treebanks into dependency trees.

The largest open community project for building dependency trees is the Universal Dependencies project at https://universaldependencies.org/ introduced above, which currently has almost 200 dependency treebanks in more than 100 languages (de Marneffe et al., 2021). Here are a few UD examples showing dependency trees for sentences in Spanish, Basque, and Mandarin Chinese:



[Spanish] Subiremos al tren a las cinco. "We will be boarding the train at five."(19.4)



[Basque] Ekaitzak itsasontzia hondoratu du. "The storm has sunk the ship."(19.5)

Aho, A. V. and J. D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentice Hall.

Bejček, E., E. Hajičová, J. Hajič, P. Jínová, V. Kettnerová, V. Kolářová, M. Mikulová, J. Mírovský, A. Nedoluzhko, J. Panevová, L. Poláková, M. Ševčíková, J. Štěpánek, and Š. Zikánová. 2013. Prague dependency treebank 3.0. Technical report, Institute of Formal and Applied Linguistics, Charles University in Prague. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Bhat, I., R. A. Bhat, M. Shrivastava, and D. Sharma. 2017. Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data. *EACL*.

Buchholz, S. and E. Marsi. 2006. Conll-x shared task on multilingual dependency parsing. *CoNLL*.

Chen, D. and C. Manning. 2014. A fast and accurate dependency parser using neural networks. *EMNLP*.

Choi, J. D. and M. Palmer. 2011a. Getting the most out of transition-based dependency parsing. *ACL*.

Choi, J. D. and M. Palmer. 2011b. Transition-based semantic role labeling using predicate argument clustering. *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*.

Choi, J. D., J. Tetreault, and A. Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. *ACL*.

Chu, Y.-J. and T.-H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Covington, M. 2001. A fundamental algorithm for dependency parsing. *Proceedings of the 39th Annual ACM Southeast Conference*.

Dozat, T. and C. D. Manning. 2017. Deep biaffine attention for neural dependency parsing. *ICLR*.

Dozat, T. and C. D. Manning. 2018. Simpler but more accurate semantic dependency parsing. *ACL*.

Dozat, T., P. Qi, and C. D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Edmonds, J. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.

Eisner, J. 1996. Three new probabilistic models for dependency parsing: An exploration. *COLING*.

Gabow, H. N., Z. Galil, T. Spencer, and R. E. Tarjan. 1986. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122.

Grünewald, S., A. Friedrich, and J. Kuhn. 2021. Applying Occam's razor to transformer-based dependency parsing: What works, what doesn't, and what is really necessary. *IWPT*.

Hajič, J. 1998. *Building a Syntactically Annotated Corpus: The Prague Dependency Treebank*, 106–132. Karolinum.

Hajič, J., M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Stranǎk, M. Surdeanu, N. Xue, and Y. Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. *CoNLL*.

Karlsson, F., A. Voutilainen, J. Heikkilä, and A. Anttila, eds. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Kiperwasser, E. and Y. Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Kudo, T. and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. *CoNLL*.

Kulmizev, A., M. de Lhoneux, J. Gontrum, E. Fano, and J. Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. *EMNLP*.

Lin, D. 2003. Dependency-based evaluation of minipar. *Workshop on the Evaluation of Parsing Systems*.

de Marneffe, M.-C., T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. *LREC*.

de Marneffe, M.-C., B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. *LREC*.

de Marneffe, M.-C. and C. D. Manning. 2008. The Stanford typed dependencies representation. *COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation*.

de Marneffe, M.-C., C. D. Manning, J. Nivre, and D. Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

McDonald, R., K. Crammer, and F. C. N. Pereira. 2005a. Online large-margin training of dependency parsers. *ACL*.

McDonald, R. and J. Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

McDonald, R., F. C. N. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. *HLT-EMNLP*.

Nivre, J. 2007. Incremental non-projective dependency parsing. *NAACL-HLT*.

Nivre, J. 2003. An efficient algorithm for projective dependency parsing. *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*.

Nivre, J. 2006. *Inductive Dependency Parsing*. Springer.

Nivre, J. 2009. Non-projective dependency parsing in expected linear time. *ACL IJCNLP*.

Nivre, J., J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007a. The conll 2007 shared task on dependency parsing. *EMNLP/CoNLL*.

Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007b. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Nivre, J. and J. Nilsson. 2005. Pseudo-projective dependency parsing. *ACL*.

Nivre, J. and M. Scholz. 2004. Deterministic dependency parsing of english text. *COLING*.

Petrov, S., D. Das, and R. McDonald. 2012. A universal part-of-speech tagset. *LREC*.

Petrov, S. and R. McDonald. 2012. Overview of the 2012 shared task on parsing the web. *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, volume 59.

Seddah, D., R. Tsarfaty, S. Kübler, M. Candito, J. D. Choi, R. Farkas, J. Foster, I. Goenaga, K. Gojenola, Y. Goldberg, S. Green, N. Habash, M. Kuhlmann, W. Maier, J. Nivre, A. Przepiórkowski, R. Roth, W. Seeker, Y. Versley, V. Vincze, M. Woliński, A. Wróblewska, and E. Villemonte de la Clérgerie. 2013. Overview of the SPMRL 2013 shared task: cross-framework evaluation of parsing morphologically rich languages. *4th Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Sleator, D. and D. Temperley. 1993. Parsing English with a link grammar. *IWPT-93*.

Surdeanu, M., R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. *CoNLL*.

Tesnière, L. 1959. *Éléments de Syntaxe Structurale*. Librairie C. Klincksieck, Paris.

Yamada, H. and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. *IWPT-03*.

Zeman, D. 2008. Reusable tagset conversion using tagset drivers. *LREC*.