May 6, 2003

Handbook for Language Engineers

Ali Farghaly (ed.)

**CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION**

Contents

May 6, 2003

1

Text Mining, Corpus Building and Testing

Karine Megerdoomian

1.1 Introduction

Approaches in computational linguistics have traditionally concentrated on the development of formal mechanisms and the enhancement of knowledge sources for language analysis. Recently, however, there has been a marked trend towards quantitative approaches, with a focus on statistical knowledge acquisition techniques using a collection of written text or recorded speech called **corpus** (plural **corpora**).

With the publication of *Syntactic Structures* (Chomsky, 1957), computational approaches to language became dominated by the theoretical perspective developed in generative linguistics. In Syntactic Structures, Chomsky argued that quantitative approaches to analyzing language and investigating linguistic patterns in a corpus cannot provide an *explanatory* theory. Chomsky's classification of the three levels of adequacy treated quantitative approaches as *descriptive* and thus having no impact on the goal of formulating an explanatorily adequate theory of language.

Chomsky (1957) criticized the inadequacies of Markov models used in statistical approaches for modeling natural language pointing to the fact that they are unable to model many recursive structures. Furthermore, he argued that a corpus cannot serve as a useful tool for the linguist:

> "Any natural corpus will be skewed. Some sentences won't occur because they are obvious, others because they are false, still others be-

Handbook for Language Engineers.
Ali Farghaly (ed.).
Copyright © 2003, CSLI Publications.

cause they are impolite. The corpus, if natural, will be so wildly skewed that the description [of language based on the corpus] would be no more than a mere list." (Chomsky, 1957, p. 159)

The result was a rift between the generative linguistic or *symbolic* approach, whose goal is to develop an explanatorily adequate theory of language with introspective data as the primary evidence, and the *statistical* approach, which was motivated by empirical coverage and used collections of naturally occurring data as primary evidence. Hence, computational approaches to language analysis from the late 1950s to late 1980s included a prominent emphasis on building knowledge sources in the form of symbolic rules and were often hostile to quantitative methods. The 1990s, however, have witnessed a resurgence of interest in corpus-based and statistical methods of language analysis, which has now become the prominent methodology within the field.

Many technological advances have contributed to the newly found popularity of quantitative models. With the dramatic increase in online information throughout the world, naturally occurring data are now readily available. There exists a preponderence of unrestricted text available through the world wide web and electronic communications. At the same time, a number of advances in the field of computer science and engineering have facilitated the analysis of these large text corpora. Faster processing and cheaper storage space provide the computational resources for successfully extracting the potential value and information from these texts. Furthermore, the advent of CD-ROMs, online dictionaries and funded data-collection initiatives have made the distribution of large sets of data much easier.

Within the last two decades, the processing speed of personal computers has also risen while costs have fallen dramatically. These changes have given personal computer users access to large sets of corpora. The resurgence of interest in corpus-driven approaches has also been fueled by the need for applications that work with language and unstructured information in a real world context. The old 'acquisition bottleneck' in collecting corpora has now been replaced by the dire need to manipulate the floods of unstructured and uncoordinated data available. In fact, an increasing number of natural language processing systems are being used in support of other computer programs – as in database management systems, text categorization, question answering systems and automatic summarizers. The need for broad coverage, robustness, and the fact that users are satisfied with less than perfect results (e.g., rough automatic translations of online text) have required natural language processing systems that employ large-scale quantitative methods.

In addition to the language technology applications for businesses and government agencies, investigations of large text corpora have been used to provide insight on a number of linguistic phenomena such as usage of collocations or idiomatic expressions, word sense disambiguation, and lexical acquisition.

This chapter presents an introduction to corpus-based approaches in computational linguistics. Section 1.2 defines various corpus types and explores some of the general applications and usages of text data mining techniques and corpus linguistics. Section 1.3 discusses the basic problems of dealing with online text such as formatting, conversion and tokenization. These include some of the low-level processes applied to a text prior to the real quantitative or research work. Though low-level, these issues are at the basis of processing and manipulating a corpus and any inaccuracies at this level affect all subsequent results. The following section describes methods for annotating the tokenized texts. In certain applications, explicit semantic or syntactic information is required before the research project begins. This section will consider some of the criteria taken into account when marking up a text. In Section 1.5, various applications of corpus-based approaches are discussed briefly. These include lexicon acquisition, partial parsing, word sense disambiguation, discourse analysis, information retrieval, text categorization and machine translation. Section 1.6 presents the basic tools and techniques needed to manipulate a language corpus. In addition, the section addresses methods used to evaluate and test a language model. Section 1.7 further studies the division between symbolic and statistical approaches to language analysis and explores the arguments for hybrid systems in computational linguistics. The last section is a summary of the chapter.

1.2 Corpus Analysis and Text Data Mining

1.2.1 Corpus Properties

A **corpus** is a collection of text or speech material that has been brought together according to a certain set of predetermined criteria. Corpora are usually used for extracting statistical and linguistic information and for testing hypotheses about natural language. In statistical approaches, a corpus is used to train and test probabilistic algorithms.

Although there is no clear consensus in the field on what counts as a corpus, there exist a number of characteristics that describe a corpus in computational linguistics. A modern corpus of computational linguistics is often collected in machine readable or electronic format.

In addition, a corpus needs to be *representative* of the domain under study and should be a *balanced* sample.

A sample is **representative** if the linguistic information found within the sample also holds for the general population of the domain under investigation. One of Chomsky's criticisms of a corpus-based approach was that any corpus used for analyzing language is bound to be skewed since language is infinite and all possible utterances could never be represented accurately in a finite collection of text. Chomsky argues that, in a corpus, rare utterances may be included several times while common words or structures may not appear.

Hence, the goal is to construct a corpus that is as representative as possible of the domain under study, by including samples from a broad range of material. A representative corpus can provide a reasonably accurate and proportional picture of the entire language population. For instance, the percentage of irregular plurals in a representative sample of English is proportional to the percentage of occurrence of irregular English plurals in general.

Another important criterion for a corpus is that it be a **balanced** sample of the general population. A balanced corpus is a collection that attempts to cover as many textual styles as possible by trying to include samples from various genres: poetry, prose, non-fiction, news, emails, references, etc. What constitutes a truly balanced corpus is an open question, but some of the parameters that need to be taken into account for constructing a balanced sample of material are genre, text type and domain. In addition, the length of the corpus and the time frame represented (e.g., 16th century English vs. Modern English) are also important criteria in constructing a balanced sample of text.

The properties of a corpus, however, vary depending on the applications for which they are to be used. For instance, it is more important for statistical natural language processing (NLP) systems to have a large set of data to train on rather than having a balanced corpus. Similarly, if corpora are assembled for a specific purpose, then they do not necessarily include samples from diverse genres. These **special corpora** are not balanced by definition and will only provide a distorted view of the language segment if used for general purposes.[1] The

[1]Note that the notion of *balanced corpus* has sometimes been used in a relative sense. A special corpus can be "balanced" within the scope of a given domain if it includes various genres containing the language phenomena under study. Hence, if the goal is to analyze language phenomena in political news articles then the corpus should be "balanced" by including news items from various newspapers or sources of the language in order to capture different styles, vocabulary and tokenization standards. But it is not crucial to gather email or poetry samples to train the

advantage of such a corpus over a *balanced corpus* is the smaller size. In addition, the content of a *special corpus* is specially targeted to include the phenomena the researcher is looking for, thus the particular wordforms or syntactic structures being investigated will occur more frequently than they might in a *balanced corpus.*

Although corpora are generally finite in size, a type of corpus known as a **monitor corpus** is formed of a nonfinite collection of text which grows regularly to reflect language changes. Monitor corpora are normally used for lexicography purposes. Furthermore, certain corpora are enriched with explicit semantic or syntactic information that facilitates retrieval and analysis of electronic text. These **annotated** or **marked up** texts will be discussed at length in Section 1.4. Some applications require monolingual corpus samples while others may need multilingual corpora (e.g., in machine translation).

Thus, although there is no consensus in the field of computational linguistics on the requirements for assembling corpora, there does exist a general agreement that attempts should be made to make a corpus *representative* and *balanced.* Report results and estimates obtained from such corpora can then be reasonably extrapolated to the domain of interest. Nevertheless, what properties a corpus should contain is determined in large part by the applications. For specific parameters and guidelines for establishing a corpus, the reader is referred to *Corpus Design Criteria* by Atkins et al. (1992), which reviews the practical stages in the process of assembling a corpus.

1.2.2 Corpora Resources

One of the first major collections of text to be used in computational linguistics was the *Brown Corpus* (Kučera and Francis 1967). The Brown Corpus was assembled in Brown University in 1963-64 and was designed as a representative sample of written American English, divided roughly evenly into genres, and including about one million tagged words. The Brown Corpus is probably the most widely known corpus and most institutions doing research on NLP have a copy available. This corpus covered only American English but its counterpart for British English was built in the 1970s and is known as the *Lancaster-Oslo-Bergen* corpus (Johansson et al 1978).

The *Penn Treebank* is a larger corpus of about 4.5 million words. All words in this corpus have been annotated with part of speech tags and it also includes labelled brackets marking syntactic analysis. This corpus was collected from the Wall Street Journal. It is very widely

statistical system or to test the output.

```
((S
    (NP (NN Implementation)
        (PP (IN of)
            (NP
                (NP (NNP Georgia))
                (POS 's) (NN automobile) (NN title) (NN law))))
    (AUX (VBD was))
    (VP (ADVP (RB also))
        (VBN recommended)
        (PP (IN by)
            (NP (DT the) (JJ outgoing) (NN jury)))))
 (. .))
```

FIGURE 1 Brown Corpus Sample.

```
(S (`` ``)
   (NP (DT A) (NN stockbroker))
   (VP (VBZ is)
       (NP (DT an) (NN example)
           (PP (IN of)
               (NP (DT a) (NN profession)
                   (PP (IN in)
                       (NP (NN trade) (CC and) (NN finance)))))))
```

FIGURE 2 Penn Treebank Sample.

used but is not available for free.

There exist a number of other English corpora sources often used in corpus analysis and computational linguistics. The *British National Corpus* (BNC) was completed in 1994 and is a 100 million word collection representing a wide cross-section of current written as well as spoken British English[2]. This corpus has been tagged for part of speech. The *Birmingham Collection of English Text* contains 20 million words and was built in 1985. The 2002 release of the *Bank of English* consists of approximately 450 million words from written and spoken language[3]. *Project Gutenberg* is another good source of online raw text freely available and consisting of over 6,000 books in English, most of

[2]www.hc.ox.ac.uk/BNC.

[3]http://titania.cobuildcollins.co.uk/boe_info.html.

which were published before 1923 and are thus free of copyright issues[4]. Also, *Reuters* makes available about 810,000 English language news stories in raw text for research purposes.

A project is currently underway to build the *American National Corpus* (ANC), to consist of over 100 million tagged words, chosen from a broad selection of contemporary written and spoken texts of American English[5]. The first release consisting of 10 million words tagged for part of speech is scheduled to be released in 2003, with the final release projected for Fall 2004. The ANC will be freely available for research purposes from the Lingustic Data Consortium (LDC).

Corpora of Western European languages are often freely available. The largest corpus of German text, for instance, is the *Mannheimer Corpus Collection* with about 2 billion running words. A smaller version is available free of charge, but only for research purposes[6]. Corpora for other languages are not as easily available and oftentimes, the researcher needs to spend a significant amount of time downloading and preparing corpus sources for the language under study.

Probably the most well-known *bilingual corpus* of parallel texts is the *Canadian Hansards*, which is the proceedings of the Canadian parliament. This corpus consists of texts in French and English that are translations of each other. This bilingual corpus has been used for statistical approaches to machine translation by training the system on the aligned parallel texts. The Canadian Hansards are available for a licensing fee. The *United Nations Parallel Text Corpus*, available at the Linguistic Data Consortium (LDC) for research purposes, was drawn from the UN electronic text archives covering the period between 1988 and 1993. This corpus contains about 2.5 gigabytes of text in English, French and Spanish.

The researcher often needs to take into account legal issues surrounding copyrights, especially if the corpora are to be made public and distributed freely for research purposes. Commercial institutions often do not make their corpora available to the public and to their competitors, although some may provide them in exchange for a licensing fee. Copyright issues do not arise if researchers use documents of public record (e.g., certain online news sources, public governmental documents), material available freely from research institutions, and old text that has fallen out of copyright (e.g., Shakespeare, the Bible).

A good place to start gathering corpora for lesser-studied languages is `www.webdopresse.ch`, which contains links to multilingual news

[4]`http://promo.net/pg/`.
[5]`http://americannationalcorpus.org`.
[6]`http://corpora.ids-mannheim.de/cosmas/`.

sources. Parallel text in all European Union languages can be downloaded from the EU webserver at `http://europa.eu.int`. Also of interest are the corpus links on Linguist List (`www.linguistlist.org`). Another good source for corpus-based research is the Linguistic Data Consortium (LDC) at the University of Pennsylvania (`www.ldc.upenn.edu`). LDC is a consortium of research laboratories, companies and Universities and is dedicated to collecting speech and text corpora and language resources for research and development.

1.2.3 Corpus Analysis

Once the data are collected into a corpus of text or speech, they can be used to investigate linguistic phenomena or to extract information on language use. As mentioned earlier, corpora can be used for a number of distinct applications and the manipulations done to a corpus depend mainly on the final goals of the researcher.

Corpus linguistics is the study of language through analysis of naturally-occurring data. Given the vast amount of language data available online, corpus linguistics usually involves computational methods and tools. Corpus linguists use the resulting patterns and the extracted information to test hypotheses about language, with the goal of developing theories of linguistics and language use. In contrast, computational linguists usually gather information in order to create more effective computational grammars and systems. In computational linguistics, the results of tests performed on corpora and the information obtained from corpus analysis is often used as a basis for improving NLP applications.

Although much information can be obtained from raw corpora, a prerequisite to most corpus analysis work is the annotation of the textual material. The tags and indices are then used to extract relevant information. The extent of the markup (e.g., words vs. phrases) and the tagsets used are determined by the application needs. Hence, the analysis of a corpus of written text allows the researcher to retrieve words, phrases or sentences by pattern matching. In order to facilitate information retrieval, the researcher could annotate the corpus with tags or indices, and statistical methods could be utilized to rank the retrieved elements. These methods can be used to analyze linguistic patterns and usage within the given corpus. For instance, such information retrieval is often used in applications in collocation analysis, machine translation, lexicon acquisition, partial syntactic analysis and text summarization. Marked-up data and the criteria for developing tagsets are discussed in Section 1.4. We will further develop a number of applications for annotated corpora in Section 1.5.

1.2.4 Text Mining

Recent years have witnessed an enormous growth in the volume of online text documents in multiple formats and languages. These documents represent **unstructured data** residing outside of relational databases, such as spreadsheets, word processor files, email message archives, presentations, PDF files, websites, graphics, or even audio and video files. The growing prevalence of online documents has led to a great interest in **Text Mining**, the goal of which is to discover knowledge from unstructured textual data. Text Mining, also known as **Text Data Mining** is a relatively new approach in computational linguistics, often combining techniques from data mining, information extraction, machine learning, natural language understanding, database theory, and visualization. Text Mining has been defined as the nontrivial extraction of implicit, previously unknown, and potentially useful information from given data (Piatetsky-Shapiro and Frawley, 1991).

Several factors have contributed to the exponential growth of unstructured data. In the 1980s, computers started to take over as the principal method for creating content and sharing knowledge. With the advent of the internet, the ease of creation of online text, and the shift from manufacturing to service economies that involve the processing of enormous volumes of information, the traditional bottleneck of corpus acquisition gave way to the problem of **unstructured data management**. Interestingly, most of the work and research done in text mining seems to originate from the industry rather than academic institutions, and there are many commercial text mining tools currently in use. Some of these tools use neural networks as the basis for text pattern recognition, while others combine a knowledge-based model with a statistical approach.

Some researchers fail to distinguish *data mining* and *text mining.* However, the difference between these two approaches lies in the type of data under investigation. The aim of data mining is to discover knowledge from databases and usually relies on the explicit structure provided by the relational templates. However, about 85% of online data do not exist in relational or tabular format but appear in unstructured form. Text mining is concerned with knowledge discovery from the large volume of unstructured natural-language documents by determining key relationships and concepts.

Text mining has also been equated with *Information Retrieval*, but Hearst (1999) distinguishes the two methodologies arguing that text data mining can derive new information from the data while Information Retrieval only extracts already existing information. Hence, while information retrieval accesses documents and matches patterns and keywords to extract the information, a text mining system actually generates new information by finding patterns across datasets. In fact, although most text mining systems include an *Information Extraction* module, knowledge discovery from unstructured data consists of several distinct techniques. In text mining, a textual document is analyzed by performing basic language processing and by extracting concepts, entities and events. The relationships between various events and entities are then organized into a well-defined taxonomy of categories and finally visualization tools are applied to this information structure to navigate and explore the content set. Hence, text mining may include information retrieval methods but it is really a nascent field at the intersection of data mining, machine learning, information retrieval and natural language processing.

The first step in text mining is information extraction (IE), which locates significant vocabulary items in natural-language documents. Typically, the elements extracted by an information extraction module include people, places, and companies; this is accomplished by recognizing proper names and noun phrases in the text documents. An information extraction module may also use a set of heuristics to identify phone numbers, dates, and technical or domain-specific terms. It is important to note, however, that information extraction cannot be applied unless the textual document has undergone some level of pre-processing such as *tokenization*, *lemmatization* or *stemming*, and *tagging*. Machine learning is often used to train the information extraction system. Once terms and events are extracted from the text, relation and hierarchical analysis is applied to determine the clustering of the documents and the structuring in databases. The extracted terms are used to **cluster** a collection of documents into groups that share common properties or related information, and to **categorize** them by assigning pre-determined categories usually based on themes or topics. Data mining or *Knowledge Discovery in Database* (KDD) techniques are applied at this point to dicover the knowledge from the databases and visualization tools are used to navigate the information.

Linguistic knowledge and corpus training are important during the first part of text mining. Most text mining approaches require the extraction of entities as well as events in order to form a meaningful conceptual structure. For example, Karanikas et al. (2000) extracts entity terms such as *Department of Computation*, combined with basic event structure information such as *take-over, company1, company2*, items that are used to describe the event depicted in the text. Similarly, the text mining model developed at Inxight Software is based on a multilingual linguistic system that includes tokenization, morphological analysis, tagging, name recognition and partial parsing. Other systems may also take into account semantic and coreference analyses. Thus, most text mining systems have a level of text analysis involving linguistic knowledge. The rest of this chapter discusses the creation of some of these linguistic modules in computational linguistics.

1.3 Tokenization

Prior to analyzing corpora, one needs to filter out particular formatting that is not relevant to the issues under study. Hence, when dealing with online documents, HTML tags need to be stripped, and paragraphs, tables and headers should be separated. The text often needs to undergo **tokenization** in order to separate the input text into distinguishable units or **tokens** and to determine sentence and word boundaries as shown below with | indicating the token boundaries:

(1)	*raw text:*	"Stop!", Dr. Tyler shouted.
	tokenized text:	"|Stop|!|"|,|Dr.|Tyler|shouted|.

In languages like English, the presence of a space or tab usually indicates a word boundary. In other languages, however, tokens are often written without an intervening space; this is particularly true in certain Asian languages. Punctuation can often be used in determining word and sentence boundaries, but it is not always clear since periods can be used to mark an abbreviation and hyphens occur inbetween two parts of a compound word. This section outlines some of the issues that arise when pre-processing corpora, with special emphasis on problems in determining word boundaries.

1.3.1 Sentence Boundaries

Labeling of sentence boundaries is a prerequisite for many NLP tasks, including part of speech tagging, sentence alignment in parallel corpora or machine translation applications. Sentential punctuation markers such as the question mark (?), exclamation point (!) and the period (.) can be used to distinguish sentence boundaries. The period, however, can be used to mark an abbreviation as in *etc.* or *K.H. Smith*, or an acronym as in *I.B.M.* The period can also mark a decimal point or an ellipsis. Furthermore, since an abbreviation can be the last token in a sentence, the period at the end could signal both a sentence break and an abbreviation; this is known as a **haplology** where one character has two simultaneous uses. In general, however, 90% of periods in English are sentence boundary markers and therefore, most tokenizing modules use generic rules to detect a sentence boundary, often encoded in terms of regular expressions, and supplement them with a list of abbreviations and acronyms. Hence, if the system knows that the word preceding a period is not an abbreviation or acronym, then the period can be confidently treated as an end-of-sentence marker. For the case of the decimal point, processors can easily identify digits and numbers and thus disambiguate the period in these instances.

In languages like English or French, capitalization can be used to disambiguate punctuation. If the token following a period is a capitalized word then the period probably marks a sentence boundary (unless the word is a proper noun), while a lowercase word would indicate that a boundary has not been reached. The case of the word cannot be used, however, to disambiguate boundaries in languages such as German where common nouns are always capitalized, or in Arabic which does not have a capitalized form.

Other issues also arise in sentence boundary detection. Sometimes other punctuation such as the colon, semicolon and dash may separate sentences. In addition, sentences may be split as in quotes in indirect speech as in:

(2) “According to the latest reports,” he explained, “the war is imminent.”

This example also demonstrates another problem arising in tokenization of English due to the typesetting standard in the United States, where the close quotation mark follows the period. Furthermore, detecting sentence boundaries is made difficult in certain corpora sources that contain sentence fragments. This occurs, for instance, in headers in news sources. Some tokenizers can recognize consecutive newline characters which are then used to identify headings as sentence fragments.

Recent approaches to sentence boundary detection have tried different methodologies for obtaining better coverage and more robust systems. Riley (1989) uses statistical classification trees trained on a pre-tagged corpus. Palmer and Hearst (1997) take advantage of part-of-speech information in a neural network to predict sentence boundaries. More recent works develop a Maximum Entropy approach to boundary detection based on the probability distribution of sentence boundaries in a text (Reynar and Ratnaparkhi, 1997, Mikheev, 1998).

1.3.2 Word Segmentation

One of the major problems in tokenizing corpus data is the recognition of what constitutes a *word*. The simplest method for delineating words in computational linguistics is to use the occurrence of whitespace or punctuation as a clue. Hence, if a token is surrounded by spaces, tabs, newline characters or punctuation marks, then it can be treated as a distinct word. Although this approach may work for English in most instances, it raises difficulties in analyzing other languages. This section addresses some of the main problems in determining word boundaries in computational linguistics.

Punctuation. In order to recognize words, it is usually desirable to separate all unambiguous punctuation into a distinct token during segmentation. However, as discussed in the previous section, a period could sometimes signal the presence of an abbreviation and is not necessarily preceded by a word. In these instances, the punctuation character should presumably be maintained as part of the word token.

In English, apostrophes are sometimes used in marking possessive nouns such as *cat's* in the singular form or *cats'* in the plural form of a noun. Apostrophes can also be part of a word in English, e.g., *rock'n roll.* To complicate matters further, single apostrophes are also used in forming contractions in English thus raising further difficulties in word segmentation. Hence, some tokenizers treat *they'll* or *can't* as single tokens while others divide these contractions into two distinct tokens that will be treated as separate words by a tagger or parser. The separated tokens *'ll* and *n't* are then treated as equivalent to *will* and *not*, respectively.

Contractions. Contractions are common in Romance languages as well. For instance, a tokenizer may need to expand the Spanish contraction *del* into *de* and *el* in order to obtain the correct syntactic structure needed at a later stage in the processing. Other systems treat *del* as a single token and decompose it into two distinct "words" or parts of speech during morphological analysis.

Hyphenation. Hyphenation is a complex issue in most languages since it can be used to divide subparts of a compound word as in *part-of-speech*, *e-mail*, or *anti-semitic*, but hyphens may also be used in joining words that are usually not treated as a single token as in *import-export* or *the 25-year-old.* To treat these hyphenated elements as a single word would tremendously increase the size of the lexicon. In addition, it may actually cause problems in parsing since it often alters the syntactic structure. Sometimes hyphenation is optional, e.g., *mark-up* vs. *markup* or *part-of-speech* vs. *part of speech*, and processors need to take these differing conventions into account. A final difficulty arises when a hyphen is used at the end of the line to split a long word. These line-final cases are problematic since it is not clear whether the word was initially hyphenated or not.

Whitespace. Another common problem in word segmentation is the treatment of compounds when two tokens separated by a space need to be treated as a single unit. For instance, the proper names *Los Angeles* and *New York* each consist of two distinct tokens which form a single word together. Other similar cases in English are multiwords such as *in order to* or *a priori* or phrasal verbs as in *pick up*. Generally, tokens separated by a space are split up by tokenizer modules and it is only in subsequent processing that multiword tokens are recognized and put back together.

Word segmentation in English is simpler than in other languages where whitespace is not always used to separate words. In the writing system of East Asian languages such as Chinese, Thai or Japanese, there are no spaces between individual words. In Chinese, for instance, whitespace usually only appears at paragraph boundaries. In implementing tokenizers for these languages, oftentimes a lexicon or an extensive wordlist is incorporated in the module and used along with a "greedy" word-matching algorithm that starts at the first character and tries to match the longest word in the wordlist. More recent approaches have applied statistics for word boundary disambiguation with algorithms that can find the most probable sequence of words according to the training model.

In German and Dutch, compound nouns are commonly written as single orthographic words. The reader may be familiar with Mark Twain's famous parody of the German language (Twain 1880) in which he provides examples of German compounding. For instance, the equivalent of the English *city council meetings* is written as *Stadtverordnetenversammlungen* in German. For processing purposes, a tokenizer may need to perform **decompounding** by dividing up these compound nouns or to somehow mark the internal structure of these words. Note that German compounds may contain embedded morphemes that need to be analyzed when tokenizing. For example, in the compound *Unabhängigkeit***s***erklärungen* (Declarations of Independence), the two tokens are joined with the morpheme "s" (shown in bold). Certain NLP systems incorporate some level of morphological analysis in the tokenizer module in order to be able to accurately recognize compounded tokens; some systems merge the tokenizer and morphological analysis module into a single component responsible for both functions.

Another tokenization problem arises with languages whose writing system does not utilize the space consistently to mark a word boundary. Although words are usually separated by intervening spaces in Persian, they are sometimes omitted resulting in two distinct words being treated as a single token. Also, some affixes in this language, such as the plural suffix, may be separated from the base noun by a space, requiring further processing to put together the separated morphemes.

Numbers. Numbers often present additional problems in tokenization of a text due to typographical differences across languages or countries. For instance, phone numbers can be written in a number of different formats using hyphens, brackets, periods, spaces, plus signs (to mark area codes) or slashes. For example, dates in European texts are in the format *day/month/year*, whereas in American English texts the day and month order is reversed and dates are presented as *month/day/year*. In addition, the number of digits in a phone number varies from one country to the next. Preprocessors generally do not have to distinguish a phone number from a date, but they do need to be able to recognize numbers from alphabetic strings (or words).

Encoding. English and most Western European languages use the ASCII character set but one of the problems in tokenizing non-European languages lies in recognizing the character encoding of the corpus data. If the natural language processing system cannot identify a document's encoding, it will not be able to process it accurately at any level. This is even more difficult for multilingual applications since there does not exist a single standard encoding for online texts in most languages. For example, Russian text can be found in any of the following four character sets: cp1251, ISO 8859-5, koi8-r or utf8. Many NLP systems try to unify the processing modules by using the Unicode standard which provides a unique number for each character and allows to handle text in practically any script or language. Hence, the encoding of the input data is converted to the Unicode character set for processing purposes throughout the system. In any NLP application, the character encoding of the language under study needs to be taken into account prior to processing.

1.3.3 Speech Corpora

The demarcation of word boundaries in speech corpora faces a whole other set of problems. Speech corpora are filled with contractions, performance issues (noise, coughs, "um" fillers), and fragments. In addition, accents and pronunciation variants affect the recognition of the speech data and even the pitch of the wave frequency can hinder the segmentation of words in these corpora. For specific issues in analyzing speech corpora and techniques for word segmentation, the reader is referred to Amtrup, Chapter 9, this volume.

1.4 Corpus Annotation

While much information can be obtained from plain text or **unannotated** corpora, many applications take advantage of an **annotated** corpus that has been enhanced with tags providing explicit linguistic information. For instance, the implicit information denoted by the verb "eats" becomes explicitly marked in a tagged corpus in which the verb may be annotated as "eats_VVZ" where VV marks a lexical verb and Z indicates a third person singular form. Such annotations facilitate the retrieval and analysis of linguistic information.

A basic mark-up scheme is to tag a corpus for formatting attributes such as page breaks, paragraphs or font information. A corpus can also be tagged with identifying information such as author, date, title, genre, register, etc. This information is usually enclosed within angle brackets and annotated with a tag that describes the general function or identity of the marked-up text. The most supported form of annotation is the *Standard Generalized Markup Language* or SGML, HTML being the most common example of SGML encoding. An SGML document contains a set of marked-up elements that are enclosed in between two tags contained within angle brackets. The end tag begins with a forward slash character. Some examples are given below:

```
<title>A simple title</title>
<author>
    <name>John Smith</name>
</author>
<Mod date=''15-Feb-2002''
     type=''Last annotation update''>
</Mod>
```

A more common set of tags are `<p>` for paragraph and `<s>` for sentence as shown:

```
<p><s>This is a short example.</s>
   <s>It contains two sentences.</s></p>
```

XML (*Extensible Markup Language*) is a simplified subset of SGML and is often used in marking up linguistic corpora. A sample annotation of a Penn Corpus sentence using XML syntax is illustrated below:

```
<S>
 <NP>
  <DT>A</DT>
  <NN>stockbroker</NN>
 </NP>
 <VP>
  <VBZ>is</VBZ>
  <NP>
   <DT>an</DT>
   <NN>example</NN>
   <PP>
    <IN>of</IN>
    <NP>
     <DT>a</DT>
     <NN>profession</NN>
     <PP>
      <IN>in</IN>
      <NP>
       <NN>trade</NN>
       <CC>and</CC>
       <NN>finance</NN>
      </NP>
     </PP>
    </NP>
   </PP>
  </NP>
 </VP>
 <.>.</.>
</S>
```

As this example demonstrates, a corpus can be annotated by tagging part of speech categories and linguistic phrases. The level of detail in corpus annotation and the size of the element tagged (i.e., word vs. phrase) is typically dependent on the objectives of the system or project.

Probably the most widespread annotation scheme is tagging lexical units for part of speech (POS). Part of speech information in a text is used in information retrieval applications by helping to select and extract nouns or other important words from a document. Part of speech information can also be used in partial parsing. For example, looking for a "Title" such as *Dr.* or *Mr.*, the system can detect proper names for information extraction applications and summarization. Statistical automatic part-of-speech taggers trained on marked-up text can help build word sense disambiguation models.

Speech corpora also take advantage of part of speech information. In English, words like *object* are pronounced differently depending on their part of speech: the noun is pronounced 'OBject' while the verb is pronounced 'obJECT'. Thus, knowledge of the part of speech can help the accuracy in a speech recognition system.

This section discusses applications of annotated corpora and describes different markup schemes and methodologies. Some of the criteria to be considered prior to selecting or designing a tagset for part of speech tagging are also addressed.

1.4.1 Annotation Coverage

The design of a tagset and the choice of the annotation scheme depends heavily on the goals of the research and the final application of the NLP system. As discussed in the last section, some corpora are annotated only with formatting tags distinguishing headers and paragraphs, but most linguistic applications often require some level of **POS tagging** whereby each word is explicitly assigned a part of speech or grammatical class. For information retrieval applications, it is often necessary to mark the boundaries of constituents or to tag noun phrases. Some applications require even more markup such as annotating full syntactic structure or tagging of word senses.

Since, as in most corpus-related decisions, there is no single standard in corpus annotation schemes and methodologies, the information to be annotated in the data and the format used are often motivated by the system application. It should be noted that for rule-based methodologies in NLP, tagged corpora are used mainly to evaluate the system. In statistical approaches, in addition to tagged corpora for testing purposes, large sets of texts need to be annotated for training the stochastic model. The following represent some examples of the possible annotation schemes:[7]

Part of speech annotation: POS tagging was one of the first types of annotation used in corpus-based work and is the most common annotation used today. Corpora tagged with POS information are often used as a prerequisite for more complex NLP applications such as information extraction, syntactic parsing or semantic field annotation. They are also used to help train statistical models. POS tagging will be dicussed in detail in the rest of this section.

Lemmatization: Certain POS taggers also perform lemmatization or **stemming** by reducing the word form in the corpus to its lexeme form. A **lexeme**, also known as *stem* or *headword* is the form of the word that is listed in a dictionary and which does not contain any inflectional morphology. For example, the lexeme for the verb forms *ate*,

[7]For links to various annotated corpora for text and speech, the reader is referred to the LDC website at `www.ldc.upenn.edu/annotation`.

eats, and *eating* is *eat*. Lemmatization requires some level of morphological analysis but combined with POS tagging, it is crucial for most complex NLP models.

Parsing: Parsed corpora, also known as **treebanks**, annotate the syntactic phrases by using the information provided by the POS tagger. Usually, labelled brackets are used to tag the constituents. The Penn Treebank project is an example of parsed corpora. The following is a sample parse where S marks sentence, VP marks the verb phrase, and NP and PP indicate the noun phrase and prepositional phrase, respectively.

(3) `[S [NP The boy NP] [VP sat [PP on [NP his bicycle NP] PP] VP] S]`

Certain applications only require *skeleton parsing* where only the most basic constituent structure information is overtly tagged. Many applications only tag noun phrases (NPs) and prepositional phrases (PPs) and do not deal with more complex or embedded structures. In contrast, *full parsing* schemas provide a very detailed analysis of the syntactic structure and include nested pairs of labelled brackets.

Although automatic parsing methods have been used in the field, the results are not as accurate as part of speech tagging and often require manual editing. Hence, it is important to have detailed guidelines when tagging complete sentences, in particular in projects where several people are involved, in order to maintain a certain level of consistency throughout the corpus.

Semantic Annotation: The kind of semantic information required by an application or deemed important by the researcher varies. One of the most common semantic annotations consists of subcategorization information that provide insight on the relations between constituents and events. For instance, fact extraction and text mining systems can utilize information of the type 'who did what to whom' as shown in the following example, explicitly marking the agent and patient relations of the verb *murder*.

(4) [NP The soldiers NP-AGENT] murdered-VVDv [NP six Jesuit priests NP-PATIENT]

Corpora are oftentimes annotated with semantic information about word senses as shown in the examples below:[8]

[8]See description of ontology systems in Chapter 2.

(5)

The	FW	= low content word; function word
soldiers	WAR_AGNT	= war and conflict, agent
put	PUT	= object-oriented physical activity
Jesuit	REL_TYPE	= religion

Discourse Analysis: Discourse processing is concerned with the coherent analysis of text segments larger than a sentence. Discourse annotation may consist of marking the text with tags distinguishing *greetings* (e.g., hello), *apologies* (e.g., excuse me, sorry), *politeness* (e.g., please), etc. One of the main researches in discourse analysis is the identification of the referents of a noun phrase. Hence, *anaphoric annotation* is one of the most important tasks in discourse markup, whereby pronominal reference is overtly encoded. The following example from the Lancaster/IBM anaphoric treebank illustrates this approach. Note that in this example, tags such as (1 1) or (2 2) denote a noun phrase that enters into a relationship with an anaphoric element tagged with REF=1 or REF=2, depending on the noun phrase that it is to be identified with.

```
A039 1 v (1 [N Local_JJ atheists_NN2 N] 1) [V want_VV0
(2 [N the_AT (9 Charlotte_N1 9) Police_NN2 Department_NNJ
N] 2) [Ti to_TO get_VV0 rid_VVN of_IO [N 3 <REF=2 its_APP$
chaplain 3) ,_, [N {{3 the_AT Rev._NNSB1 Dennis_NP1
Whitaker_NP1 3} ,_, 38_MC N]N]Ti]V] ._.
```

Speech Annotation: Speech corpora can be tagged to reflect the prosody of the spoken text, such as intonation, tone or stress patterns. Typically, only the most prominent intonations are annotated. It is extremely difficult to automatically tag an intonation or tone pattern, hence speech annotations need to be performed by humans. This gives rise to a number of difficulties that do not exist with written text annotations. First, it is very hard to agree on prosodic patterns and provide a consistently annotated corpus, since judgments are more impressionistic in nature. Furthermore, since prosodic features are a property of syllables rather than whole words, markup often appears in between the word parts making it difficult to automatically recover the original raw text. The following is a small example from the London-Lund corpus:

```
1 8 14 1530 1 1 B 11 ^quite a nice .room to !s\it in# /
1 8 14 1540 1 1 B 11 *^\isn't* it#                    /
1 5 15 1550 1 1 A 11 *^y/\es#* - - -                  /
```

The codes used in this example are:

`#`	end of tone group
`^`	onset
`\`	falling nuclear tone
`/\`	rise-fall nuclear tone
`.`	normal stress
`!`	booster: higher pitch than preceding prominent syllable
`(( ))`	unclear
`* *`	simultaneous speech
`-`	pause of one stress unit

As is clear from the various annotation approaches presented here, the material being tagged and the markup set selected depend strongly on the research objectives and application needs. Since there does not exist a standard tagset for each application (besides a general notion that tagsets should have broad coverage and remain theory-neutral), the researcher typically designs a set of tags that would be relevant to the distinctions he or she needs to capture within the corpus or the language domain under study.

1.4.2 Tagset Design

The most common type of corpus annotation is part of speech tagging where the aim is to assign a grammatical category to each lexical unit. POS tagging is the foundation for further forms of linguistic analysis but there is no generally accepted convention with respect to the tags used or even the level of detail encoded.

A human responsible for annotating a raw text needs to be provided with a **tagset** that includes all the annotation tags to be used within the corpus and that has been designed for the purpose of the project. A tagging algorithm also requires a specified tagset as input in order to automatically annotate a wordlist derived from a raw corpus.

The most widely used tagsets in computational linguistics have been the ones used in the Brown Corpus, the Penn Treebank and the British National Corpus (or CLAWS). Manning and Schütze (1999) provide an interesting comparison of these three tagsets clearly demonstrating the distinct tag format chosen in each case and the different tagset sizes. For instance, both the Brown Corpus and the BNC distinguish ordinal number adjectives (e.g., *sixth*) from regular adjectives, by marking them with different tags. In addition, both corpora have a special tag for adjectives that are not morphologically marked as a superlative but which have a superlative meaning (e.g., *chief, top*). The Penn Corpus, however, has one single tag for a regular adjective, an ordinal adjective and a semantically superlative adjective. The Penn Corpus, on the other hand, distinguishes the base present form of the auxiliary *do* from the infinitive form tagging them as VBP and VB, respectively. The Brown Corpus and the BNC annotate both instances of *do* with the same tag. In contrast, the Penn Treebank annotates auxiliaries with the same tag as that of verbs, whereas the Brown Corpus and the BNC mark each auxiliary with a different tag. Clearly, each project determines which distinctions need to be made and there is no single standard method for corpus annotation throughout the field.

However, several guidelines are usually taken into account when designing a tagset. The first objective of the tagset is to define an annotation set that can provide the relevant linguistic information to the user about the syntactic or semantic properties of the word. In addition, the tagsets often include features that will be useful at further stages of processing or that will be needed for predicting the behavior of nearby words. The more fine-grained the distinctions made in a tagset, however, the larger the size. The rest of this subsection discusses some of the issues to be considered when designing a tagset.

The tags selected for annotating a corpus should be able to communicate the relevant linguistic information to the user. Tags are generally short and often consist of a two or three letter markup. For instance, the BNC tag for a comparative adjective is AJC and the tag for a superlative adjective is AJS. In companies whose products are used by non-linguists, sometimes a more user-friendly tagset is adopted such as *Nn* for a singular noun and *Nn-Pl* for plural nouns.

In certain cases, the lexical unit consists of more than one token such as *part of speech, hang on, New York* or *carriage return.* Multiword expressions cannot be properly analyzed if they are not recognized as single units. Most taggers are equipped with a lexicon that recognizes multiword tokens and idiomatic expressions, and tags them as a single word, hence *part of speech* will be marked as a noun and *New York* as a proper noun.

Certain multiwords, however, may be discontinuous in a sentence. For example, the English expression *either... or...* consists of two distinct tokens *either* and *or*, which appear together within a sentence but are always separated by intervening material. Automatically tagging such elements as a single unit is quite difficult and usually requires complex lexical structure that can allow intervening variables. This is particularly true of phrasal verbs in English as in *pick up* (e.g., *pick it up*) or *keep tabs* (e.g., *keep close tabs*). In order to capture the relation between the two parts of an idiom, however, many tagsets include tags that are linked or that refer back to each other.

Certain words have ambiguous parts of speech. For instance, *book* can be interpreted as a noun or as a verb, and both part of speech tags should be assigned to the word *book* in the corpus. Statistical taggers attempt to disambiguate these multiple tags based on weights attributed to the probability of occurrence of nearby words and select the most probable tag given the context.

In order to help the disambiguation in statistical approaches, the researcher can select tags that classify the part of speech of the lexical item but which also behave as distributional tags by providing information on the behavior of other words in the context. Hence, distinguishing the pronoun *they* from the possessive form *their* by annotating them with separate markers can help determine the tag of the following element in the sentences below, where *left* is ambiguous between a verb and an adjective.

(6) a. 'They left'
    b. 'Their left hands...'

Such distributional information is then used by the statistical algorithm to disambiguate nearby tags. Thus, if the tagger knows that a possessive pronoun can only be followed by the elements in a noun phrase and not by a verb, the word "left" can be easily disambiguated. Another example comes from Romance languages where adjectives can appear before the noun or following it depending on the properties of the adjectival. Annotating the two pre-nominal and post-nominal adjectival types with distinct tags can help the tagger disambiguate nearby elements since it would expect a noun after the premodifying adjective and a noun before the postmodifying adjective.

Sometimes, tags are differentiated in order to provide finer-grained information that would help in later stages of corpus analysis. For example, in certain languages, case-marking can be used to indicate a noun phrase boundary. If the case-marked noun is simply tagged as Nn, it would not be distinguished from a non-case marked nominal. Hence, if the goal is to group noun phrases or to perform parsing, it may be advantageous to have a separate tag Nn-Case or Nn-NPB to signal that a noun phrase boundary has been reached. A more detailed tagset could ascertain that no useful information is lost at later stages of processing.

We saw that more fine-grained tagsets may be needed to provide distributional information or to capture important morphological information. A more detailed tagset, however, also means a larger tagset which may make it harder to train a statistical model, giving rise to more errors. As always, the researcher needs to find the balance between improving the prediction ability of the tagger and facilitating the classification task, since there is no straightforward relationship between the size of the tagset and the performance of the automatic tagger.

Other issues that the researcher needs to consider when designing a tagset include tags for foreign words, numbers and punctuation. For instance, although normally an end of sentence punctuation is distinguished from others, taggers vary depending on whether they choose to make distinctions among the other punctuation types (for instance, the Penn Treebank tagset has 9 distinct punctuation tags whereas the BNC distinguishes only 4).

1.4.3 Tagging Methods

Although small corpora are often tagged manually, the size of most corpora today, with millions of running word forms, calls for an automatic approach. Automatic taggers can be rule-based or probabilistic while some modern taggers combine the two approaches.

Rule-based taggers, also known as knowledge-based taggers analyze corpus data using a grammatical model. Hence, the information about morphological and grammatical structures is encoded in the program (possibly using a meta language) rather than being "learned" from a training corpus. The first automatic tagger, TAGGIT, is an example of a rule-based tagger (Greene and Rubin, 1971). Rule-based taggers can often correctly analyze complex and long structures, but they are generally unable to provide tags for constructions that have not been recognized.

Probabilistic taggers use statistical algorithms to analyze a corpus. These taggers first need to be trained on a pre-tagged corpus (often tagged manually). Based on this training corpus, probabilistic taggers build a *probability matrix* that stores the probability of an individual word belonging to a certain grammatical class or part of speech. In addition, the matrix stores the probability of having a word of a certain part of speech follow a word from another particular part of speech; this is known as *bigram analysis* since the tagger stores information on pairs of tags. Some taggers can analyze a larger context by storing information about *trigrams* or by gathering information on several nearby wordforms. However, training a probabilistic parser on a bigger context requires much larger training corpora and more memory.

The advantage of probabilistic taggers is that when the tagger encounters an unknown word, it can use the distributional information gathered from the n-grams to determine (or guess) the grammatical class of the unknown word given its nearby context. Statistical taggers can reach high accuracies; however, the results often saturate, at which point the performance of the system can no longer be improved.

Many modern taggers combine both statistical and rule-based methodologies; these systems are known as **hybrid taggers**. Since rule-based taggers can accurately annotate large grammatical constructions, they are used to analyze and mark up a given corpus. Statistical taggers are then applied to disambiguate the results or to guess the tags for the unknown words.

1.5 Applications in Corpus Linguistics

Most often, tagging is not treated as a self-sufficient processing module, but is rather considered as a prerequisite for deeper analysis of text. Hence, taggers are an important and integral part to a number of natural language processing applications, some of which are discussed in more detail in this section. The section also outlines some of the main applications in natural language processing that employ text corpora for extracting information or for training purposes.

1.5.1 Lexicon Acquisition

Lexical acquisition has become an essential phase in building a natural language processing system because the final performance of the model is often dependent on the quantity of the computational lexicon associated with it. The morphological, syntactic, semantic and pragmatic information provided in the lexicon play an important role in applications such as information extraction, document summarization or machine translation. Hence, a computational lexicon should contain information about all potential word forms that the system may encounter in order to guide the text analysis. Different natural language processing tasks, however, may require varying levels of detail or types of information.

Taggers, and in particular part of speech taggers, are the preliminary phase to most NLP applications. To accomplish this task successfully, the system should be able to identify the possible wordforms in the language. This could be accomplished by listing all morphological forms of the words in the lexicon, of course, but most systems include a stemming or morphological analysis module that reduces each word to its lexeme form. Lexemes are then looked up in the computational lexicon.

Typically therefore, the lexicon associated with a POS tagger consists of the lexemes or stem forms of each potential word along with information on their grammatical categories (i.e., noun, adjective, verb, conjunction, etc.). Semantic or subcategorization information is not needed in POS tagging tasks unless the result is to be used as the input to a parser or other more complex applications. If the tagger is to be used as part of a machine translation system, the lexemes should also be associated with translations for the target languages. In addition, lexemes for irregular elements (e.g., irregular plurals in English) may need to be marked explicitly in order to constrain morphological or syntactic analyses.

Most taggers use a statistical model to select between ambiguous tags. These approaches can achieve a high level of accuracy (close to 96-98%) but not if the original POS tags were wrong in the lexicon. Although statistical taggers can 'guess' the part of speech for words that were not found in the lexicon, this reduces the accuracy of the tagged corpus.

Since tagging is generally the first step in many NLP applications, and its output is directly fed into more complex tasks, taggers should have a very low error-rate. And to achieve that, it is crucial that the module be associated with a lexicon with high coverage of the words in the language and an accurate list of attached parts of speech.

If the tagger output is used in a parsing application, then it is important to also include subcategorization information, which associates a verb or noun with its related constituents. This information is of particular importance in rule-based systems since the subcategorization knowledge can restrict the number of potential rules that could apply during parsing. This information is also utilized by statistical approaches in disambiguation of phrasal boundaries since the syntactic context can constrain the choices. Thus, a more detailed lexical entry can constrain the possible parses, but it may also give rise to more complexity and less speed during parsing. As always, each decision made about the structure of the lexicon should attempt to find a balance between the complexity of the lexical items and the overall performance of the system.

Certain natural language understanding applications such as a question answering system, as well as information extraction tasks, use information from word senses to draw the relevant inferences. The computational lexicon associated with these models needs to reflect semantic relations such as synonymy/antonymy as well as the hierarchical relations between words in order to provide an analysis for the 'meaning' of the sentence.

The information contained in a computational lexicon can range from a list of wordforms or a list of lexemes to a highly structured and fine-grained lexicon with full part of speech, morphological, semantic and subcategorization information. More recent works such as the Generative Lexicon approach have argued for a more structured lexicon that incorporates insights from linguistic theory and studies on event structure to represent lexical items (Pustejovsky, 1995).

Traditionally, lexicons in computational linguistics were small and were constructed mainly by hand. However, as we saw in this section, a complete and accurate lexicon can immensely improve natural language processing. Furthermore, building a lexicon manually is extremely labor intensive and costly. Nowadays, lexical information is readily available in the form of *machine readable dictionaries* (or MRDs). Yet MRDs may lack information required by a computational application. Furthermore, MRDs are not available in all languages and oftentimes, lexical resources for lesser-studied languages need to be prepared from scratch. MRDs and regular dictionaries are static in nature and sometimes do not cover new words in different language registers. It is even more difficult to find a bilingual lexicon that contains direct and unambiguous translations for machine translation purposes. Thus, the fundamental problem of lexical acquisition is how to provide the lexical information and quality required for computational systems.

Recently, attempts have been made in the lexicon acquisition community to gather lexical information from text corpora. The appeal for using automatic methods to extract lexical knowledge is in that corpora are now widely available allowing the researcher to collect a representative sample of written or spoken language that reflects the dynamic and changing nature of language. In addition, the ability to automatize the process facilitates the creation of large lexicons without the labor-intensive manual work traditionally involved in the process.

There is a vast field of corpus-based lexicon acquisition that has utilized online textual material to extract lexical knowledge. The following quote outlines some of these investigations:

> "Corpus processing techniques demonstrated, among other things, how certain categories of lexical properties could be identified by means of empirical studies of word occurrences in large bodies of text. For instance, paired corpora in two languages provide evidence that a lexicon could be induced from alignment of texts which are translations of each other; word collocations can be distinguished from coincidental co-occurrences; semantic analysis of phrasal segments points to evidence for regular behavior of certain word classes; conversly, analysis of patterns of parsed (or otherwise structurally annotated) texts revealed the potential for deducing semantic information about lexical classes." (from Boguraev and Pustejovsky (1995), page 3)

1.5.2 Partial Parsing

Partial parsing (or underspecified analysis) is often used in natural language processing to tag chunks of linguistic text rather than the full syntactic structure. Typically, partial parsers are limited to locating noun phrases in a corpus, but have also been applied to prepositional phrases. In certain cases, the parser may assign grammatical functions such as subject or object to a noun phrase. These partial parses, based on linguistic constituents, are crucial for improving the quality of entity and fact extraction modules and are often used in question answering systems as well. Partial parsing has also been used to train probabilistic algorithms to automatically learn grammar rules or to disambiguate tagger results.

Partial parsing emerged as a response to the difficulties inherent in full syntactic parsing due to the ambiguity of grammars and length of sentences. Instead, partial parsing techniques provide reliable syntactic information while sacrificing the depth of analysis or the completeness that could be obtained from a full parse.

In many systems, the partial parser is built on top of a tagger. *Chunks* of sentential structure can be recovered by searching for patterns of regular expressions over the output of the tagger. In partial parsing, there is no resolution of attachments or subcategorization information. Typically, relative clauses are not handled in a partial parser either, rather the NP chunks are recognized and tagged separately. An example of a partially parsed sentence is given below (from (Abney, 1996a)).

```
[S
  [NP The resulting formations]
  [VP are found]
  [PP along [NP an escarpment]]
][RC
  [WhNP that]
  [VP is known]
  [PP as [NP the Fischer anomaly]]
]
```

The simplest method used for partial parsing is to delimit the boundaries of a phrase. The first step for partial parsing of a noun phrase, in that instance, would be to distinguish the elements that could not belong to a simple NP (e.g., conjunctions or verbs). In many languages, morphological cues are available for determining the delimiters of a noun phrase such as case-marking. In other languages, the lack of a boundary may be morphologically marked such as in "idafa" constructions in Arabic, which are equivalent to possessive constructions in English as far as the meaning is concerned.

If partial parsing is used as a precursor to full syntactic or semantic analysis, then the chunks recognized can be linked to each other by lexical association or knowledge of syntactic frames. In certain applications, however, a full noun phrase or prepositional phrase parse is not required. For instance, in recovering noun phrase chunks for information extraction modules, the partial parse of a noun phrase should consist solely of the "meaningful" elements. In other words, function words such as determiners or even adjectives are not included in the rules for partial parses of the noun phrase, since they do not contribute to the main meaning of the nominal element being searched by the information retrieval machine. Hence, in these applications, a "noun phrase" may refer to something that is very different from the traditional linguistic usage. A detailed account of partial parsing is provided in Butt and King, Chapter 6, this volume.

Partial parsing modules can be built as manual symbolic rules that are used by the parser to recognize syntactic chunks. But many contemporary systems use machine learning techniques in an attempt to acquire simple heuristics. Others may use a statistical tagger as the underlying module but use rules for special cases. The statistical and machine learning methods for partial parsers generally require a large training corpus of manually tagged text. Hence, annotated corpora with explicit boundary markers for noun phrases is a prerequisite for partial parsing applications.

1.5.3 Word Sense Disambiguation

The word *bat* means a nocturnal animal, a baseball instrument or the blinking of an eye. The goal of automatic **word sense disambiguation** (WSD) models is to select the appropriate meaning to a given word based on the linguistic context. Hence, the system should be able to determine that *bat* in the sentence *he picked up the bat and hit the ball* refers to the sports apparatus meaning of the word, while in the sentence *he saw the bats fly out of the cave*, *bat* refers to the animal. The automatic resolution of ambiguous meanings is a very difficult task for computer systems, yet it stands at the basis of a number of very important natural language applications, such as machine translation, information retrieval and parsing.

The meaning of a word in a particular usage can only be determined by examining its context. Thus, by looking at an ambiguous word such as *bank* – the often cited example of polysemy, the disambiguator is supposed to determine the sense invoked in the text, e.g., whether the word refers to a financial institution or to the rising ground bordering a river. Word sense disambiguation is an essential task in machine translation applications since the system needs to be able to distinguish the sense of *bank* in an English text and determine whether it should be translated into the French *banque* (for the financial institution meaning) or *rive* (for the river bank meaning). An information retrieval system needs to be able to detect only the documents where the keyword being searched is used in the appropriate sense. For instance, if the query is for 'financial banks' then the system should eliminate occurrences of *bank* used in the river edge sense. Word sense disambiguation has mostly been used to enhance part of speech tagging by distinguishing senses among homographs belonging to the same syntactic category, or to restrict the possible parses of a given sentence. Content or stylistic analyses often use WSD to analyze the distribution of words signalling a given concept or theme.

Clearly, a part of speech tagger and shallow parser will simplify the process of word sense disambiguation by distinguishing the meanings of the different syntactic categories. Hence, the WSD module does not need to differentiate the verb meaning of *bat*, as in *he didn't bat an eye*, from the noun meanings. In addition, selectional restrictions can help determine the sense intended for a given word. For example, the word *star* can refer to a celebrity or to a celestial body. Given the sentence *the astronomer married the star*, however, if the system knows that the direct object of the verb *marry* is constrained to being ANIMATE, then only the celebrity meaning of *star* will be selected since it belongs to the class of ANIMATE things. Word sense disambiguation models thus often take advantage of world knowledge information in order to determine the intended meaning of a particular word. Other WSD systems also use less rigid *selectional preferences* or *associations* in disambiguating polysemous words in a text. An interesting example that clearly demonstrates the use of word associations is provided by the sentence *John put the pot in the dishwasher*, where the word *pot* is ambiguous. Using meaning associations, the word sense disambiguation system should be able to determine that the sense of COOKING VESSEL for *pot* and MACHINE FOR WASHING DISHES of *dishwasher* are closely related, which would inhibit the linking of the sense of HERB or ILLEGAL DRUG of *pot.* However, if the full context of the sentence is *John put the pot in the dishwasher; he could see that a police car had pulled up in his driveway*, then the meanings associated with the word *police car* would activate the ILLEGAL DRUG meaning of *pot* overriding the previous selection.[9]

There have been three main approaches to word sense disambigua-

[9]Example taken from syllabus by a course taught by Michael Gasser at Indiana University.

tion. *Knowledge-based* approaches employ lexical knowledge sources and ontologies with information on selectional preferences, concept associations, and syntactic categories. In *corpus-based* disambiguation methods, the semantics of each polysemous lexical item is marked in a training corpus. A machine learning algorithm is then used to form a representation for each of the senses. *Hybrid* approaches combine both methodologies. As in all other NLP applications, the use of corpora has increased in recent years and a manually annotated training corpus is often used to train statistical sense disambiguating modules. The semantic features determining the meaning of each ambiguous word is marked manually in the training corpus. Some researchers have tried to match senses using an *aligned parallel corpus* (see next section) in order to avoid hand-tagging the text.

Statistical based models that use *supervised learning* to attribute senses to ambiguous words need a sense-tagged corpus for training purposes. The sense with the highest probability computed on the basis of the training data is selected by the system. As in other corpus-based applications, a statistical model for WSD requires a large training set. For word sense disambiguation applications, in particular, enormous amounts of text are needed in order to ensure that all senses of a polysemous word are represented. There exist two possible methods for assigning a sense to an ambiguous word: the use of **distributional information** and the use of **context words**. *Distributional information* is the frequency distribution of a word's senses, while *context words* refers to the window of words found to the right and left of a certain word. The statistical machine trains on the information about the sense frequency as well as the contexts of previously disambiguated instances of a word in the training corpus.

The training data in a text corpus are annotated with sense labels that distinguish the various meanings of a word. For instance, to differentiate the two senses of the word *sentence*, the item may be tagged as *sentence-judicial* or *sentence-grammatical* depending on the context and the meaning. One of the resources most often used in WSD applications is WordNet, created at Princeton University, which organizes lexical information about English nouns, verbs, adjectives and adverbs, in terms of word meanings rather than word forms. The lexical items are arranged into synonym sets, each representing one underlying lexical concept.[10] Similar resources are available for specific domains. For example, the U.S. National Institute of Health has constructed an an-

[10]The home page for WordNet is located at
`http://www.cogsci.princeton.edu/~wn/`. To search for various senses of words in WordNet, visit `http://www.vancouver-webpages.com/wordnet`.

notated training corpus for the Biomedical domain consisting of about 5000 instances of highly frequent ambiguous medical concepts[11].

1.5.4 Discourse Analysis

The goal of discourse analysis is to analyze text segments larger than the sentence. This is in contrast with linguistic analyses that are concerned with the study of grammar or of word structure. Researchers working in this field of computational linguistics examine the larger context of the discourse in order to understand how it affects the meaning of the sentence or of the document. Computational approaches to discourse analysis have been increasingly focused on empirical and corpus-based methodologies to investigating textual meaning. These approaches use a tagged corpus that has been annotated with various discourse phenomena in order to induce algorithms for statistical discourse analysis modules. Hence, the collection and annotation of corpora is a prerequisite to investigations in discourse.

Newspaper articles and third person narratives may contain passages that objectively narrate events, but they also comprise sections that express the characters' or the writer's thoughts, perceptions and inner states. One of the main research topics in discourse analysis is concerned with determining the psychological *point of view* (POV) of the author or character of a text. This could be useful, for instance, if an information analyst is looking for opinions in the press about a specific event, or if the goal is to separate fact from beliefs and attitudes. The current approaches to determining the point of view in computational linguistics have focused on creating annotated corpora which are then used to train a statistical discourse tagger module to automatically mark a corpus for point of view expressions. However, annotating corpora for such discourse items is extremely difficult.

In order to annotate corpora for point of view analysis, humans need to manually mark up texts with relevant tags to determine whether a certain use of language is opinionated or expresses a certain attitude. There are really no formal criteria for identifying these point of view properties. Although several schemes have been proposed in the literature to tag elements that express the opinion of a person, an emotional state or a mental perception, the annotation depends heavily on human intuition. Furthermore, words should be analyzed within the context of use, since they may convey a different perspective or state based on the usage within a sentence or in the larger discourse. GATE is a graphic interface tool that is available freely from Sheffield University

[11] http://wsd.nlm.nih.gov.

and which can be used for annotating opinions in newspaper articles (see `http://gate.ac.uk`).

As an example, let us consider the annotation standards used at the University of Pittsburgh computer science department for tagging discourse corpora. In general, annotations are applied to expressions of opinion or attitude (e.g., *John hates Bill* or *what an idiot!*). Tags are also used to mark the source (i.e., the person whose opinion is being expressed) and to determine the type of opinion (e.g., negative or positive). Some opinions or states can be inferred. For instance, in the sentence *the people in the front of the room began to spontaneously applaud the President*, the expression *applaud* indicates a positive emotion or evaluation. Sometimes, words expressing an opinion are more subtle. For instance, the italicized elements in 'the *so-called* expert' or 'John *blathered on* about something' express the writer's negative evaluation or perception. Another instance of a writer's attitude is represented by the famous example of *terrorist* vs. *freedom-fighter*, since the choice between these two words for describing a particular political group can clearly indicate the opinion held by the writer or speaker.

Another key task in natural language processing applications has as its goal to resolve anaphoric references in texts. This is of particular importance in applications such as machine translation, information extraction and question answering systems. Recently, the computational treatment of anaphora has moved away from methods using extensive linguistic knowledge and has instead focused on corpus-driven methodologies. Corpora are often annotated with coreferential links that can then be used for empirical research, for training a statistical machine to develop new rules and patterns, or for testing and evaluation of the implemented approaches.

Mitkov et al. (2000) define **anaphora** as 'the linguistic phenomena of pointing back to a previously mentioned item in the text' and **coreference** is described as 'the act of referring to the same referent in the real world'. According to these definitions, there may exist anaphora that do not involve coreference (e.g., *every participant*$_i$ *had to present his*$_j$ *paper*). In theoretical linguistics, anaphoric resolution is mainly concerned with the occurrence of anaphora and their antecedents within a sentence. In computational discourse research, the resolution of anaphora studies the context beyond a sentence and often requires the use of world knowledge. Hence, in computational treatments, anaphora normally deal with the content of a document, whereas coreference can operate across documents.

Annotating anaphora in a corpus is an extremely difficult and labor-intensive process, since determining which items should be tagged and what type of annotation scheme should be used is not clear-cut and there are no standards. This task is particularly difficult when the antecedent is located beyond the boundaries of the immediate sentence in which the anaphor appears. Coreference annotations often vary since they are subject to human judgment and require an incredible amount of concentration. In addition, certain annotation schemes only tag coreferential elements while others also tag quantificational noun phrases, such as *every linguist* or *many students*, which are sometimes treated as non-referring noun phrases.

The *Lancaster Anaphoric Treebank*, which consists of a 100,000 word corpus collected from Associated Press sources in 1991, is tagged for noun phrase and pronominal anaphora coreferences and ellipsis. There is ongoing work in Xerox Parc and University of Stendahl in Grenoble which is expected to deliver a one million word annotated corpus in French. Parallel corpora annotated for anaphoric resolution are extremely rare. For further information on various annotation schemes and tools, the reader is referred to Mitkov et al. (2000).

1.5.5 Machine Translation and Parallel Corpora

One of the most important applications in natural language processing is Machine Translation (MT), the goal of which is to automatically translate text or speech from one language to another. There are numerous approaches to the MT problem ranging from full knowledge-based systems for a single language to *interlingua* methods, with knowledge representation formalisms for various languages, to purely statistical approaches to machine translation. Most current systems are a mix of probabilistic and non-probabilistic modules.

Machine Translation includes many of the NLP components already discussed such as a multilingual lexicon, a tagger, a parser and a word-sense disambiguation module, all of which take advantage of text corpora either for extracting knowledge or for training the statistical components of the system. Specific to MT is the use of **parallel corpora**, which consist of the same text translated into several languages.

Before a parallel corpus can be used, however, it needs to be aligned. The objective in **text alignment** is to match sentences, phrases and words in the two texts. Once we create an explicit link between the elements that are mutual translations, we obtain an **aligned corpus** that can be used to create bilingual dictionaries or parallel grammars. An aligned corpus can also be helpful as a knowledge source for word sense disambiguation or for multilingual information retrieval.

Text alignment is not a simple task. Often translations are not reflecting the original structure or wording, since human translators often rearrange the text to obtain a better flow in the target language or to convey certain idiomatic usages. Oftentimes, languages organize their sentence structure quite differently. One of the best-known examples in linguistics is the way in which English and French represent manner of motion. As the examples below illustrate, the manner of motion is expressed by the verb in English while it is represented by the adverb in French.

(7) The boat floated across the river.

(8) Le bateau a traversé la rivière en flottant.
the boat crossed the river floating

Furthermore, translations do not necessarily correspond directly at the word level. For instance, the English verbal expression *make an announcement* would be translated as the single verb *annoncer* in French.

In general, the more accurate and literal the translations are in the parallel corpora, the easier it would be to automatically align sentences and lexical items. Section 1.2.1 introduced parallel corpora obtained mainly from governmental sources which provide a large supply of multilingual texts with rather literal translations of each other. Other multilingual online texts, such as religious texts or translations of literary works, can also serve as parallel corpora, but these texts usually contain less literal translations. In addition, texts comprising languages that are very different from each other (i.e., do not have cognates, are not written in the same alphabet, and have very distinct syntactic structures) will be more difficult to align automatically.

Many researchers have examined methods for aligning sentences in multilingual parallel texts. Some aligned large quantities of text by comparing the lengths of textual units. Other approaches use lexical content to match sentences across languages. For instance, Church (1993) aligns texts by matching cognates (such as the French *flotter* and the English *float*) and similar character strings. Kay and Röscheisen (1993) uses a parallel alignment of lexical items in order to match up sentences in parallel texts. Making matches at the level of the word can be done using a small bilingual dictionary or by examining the distribution of the words within sentences in the two corpora. Once sentences are aligned in parallel texts, the words and phrases can be matched in order to extract corresponding translations for bilingual dictionaries. In addition to developing bilingual lexicons, parallel corpora have been used for extracting grammar rules or structural patterns from the text.

Statistical machine translation employs probabilistic methods for learning from aligned corpora. As in all cases discussed earlier, the quality of a statistical machine is directly related to the accuracy of the training corpus, and therefore, the alignment of the textual corpora is crucial for training a statistical MT system. In order to make serious improvements in translation quality, however, the statistical translation model needs to be able to train on more detailed linguistic features such as the part of speech, the subcategorization frames of a verb, and lexical co-occurrence probabilities for nonadjacent items. Hence, to improve statistical MT models, parallel corpora are usually annotated to varying degrees.

The Natural Language Group at the University of Maryland, for example, takes advantage of existing lexicons, morphology tools, ontologies and other resources for English. The English corpus is manually annotated but the target language annotation is created via a projection from the English side using automatic alignments of the parallel texts. The target language corpus is then used as a training corpus for various machine learning components for taggers, morphological analyzers, dependency parsers, noun phrase groupers, word sense taggers and translation models.

Corpus-based, statistical methods are also used in speech translation applications where the translation rules are learned automatically from a parallel corpus of spoken language. Although the probabilistic translation paradigm reduces the time spent in building a machine translation system, it also relies completely on the availability of an accurate parallel corpus; thus, MT models for speech provide good results in domain-specific applications mainly.

1.6 Corpus Processing Tools

This section introduces the reader to some of the basic tools needed for working with corpora. There are a number of techniques and operations that can be performed on unmarked language corpora in order to analyze the data. Basic operations are used to count words or to find the common contexts in which words appear. Typically, existing tools and some limited programming is enough to manipulate text corpora and perform low-level analysis. Since the most common and important set of tools are the UNIX system tools, the main chunk of this section is dedicated to the description of their usage.

1.6.1 UNIX Tools

Almost anyone who works with language corpora will need to use the UNIX tools at some point, hence it is important that researchers in computational linguistics be familiar with them. Usually, one can find a pre-installed text editor, such as Emacs or vi, on UNIX systems that can be used to look at the files and to edit their contents.[12]

Generally, UNIX tools use **regular expressions** to detect certain patterns in a text. Regular expressions are a powerful tool that allow the researcher to find complex patterns when a match against a full word or sequence of characters is not possible. For instance, to search for any English word containing only capital letters, the regular expression `[A-Z]+` can be used, which allows for any sequence of one or more capital letters between A and Z. If you are looking for the word *allottment* in a corpus but realize that there may be various misspellings or forms of the word, you may want to try the following regular expression pattern:

```
(a|A)l+ot+[a-z]*
```

The `(a|A)` will match either an uppercase or lowercase letter 'a', followed by a letter 'l' which may appear one or more times (indicated by the +). Similarly, the pattern allows for one or more 't' letters in the word. The string *allott* can also be followed by other lowercase letters or not; the Kleene star * means zero or more occurrences of the immediately previous character or regular expression. For an introduction to regular expressions, see Jurafsky and Martin (2000), Chapter 2.

1.6.2 Word Counts

Once you have downloaded some text, you may wish to perform a **word count**, by counting the total number of words in a text or by counting the number of times each unique word appears in a text. This could then be used, for instance, to extract only words that appear in the text more than a certain number of times. Also, frequency numbers associated with word lists are often used by statistical machines to assign weights in order to disambiguate information. UNIX already contains some tools that would facilitate this task.

In order to count the number of words in a text, the command `wc` (for word count) is used. Consider the text file below named `buckyball.txt`.

```
A few years ago, scientists made an exciting breakthrough
that could revolutionize the world of science, with
particular emphasis on chemistry. Called
buckminsterfullerene, or the buckyball, for short, this
newly-discovered molecule has scientists the world over
scrambling to find new and unusual uses for this unique
molecular oddity.
```

[12] A convenient version of these tools for use on Windows machines can be found at `www.cygwin.com`.

```
molecular oddity.
```

To find out how many words exist in your corpus, type

```
wc -w buckyball.txt
```

You will see that the result is 51 word tokens. You may have noticed, however, that the last line of this file is repeated twice. One option is to use the command `uniq` which can be used to remove duplicate adjacent lines. So typing

```
uniq buckyball.txt | wc -w | more
```

removes the last line, counts the number of words and then prints the result on the screen.[13] This time there are 49 word tokens in the file.

However, there is a difference between a **word token** and a **word type**. If one is interested in the length of a text, then a word token count will suffice. But if we are interested in determining the number of different words in a text, then we need to count the occurrence of word types.

The command `tr` is used in UNIX to translate sets of characters. For example, the following command will replace all the occurrences of the vowels *e*, *o* or *i*, in the text file by *X*. The left angle bracket indicates that `buckyball.txt` is the input file.

```
tr ‘eoi’ ’X’ < buckyball.txt | more
```

The result of this command is shown below:

```
A fXw yXars agX, scXXntXsts madX an XxcXtXng brXakthrXugh
that cXuld rXvXlutXXnXzX thX wXrld Xf scXXncX, wXth
partXcular XmphasXs Xn chXmXstry. CallXd
buckmXnstXrfullXrXnX, Xr thX buckyball, fXr shXrt, thXs
nXwly-dXscXvXrXd mXlXculX has scXXntXsts thX wXrld XvXr
scramblXng tX fXnd nXw and unusual usXs fXr thXs unXquX
mXlXcular XddXty.
mXlXcular XddXty.
```

`tr` can also be used to tokenize a text, by displaying each word on a separate line. This can be achieved by ‘translating’ all non-word items into a new line character (ASCII code 012 or `\n`), where non-word items include punctuation marks, numbers and space characters, and then taking the complement. Hence consider a new shorter file called `buckytest.txt` with the following content:

[13]The pipeline or | is used to represent a chain of commands in UNIX whereby the output of the first command is fed to the second one, and so on.

TABLE 1 Word Tokens from Buckyball file.

The	the	known
buckyball	early	forms
is	s	of
a	Before	pure
new	this	carbon
form	exhilerating	to
of	discovery	be
carbon	there	found
previously	were	on
undiscovered	only	Earth
until	two	

```
The buckyball is a new form of carbon, previously
undiscovered until the early 1980's. Before this
exhilerating discovery, there were only two known
forms of "pure" carbon to be found on Earth.
```

The following command will produce a list of new lines (or blank lines) with several punctuation marks in between.

```
tr 'a-zA-Z' \n < buckytest.txt | more
```

Since we actually want the words and not the punctuation marks we use option `-c` to map every non-letter or complement of letter into a new line.

```
tr -c 'a-zA-Z' \n < buckytest.txt | more
```

This in effect replaces all punctuation, numbers and space characters by a new line. If you try this command you will notice that where we once had punctuation or number characters, we now have blank lines. To eliminate these blank lines, the option `-s` is used to make sure that multiple cases of `\n` do not occur. The command

```
tr -cs 'a-zA-Z' \n < buckytest.txt | more
```

produces the result shown in Table 1.

Table 1 is a wordlist of tokens that appear in the file `buckytest.txt`, but it includes duplicates of the words *of* and *carbon*. This can be seen more clearly if we sort the words alphabetically by using the `sort` command.[14]

```
tr -cs 'a-zA-Z' \n | sort | more
```

[14]Note that this command only sorts according to the English alphabet.

TABLE 2 Sorted unique word tokens from Buckyball file.

Before	forms	the
Earth	found	there
The	is	this
a	known	to
be	new	two
buckyball	of	undiscovered
carbon	on	until
discovery	only	were
early	previously	
exhilerating	pure	
form	s	

In order to remove these duplicates and obtain a list of word types only, the following command can be used:

```
tr -cs 'a-zA-Z' \n | sort -u | more
```

The option -u refers to *unique*. The result of the unique-sort is shown in Table 2.

One of the most common operations performed on a text file is to compute the number of times a certain word occurs in the text. This can be achieved by the command `uniq -c`. The option `-c` gives the number of times a line occurred. Hence, if you sort a wordlist, then run the `uniq` command to strip off duplicate lines, the option `-c` can be used to announce how many times the line occurred originally. This command is very useful in determining the frequency of occurrence of words in a text file as illustrated below.

```
tr -cs 'A-Za-z' \n < buckytest.txt | sort |
                             uniq -c > buckyball-freq.txt
```

The result in this case is illustrated in Table 3.

The frequency file can be sorted according to the frequency of the words. `sort -n` allows numerical ordering and `sort -n -r` puts the word with the highest frequency on top.

By looking at the results in Table 3, you will notice that there exist a number of errors or redundancies. For instance, you will notice that *the* and *The* are treated as distinct words. If we wish to treat both capitalized and lowercase instances of a word as a single word type, then we could convert all words to either uppercase or lowercase. This is appropriate in determining word frequency or when analyzing the usage of definite articles in English noun phrases. However, removing

TABLE 3 Word frequency from Buckyball file.

1	Before	1	form	1	pure
1	Earth	1	forms	1	s
1	The	1	found	1	the
1	a	1	is	1	there
1	be	1	known	1	this
1	buckyball	1	new	1	to
2	carbon	2	of	1	two
1	discovery	1	on	1	undiscovered
1	early	1	only	1	until
1	exhilerating	1	previously	1	were

distinctions between upper- and lowercase letters may raise ambiguities at later stages. For example, this operation may remove the capitalization from proper names thus making it difficult to recognize them as such.

Another redundancy in the wordlist in Table 3 is that the singular and plural forms of the noun *form* are treated separately. In more involved cases, we may even wish to treat *were* and *be* as a representative of the same word in order to get an accurate count of word types.

This can be achieved by *lemmatization* or *stemming*, which strips off the affixes on a word to return its lexeme or stem form. Thus, a stemmer module needs to be applied to our `buckyball-freq.txt` file in order to strip off the plural ending of *forms* and to represent the irregular verb form *were* as *be*. Note that a stemmer does not provide a morphological analysis of the word; it simply strips off the inflectional affixes and returns the corresponding lexeme(s). For example, a stemmer need not know whether the use of the form *lying* is related to the verb meaning 'telling untruth' or the verb meaning 'to prostrate oneself'. All the stemmer needs to know is that the stem form is *lie*.

Finally, perhaps we do not want to consider the *s* character (remaining from the 1980's) as a separate word. This brings up the discussion addressed in the section on Tokenization on how to treat cliticized elements like *n't* or *'ll*. Some systems apply a preprocessor to separate these into the words *not* and *will* respectively, before removing all punctuation from the text. The *'s* is a more difficult problem since it could represent an abbreviated form of the verb *is*, the possessive marker, or in the Buckyball example a marker of decades or dates. In such cases, either preprocessing is required to distinguish these cases or a morphological analyzer is used. In some cases, you may want to include

TABLE 4 Word Type frequency from Buckyball file.

2	the	1	only
2	of	1	on
2	form	1	new
2	carbon	1	known
2	be	1	is
1	until	1	found
1	undiscovered	1	exhilerating
1	two	1	earth
1	to	1	early
1	this	1	discovery
1	there	1	buckyball
1	pure	1	before
1	previously	1	a

the frequency of occurrence of the *'s* regardless of its function – for instance, if you would like to know what type of words should be accounted for in your lexicon or stemmer. For our purposes here, we could simply eliminate all *'s* occurrences after numbers prior to applying the `tr` command.

If we apply all these changes to our sample file and run a frequency count on the result, we will obtain the results shown in Table 4.

Another useful command in UNIX is `cut` which allows one to extract a whole column from a text. For example, in the previous results, the word types are associated with a frequency number. But let's say you are interested in knowing the word types of the text file but do not care about the frequency numbers. In that case, you can use `cut` to extract only the second column of word types and place it into a new output file.

```
cut -f2 -d' ' < buckyball-freq.txt > buckywords.txt
```

The option `-d` is used to indicate the type of delimiter separating the columns. The default delimiter for the cut command is the tab, hence if the text contains a different type of delimiter such as the space in this case, it has to be explicitly provided inbetween the apostrophes following the option `-d`. In other words, if the delimiter in the file is a comma, then the command would be written as

```
cut -f2 -d',' < buckyball-freq.txt > buckywords.txt
```

In analyzing corpora, it is sometimes useful to be able to locate a certain pattern in a particular place in the text. The UNIX command `grep\index{grep}` can be used for this purpose. Some useful options available with `grep` are `-i` for ignoring the case of the character. The option `-c` displays the number of matching lines and `-v` displays all lines except for the ones containing the pattern specified. Some examples:

`grep 'support'`	find all lines containing the word ‘support’
`grep ^[0-9]`	find all lines beginning with a number
`grep -i ^[aeiou]`	find all lines starting with a vowel, regardless of case
`grep -v 'support'`	find all lines that do not contain the word ‘support’

In addition to the UNIX tools described here, the shell languages SED and AWK are also very powerful in manipulating text. However, the most widely used programming language for corpus manipulation is PERL, which is available free of charge. PERL is very easy to pick up and allows the user to utilize regular expressions to search for patterns and perform powerful string manipulations.

1.6.3 Concordances

When building a computational lexicon or while developing a grammar for a language, it may be beneficial to investigate the context in which certain words appear. By studying the context of a verb, for instance, it is often possible to figure out what the selectional requirements of the lexical item are and whether there are intervening elements between the verb and its arguments. This can be done by automatically extracting any occurrence of the word under investigation, along with a number of the tokens to its left and right. In other words, the word of interest is given with its context of use. The surrounding context can then be used to extract information manually by studying the data or as a learning tool for statistical parsers. The tool typically used for such studies is the *Key Word in Context* or KWIC **concordancing** program. This tool extracts all occurrences of the word of interest and displays it with the word lined up in the center and the surrounding context on the two sides. A good place for getting a sense of what concordance results look like is the Bank of English demonstration page[15].

As an example, consider we are interested in determining what type of particles are associated with the verb *pick* in English. In order to accomplish this, we search for all occurrences of the verb within our corpus. Some of the Bank of English results for this query are displayed

[15]`http://titania.cobuild.collins.co.uk/form.html`.

Why by 10 Celsius, to	pick	a number, and not by 12 or 8 ? And,
somebody was trying to	pick	a fight, I would always see to it
blow and my neck ached.	Pick	her up and carry her up the hill,"
isn't going to	pick	him up. But the military radar at
then the state has to	pick	it up. Somebody has to pay for it.
are necessary, then	pick	one or two stocks or equity funds,
Take your time,	pick	out your targets. Let's keep them at
the bristles retract. We	pick	pink, green and/or red for you.
behaviors. They	pick	the baby up when he cries, wait for
unannounced, waiting to	pick	up the keys to the cabin they rented
it means that teens	pick	up most of their sexual knowledge
that we talked about.	Pick	up the pace as quickly as possible
here and that you can	pick	up the telephone and call her or–
a Volvo 340, a Sierra	pick	-up and a Ford Capri. All were local
or stripping only.	Pick	-up and delivery. Fast service.
schedule until the 9 [f] p.m.	pick	-up time. [p] Facilities for
were only too glad to	pick	up the tab - not only for this
investors are starting to	pick	up, but you also have France and

FIGURE 3 Concordance results for the verb *pick*.

in Figure 3.

As is clear from these partial concordance results, the verb *pick* is most often used with the particle *up* in a transitive setting, but it also appears with the particle *out* as in *pick out your targets*. In addition, these results show that *pick* can also be used on its own as a transitive verb as in *pick one or two stocks*. Hence, if the lexicographer is to mark subcategorized particles for this verb, he or she will need to include *up* and *out* as particles associated with the lexical entry.

Note also that *pick-up* with a hyphen is used as a noun in *Pick-up and delivery*. In the automotive meaning, *pick-up* can be used either as a noun (*Sierra pick-up*) or as an adjective (*pick-up truck*). More interestingly, however, the results of a concordance tool can also demonstrate whether there are intervening materials between the verb and the particle. This information is useful for developing language grammars. In the examples in Figure 3, *pick* and *up* are sometimes separated by an intervening pronoun as in *pick her up* or by a full noun phrase as in *pick the baby up* or *pick some things up*. Yet further investigation would show that this is directly related to the meaning of the verb, since the particle cannot be separated from the verb in intransitive contexts such as *regional investors are starting to pick up*.

Furthermore, this kind of concordance tools can be used to locate

fit over built-in appliances to	give	a fully fitted look. Note that
people, the ones who don't really	give	a damn, and the smartasses
have a tangy, salty taste and	give	a pleasant piquance to salads
would result if Pakistan were to	give	active support to the Kashmiri
of your travel arrangements and	give	assistance around the world
soldiers: they had wanted him to	give	away the position of a partisan
is mistaken.j Even if we could	give	credence to the idea of
advisors will also be on hand to	give	free advice and information on
afterwards when he came to	give	her daughter piano lessons. 'He
opposite–come in off the road and	give	herself over to prayer–I suspect
with the other kids. All I did was	give	him a telephone number where
over him at his christening [p] [p]	Give	him an incentive! There will
it was to keep Henry alive, to	give	him heart, for his drowned
Reynolds says it needs them to	give	insight into the motivation of
happily enough. He said he would	give	me a call - he always called.
determined that she would not	give	up her career. [p] A charity for

FIGURE 4 Concordance results for the verb *give*.

certain idiomatic expressions such as *pick up the pace*, *pick up the tab* or *pick a fight*. More examples of idiomatic expressions are available for the verb *give* in Figure 4. By extracting the context in which the verb appears, we can locate a number of idiomatic uses for this verb as in *give a damn*, *give a look*, *give support PP[to]*, *give assistance*, *give away*, *give heart*, *give credence PP[to]*, etc.

Although KWIC is the most common method for looking at concordances, one may also choose to simply extract all the sentences in which a certain word appears, rather than looking at the few tokens to the left and right of the keyword. The programming approach to making a KWIC index is described in detail in Aho et al. (1988).

1.6.4 Collocations

Concordance analysis is a useful tool for investigating the context in which keywords appear, but it is not very efficient for real quantitative analysis. Concordance results are not ordered by frequency and the number of hits may be very high. In order to inspect the context in which a particular keyword appears more efficiently, researchers often turn to **collocation analysis**.

A *Collocation* is an expression that consists of a number of words within a short distance of each other. These words generally correspond to a conventional way of expressing the concept. The degree of idiomaticity of a collocation is not very clear. For instance, an expression such as *decaffeinated tea* or *file a lawsuit* is a collocation as well as the compound expression *weapons of mass destruction* or the phrasal verb *show off.*

In general, however, a collocation is described as lacking **compositionality**. If an expression is compositional, then its meaning can be derived by taking into account the meaning of its parts. An example of a compositional expression is *television show.* Idioms are the uncompositional expressions such as *kick the bucket* meaning 'to die' or *take a shower.* Hence, collocations are generally construed as noncompositional in character. It is crucial to locate such noncompositional expressions in computational lexicography. Since the meaning of the whole cannot be constructed based on the meaning of the parts, such expressions are usually listed in the lexicon. Identifying collocations is also important in parsing in order to ensure that the collocations are given preference and thus the correct parse tree is selected. The study of collocations is also central to research on a contextual theory of meaning or language use.

There are many works within the field of linguistics focusing on what constitutes an idiom or *complex predicate* and in order to determine its properties. In computational linguistics, collocations are sometimes identified by attempting to translate them into another language. If the translation cannot be performed by translating the subparts, then it is to be treated as a collocation. This test is particularly true for building lexicons for machine translation purposes. For example, the English expression *take a test* cannot be translated into French as *prendre un examen*, but rather it should be translated as *passer un examen.* In such circumstances, the lexicographer may choose to list these expressions in the lexicon. Sometimes, collocations are defined in computational linguistics as expressions that are static. In these cases, the test is to see whether the internal elements within an expression could be modified or whether the two elements may be separated from each other by intervening material. For instance, the two parts of the phrasal verb *pick up* seen earlier in the section can often be separated by the direct object argument. Similarly, in all the phrasal verbs (as in *pick up*) as well as in light verb constructions (as in *make a decision*), the first (i.e., verbal) element can take inflectional morphology. If these expressions were listed in the lexicon as a static collocation, then they could not undergo morphological analysis correctly, since they would be treated as a single unit. Within the field of computational linguistics, sometimes proper names are treated as collocations as in *Vaclav Havel* or *New York Stock Exchange.*

There are several methods for computing collocations in a text corpus, the simplest of which is to simply calculate the frequency or the number of times the elements within an expression appear next to each other within the text. It has been argued, however, that neither frequency nor adjacency are sufficient parameters for locating collocations. In a moderately large corpus, a true collocation may only appear a few times: In a 22 million word corpus made of Wall Street Journal and San Jose Mercury Times articles, the expression "emotional baggage" occurred only 3 times; certainly not frequent, but it is still considered a collocation. Also, elements that are considered to be parts of collocations may allow intervening material as in *filing a class action lawsuit.* Some approaches have overcome the word adjacency limitation by allowing a window of up to 4 or 5 words for collocation searches. However, although *dog* and *bark* often appear very close to each other within a sentence, they are not to be treated as a collocation. Some researchers have opted to associate more weight to collocates that occur closer to each other.

Current systems for extracting collocations from text corpora first locate the word under investigation and identify possible concordances. The second step is then to use frequency counts and make more complex statistical observations about the word in question as well as its collocates. What a "statistically significant word pair" consists of, however, is not completely clear. Chapter 5 in Manning and Schütze (1999) presents some of the methods and computations used to detect collocations such as *Mutual Information* and *t-scores.*

Let us take the results shown in Table 5 for the query word *show.* In the first column, the table lists the words that appeared in the corpus with *show.* The second column lists the frequency with which the collocate word appeared within the text; as can be seen from the numbers, the definite article *the* has a very high frequency of occurrence. The third column lists the Joint Frequency which refers to the bigram frequency; in other words, this column refers to the number of times *show* and the collocate appeared next to each other (i.e., within four words of each other). However, just looking at bigrams is not a very accurate method of identifying collocations. As the reader can see from the results in Table 5, the most frequent bigram seems to be *the show*, which is certainly not to be treated as a collocation. Hence, simply taking into account the raw frequency of cooccurrence is not sufficient to detect collocations since it would most often place function words such as *the* and *of* at the top of the list simply because they are the most frequent words in English language in general.

The **Mutual Information** score is often used to extract collocations and it characterizes the extent to which the observed frequency

TABLE 5 Collocate t-score for *show* (Bank of English corpus).

Collocate	**Corpus Freq**	**Joint Freq**	**t-score**
the	2872094	10527	28.968403
to	1375856	5690	27.454145
on	393554	2048	22.379674
tv	9132	384	18.370103
how	58166	572	17.519233
that	594996	2407	17.160445
will	152423	886	16.296037
figures	4908	282	16.024071
at	292324	1311	14.970991
off	52036	397	13.055228
motor	3207	180	12.787644
radio	9684	203	12.459955
chat	1145	158	12.330197
polls	1313	142	11.626544
a	1228514	3925	11.069379
late	11128	161	10.381677

of cooccurrence is different from when there is no collocation in the context of the word under investigation. In other words, the question is whether the word *show* has an effect on the presence of e.g., *the* or *figures* or *off*. If *the* has an overall relative frequency of 1 in 20, then we would expect it to occur with the same relative frequency regardless of the presence of *show* in its context. If, on the other hand, Mutual Information, or rather the difference between the default or expected relative frequency and the actual frequency of cooccurrence with the word of interest, is very large, then this is indicative of a strong collocation possibility.

This approach, however, allows a nonfrequent element that may cooccur with the word under investigation without forming a collocation with it to give rise to very high Mutual Information scores. For instance, the adjective *serendipity* may appear within the context of *show*. Since it is a word of very low frequency (i.e., has low expected frequency), its presence in the vicinity of *show* would be given a very high value (i.e., high observed frequency). To put it simply, there isn't enough evidence for the statistical machine (or for a child learning a language for that matter) to be able to determine with confidence whether two cooccurring words that have been observed only once are to be associated strongly with each other or not. **T-score** is then used to indicate the level of confidence in the decision or how probable a particular proposed collocation really is. Since the frequencies of word occurrences are used to compute the t-score for a collocation, the score is higher for more frequent lexical items.

The choice between Mutual Information and t-score depends on the corpus being analyzed and the purpose. The Bank of English collocation finder takes into account both Mutual Information and t-score results:

> "In practical terms, raw frequency or j(x) won't tell you much at all about collocation: you'll simply discover what you already knew that "the" is a *very* frequent word and seems to co-occur with just about everything. MI is the proper measure of strength of association: if the MI score is high, then observed j(x) is massively greater than expected, BUT you've got to watch out for the low j(x) frequencies because these are very likely to be freaks of chance, not consistent trends. T-score is best of the lot, because it highlights those collocations where j(x) is high enough not to be unreliable and where the strength of association is distinctly measurable [...] If a collocate appears in the top of both MI and t-score lists it is clearly a humdinger of a collocate, rock-solid, typical, frequent, strongly associated with its node word, recurrent, reliable, etc etc etc."

1.6.5 Testing and Evaluation

In addition to a collection of corpora for research and training purposes, there is also a need for consciously created collections of data to be used for evaluating and testing the performance of NLP systems. In particular, when using statistical algorithms, it is crucial to construct a set of **test corpora** distinct from the set of data the statistical machine was trained on, in order to obtain an accurate evaluation. Just as corpora designed for training or research purposes need to be representative and balanced, so do test data.

Typical test collections are annotated for the problem under study. Hence, a **truthfile** or annotated test corpus for a tagger should contain the relevant part of speech labels and the test corpus for a parser should consist of accurately tagged syntactic phrases. The standard measurements to evaluate an information retrieval system are *recall*, the probability that an item may be retrieved, and *precision*, the probability that a retrieved item may be relevant. Other important evaluation measurements are the notions of *accuracy* and *coverage*:

Accuracy = C/P and Coverage = P/T

where C is the number of items correctly inferred, P is the number of items that have been tagged or for which the machine has made a prediction, and T represents the total number of items in the test data.

The number of correct results are determined by comparing the NLP module's output to the truthfile prepared for testing.

For example, when measuring the results of a part of speech tagger for a test corpus of 100 words where 50 tags have been assigned a POS, of which 2 are wrong and 48 are correct, we obtain the following numbers:[16]

Accuracy = 48/50 = 96%
Coverage = 50/100 = 50%

In general, however, in order to obtain a fair evaluation of how the NLP model is functioning, it must be tested on a set of data that have not been seen before. The statistical models, in particular, expect phenomena that they have already been trained on, hence it is important to test on a new collection of data. Furthermore, it is also beneficial to evaluate the model on a structure or word or other types of phenomena that it has never encountered before in order to evaluate the statistical machine's ability to infer in new contexts. Test data should be large enough to be representative of the domain in question and the complexity of the language phenomena under study.

It is generally a good idea to separate a set of data into a training set and a test set before beginning the project. The test data could consist of only about 5 to 10% of the total corpora as long as it is large enough to be representative and reliable. Although some researchers consciously select specific phenomena to test the NLP system on, test data are most often a random collection of corpus material for the language or domain under study. A better methodology would be to have two distinct test sets as well where one is used as a *development test set*. This set will be used for successive trials of the system, the results of which are used to improve the NLP model. The second test set will be used as a *final test collection*.

When constructing a test set or truthfile, the following criteria should be taken into account:

- **Consistency:**
  The researcher needs to provide a predefined annotation scheme that would best reflect the linguistic phenomena being investigated or language properties that need to be marked for the user. This markup scheme is used to create a large enough truthfile in order to evaluate the output of the NLP module.
- **Size of Tagset:**

[16]Sometimes, the researcher may also need to take into account the ambiguity of the tags if the system does not produce disambiguated results.

Although a bigger tagset may capture more detailed distinctions, it is also harder to train for a statistical machine since larger corpora are required, and therefore, the model may give rise to a higher error rate. Note that a larger tagset does not make much difference in manual tagging.

- **Linguistic Content:**
  It is important that the tags or the phrase brackets used in the NLP module correspond to the linguistic phenomena that need to be captured. The contribution that a particular tag or bracketing distinctbeforeion makes to the successive applications should be taken into account when designing the annotation scheme.

The preparation of a training set and that of a test set both require labor-intensive work, since the items in the corpus need to be manually annotated in each case. The training set is used to train the statistical machine in order to induce the patterns in the language phenomena under study, and the truthfile or test set is used to evaluate the final performance of the system. Hence, although statistical systems do not require time spent creating knowledge sources and grammar rules, they do need a lot of time in creating the various corpus collections used for training and testing the statistical algorithms.

1.7 Symbolic and Statistical Paradigms

The field of computational linguistics has historically been characterized by the conflict between the symbolic and statistical approaches to language. This rift, which was triggered by the publication of *Syntactic Structures* in 1957, has at its core the questions concerning the role of quantitative measures and the type of data to be investigated in modeling natural language. The computational approaches to language since the 1960s were mostly dominated by the perspectives of generative theory and concentrated on building of linguistic models based on formal grammars and knowledge sources. Since the 1990s, however, the renewed interest in statistical models, coupled with recent technological advances (see 1.1), have given rise to the importance of large corpora sets within the field of computational linguistics. Other related aspects of the debate have centered on the nature of language, the role of probability in human language computation, and the competence-performance distinction.

These different computational programs that have been at odds for decades have actually emerged from two different approaches to language with very distinct goals and methodologies that are, in fact, complementary since they try to account for different aspects of human language computation. Ronald Kaplan of PARC notes:

> "The statistical and symbolic approaches to language have emerged from different starting points and methodologies and have tended to focus on different goals. The resulting tension and confusion has obscured the fact that both approaches can make crucial and often complementary contributions to a deeper understanding of how language works." (Klavans and Resnik (1996), back cover)

In 1994, a workshop entitled *The Balancing Act* was held at the 32nd Annual Meeting of the Association for Computational Linguistics at the New Mexico State University in Las Cruces, with the goal of starting a dialogue between researchers on both sides of the debate and emphasizing the complementary nature of the two approaches. Since then, the field has seen an increase in the number of hybrid approaches combining symbolic and statistical methodologies to modeling natural language by bringing together the more linguistically motivated approaches and the more corpus-driven systems.

1.7.1 Distinct Goals

In *Syntactic Structures*, Chomsky argued that broad coverage of language data gathered from naturally occurring textual or spoken material is not able to contribute to an 'explanatory theory' of language.[17] According to Chomsky, quantitative approaches would fall under the category of 'descriptive adequacy'. Indeed, most computational linguistic work aim at determining the patterns that govern the observed language data and there is little concern in the field with how humans process or acquire languages. Hence, the main goal of current computational approaches is to obtain a system to be used for practical applications, which can be robust, display high speed and have broad coverage. Since the ultimate goal of these systems is not necessarily advancing the scientific understanding of human language computation, or providing an explanatory theory of the language faculty, then the methodology used for developing these systems can indeed consist of

[17]Chomsky developed a classification of linguistic investigation by outlining three levels of adequacy– observational, descriptive, and explanatory. *Observational adequacy* refers to an approach that provides a description of linguistic facts without attempting to draw a generalization. A *descriptively adequate* theory of language will have as its goal to account for the phenomena of particular languages by formulating a general principle to capture the patterns observed within that language. A grammar or theory of linguistics is considered *explanatorily adequate* if it can explain how this knowledge of the principles of the internal grammar can be acquired by the speakers. In other words, the goal of generative grammar is to formulate a common internal structure for the human language faculty and explain how this internal structure comes into existence in the mind of a speaker.

quantitative approaches.

Thus, once we realize that the goals of theoretical generative linguistics and of most current computational approaches to language differ, then it also follows that different goals may require distinct methodologies. Accordingly, a computational model concerned about practical applications and efficient coverage of observable online data would more readily utilize quantitative methods and large corpora for building the model.

1.7.2 Theoretical Approaches in Computational Linguistics

It should be pointed out, however, that certain computational approaches are more concerned with modeling a linguistic theory to test its hypotheses and predictions than focusing on practical applications of the system. Researchers working within these approaches, although rare in the current state of the field, may indeed be interested in developing a theory of the human language faculty and would fall under the label of 'explanatory' within Chomsky's classification. I will not be discussing these approaches in this section but whether these models use a symbolic, statistical or hybrid approach depends on the type of data being tested and the theories of language adopted in each case.

For instance, modeling the principles defining the internal structure of the language computation system within generative grammar would be symbolic, using the results of the research within theoretical linguistics. On the other hand, a model trying to capture the setting of the parameters in a language acquisition model might best be captured by a probabilistic module, taking advantage of naturally occurring spoken corpora. Whether language should be treated as symbolic in nature or probabilistic is still an ongoing debate in the theoretical approaches to the field. For some discussion of the complementary nature of symbolic and probabilistic methodologies in linguistic theory, see Abney (1996b).

An interesting viewpoint on the rift between symbolic and statistical paradigms, and on the concept of the complementary nature of the various approaches comes from Robert Simmons, suggesting that different aspects or perspectives of language are to be investigated with different methods:

> "My own perceptions of the state of computational linguistics during that period were given in "On Seeing the Elephant" in the Finite String, March-April 1972. I saw it as a time of confusion, of competition among structuralists, transformationalists, and the new breed of computerniks. "On Seeing the Elephant" was a restatement of the old Sufi parable that suggested that we each perceived only isolated parts of our science." (Simmons, 1982)

1.7.3 Hybrid Approaches

In recent years, the two approaches within the field of computational linguistics have become less at odds with each other, and researchers have begun to search for a coherent combination of the two methodologies. There is now a consensus within the field that each approach can provide properties and expertise to achieve their common goals. The use of corpora has been an integral part of these hybrid systems.

One of the main applications of the statistical methodologies has been in the domain of disambiguation. Distributional approaches have demonstrated great success in disambiguating parts-of-speech, subcategorizations, or word senses. Thus, most hybrid approaches have taken advantage of statistical algorithms to detect the correct parse from a set of possible ones provided by a symbolic, rule-based system.

Statistical approaches are more robust and show a high degree of error tolerance. Online textual material often contain errors, misspellings, foreign language words, as well as structural or agreement mistakes, e.g., 'he leave'. A symbolic, rule-based system will not generally be able to process unknowns or process ungrammatical structures. A statistical model, however, may still produce a resulting analysis. In addition, statistical models are better at providing partial parses for linguistic structures that are missing in the grammar.

On the other hand, most statistical models use some kind of symbolic and knowledge information in any case. Hence most models assume a binary branching tree or make use of a language lexicon. It has also been shown that the addition of knowledge sources usually improves the results.

Statistical approaches have been often used to acquire knowledge. For instance, probabilistic algorithms and the availability of large corpora have helped the automated or semi-automated acquisition of lexical knowledge (see section 1.5.1), providing a faster development time of such knowledge sources.

1.8 Summary

This chapter provides an introduction to corpus linguistics. It begins with a brief outline of the history of corpus-based analyses within computational linguistics leading to the recent resurgence of interest in corpus-driven approaches. Various corpus types are defined and corpora resources are presented. The chapter enumerates a number of issues that arise in segmenting a corpus and in determining the word boundaries in computational linguistic applications. Several corpus annotation schemes are discussed, providing general guidelines for designing tagsets.

A large part of the chapter is dedicated to the various computational applications in which language corpora play a crucial role. This is, in essence, the theme that is common throughout all the sections of this chapter since the application of a project often determines the decisions made at each step of corpus collection and processing.

The chapter provides an introduction to various tools and programs used for the analysis of corpora and the performance of text mining. In addition, guidelines for the testing and evaluation of computational modules employing annotated corpora are provided. The final section presents some of the arguments that have led to the recent development of hybrid paradigms in computational linguistics by combining symbolic and statistical methodologies.

References

Abney, Steven. 1996a. Part-of-speech tagging and partial parsing. In S. Young and G. Bloothooft, eds., *Corpus-based methods in language and speech processing*, pages 118–136. Dordrecht: Kluwer Academic.

Abney, Steven. 1996b. Statistical methods and linguistics. In J. L. Klavans and P. Resnik, eds., *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pages 1–26. Cambridge, Mass.: MIT Press.

Aho, Alfred V., Brian W. Kernighan, and Peter J. Weinberger. 1988. *The AWK Programming Language*. Addison-Wesley.

Atkins, Sue, Jeremy Clear, and Nicholas Ostler. 1992. Corpus design criteria. *Literary and Linguistic Computing* 7(1):1–16.

Boguraev, Branimir and James Pustejovsky, eds. 1995. *Corpus Processing for Lexical Acquisition*. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 1957. *Syntactic Structures*. MIT Press.

Church, Kenneth Ward. 1993. Char_align: A program for aligning parallel texts at the character level. In *ACL 31*, pages 1–8.

Greene, Barbara B. and Gerald M. Rubin. 1971. Automatic grammatical tagging of english. Department of Linguistics, Brown University.

Hearst, Marti A. 1999. Untangling text data mining. In *Proceedings of ACL'99*. University of Maryland.

Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing*. Upper Saddle River, NJ: Prentice Hall.

Karanikas, Haralampos, Christos Tjortjis, and Babis Theodoulidis. 2000. An approach to text mining using information extraction. In *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*. Lyon, France.

Kay, Martin and Martin Röscheisen. 1993. Test-translation alignment. *Computational Linguistics* 19:121–142.

Klavans, Judith L. and Philip Resnik, eds. 1996. *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. Cambridge, Mass.: MIT Press.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Mikheev, Andrei. 1998. Feature lattices for maximum entropy modelling. In *ACL*, vol. 36, pages 848–854.

Mitkov, Ruslan, Richard Evans, Constantin Orasan, Catalina Barbu, Lisa Jones, and Violeta Sotirova. 2000. Coreference and anaphora: developing annotating tools, annotated resources and annotation strategies. In *Proceedings of Discourse, Anaphora and Reference Resolution Conference (DAARRC) 2000*. Lancaster University.

Palmer, David D. and Marti A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics* 23:241–267.

Piatetsky-Shapiro, G. and W. J. Frawley, eds. 1991. *Knowledge Discovery in Databases*. AAAI Press / MIT.

Pustejovsky, James. 1995. *The Generative Lexicon*. Cambridge, MA: MIT Press.

Reynar, Jeffrey C. and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *ANLP*, vol. 5, pages 16–19.

Riley, Michael D. 1989. Some applications of tree-based modeling to speech and language indexing. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 339–352. Morgan Kaufman.

Simmons, Robert F. 1982. Themes from 1972. In *Proceedings of ACL'82*. 20th Annual Meeting of the Association for Computational Linguistics, University of Toronto, Canada.

Index