Introduction to natural language processing

Contents

Preface

About this half unit

This half unit course combines a critical introduction to key topics in theoretical and computational linguistics with hands-on practical experience of using existing software tools and developing applications to process texts and access linguistic resources. The aims of the course and learning outcomes are listed in Chapter 1. This course has no specific prerequisites. There will be some programming involved and you will need to acquire some familiarity with the Python language, but you will not be expected to develop substantial original code or to encode specialised algorithms. The course involves some statistical techniques, but the only mathematical knowledge assumed is an understanding of elementary probability and familiarity with the concept of logarithms.

Before the advent of the world wide web, most machine-readable information was stored in structured databases and accessed via specialised query languages such as Structured Query Language (SQL). Nowadays the situation is reversed: most information is found in unstructured or semi-structured natural language documents and there is increasing demand for techniques to 'unlock' this data. Computing graduates with knowledge of natural language processing techniques are finding employment in areas such as text analytics, sentiment analysis, topic detection and information extraction.

Assessment

The course is assessed via an unseen written examination. A sample examination paper is provided in the Appendix at the end of this subject guide, with some guidelines on how to answer the questions. You will be required to attempt three questions out of a choice of five. The questions will cover 'book knowledge', problem solving and short essays on more theoretical topics. The examination is not a memory test but will be designed to assess your understanding of the course content. There will also be coursework which will include a similar mix of questions, but with a stronger focus on practical problem-solving.

You will be expected to provide electronic copies of your coursework for plagiarism checking purposes. It is very important that any material that is not original to you should be properly attributed and placed in quotation marks, with a full list of references at the end of your submission. You should follow the style used in this subject guide for citing references, for example:

> *Segaran (2007, pp.117–118) discusses some problems with rule-based spam filters.*

Answers which consist entirely or mostly of quoted material are unlikely to get many marks even if properly attributed, as simply reproducing an answer in someone else's words does not demonstrate that you have fully understood the material.

In order to give you some practice in problem-solving and writing short essays, there are a number of Activities throughout this subject guide. The Appendix includes a section 'Answers to selected activities', although these will not always provide complete answers to the questions but are intended to indicate how particular types of questions should be approached. Sample examination questions are provided at the end of each chapter. Some, but not all, of these are included in the sample examination paper with suggested answers at the end of the guide.

The subject guide and other learning resources

This subject guide is not intended as a self-contained textbook but sets out specific topics for study in the CO3354 half unit. There is a recommended textbook and a number of other readings are listed at appropriate places. There are also links to websites providing useful resources such as software tools and access to online linguistic data. The learning outcomes listed in the next chapter assume that you are working through the recommended readings, activities and sample examination questions. It will not be possible to pass this half unit by reading only the subject guide. Please refer to the Computing VLE for other resources, which should be used as an aid to your learning.

Suggested study time

The Student Handbook states that 'To be able to gain the most benefit from the programme, it is likely that you will have to spend at least 300 hours studying for each full unit, though you are likely to benefit from spending up to twice this time'. Note that this subject is a half unit.

The course is designed to be delivered over a ten-week term as one of four concurrent modules, and this guide has six chapters. Chapter 1 goes into more detail about the structure of the guide and the course, while Chapters 2 to 6 are each dedicated to a particular topic. It is suggested that you spend about two weeks on Chapters 1 and 2 together and each of Chapters 3 to 6, including the associated reading and web-based material, and work through the activities and sample examination questions during this time.

Acknowledgement

This subject guide draws closely on:

Bird, S., E. Klein and E. Loper, *Natural Language Processing with Python*. (O'Reilly Media 2009) [ISBN 9780596516499; http://nltk.org/book].

You will be expected to draw on it in your studies and to use the accompanying software package, the Natural Language Toolkit, which requires the Python language. *Natural language processing with Python* has been made available under the terms of the Creative Commons Attribution Noncommercial No-Derivative-Works 3.0 US License: *http://creativecommons.org/licenses/by-nc-nd.3.0/us/legalcode* (last visited 13th April 2013).

Chapter 1

Introduction: how to use this subject guide

1.1 Introduction

The idea of computers being able to understand ordinary languages and hold conversations with human beings has been a staple of science fiction since the first half of the twentieth century and was envisaged in a classic paper by Alan Turing (1950) as a hallmark of computational intelligence. Since the start of the twenty-first century this vision has been starting to look more plausible: artificial intelligence techniques allied with the scientific study of language have emerged from universities and research laboratories to inform a variety of industrial and commercial applications. Many websites now offer automatic translation; mobile phones can appear to understand spoken questions and commands; search engines like Google use basic linguistic techniques for automatically completing or 'correcting' your queries and for finding relevant results that are closely matched to your search terms. We are still some way from full machine understanding of natural language, however. Automated translations still need to be reviewed and edited by skilled human translators while no computer system has yet come close to passing the 'Turing Test' of convincingly simulating human conversation. Indeed it has been argued that the Turing Test is a blind alley and that research should focus on producing effective applications for specific requirements without seeking to generate an illusion that users are interacting with a human rather than a machine (Hayes and Ford, 1995). Hopefully, by the time you finish this course you will have come to appreciate some of the challenges posed by full understanding of natural language as well as the very real achievements that have resulted from focusing on a range of specific, well-defined tasks.

1.2 Aims of the course

This course combines a critical introduction to key topics in theoretical linguistics with hands-on practical experience of developing applications to process texts and access linguistic resources. The main topics covered are:

- accessing text corpora and lexical resources
- processing raw text
- categorising and tagging
- extracting information from text
- analysing sentence structure.

1.3 Learning outcomes

On successful completion of this course, including recommended readings, exercises and activities, you should be able to:

1. utilise and explain the function of software tools such as corpus readers, stemmers, taggers and parsers
2. explain the difference between regular and context-free grammars and define formal grammars for fragments of a natural language
3. critically appraise existing Natural Language Processing (NLP) applications such as chatbots and translation systems
4. describe some applications of statistical techniques to natural language analysis, such as classification and probabilistic parsing.

Each main chapter contains a list of learning outcomes specific to that chapter at the beginning, as well as a summary at the end of the chapter.

1.4 Reading list and other learning resources

This is a list of textbooks and other resources which will be useful for all or most parts of the course. Additional readings will be given at the start of each chapter. See the bibliography for a full list of books and articles referred to, including all ISBNs. In some cases several different books will be listed: you are not expected to read all of them, rather the intention is to give you some alternatives in case particular texts are hard to obtain.

Essential reading

Bird, Klein, and Loper (2009): *Natural Language Processing with Python*. The full text including diagrams is freely available online at *http://nltk.org/book* (last visited 13th April 2013). The main textbook for this course, *Natural Language Processing with Python* is the outcome of a project extending over several years to develop the Natural Language Toolkit (NLTK), which is a set of tools and resources for teaching computational linguistics. The NLTK comprises a suite of software modules written in Python and a collection of corpora and other resources. See section 1.5 below for advice on installing the NLTK and other software packages.

In the course of working through this text you will gain some experience and familiarity with the Python language, though you will not be expected to produce substantial original code as part of the learning outcomes of the course.

Recommended reading

Pinker (2007). *The Language Instinct*. This book is aimed at non-specialists and deals with many psychological and cultural aspects of language. Chapter 4 is particularly relevant to this course as it provides a clear and accessible presentation of two standard techniques for modelling linguistic structure: finite-state machines and context-free grammars (though Pinker does not in fact use these terms, as we will see in Chapter 2 of the subject guide).

Jurafsky and Martin (2009): *Speech and Language Processing*, second edition. Currently the definitive introductory textbook in this field, covering the major topics in a way which combines theoretical issues with presentations of key technologies, formalisms and mathematical techniques. Much of this book goes beyond what you will need to pass this course, but it is always worth turning to if you're looking for a more in-depth discussion of any particular topics.

Perkins (2010): *Python Text Processing with NLTK 2.0 Cookbook*. This book will be suitable for students who want to get more practice in applying Python programming to natural language processing. Perkins explains several techniques and algorithms in more technical detail than Bird et al. (2009) and provides a variety of worked examples and code snippets.

Segaran (2007) *Programming Collective Intelligence*. This highly readable and informative text includes tutorial material on machine learning techniques using the Python language.

Additional reading

Russell and Norvig (2010) *Artificial Intelligence: a modern approach*, third edition. This book is currently regarded as the definitive textbook in Artificial Intelligence, and includes useful material on natural language processing as well as on machine learning, which has many applications in NLP.

Mitkov (2003) *The Oxford Handbook of Computational Linguistics*. Edited by Ruslan Mitkov. A collection of short articles on major topics in the field, contributed by acknowledged experts in their respective disciplines.

Partee et al. (1990) *Mathematical Methods in Linguistics*. A classic text, whose contents indicate how much the field has changed since its publication. A book with such a title nowadays would be expected to include substantial coverage of statistics, probability and information theory, but this text is devoted exclusively to discrete mathematics including set theory, formal logic, algebra and automata. These topics are particularly applicable to the content of Chapters 2 and 6.

Websites

Introductory/Reference *The Internet Grammar of English* is a clear and informative introductory guide to English grammar which also serves as a tutorial in grammatical terminology and concepts. The site is hosted by the Survey of English Usage at University College London (http://www.ucl.ac.uk/internet-grammar/home.htm, last visited 27th May 2013).

Hands-on corpus analysis

BNCWeb is a web-based interface to the British National Corpus hosted at Lancaster University which supports a variety of online queries for corpus analysis (http://bncweb.info/; last visited 27th May 2013).

The Bank of English forms part of the Collins Corpus, developed by Collins Dictionaries and the University of Birmingham. Used as a basis for Collins Advanced Learner's Dictionary, grammars and various tutorial materials for learners of English. Limited online access at http://www.collinslanguage.com/wordbanks; (last visited 27th May 2013).

Journals and conferences

Computational Linguistics is the leading journal in this field and is freely available at http://www.mitpressjournals.org/loi/coli (last visited 27th May 2013).

Conference Proceedings are often freely downloadable and many of these are hosted by the ACL Anthology at http://aclweb.org/anthology-new/ (last visited 27th May 2013).

1.5 Software requirements

This course assumes you have access to the Natural Language Toolkit (NLTK) either on your own computer or at your institution. The NLTK can be freely downloaded and it is strongly recommended that you install it on your own machine: Windows, Mac OSX and Linux distributions are available from `http://nltk.org` (last visited April 10th 2013) and some distributions of Linux have it in their package/software managers. Full instructions are available at the cited website along with details of associated packages which should also be installed, including Python itself which is also freely available. Once you have installed the software you should also download the required datasets as explained in the textbook (Bird et al., 2009, p. 3).

You should check the NLTK website to determine what versions of Python are supported. Current stable releases of NLTK are compatible with Python 2.6 and 2.7. A version supporting Python 3 is under development and may be available for testing by the time you read this guide (as of April 2013).

1.6 How to use the guide/structure of the course

This section gives a brief summary of each chapter. These learning outcomes are listed at the beginning of each main chapter and assume that you have worked through the recommended readings and activities for that chapter.

1.6.1 Chapter 2: Introducing NLP: patterns and structures in language

This chapter looks at different approaches to analysing texts, ranging from 'shallow' techniques that focus on individual words and phrases to 'deeper' methods that produce a full representation of the grammatical structure of a sentence as a hierarchical tree diagram. The chapter introduces two important formalisms: **regular expressions**, which will play an important part throughout the course, and **context-free grammars** which we return to in Chapter 6 of the subject guide.

1.6.2 Chapter 3: Getting to grips with natural language data

This chapter looks at the different kinds of data resources that can be used for developing tools to harvest information that has been published as machine-readable documents. In particular, we introduce the notion of a 'corpus' (plural *corpora*) – for the purposes of this course, a computer-readable collection of text or speech. The NLTK includes a selection of excerpts from several well-known corpora and we provide brief descriptions of the most important of these and of the different formats in which corpora are stored.

1.6.3 Chapter 4: Computational tools for text analysis

The previous chapter introduced some relatively superficial techniques for language analysis such as concordancing and collocations. This chapter covers some fundamental operations in text analysis:

- tokenisation: breaking up a character string into words, punctuation marks and other meaningful expressions;
- stemming: removing affixes from words, e.g. *mean+ing*, *distribut+ion*;
- tagging: associating each word in a text with a grammatical category or part of speech.

1.6.4 Chapter 5: Statistically-based techniques for text analysis

Statistical and probabilistic methods are pervasive in modern computational linguistics. These methods generally do not aim at complete understanding or analysis of a text, but at producing reliable answers to well-defined problems such as sentiment analysis, topic detection or recognising named entities and relations between them in a text.

1.6.5 Chapter 6: Analysing sentences: syntax and parsing

This chapter resumes the discussion of natural language syntax that was introduced in Chapter 2, concentrating on context-free grammar formalisms and various ways they need to be modified and extended beyond the model that was presented in that chapter. Formal grammars do not encode any kind of processing strategy but simply provide a declarative specification of the well-formed sentences in a language. Parsers are computer programs that use grammar rules to analyse sentences, and this chapter introduces some fundamental approaches to syntactic parsing.

1.6.6 Appendices

The Appendices include:

A. A bibliography listing all works referenced in the subject guide, including publication details and ISBNs.
B. A glossary of technical terms used in this subject guide.
C. Answers to selected activities.
D. A trace of a recursive descent parse as described in Chapter 6 of the subject guide.
E. A sample examination paper with guidelines on how to answer questions.

1.7 What the course does not cover

The field of natural language processing or computational linguistics is a large and diverse one, and includes many topics we will not be able to address in this course. Some of these are listed below:

- speech recognition and synthesis
- dialogue and question answering
- machine translation
- semantic analysis, including word meanings and logical structure
- generating text or speech from non-linguistic inputs.

However, the course should provide you with a basis for investigating some of these areas for your final year project. Some of these topics are dealt with in the later chapters of Bird et al. (2009) and most of them are touched on by Jurafsky and Martin (2009).

Chapter 2

Introducing NLP: patterns and structure in language

Essential reading

Steven Pinker (2007), *The Language Instinct*, Chapter 4.

Recommended reading

Jurafsky and Martin (2009), *Speech and Language Processing* second edition, Chapters/Sections 2 'Regular Expressions and Automata', 5.1 '(Mostly) English Word Classes', 12.1 'Constituency', 12.2 'Context-Free Grammars', 12.3 'Some Grammar Rules for English'.

Additional reading

- The Internet Grammar of English; *http://www.ucl.ac.uk/internet-grammar/home.htm* especially sections 'Word Classes' and 'Introducing Phrases'.
- Partee, ter Meulen and Wall, (1990), *Mathematical Methods in Linguistics*, Chapters/Sections 16.1–4, 17.1–3 (omitting 17.1.2–5, 17.2.1), 18.2, 18.6.

2.1 Learning outcomes

By the end of this chapter, and having completed the Essential reading and activities, you should be able to:

- explain the concept of **finite state machines** (FSMs) and their connections with regular expressions; work through simple FSMs
- write regular expressions for well-defined patterns of symbols
- analyse sentences in terms of parts of speech (POS) and constituent structure, including the use of tree diagrams
- write regular and context-free grammars for small fragments of natural language
- explain the concept of **stemming** and specify word-formation rules for given examples.

2.2 Introduction

People communicate in many different ways: through speaking and listening, making gestures, using specialised hand signals (such as when driving or directing traffic), using sign languages for the deaf, or through various forms of **text**.

By text we mean words that are written or printed on a flat surface (paper, card, street signs and so on) or displayed on a screen or electronic device in order to be read by their intended recipient (or by whoever happens to be passing by).

This course will focus only on the last of these: we will be concerned with various ways in which computer systems can analyse and interpret texts, and we will assume for convenience that these texts are presented in an electronic format. This is of course quite a reasonable assumption, given the huge amount of text we can access via the World Wide Web and the increasing availability of electronic versions of newspapers, novels, textbooks and indeed subject guides. This chapter introduces some essential concepts, techniques and terminology that will be applied in the rest of the course. Some material in this chapter is a little technical but no programming is involved at this stage.

We will begin in section 2.3 by considering texts as strings of characters which can be broken up into sub-strings, and introduce some techniques for informally describing patterns of various kinds that occur in texts. Subsequently in section 2.4 we will begin to motivate the analysis of texts in terms of hierarchical structures in which elements of various kinds can be embedded within each other, in a comparable way to the elements that make up an HTML web document. This section introduces some technical machinery such as: finite-state machines (FSMs), regular expressions, regular grammars and context-free grammars.

2.3 Basic concepts

2.3.1 Tokenised text and pattern matching

One of the more basic operations that can be applied to a text is **tokenising**: breaking up a stream of characters into words, punctuation marks, numbers and other discrete items. So for example the character string

> *"Dr. Watson, Mr. Sherlock Holmes", said Stamford, introducing us.*

can be tokenised as in the following example, where each token is enclosed in single quotation marks:

```
'"' 'Dr.' 'Watson' ',' 'Mr.' 'Sherlock' 'Holmes' '"' ','
'said' 'Stamford' ',' 'introducing' 'us' '.'
```

At this level, words have not been classified into grammatical categories and we have very little indication of syntactic structure. Still, a fair amount of information may be obtained from relatively shallow analysis of tokenised text. For example, suppose we want to develop a procedure for finding all personal names in a given text. We know that personal names always start with capital letters, but that is not enough to distinguish them from names of countries, cities, companies, racehorses and so on, or from capitalisation at the start of a sentence. Some additional ways to identify personal names include:

- Use of a title *Dr., Mr., Mrs., Miss, Professor* and so on.
- A capitalised word or words followed by a comma and a number, usually below 100: this is a common way of referring to people in news reports, where the number stands for their age – for example *Pierre Vinken, 61, . . .*
- A capitalised word followed by a verb that usually applies to humans: *said, reported, claimed, thought, argued . . .* This can over-generate in the case of country or organisation names as in *the Crown argues* or *Britain claimed*.

We can express these more concisely as follows, where | is the disjunction symbol, *Word* stands for a capitalised word and *Int* is an integer:

- (Dr. | Professor | Mr. | Mrs. | Miss | Ms) *Word*
- *Word Word, Int*
- *Word* (said | thought | believed | claimed | argued | ...)

Learning activity

1. Write down your own examples of names that match each of the above patterns.
2. Pick a newspaper article or webpage that provides a variety of examples of people's names. Do they match the patterns we have encoded above? If not, see if you can devise additional rules for recognising names and write them out in a similar format.

2.3.2 Parts of speech

A further stage in analysing text is to associate every token with a grammatical category or **part of speech** (POS). A number of different POS classifications have been developed within computational linguistics and we will see some examples in subsequent chapters. The following is a list of categories that are often encountered in general linguistics: you will be familiar with many of them already from learning the grammar of English or other languages, though some terms such as *Determiner* or *Conjunction* may be new to you.

Noun fish, book, house, pen, procrastination, language

Proper noun John, France, Barack, Goldsmiths, Python

Verb loves, hates, studies, sleeps, thinks, is, has

Adjective grumpy, sleepy, happy, bashful

Adverb slowly, quickly, now, here, there

Pronoun I, you, he, she, we, us, it, they

Preposition in, on, at, by, around, with, without

Conjunction and, but, or, unless

Determiner the, a, an, some, many, few, 100

Bird et al. (2009, pp. 184–5) make the standard distinction that nouns 'generally refer to people, places, things or concepts' while verbs 'describe events or actions'. This may be helpful when one is starting to learn grammatical terminology but is something of an over-simplification. One can easily find or construct examples where the same concept can be expressed by a noun or a verb, or by an adjective or an adverb. And on the other hand, there are many words that can take different parts of speech depending on what they do in a sentence:

1. Rome *fell swiftly*.
2. The *fall* of Rome was *swift*.
3. The enemy *completely destroyed* the city.
4. The enemy's *destruction* of the city was *complete*.
5. John likes to *fish* on the river bank.
6. John caught a *fish*.

Additionally, some types of verbs do not correspond to any particular action but serve a purely grammatical function: these include the auxiliary verbs such as *did*, *shall* and so on. So in summary, we can often only assign a part of speech to a word depending on its function in context rather than how it relates to real things or events in the world.

Learning activity

Identify parts of speech in these examples:

1. The cat sat on the mat.
2. John sat on the chair.
3. The dog saw the rabbit.
4. Jack and Jill went up the hill.
5. The owl and the pussycat went to sea.
6. The train travelled slowly.

2.3.3 Constituent structure

You will have noticed several recurring patterns in the above examples: *Det Noun*, *Prep Det Noun* and so on. You may also have noticed that some types of phrase can occur in similar contexts: *(John | the cat) sat*, a *Proper Noun* or a sequence *Det Noun* can come before a *Verb*. Some of these possibilities can be captured using the pattern-matching notation introduced above, for example:

(((the | a)(cat | dog))(John | Jack | Susan))(barked | slept)

This will match any sequence which ends in a verb *barked* or *slept* preceded by **either** a Determiner *a* or *the* followed by a Noun *cat* or *dog* **or** a proper name *John, Jack* or *Susan*.

Patterns that have similar distributions (meaning that they can occur in similar contexts) are standardly identified by phrasal categories such as *Noun Phrase* or *Verb*

Phrase. A common way to represent information about constituent structure is by means of production rules of the form $X \rightarrow A, B, C \ldots$. Using rules of this form, grammatical sentences can be broken down into constituent phrases consisting of various combinations of POS:

Sentence $\rightarrow$ Noun Phrase, Verb Phrase

Noun Phrase $\rightarrow$ Determiner, Noun (**Example:** the, dog)

Noun Phrase $\rightarrow$ Proper Noun (**Example:** Jack)

Noun Phrase $\rightarrow$ Noun Phrase, Conj, Noun Phrase (**Examples:** Jack and Jill, the owl and the pussycat)

Verb Phrase $\rightarrow$ Verb, Noun Phrase (**Example:** saw the rabbit)

Verb Phrase $\rightarrow$ Verb, Preposition, Noun Phrase (**Examples:** went up the hill, sat on the mat)

Learning activity

Read through the recommended sections of the UCL 'Internet Grammar of English'. Write production rules that cover some of the examples in these sections.

2.4 A closer look at syntax

This section aims to motivate the idea that texts can be analysed as hierarchical structures rather than 'flat' sequences whose elements are organised in various patterns. The Essential reading for this chapter by Steven Pinker gives a concise and accessible introduction to some fundamental distinctions we will make in this section, from the point of view of Chomskyan linguistics (compare Chomsky, 1957/2002). Chomsky and his followers argue that some components of our knowledge of language are innate, and Pinker (2007, chapter 4) sketches some arguments in support of this claim. This position is considered to be contentious by many linguists and we will not address it in this course. However, Pinker's chapter provides a useful introduction to syntactic analysis and clearly distinguishes between two formal techniques for modelling grammatical knowledge, which underlie **regular** and **context-free** grammars respectively (these terms will be explained as we go along).

Learning activity

If you have access to it, read through the recommended chapter by Pinker and make notes, and have it to hand while working through the remainder of this section.

Pinker notes that language makes 'infinite use of finite means', in Humboldt's phrase[1]. That is, there is no principled upper limit to the length of a grammatical sentence: we can always add another phrase, even if it's a banal one like 'one could say that', 'and that's a fact' or 'and you can tell that to the Marines'. A large

[1]'Sie [die Sprache] muss daher von endlichen Mitteln einen unendlichen Gebrauch machen' (von Humboldt, 1836, p. 122).

proportion, perhaps most of the sentences we read, hear or speak every day may be entirely novel, at least to us. Consequently, knowledge of a language seems to consist in knowing rules that specify what sentences belong to the language, rather than memorising long lists of sentences to be produced on appropriate occasions.

Pinker considers two different formal systems for generating or recognising sentences in English:

- 'wordchain' devices, equivalent to finite state machines. These devices incorporate three distinct operations: **sequence**, **selection** and **iteration**.
- Phrase structure grammars, which include the additional operation of **recursion**.

Note that Pinker deliberately uses the more descriptive expression 'wordchain' as he is concerned to avoid the use of forbidding technical terminology. In what follows we will stick to the standard term *finite-state machine* which you are more likely to find in textbooks. You may also encounter the terms *finite-state automaton* or just *finite automaton*.

2.4.1 Operation of a finite-state machine

A wordchain device or finite-state machine (FSM) can be seen as a set of lists of symbols (such as words or fixed phrases) and rules for going from list to list. A simple example:

Word lists

1. The, a, one
2. Cat, dog, fish
3. Barked, slept, swam

Rules

It is important to keep in mind that FSMs are neutral between accepting and generating strings. That is to say, one way to operate a FSM is to read a string, one symbol at a time, and determine whether the symbol is found in the list at the current state of the machine. If it is, we advance to the next state and read the next symbol. Alternatively, this FSM could be used to generate strings by picking one word from each list in sequence. Some possible matching strings are:

- The dog swam
- A cat barked
- A fish slept
- . . .

A more complex example:

1. *John/Mary/Fred* **OR**
   1a. *the/a/one*
   1b. *cat/dog/fish*

2. (optional): *and/or* **GO TO 1**
3. *slept/barked/swam* **OR**
   3a. *sat/walked*
   3b. *on*
   3c. *a/the*
   3d. *mat/hill*
4. (optional) *and/or* **GO TO 3**
5. (optional) *and/or/but* **GO TO 1**.

This formulation involves the basic operations of sequence, selection and iteration as follows:

SEQUENCE

Moving from list to list in numerical order: 1, 2, 3 . . .

SELECTION

Choosing an item from a list, for example *cat*, *dog* or *fish*; choosing between lists.

ITERATION

Repeating particular sequences, for example:

- John and Mary or a fish *(repeats step 1.)*
- The cat barked but Fred walked on the hill. *(Repeats steps 1–5, omitting step 4.)*

Learning activity

1. Find the shortest sentence generated or accepted by the above FSM.
2. Write out four sentences between six and 20 words long which are accepted by the FSM.

2.4.2 Representing finite-state machines

There are various conventional ways of representing a non-deterministic FSM in terms of a number of states and the permissible transitions between states. In our informal exposition above, the numbered steps represent states and each symbol or word in a list counts as a possible transition to the next state. Pinker adopts a graphical convention where states are depicted as nodes in a graph and transitions are directed, labelled arcs between the nodes; see also Partee et al. (1990, p. 457 and following). Alternatively, the states and transitions can be shown in tabular form as in Table 2.1 where $q1$ is the initial state and $q4$ the final state:

q1	john	q2
q1	mary	q2
q1	the	q1a
q1	a	q1a
q1a	cat	q2
q1a	dog	q2
q2	slept	q3
q2	barked	q3
q2	swam	q3
q3	and	q1
q3	or	q1
q3	.	q4

Table 2.1: A finite-state machine represented as a state-transition table.

2.4.3 Declarative alternatives to finite-state machines

The FSMs shown above combine a formal specification of a language with a processing strategy. It is often convenient to separate the two and define the language using expressions from a declarative formalism which can be manipulated using various different algorithms. This section considers two such formalisms: *regular expressions* and *regular grammars*.

Regular expressions (REs) provide a simple but powerful means of identifying patterns in text and are widely used in various applications of computer science. REs are based on three fundamental concepts which as we have seen are characteristic of finite-state machines:

sequence – to do with the order in which items occur: may include a wildcard character which is written as the period or full stop '.' and may be replaced by any character.

selection – specifying a choice between alternative items or sequences, indicated by the '|' operator

iteration – repetition of items or sequences, indicated by the '*' operator, meaning zero or more occurrences of whatever precedes the star.

Some simple examples:

a* matches sequences of zero or more **a**'s: **a**, **aaaa**, **aaaaaaaaaaa** and so on. A sequence of zero elements is known as the 'empty string' and conventionally denoted by the Greek letter *epsilon* or ϵ.

aa* sequences of one or more **a**'s

ab* sequences of one **a** followed by zero or more **b**'s: **a**, **ab**, **abbbb**, ...

(ab)* sequences of zero or more pairs **ab**: ϵ, **ab**, **abab**, **ababab** ...

(ab)|(ba) **ab** or **ba**

((ab)|(ba))* possibly empty sequences of **ab** and **ba** pairs: ϵ, **ab**, **abab**, **baab**, **bababa**, **abba** ... Note that parentheses operate in the usual manner as in mathematical or logical expressions, to denote the **scope** of operators.

b.*a all strings that start with **b** and end with **a**: **ba**, **bbbaaaa**, **bcccccccca** ...

Programming languages such as Java, Perl and Python implement extensions of REs with operators which are mostly redundant in that they can be reduced to combinations of the above operations, but can make programs much more compact and readable, including:

+ – one or more of the previous item

? – the previous item is optional

[A-Z], [0-9] – this expression matches one of a range of characters

ˆabc – matches pattern *abc* at the start of a string

abc$ – matches pattern *abc* at the end of a string.

See also Bird et al. (2009, Table 3.3) and the other recommended readings on this topic.

Here are some examples of our suggested ways of recognising personal names coded as regular expressions. These are intended to be applied to tokenised text and every sequence enclosed by angled brackets < . . . > stands for an individual token. In Examples 1, 3 and 4 below, the material within parentheses represents the target string and sequences outside parentheses provide the context.

1. <Mrs?>(<.+>) matches 'Mr' or 'Mrs' followed by any string. The first token consists of the sequence *Mr* followed optionally by the character *s*. The second consists of a sequence of one or more characters: any character may occur in place of the wildcard '.'.
2. <[A-Z][a-z]+>+ matches any sequence of one or more capitalised words.
3. (<[A-Z][a-z]+>+)<,><[0-9]+> matches capitalised word(s) followed by a comma and a number (age).
4. (<[A-Z][a-z]+>+)<said|reported|claimed>.

Learning activity

1. Write a regular expression for all strings consisting of an odd number of **a**'s followed by an even number of **b**'s.
2. Write a regular expression for all sequences of **a**'s and **b**'s of length 3.
3. Write a regular expression for all strings that contain **abba** somewhere within them.

As you have probably observed, the pattern-matching notation we used in section 2.3 employed a subset of the RE syntax, and we could in principle use regular expressions to encode simple grammars as presented in that section. For example:

((John|Mary|Fred) | ((the|a)(cat|dog|fish))

(barked | slept | swam)

*((and | or) (barked | slept | swam))**

matches sentences like:

1. John slept
2. The cat barked or swam
3. Mary swam and barked or slept

4. ...

It can be seen that even conceptually simple REs can rapidly become almost unreadable. A more manageable formalism is a **regular grammar**, made up of production rules or rewrite rules of the kind you have seen in the previous section:

S $\rightarrow$ John | Mary | Fred VP

S $\rightarrow$ the | a S1

S1 $\rightarrow$ cat | dog | fish VP

VP $\rightarrow$ barked | slept | swam VP1

VP1 $\rightarrow$ and | or VP

VP1 $\rightarrow$ ϵ

A sequence of words forms a grammatical sentence according to a grammar of this type if one can draw a tree diagram like Figure 2.1 such that:

1. The root node is *S* or *Sentence*.
2. For every node that matches the left hand side (LHS) of a grammar rule, one can draw a subtree with the items on the right hand side (RHS) as daughter nodes.
3. When no more grammar rules can be applied, every leaf node of the tree matches a word in the language or the empty string ϵ.

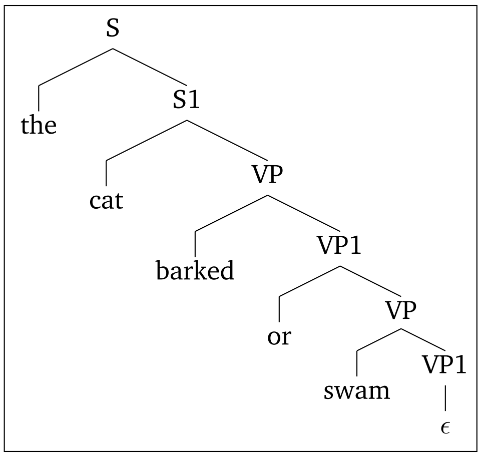

Figure 2.1: A right-branching tree.

Symbols which only occur on the right-hand side of rules, and so can only appear as leaf-nodes in a tree, are known as **terminal symbols**. Regular grammars have the restriction that when non-terminal symbols appear on the RHS they must either always be the rightmost symbol, or always the leftmost. These classes of grammars are known as **right-linear** and **left-linear** respectively. A right-linear grammar will always result in a right-branching tree as in the above example.

Learning activity

Draw tree diagrams according to the above grammar for the sentences:

1. The dog slept.
2. Mary swam and barked or slept.

2.4.4 Limitations of finite-state methods – introducing context-free grammars

Pinker (2007, p.86) gives an example of a 'wordchain device' or FSM which is intended to show the limitations of finite-state methods for handling natural language. The procedure is apparently designed to deal with complex sentences including constructions like *If. . . then. . .* and *Either. . . or* . . . . If we look at a few possible matching strings, we see clearly that some are grammatical sentences but others are nonsensical. (Following a standard convention in linguistics, the unacceptable cases are marked with an asterisk '*'.)

- Either a happy girl eats ice cream or the boy eats hot dogs.
- *Either a happy girl eats ice cream then one dog eats candy.
- If a girl eats ice cream then the boy eats candy.
- *If a girl eats ice cream or the boy eats candy.

Learning activity

1. Write a regular grammar that is equivalent to the FSM in Pinker (2007, p. 86).
2. Convince yourself that it allows you to draw well-formed trees for the ungrammatical examples above.
3. What characteristic of the grammar prevents it from ruling out the ill-formed examples?

See 'Answers to Activities' in Appendix C (p. 98) for further discussion.

In order to handle these kinds of sentences correctly we need to add new kinds of rewrite rules, going beyond the class of right- or left-linear grammars:

1. To match pairs of words like *if . . . then*, *either . . . or*, we need rules where a non-terminal symbol on the RHS can have additional material on both sides as in the first two rules below.
2. In order to allow for indefinite nesting – *if either John will come if Mary does, or* . . . we need rules where the same symbol can occur on both sides of the arrow. This is known as **self-embedding** or **centre-embedding**.

Note that centre-embedding is an instance of **recursion** in grammar; right-linear grammars may also include recursive rules but they can always be processed iteratively rather than recursively (Jurafsky and Martin, 2009, p. 447).

S →Either S or S

S →If S then S

S → NP VP

NP → Det N

Det → a | the | ϵ

N → girl | boy | dog | cat | burgers | candy | cream | cake

VP → V NP

VP → V PP

PP → P NP

V → eats | likes | sat

P → on

The above grammar handles these cases correctly as well as simple sentences like *The cat sat on the mat*:

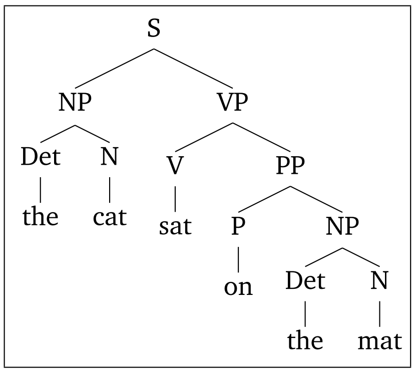

Figure 2.2: Tree diagram for *The cat sat on the mat.*

It is also acceptable to represent trees using *labelled bracketed strings* as in the example below:

```
(S
(NP (Det the ) (N cat ))
(VP (V sat )
 (PP
  (P on )
  (NP
  (Det the ) (N mat ) )
  )
   )
   )
```

Figure 2.3 is an example of self-embedding.

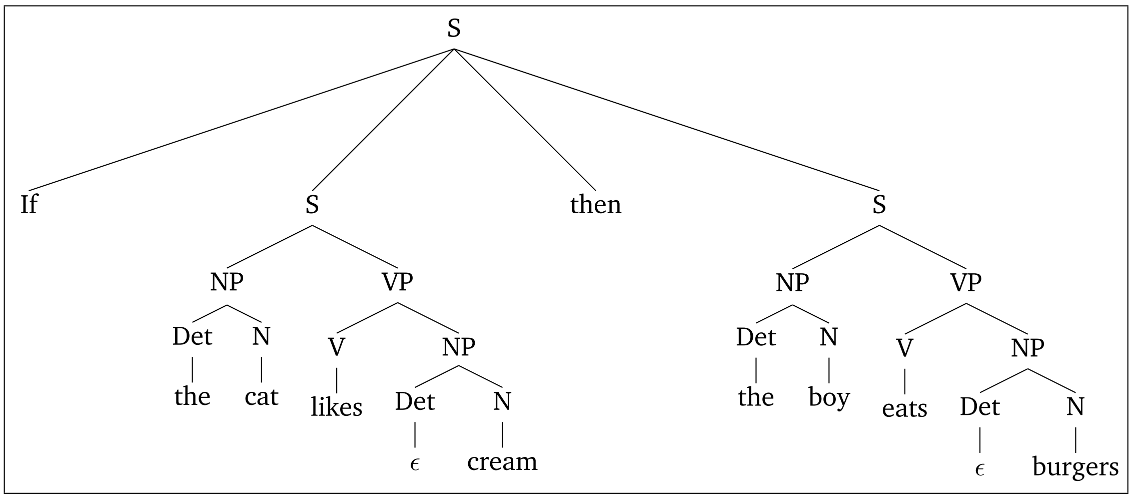

Figure 2.3: Tree diagram with self-embedding.

Learning activity

1. Trace through the grammar rules and satisfy yourself that Figure 2.3 represents the structure of the sentence *If the cat likes cream then the boy eats burgers* according to the grammar.
2. What is the shortest sentence generated by the above grammar?
3. Using the above grammar, draw complete tree diagrams for:
   (a) If the girl likes cake then either the boy eats burgers or the boy eats candy.
   (b) If either the boy likes cake or the girl likes burgers then the dog eats candy.
4. Think of ways to modify the grammar to generate more natural-sounding sentences.

2.4.5 Looking ahead: some further uses of regular expressions

In this chapter we have so far looked at finite-state formalisms as techniques for generating or recognising short phrases as well as whole sentences, and found them to be wanting. Many current applications in language technology do not, in fact, require complete analysis of sentences but proceed by looking for patterns of interest within a text and discarding what does not match these patterns. Finite-state methods are often quite adequate for these applications and you will see many uses for regular expressions in later chapters of this guide. Some examples we will look at in more detail in later chapters are:

- stemming: extracting the 'base form' of a word as informally presented in section 2.5 of this chapter
- tagging: automatically assigning POS or other forms of mark-up to elements in a text
- chunking: grouping together a sequence of words as a phrase
- information extraction: identifying chunks that denote meaningful entities, events or other items of interest.

2.4.6 Looking ahead: grammars and parsing

The pseudocode and graphical representations of wordchains (FSMs) combine a specification of the well-formed sentences in a language fragment with a processing strategy. It is important to keep in mind that formal grammars made up of a series of production rules do **not** encode a processing strategy. As stated above, a grammar is a declarative specification of the strings that make up a language while parsers use a variety of algorithms to apply the grammar rules. We will look at some of these parsing strategies in Chapter 6 of this subject guide.

2.5 Word structure

Words combine in different orders to form sentences and phrases; they also have internal structure. Nouns in English may have different endings according to whether they are singular (*a box*) or plural (*some boxes*) while in some languages this information may be expressed at the start of the word, for example Swahili *ziwa* ('lake') vs *maziwa* ('lakes'). In English, endings of verbs can indicate person, number, tense and mood[2], while other languages may make different dictinctions. Nouns and verbs are sometimes classified as regular or irregular according to whether their inflected forms can be derived by following simple rules. Table 2.2 shows examples of some common past tense forms in English.

Present	**Past**
become	became
come	came
mistake	mistook
misunderstand	misunderstood
ring	rang
sell	sold
shake	shook
sing	sang
sink	sank
stand	stood
take	took
tell	told
travel	travelled
understand	understood
withstand	withstood

Table 2.2: Past tense forms (1).

The subfield of linguistics known as **morphology** is concerned with the structure of words and is concerned, among other things, with formulating rules for deriving different forms of a word according to its grammatical role. Here are some rules which appear to cover the examples in the table:

[2] See the Internet Grammar of English at http://www.ucl.ac.uk/internet-grammar/verbs/verbs.htm (last visited 27th May 2013) for explanations of these terms.

Some rules for past-tense formation

-come → -came
-take → -took
-ing → -ang
-ink → -ank
-ell → -old
-and → -ood
-el → -elled

Some of these rules could be made more general: we could combine the *-ing* and *-ink* rules to a single rule, -in_ → -an_. On the other hand, some rules which work for these particular examples would fail if applied to a wider range of data: we have *come* → *came*, *become* → *became* but not *welcome* → **welcame*. This is an example of rules **overfitting** the data.

Learning activity

Modify the above rules so that they will account for the past tense forms in Table 2.3 as well as in Table 2.2.

Present	**Past**
bake	baked
command	commanded
bring	brought
sling	slung
smell	smelt
think	thought
wake	woke

Table 2.3: Past tense forms (2).

A natural language application such as a machine translation system will typically include a database of words or *lexicon* along with rules for deriving word endings: for example, a translation from English into Dutch might handle the word *brought* as follows:

1. Find the stem of *brought* and interpret the inflection: *bring+past*
2. Find the Dutch equivalent of *bring*: *brengen*
3. Find the past tense of *brengen*: *bracht*

The process of removing affixes from words to derive the basic form is called *stemming*. We will look at some tools for doing this in Chapter 4, and you will also have the opportunity to encode your own rules as regular expressions.

2.6 A brief history of natural language processing

The field of natural language processing or computational linguistics builds on techniques and insights from a number of different disciplines, principally theoretical linguistics and computer science but with some input from mathematical logic and psychology.

The notions of a finite-state machine and context-free grammar (CFG) were first introduced to linguistics by Chomsky (1957; see Pullum (2011) for a somewhat critical reappraisal). Chomsky argued that both formalisms were inadequate for modelling natural language and proposed an additional operation of transformations, which could essentially permute the output string of a CFG in various ways. Chomsky's work introduced a methodology which was to dominate theoretical linguistics for the next couple of decades: linguists concentrated on postulating formal rules of grammar which were tested against their own intuitions or those of native speakers of other languages, rather than seeking to induce rules from large collections of data. Part of the rationale for this was that native speakers of a language are able to recognise whether a sequence of words makes up an acceptable sentence in their language, even if they have never encountered those words in that particular order before. Prior to what was to become known as the Chomskyan revolution, corpus-based approaches had been the norm in general linguistics. This tradition was overshadowed for a time by so-called 'generative' linguistics, but corpus-based research continued in some quarters until its resurgence in the 1980s, including the development of the first machine-readable corpus by the Jesuit priest Fr Robert Busa. Busa developed a 10 million-word corpus of medieval philosophy on punch-cards, with the support of Thomas Watson of IBM (McEnery and Wilson, 2001, pp. 20–21).

Work in formal grammar tended to assume a 'backbone' of context-free rules, augmented with various mechanisms to handle data that appeared to go beyond the context-free model; some important developments were Generalised Phrase Structure Grammar (Gazdar et al., 1985) and Head-driven Phrase Structure Grammar (Pollard and Sag, 1994). We will see examples of these extra mechanisms in Chapter 6.

Early work on automated language processing was essentially procedural in its methodology, working with a type of finite-state machine called **transition networks** which were extended as augmented transition networks to cope with various linguistic constructions (Woods, 1970). Later work based on declarative grammar formalisms employed techniques including **deductive parsing** (Pereira and Warren, 1983) and **unification** (Kay, 1984). The former adopts techniques from the AI field of automated reasoning: the core idea is that parsing a sentence can be seen as constructing a logical proof that a particular sequence of words forms a proper sentence according to a given set of grammar rules. Unification grammars treat linguistic objects as sets of attributes or features with a finite range of values, and grammar rules specify that particular items in a sentence must have the same or compatible values for certain features. For example, the subject and main verb of a sentence should have the same value for the *number* feature. We will consider detailed examples in Chapter 6.

Meanwhile, substantial progress was made in lower-level tasks such as speech recognition and morphological analysis using probabilistic techniques and finite-state models. During the late 1990s these techniques were extended to cover tasks such as parsing, part-of-speech tagging and reference resolution (recognising whether or not different expressions in a document referred to the same person or entity). These developments were driven by a number of factors: the continuing increase in the processing speed and memory capacity of computers; the availability of massive amounts of spoken and written material, both in unstructured form on the world wide web and with various types of annotation in corpora such as the

Penn Treebank[3] or the British National Corpus[4], and events such as the Message Understanding Conferences[5] which were initially sponsored by the US Department of Defense to measure and foster progress in extracting information from unstructured text.

Much work since around the year 2000 has involved the use of machine learning techniques such as Bayesian models and maximum entropy (see Chapter 5). This has involved using annotated corpora to train systems to segment and annotate texts according to various morphological, syntactic or semantic criteria. These techniques have been systematically applied to particular tasks such as parsing, word sense disambiguation, question answering and summarisation.

2.7 Summary

1. This chapter has characterised the subject matter of the course as being concerned with various ways of using computer programs to analyse **text**, by which we mean words, numbers, punctuation and other meaningful symbols that are printed on paper or some other flat surface, or displayed on a screen.
2. Some fundamental operations in text analysis include **tokenisation**, which involves extracting these meaningful elements from a stream of electronic characters and discarding white space, line feed characters and other material which has no explicit meaning for a human reader, and using regular expressions to identify patterns in a text.
3. Regular expressions are composed of the three basic operations of sequence, selection and iteration, and have many applications in computational linguistics and computer science at large. A finite-state machine is a process whose operations can be specified by means of regular expressions. A regular grammar is a set of production rules or rewrite rules that defines the sentences that make up a language, and any language defined by a regular grammar can be processed by a finite state machine or described using a regular expression.
4. A complete syntactic analysis of natural language sentences is generally held to require the additional operation of centre-embedded recursion, which is beyond the power of finite-state methods. Recursion is formally encoded in context-free grammars.
5. Not only do words combine in various patterns and structures to form sentences; they also have internal structure which can be described to an extent using rules for regular and irregular forms.
6. The current state of NLP or computational linguistics builds on research results and concepts from many different fields, and we have sketched some of the highlights in a very short history of the discipline.

2.8 Sample examination questions

You can expect a list of RE operators to be included as an appendix in the examination paper.

[3]http://www.cis.upenn.edu/~treebank/; last visited 27th May 2013
[4]http://www.natcorp.ox.ac.uk/; last visited 27th May 2013
[5]http://www.itl.nist.gov/iaui/894.02/related_projects/muc/; last visited 27th May 2013

1. S → NP VP
   NP → Det N
   NP → PN
   VP → V
   VP → V NP
   VP → V NP PP
   VP → VP Adv
   PP → P NP
   Det → the | a
   N → waiter | chairs | tables | hotel | manager
   PN → Oscar | Paris
   V → died | put | saw | called
   Adv → suddenly | quickly | slowly
   P → in | on
   (a) Using the grammar rules above, draw syntax trees for:
      i. Oscar died suddenly.
      ii. The waiter put the chairs on the tables.
      iii. Oscar called the waiter.
   (b) Modify the grammar so that it generates the unstarred sentences below as well as (i–iii) above but not the starred ones. Explain the reasons for your modifications.
      i. Oscar died in Paris.
      ii. Oscar died in a hotel in Paris.
      iii. The waiter came to the table when Oscar called him.
      iv. When Oscar called him the waiter came to the table.
      v. * Oscar put
      vi. * The waiter saw on the tables
      vii. * The waiter put in the chairs
      viii. * The waiter put the chairs
      ix. * Oscar died the table
      x. * When Oscar called him when the waiter came to the table.
2. Write a regular expression that will identify male and female names in context, in an English-language text. Discuss ways in which this might over- or under-generate.
3. Explain the difference between regular and context-free grammars and discuss the claim that natural language grammars need at least context-free power.
4. (a) Write a regular grammar which generates the following sentences:
      i. This is the kid that my father bought.
      ii. This is the cat that killed the kid that my father bought.
      iii. This is the dog that worried the cat that killed the kid that my father bought.
      iv. This is the stick that beat the dog that worried the cat that killed the kid that my father bought.
   (*Brewer's Dictionary of Phrase and Fable*, 1896)
   (b) Write out three more sentences generated by your grammar.

Chapter 3

Getting to grips with natural language data

Essential reading

Bird et al. (2009) *Natural Language Processing with Python* Chapters 1 and 2 particularly: 1.1–1.3, 2.1–2.2, 2.5.

Recommended reading

McEnery (2003) 'Corpus Linguistics' in Mitkov (2003) is a succinct overview of the topic from one of the leading scholars in the field.

Additional reading

McEnery and Wilson (2001) *Corpus Linguistics* is an established undergraduate textbook; Chapters 2 and 3 are especially relevant for this topic.

McEnery and Hardie (2011) *Corpus Linguistics: Method, Theory and Practice*. Chapter 3 addresses the important topic of research ethics for corpus linguistics, which is often neglected in technical or academic presentations of the subject.

3.1 Learning outcomes

By the end of this chapter, and having completed the Essential reading and activities, you should be able to:

- explain what is meant by a corpus in the context of natural language processing, and describe some different types, structures and uses of corpora
- describe the characteristics of some well-known corpora and other language resources such as the Brown corpus, Penn treebank, Project Gutenberg and WordNet
- Use online interfaces and other software tools to do some basic corpus analysis, including concordancing and finding collocations
- locate and format raw text documents and analyse them using corpus tools.

3.2 Using the Natural Language Toolkit

As stated in Chapter 1, this subject guide is not intended as a stand-alone tutorial in using the NLTK or the Python language. You are advised to read through the recommended sections of Bird et al. (2009) and work through the exercises marked

Your turn. You may also find it useful to attempt some of the exercises provided at the end of each chapter.

From this chapter onwards you will be running Python sessions and using the NLTK. You should get into the habit of starting sessions with the following commands:

```
>>> from __future__ import division
>>> import nltk, re, pprint
```

One of the features that makes the Python language suitable for natural language applications is the very flexible treatment of data structures such as lists, strings and sequences. You should be familiar with these structures from previous programming courses, but should ensure you understand the way they are handled in Python. For this chapter, only lists are relevant and you should study Bird et al. (2009, section 1.2) before trying any of the learning activities in this chapter.

3.3 Corpora and other data resources

As explained in the previous chapter, much natural language processing relies on large collections of linguistic data known as *corpora* (plural of *corpus*). A corpus can be simply defined as no more than a collection of language data, composed of written texts, transcriptions of speech or a combination of recorded speech and transcriptions.

Corpora fall into three broad categories (McEnery, 2003, p.450):

- **Monolingual** corpora consist, as the name suggests, of data from a single language.
- **Comparable** corpora include a range of monolingual corpora in different languages, preferably with a similar level of balance and representativeness, and can be used for contrastive studies of those languages.
- **Parallel** corpora include original texts in one language with translations of those texts in one or more different languages. Parallel corpora can be used to train statistical translation systems.

A corpus is generally expected to have additional characteristics: corpora are usually constructed so as to be *balanced* and *representative* of a particular domain (McEnery and Wilson, 2001, pp. 29–30). (Sometimes the term is used more loosely to cover any large collection of language data which need not have been compiled systematically, as in the phrase ‘the web as corpus’.) **Sampling theory** is a branch of statistics that deals with questions such as: how many respondents are needed in an opinion poll for the results to be considered to represent public opinion at large? Similar considerations arise in corpus linguistics. This is particularly important if a corpus is to be used for quantitative analysis of the kind described in Chapter 5: if the corpus data is skewed or unrepresentative then results of the analysis may not be reliable. These considerations may be less important if the corpus is collected for the literary or historical interest of the documents that make it up, as is the case with Project Gutenberg for example.

For example, Bird et al. (2009, pp. 407–412) refer to the TIMIT corpus, an annotated speech corpus developed by Texas Instruments and MIT. To ensure representativeness, it was designed to include a wide coverage of dialect variations. Corpus builders need to exercise expert judgment in deciding on the *sampling frame*, or ‘the entire population of texts from which we will take our samples’ (McEnery and Wilson, 2001, p. 78) and calculating the size of the corpus that is necessary for it to be maximally representative. The sampling frame may, for example, be *bibliographic*, based on some comprehensive index or the holdings of a particular library, or *demographic*, selecting informants on the basis of various social categories as is often done in public opinion research.

Corpora have tended to be of a finite size and to remain fixed once they have been compiled. There are also what is known as *monitor corpora* which are continually updated with new material. This is particularly useful for compilers of dictionaries who need to be able to track new words entering the language and the changing or declining use of old ones. An example of a monitor corpus is the COBUILD Bank of English^{TM}, which had around 300 million words when it was referred to by McEnery (2003) and has since more than doubled in size to 650 million words.

A further distinction is between corpora consisting solely of the original or ‘raw’ text and those that have been marked up with various annotations. One common technique is *standoff annotation* where the mark-up is stored in a different file from the original text (McEnery and Wilson, 2001, p.38); (Bird et al., 2009, p.415).

Finally, corpora can be further classifed according to their structure:

Isolated – an unorganised collection of individual texts such as the Gutenberg online collection of literary works.

Categorised – texts are organised by categories such as genre; an example is the Brown corpus described below.

Overlapping – some categories overlap. A news corpus such as Reuters may contain stories which cover both politics and sport, for example.

Temporal – texts indicate language use over time. Examples are the Inaugural corpus of all inaugural speeches by US Presidents, and the Helsinki Diachronic corpus of about 1.6 million words of English dating from the early 9th century CE to 1710.

Some examples of corpora, which will be described in more detail later in the chapter, are:

Brown Developed at Brown University in the early 1960s.

BNC British National Corpus, created and managed by BNC consortium which includes Oxford and Lancaster universities, dictionary publishers OUP, Longmans and Chambers, and the British Library.

COBUILD (Bank of English) The Bank of English^{TM}forms part of the Collins Corpus, developed by Collins Dictionaries and the University of Birmingham, and contains 650 million words.

Gutenberg An archive of free electronic books in various formats hosted at `http://www.gutenberg.org/`

Penn Treebank A corpus of reports from the *Wall Street Journal* and other sources in various different formats.

3.4 Some uses of corpora

McEnery and Wilson (2001, Chapter 4) discuss a variety of uses for corpora, some of which are briefly discussed below.

3.4.1 Lexicography

Modern dictionaries such as Chambers, Collins and Longmans now rely heavily on corpus data in order to classify and inventorise the various ways words can be used in contemporary English as well as any ways these uses may have changed. For example, a lexicographer who wishes to determine whether the words *scapegoat*, *thermostat* or *leverage* can be used as verbs can enter the appropriate search string and be presented with examples such as the following (from the BNC):

- ***Scapegoating*** *an individual prevents the debate and delays community understanding.*
- *The measuring cell is immersed in a vat of liquid, usually benzene or xylene which can be* ***thermostatted*** *at temperatures between 273 and 400 K.*
- *These one-time costs once met could be* ***leveraged*** *over much more business activity around the globe than we then enjoyed.*

McEnery and Wilson (2001, p. 107) discuss a case where they claim that two well-known dictionaries had 'got it wrong' by listing *quake* as a solely intransitive verb, while examples in a transitive construction can in fact be found through a corpus search:

- These sudden movements quake the Earth. (BNC)

It is perhaps debatable whether the dictionaries in question were 'wrong' to disregard examples like this, or the compilers may have judged this to be an idiosyncratic usage which did not merit being included in a work of reference with the status of standard usage.

3.4.2 Grammar and syntax

Large-scale grammars for pedagogic and reference use such as the *Comprehensive Grammar of the English Language* (Quirk et al., 1985) or the *Cambridge Grammar of the English Language* (Huddleston and Pullum, 2002) use corpora among their sources of information along with results of linguistic research and the compilers' own subjective intuitions as competent speakers of the language, although this has tended to involve qualitative rather than quantitative analysis. Recent advances in computational power and developments in parsed corpora and tools to analyse them have made it possible for researchers to carry out quantitative studies of various kinds of grammatical frequency, such as the relative frequency of different clause types in English. Other studies use corpora to test predictions made by formal grammars that have been developed within the generative school of linguistics. The COBUILD project which provided the resources for Collins English dictionaries has also resulted in a series of small handbooks covering various kinds of grammatical construction, which are useful both for advanced language learners and for linguists in search of examples.

3.4.3 Stylistics: variation across authors, periods, genres and channels of communication

The notion of *style* in communication is that people generally have a choice between different ways of expressing themselves and not only do individuals tend to make similar choices each time they communicate, but their particular choices may be more characteristic of particular genres (romantic fiction, financial news, court reports and so on), time periods and channels of communication. By **channels** we mean distinctions such as written text compared with spoken discourse, both of which can be further subdivided: people will make different choices when composing emails, sending text messages or (rarely) writing a letter by hand. We probably also adopt different styles when talking face-to-face and on the telephone. Literary scholars as well as law enforcement and intelligence agencies may have an interest in identifying the author of a document from internal evidence. There have been various famous and less well-known instances of controversial attribution of authorship: for example, various figures have been put forward as the authors of Shakespeare's plays.

3.4.4 Training and evaluation

In addition to the applications listed above, corpora are routinely used in linguistic research for training and testing machine-learning systems for specific tasks in *text analytics* such as:

- detecting the topic of a document
- analysing the sentiments expressed for or against some product or policy
- identifying individuals described in a text, relations between them and events they participate in
- statistical parsing
- statistical machine translation.

For example, the Brown corpus and the WSJ corpus are typically used for evaluating text segmentation among other text processing tasks (Mikheev, 2003, p. 203).

These topics will be covered in more detail in Chapter 5 of this subject guide, where you will be introduced to various machine-learning techniques. These will all be types of **supervised learning**, where a system is trained on 'corpora containing the correct label for each input' (Bird et al., 2009, p. 222), as opposed to **unsupervised learning** where the system is designed to detect patterns in the input without any feedback. This normally means that the corpus has been marked up by human annotators. Standard practice is to divide a corpus into training and test sets; the test set is considered a **gold standard** for comparing the accuracy of trained learning systems with that of the annotators. Of course humans are fallible, and it is good practice to use multiple annotators for at least a sample of the corpus and report the level of **inter-annotator agreement** that was achieved. This will set an upper bound for the performance that can be expected from the system, as it seems meaningless to say that a computer program can achieve 100 per cent accuracy on tasks where human annotators disagree (see Jurafsky and Martin, 2009, p. 189).

3.5 Corpora

This section provides brief descriptions of various corpora, some of which are distributed in full or in part with the NLTK and others can be accessed online.

3.5.1 Brown corpus

This was one of the first 'large-scale' machine readable corpora, though it may seem rather small by today's standards, consisting of one million words. It was developed at Brown University from the early 1960s and took around two decades to complete. It was intended as a 'standard corpus of present-day edited American English' and is caterorised by genre under headings such as:

News *Chicago Tribune*: Society Reportage

Editorial *Christian Science Monitor*: Editorials

Reviews *Time Magazine*: Reviews

Government US Office of Civil Defense: *The Family Fallout Shelter*

Science Fiction Heinlein: *Stranger in a Strange Land*

Humour Thurber: *The future, if any, of comedy*.

The Brown corpus is provided with the NLTK in tagged and untagged versions and can be accessed using the various methods listed by Bird et al. (2009, Table 2.3, p. 50).

3.5.2 British National Corpus

The BNC is created and managed by the BNC consortium, which includes Oxford and Lancaster universities, dictionary publishers OUP, Longmans and Chambers, and the British Library. It was developed between 1991 and 1994 and consists of 100 million words: 90 per cent written and 10 per cent transcriptions of speech. This was one of the first corpora to include spontaneous spoken English. The corpus was marked up using an automated part-of-speech tagger which resulted in a significant saving of time and expense compared with manual annotation by competent speakers of the language, but means that there is inevitably a degree of error – as you may discover in the course of the exercise given later in this chapter.

You can access this corpus online and perform various kinds of analysis using the Simple Query language. Registration is required via the following link but there is currently no charge:

`http://bncweb.lancs.ac.uk/bncwebSignup/user/login.php` (last visited 27th May 2013)

3.5.3 COBUILD Bank of English

The COBUILD project involved Collins Dictionaries and the University of Birmingham. The Collins corpus is a 2.5-billion word analytical database of English. It contains written material from websites, newspapers, magazines and books published around the world, and spoken material from radio, TV and everyday conversations. The Bank of English^{TM}forms part of the Collins Corpus and contains 650 million words. It was used as a basis for the Collins Advanced Learner's Dictionary, grammars and various tutorial materials for learners of English. It is not included in the NLTK but there is limited online access via `http://www.collinslanguage.com/wordbanks`.

3.5.4 Penn Treebank

The Penn Treebank with its various offshoots is one of the widely used linguistic resources among empirical researchers.

It includes a collection of texts in four different formats:

- Raw text (original).
- Tagged with POS using a tagset which was developed as part of the project.
- ‘Parsed’; that is, marked up with constituent structure.
- Combined, including both POS tags and constituent structure.

The Penn Treebank project . . . has produced treebanks from the Brown, Switchboard, ATIS and *Wall Street Journal* corpora of English, as well as treebanks in Arabic and Chinese.

Jurafsky and Martin (2009, p. 438)

The project began at the University of Pennsylvania in the 1990s and the results have been used as a basis for further annotation efforts involving semantics and rhetorical structure. The NLTK includes a selection from the *Wall Street Journal (WSJ)* component of the Treebank, which can be accessed in each of the above formats and additionally with a simplified POS tagset (Bird et al., 2009, Table 5-1, p. 183). Here is an excerpt showing the four different formats:

Raw text

```
Pierre Vinken, 61 years old, will join the board as a
nonexecutive director Nov. 29.
```

Tagged

```
[ Pierre/NNP Vinken/NNP ]
,/,
[ 61/CD years/NNS ]
old/JJ ,/, will/MD join/VB
[ the/DT board/NN ]
as/IN
[ a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ]
./.
```

Parsed

```
( (S (NP-SBJ (NP Pierre Vinken)
             ,
             (ADJP (NP 61 years) old)
```

```
                ,)
        (VP will
            (VP join
                (NP the board)
                (PP-CLR as
        (NP a nonexecutive director))
        (NP-TMP Nov. 29)))
        .))
```

Combined

```
( (S
     (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

3.5.5 Gutenberg archive

The NLTK includes a small selection of out-of-copyright literary texts from Project Gutenberg, an archive of free electronic books hosted at *http://www.gutenberg.org/* Some of the texts included are:

Jane Austen: *Emma*, *Persuasion*

GK Chesterton: *Father Brown stories*, *The Man Who Was Thursday*

William Blake: *Poems*

Milton: *Paradise Lost*

Shakespeare: *Julius Caesar*, *Macbeth*, *Hamlet*

3.5.6 Other corpora

Some further corpora included with the NLTK are:

- The Reuters corpus distributed with the NLTK contains 10,788 news documents totalling 1.3m words, partitioned into 'training' and 'test' sets. This split is for training and testing machine learning algorithms: we return to this topic in Chapter 5 of this subject guide.

- US Presidents' inaugural and State of the Union addresses, organised as separate files.
- UN Declaration of Human Rights in 300+ languages. Here are a few excerpts:
  - All human beings are born free and equal in dignity and rights.
  - *Abantu bonke bazalwa bekhululekile njalo belingana kumalungelo abo.*
  - *Todos os seres humanos nascem livres e iguais em dignidade e em direitos.*

Other corpora with online query interfaces include:

1. The Corpus of Contemporary American English, hosted at Brigham Young University, is claimed to be 'the only large and balanced corpus of American English'. *http://corpus.byu.edu/coca/* (last visited 27th May 2013)
2. The Intellitext project at the University of Leeds 'aims to facilitate corpus use for academics working in various areas of the humanities' and currently provides access to monolingual and parallel corpora in several European and Asian languages. *http://corpus.leeds.ac.uk/it/* (last visited 27th May 2013)

Learning activity

1. Pick two or three of the corpora mentioned above and research them online, focusing on questions such as:
   - how large is the corpus?
   - what language(s) and genre(s) does it represent?
   - when was it constructed and what is its intended use?
   - what is the sampling frame?
   - what level of interannotator agreement was achieved, if reported?
2. Logon to the BNC Web (free registration needed) or another online corpus. Study the documentation provided and search for data to answer the following questions:
   - What syntactic categories can the following words have? *total*, *pancake*, *requisition*, *acquisition*.
   - The guide to Simple Query Syntax provided with the BNC warns that 'part-of-speech tags have been assigned by an automatic software tool and are not always correct'. Have your answers to the previous question shown up any examples of incorrect classification, in your view?
   - What prepositions can follow the verb *talk*? Give an example in each case.

3.5.7 WordNet

The NLTK also includes English WordNet, with 155,287 words and 117,659 synonym sets or *synsets*. A synset consists of a set of synonymous words paired with a definition and linked to words with more general or more specific meanings. For example, *table* has various meanings:

```
table.n.01 ['table', 'tabular\_array'], "a set of data arranged in
rows and columns"

table.n.02 ['table'], "a piece of furniture having a smooth flat top
```

```
that is usually supported by one or more vertical legs"

table.n.02 hyponyms: drop-leaf\_table, coffee\_table, pool\_table, altar,

table.n.02 hypernyms: furniture
```

3.6 Some basic corpus analysis

This chapter describes some relatively simple techniques, extracting various kinds of data in suitable formats for human interpretation of the results. Chapters 4 and 5 of the subject guide will look at ways the analysis and interpretation itself can be automated to varying degrees.

Concordancing involves locating every instance of a word or phrase within a text or corpus and presenting it in context, usually a fixed number of words before and after each occurrence.

Collocations are pairs of sequences of words that occur together in a text more frequently than would be expected by chance, and so provide a rough indication of the content or style of a document.

Conditional frequency distributions support an elementary form of statistical analysis. A **frequency distribution** counts observable events and a **conditional frequency distribution** pairs each event with a condition. Some sample applications are:

- Comparing the use of particular words in different genres.
- Comparing word lengths in different languages.

3.6.1 Frequency distributions

The following worked example displays some rudimentary stylistic analysis by ranking the POS tags in a corpus according to frequency.

Calculating tag frequency

1. Import the Brown corpus.
2. List the different categories within the corpus.
3. Count the number of sentences in the science fiction category.
4. Extract all the word tokens from the science fiction category, paired with their tags, and store them in the variable `bsf`. Note that the simplified tagset is selected.
5. Calculate a frequency distribution of the tags: this gives an ordered list of the tags paired with their frequency in the variable `sf_tag_fd`. (Only the 12 most frequent are shown.)

```
>>> from nltk.corpus import brown
>>> brown.categories()
['adventure', 'belles_lettres', 'editorial', 'fiction',
'government', 'hobbies', 'humor', 'learned', 'lore',
'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']
>>> sf_sents = brown.sents(categories = 'science_fiction')
>>> len(sf_sents)
948
>>> bsf = brown.tagged_words(categories = 'science_fiction',
simplify_tags=True)
>>> sf_tag_fd = nltk.FreqDist(tag for (word,tag) in bsf)
>>> sf_tag_fd.keys()
['N', 'V', 'DET', 'PRO', 'P', '.', 'ADJ', ',', 'CNJ', 'ADV', 'VD', 'NP',  ]
>>> sf_tag_fd.tabulate()
N    V   DET  PRO  P    .     ADJ  ,    CNJ  ADV   VD   NP
2232 1473 1345 1268 1182 1078  793  791  685  644   531  467
```

Learning activity

1. Repeat the above process for other categories such as romance, news and religion. How do the frequency distributions and sentence counts enable you to compare the literary styles of these genres? Explain any assumptions you make.
2. Having read through Bird et al. (2009, section 2.1), with particular attention to Table 2-3 on page 50, answer the following questions:
   (a) Summarise the README file from the Reuters corpus.
   (b) Create a variable `soysents` containing all sentences from reports concerning soy products.
   (c) Display the first ten sentences in `soysents`.
   (d) Create a variable `metalwords` containing all words from reports concerning metals.
   (e) What are the most frequently mentioned metals in this collection? Caution: why might this result be less than 100 per cent reliable?

3.6.2 DIY corpus: some worked examples

NLTK's plain text corpus reader can be used to build a 'corpus' from a collection of text files. The resulting corpus will be formatted for access as raw text, lists of words or lists of sentences and can be re-formatted for other functions such as concordancing and finding collocations.

The first example involves a one-text 'corpus' of the recent report from the UK Equality and Human Rights Commission: *How fair is Britain*?

Step 1 Download the report as a PDF from `http://www.equalityhumanrights.com`

Step 2 Manually extract text using Adobe Acrobat or another PDF reader and save as a `.txt` file

Step 3 Point the corpus reader to the directory where you have saved the text file.

```
>>> import nltk
>>> from nltk.corpus import PlaintextCorpusReader
>>> corpus_root = 'C:\NLP-stuff\Corpora'
>>> mycorpus = PlaintextCorpusReader(corpus_root,'.*')
```

We can now use the `raw`, `words` and `sents` methods to display the content in different formats:

```
>>> mycorpus.fileids()
['howfair.txt']
>>> mycorpus.words('howfair.txt')
['Equality', 'and', 'Human', 'Rights', 'Commission', ]
>>> hf_raw = mycorpus.raw('howfair.txt')
>>> hf_raw[:100]
'Equality and Human Rights Commission\r\nTriennial
Review 2010\r\nExecutive Summary\r\nHow fair\r\nis Britain'
>>> mycorpus.sents('howfair.txt')
[['Equality', 'and', 'Human', 'Rights', 'Commission', 'Triennial',
'Review', '2010', 'Executive', 'Summary', 'How', 'fair', 'is',
'Britain', '?'], ...
```

Concordancing and collocations

The `Text` method formats the content in a way which can be accessed by the `concordance` and `collocation` methods. Note that concordancing will always return fixed-length strings which include your target text as a substring, and so may be cut off in the middle of a word.

```
>>> fair=nltk.Text(mycorpus.words('howfair.txt'))
>>> fair.concordance('equal')
Building index...
Displaying 3 of 3 matches:
has narrowed considerably since the equal Pay Act 1970 came into force in 1975
sonal circumstances , should have an equal opportunity to have a say in decisio
that every individual should have an equal chance to make the most of their tal
>>> fair.collocations()
Building collocations list
Human Rights; Rights Commission; Significant findings; Headline data;
Executive Summary; less likely; ethnic minority; life expectancy;
0845 604; domestic violence; hate crime; labour market; disabled people;
mental health; Black Caribbean; different groups; religiously motivated;
sexual assault; minority groups; formal childcare
```

Conditional frequency distribution

Recall that a **frequency distribution** is a set of ordered pairs $< event, \ count >$ where *count* is the number of times *event* occurs. In our context *event* is a word-type and *count* is the number of tokens of that type in a text. A **conditional frequency distribution** is a collection of frequency distributions, each one for a different condition.

For this example we add a second document to the corpus, extracted from a PDF 'Guide to data protection'.

Step 1 Create a single variable `text_word` consisting of pairs of each word-token with the fileid of the document it occurs in.

Step 2 Create a conditional frequency distribution, which will tell you the frequency of each word in both texts.

Step 3 Pick a sample of words which are likely to occur in both documents, and tabulate their comparative frequency.

```
>>> text_word = [(text,word) for text in ['howfair.txt','guide.txt']
     for word in mycorpus.words(text)]
>>> text_word[:10]
[('howfair.txt', 'Equality'), ('howfair.txt', 'and'),
('howfair.txt', 'Human'),('howfair.txt', 'Rights'),
('howfair.txt', 'Commission'), ('howfair.txt', 'Triennial'),
('howfair.txt', 'Review'), ('howfair.txt', '2010'),
('howfair.txt', 'Executive'),
('howfair.txt', 'Summary')]
>>> cfd = nltk.ConditionalFreqDist(text_word)
>>> cfd
<ConditionalFreqDist with 2 conditions>
>>> cfd.conditions()
['guide.txt', 'howfair.txt']
>>> cfd['howfair.txt']
<FreqDist with 16391 outcomes>
>>> cfd['guide.txt']
<FreqDist with 47064 outcomes>
Testing the CFD
>>>  cfd['guide.txt']['fair']
31
>>> cfd['howfair.txt']['fair']
30
>>> keywords = ['fair','police','crime','office','equal','privacy']
>>>>cfd.tabulate(conditions=['howfair.txt','guide.txt'],
samples=keywords)
              fair police crime office equal privacy
howfair.txt   30   15     29    4      2     0
guide.txt     31   16     17    2      0     26
```

Learning activity

Find some suitable electronic documents and follow the above techniques to construct a 'mini-corpus'. The documents in these examples were sourced from UK government websites: you may find similar documents on the website of your own country's government, or of transnational organisations like the European Union or the United Nations. Think of some terms which are likely to occur in several of these documents and compare them using a conditional frequency distribution. If you can find a lengthy report which is issued along with a shorter summary, it is an interesting exercise to pick some key terms and see if their comparative frequency is the same or similar in the original document and the summary.

3.7 Summary

1. A corpus is a collection of linguistic data which was not originally produced for the purposes of linguistic analysis. Properly speaking it should be constructed so as to be balanced and representative. If a corpus includes any kind of

annotation, it is good practice to use multiple annotators for at least a sample of the corpus and report the level of **inter-annotator agreement** that was achieved.

2. Some uses of corpora include:
   - Lexicography (compiling dictionaries).
   - Compiling grammars for education and reference purposes.
   - Stylistics: developing techniques to identify the author or genre of a document; investigating the effect on language use of different channels such as email, chat, face-to-face conversation, telephone calls and hand-written letters.
   - Training and evaluation in linguistic research, using machine learning techniques.
3. This chapter includes brief descriptions of several well-known and widely-used corpora such as the Brown corpus, the BNC and the Penn Treebank.
4. Students on this course can access a variety of corpora through online interfaces or by using corpus tools provided with the NLTK.
5. Some simple techniques for analysing corpora include concordancing, collocations and (conditional) frequency distributions. None of these techniques involve automated linguistic analysis: the interpretation of the outputs is down to the analyst.

3.8 Sample examination question

a) Explain what is meant by the following types of corpus, and describe an example in each category that you have encountered during this course:

- isolated
- categorised
- overlapping
- temporal.

b) What applications would a **tagged** and **parsed** corpus be suitable for? What are some advantages and disadvantages of using an automated tagger to build such a corpus?

c) Suppose the following lists show the number of sentences and the most commonly occurring part-of-speech tags in three different categories of text in a corpus, with their frequency of occurrence in brackets. What can you say about the styles of these documents from studying these results? Discuss any assumptions you make.

Category A (4623 sentences) N (22236) P (10845) DET (10648) NP (8336) V (7346) ADJ (5435) ‘,’ (5188) ‘.’ (4472) CNJ (4227) PRO (3408) ADV (2770) VD (2531) . . .

Category B (948 sentences) N (2232) V (1473) DET (1345) PRO (1268) P (1182) ‘.’ (1078) ADJ (793) ‘,’ (791) CNJ (685) ADV (644) VD (531) NP (467) . . .

Category C (1716 sentences) N (7304) P (4364) DET (4239) V (3863) ADJ (2620) CNJ (2324) PRO (2243) ‘,’ (1913) ‘.’ (1892) ADV (1485) NP (1224) VN (952) . . .

Appendix C

Answers to selected activities

This section includes model solutions to some of the exercises and activities where appropriate. In other cases there is no 'correct' answer and the point of the activity is to stimulate you to engage in independent self-directed learning.

Chapter 2: Introducing NLP: patterns and structure in natural language

Identify parts of speech, page 14

- Jack *(Proper Noun)* and *(conjunction)* Jill *(Proper Noun)* went *(verb)* up *(preposition)* the *(determiner)* hill *(noun)*.
- The *(determiner)* owl *(noun)* and *(conjunction)* the *(determiner)* pussycat *(noun)* went *(verb)* to *(preposition)* sea *(noun)*.

Operation of a finite-state machine, page 17

1. John swam.
2. (a) John and Mary walked on the hill.
   (b) The cat sat on the mat and slept.
   (c) John or a fish walked on a hill and barked.
   (d) . . .

Coding regular expressions, page 19

1. a(aa)*(bb)*
2. (aaa)|(aab)|(abb)|(bbb)|(bba)|(baa)|(aba)|(bab)

Regular grammars, page 21

S $\rightarrow$ either | if S1

S $\rightarrow$ the | a | one S2

S2 $\rightarrow$ happy S2

S2 $\rightarrow$ (boy | girl | dog) eats ('ice cream' | 'hot dogs' | candy) S3

S3 $\rightarrow$ or | then S2

S3 $\rightarrow \epsilon$

This is a slightly idealised rendering of Pinker's state diagram which appears to have no halting state.

The problem with this grammar can be clearly seen: the rule *S3 $\rightarrow$ or | then S2* has no connection with the rule that introduces *either* or *if* and so there is no way to ensure that the appropriate conjunction is used.

Past tense forms, page 25

The general idea is that rules need to be **conditional** in order to handle non-standard cases before applying general regularities: so a reasonable rule based on this data set might be:

welcome $\rightarrow$ welcomed **else**
-come $\rightarrow$ -came

Chapter 3: Getting to grips with natural language data

Online corpus queries, page 37

Examples of incorrect tagging: search on `to total/V` gives examples like:

- . . . a ticket **to total** oblivion.
- . . . to describe it **to total** strangers.

as well as 'correct' examples like

- . . . thought **to total** about 1,500 families . . .
- . . . a great opportunity **to total** and celebrate all the small wins made over the year

Using NLTK tools, page 39

1. 'Stylistic' analysis

Science Fiction (948 sentences) N (2232) V (1473) DET (1345) PRO (1268) P (1182) '.' (1078) ADJ (793) ',' (791) CNJ (685) ADV (644) VD (531) NP (467) ...

News (4623 sentences) N (22236) P (10845) DET (10648) NP (8336) V (7346) ADJ (5435) ',' (5188) '.' (4472) CNJ (4227) PRO (3408) ADV (2770) VD (2531) ...

Religion (1716 sentences) N (7304) P (4364) DET (4239) V (3863) ADJ (2620) CNJ (2324) PRO (2243) ',' (1913) '.' (1892) ADV (1485) NP (1224) VN (952) ...

Some counts:

SF: N 2232, PRO 1268, NP 467 ADJ 793

NE: N 22236, NP 8336, PRO 3408 ADJ 5435

RE: N 7304, PRO 2243 NP 1224 ADJ 2620

SF: S 948, COMMA 791, CNJ 685

NE: COM 5188 S 4623 CNJ 4227

RE: CNJ 2324 COM 1913 S 1716

Some tentative conclusions:

1. Reference and description: both SF and RE use pronouns more than proper names; News has more proper names. Hard to interpret without further analysis: if an SF work includes a lot of dialogue for example, it might be more natural for characters to refer to each other as I, we, you and so on rather than by name. And news stories tend to be about named individuals.
2. Syntactic complexity: we cannot be very precise with this data but it looks as if the SF genre has the least syntactic complexity and the RE genre the highest, judging from the numbers of commas and conjunctions per sentence. Of course we cannot tell whether these tokens are connecting clauses or other types of phrases.

In an examination or coursework question, you would get credit for discussing these and other characteristics in the light of your impressionistic understanding of the typical styles for these genres.

2. Reuters

- Display the README file from the Reuters corpus.

```
from nltk.corpus import reuters
desc = reuters.readme()
print desc
```

- Create a variable `soysents` containing all sentences from reports concerning soy products.

  ```
  reuters.categories()
  ```

  Pick out categories relating to soy:

  ```
  soysents = reuters.sents(categories=['soy-meal', ...])
  ```

- Display the first ten sentences in `soysents`.

  ```
  print soysents[:10]
  ```

- Create a variable `metalwords` containing all words from reports concerning metals.

  ```
  metalwords = reuters.words(categories = ['alum','copper','gold', ...])
  ```

  (Note that inspection of texts in the *alum* category confirms that they are about aluminium.)

- What are the most frequently mentioned metals in this collection? Caution: why might this result be less than 100 per cent reliable?

  ```
  from nltk.book import *
  freqmetal = FreqDist(metalwords)
  freqmetalkeys = freqmetal.keys()
  freqmetal[:100]
  ```

  Remember that the contents of a frequency distribution are listed in the order of their frequency of occurrence. By scanning the output you should see that the first three metals listed are gold, copper and steel. However caution is in order as Reuters is an overlapping corpus, so we may be double-counting some occurrences. These metals may also be mentioned under the category `strategic-metal`, or some reports may mention more than one kind of metal and so come under multiple categories.

Chapter 4: Computational tools for text analysis

Comparing stemmers, page 48

Lancaster rules
- Remove *ist/s/e/ing/en/th/ity/ate/al/a/ed/ment/ation*.
- Replace *-ies* with *-y*.
- Some motivations: reduce verbs to stem form, remove plural affix and return stem in irregular cases (*study*), remove ordinal affix *-th*, remove affixes that form nominalisations or adjectives: *-ment*, *-al*.

Porter rules
- Remove *s/ing/ity/e/ate/ed/ment/ational*.
- Replace *-y* with *-i*.
- Motivations: similar to Lancaster rules but applied more sparingly.

Errors Lancaster removes *-th* from *south* as if it were an ordinal and *-e* from *are* though *ar* is not a stem here; not clear why *-a* removed from *area*. Lancaster and Porter both treat some proper names as common nouns; for example, by removing the last letter from *Lyons* and *Stanhope* (both) or by postulating an *-i* stem for names ending in *-y* (Porter).