The Linguist's Guide to Statistics

Don't Panic

Brigitte Krenn
Universität des Saarlandes
FR 8.7, Computerlinguistik
Postfach 1150
D-66041 Saarbrücken
Germany
krenn@coli.uni-sb.de

Christer Samuelsson
Bell Laboratories
600 Mountain Ave
Room 2D-339
Murray Hill, NJ 07974
USA
christer@research.bell-labs.com

http://www.coli.uni-sb.de/{~krenn,~christer}

December 19, 1997

In a review of Eugene Charniak's book "Statistical Language Learning" in Computational Linguistics Vol. 21, No. 1 of March, 1995, David Magerman writes:

> The \$64,000 question in computational linguistics these days is: What should I read to learn about statistical natural language processing? I have been asked this question over and over, and each time I have given basically the same reply: there is no text that addresses this topic directly, and the best one can do is find a good probability-theory textbook and a good information-theory textbook, and supplement those texts with an assortment of conference papers and journal articles.
>
> ...
>
> The overriding concern should be to learn (and teach) the mathematical underpinnings of the statistical techniques used in this field. The field of statistical NLP is very young, and the foundations are still being laid. Deep knowledge of the basic machinery is far more valuable than the details of the most recent unproven ideas.

So let's get down to it!

Contents

Preface

This compendium is the result of the authors' attempts at teaching courses on statistical approaches in Computational Linguistics and Natural Language Processing, and it is continuously evolving and undergoing revision. Although we have already put considerable effort into writing, correcting and updating this compendium, there are numerous errors and omissions in it that we hope to deal with in the nearest future. The most recent release of this compendium, in the form of a uuencoded gzipped PostScript file, `stat_cl.ps.gz.uu`, or as an ordinary PostScript file, `stat_cl.ps`, can be retrieved from one of the authors' WWW homepage at

`http://www.coli.uni-sb.de/{~christer,~krenn}`

The ambition is to cover most statistical, stochastic and probabilistic approaches in the field. As will be obvious by inspecting the table of contents, this is by no means yet the case.

The first three chapters have a distinct text-book character: Chapter 1 provides the necessary prerequisites in Probability Theory and Statistics, Chapter 2 describes some statistical models that are much in use in the field, and Chapter 3 constitutes an introduction to Corpus Linguistics. Chapters 4 and 5 discuss how statistical models and techniques are applied to various task in Computational Linguistics and Natural Language Processing, relating the material presented in the previous chapters to relevant scientific articles.

Feel free to use this compendium, but please do acknowledge the source. Also, any comments or suggestions to improvements are more than welcome, and are most likely to enhance future versions of the compendium. We are already greatly indebted for this to Rens Bod, Bob Carpenter, John Carroll, Ted Dunning, Jussi Karlgren, Kimmo Koskenniemi, David Magerman, David Milward, Joakim Nivre, Khalil Sima'an, Atro Voutilainen and numerous students at the University of the Saarland, Uppsala University, the University Helsinki and to course participants at the ESSLLI-97 summer school in Aix-en-Provence. Special credit is due to Thorsten Brants, who wrote the first version of the section on Hidden Markov Models, and to our online mathematician Åke H. Samuelsson. Parts of the compendium are used in a web-based introductory course on statistical natural language processing which is set up by Joakim Nivre at Göteborg University. It can be accessed via `http://www.ling.gu.se/~nivre/kurser/wwwstat/`.

Saarbrücken New York

December 1997

Brigitte Krenn Christer Samuelsson

Chapter 1

Basic Statistics

1.1 The Stability of the Relative Frequency

Although an elegant theory in its own right, the main reason that statistics has grown as popular as it has is something that is known as "The stability of the relative frequency". This is the empirical observation that there is some structure also in random processes. If we for example flip a coin a large number of times, we will note that in approximately half the cases the outcome is "heads" and in approximately half the cases it is "tails". If we flip a coin a small number of times, say only once, twice or three times, this is not consistently the case.

The proportion of times a certain outcome occurs is called the *relative frequency* of the outcome. If n_u is the number of times the outcome u occurs in n trials, then $\frac{n_u}{n}$ is the relative frequency of u. The relative frequency is often denoted f_n.

Empirically, there seems to be some number which the relative frequency stabilizes around after a large number of trials. A fundamental assumption in statistics is that such numbers exist. These numbers are called *probabilities*.

1.2 Elementary Probability Theory

Introductory presentations of statistics often disguise probability theory in set theory, and this is no exception.

1.2.1 Sample Space

The *sample space* is a set of *elementary outcomes*. An *event* is a subset of the sample space. Sample spaces are often denoted Ω, and events are often called A, B, C, etc. Let's get this grounded with an example:

> **Example:** For a normal die[1], the six elementary outcomes are One, Two, Three, Four, Five and Six, and thus the sample space Ω is the set $\{\text{One}, \text{Two}, \text{Three}, \text{Four}, \text{Five}, \text{Six}\}$. The events are various subsets of this set, e.g., "the outcome is One", $\{\text{One}\}$; "the outcome is less than four", $\{\text{One}, \text{Two}, \text{Three}\}$; the outcome is odd, $\{\text{One}, \text{Three}, \text{Five}\}$; etc. In fact, there are $2^6 = 64$

[1] "Die" is the singular form of "dice", a sort of mechanical random generators used in games like Craps, Monopoly and Fia-med-knuff.

different subsets of Ω, i.e., there are 64 distinct events in Ω, including the empty set $\emptyset$ and Ω itself, see Section 1.2.7.

Statisticians are a kind of mathematicians, and like to distinguish between for example the outcome One, which is a basic element, and the event $\{\text{One}\}$, which is a set consisting of one element. We will try to keep this up for a while.

1.2.2 Probability Measures

A *probability measure* P is a function from events in the sample space Ω, i.e., from the set of subsets of Ω, [2] to the set of real numbers in $[0, 1]$ that has the following properties:

$$\begin{array}{ll} 1) & 0 \leq P(A) \leq 1 \quad \text{for each event } A \subseteq \Omega \\ 2) & P(\Omega) = 1 \\ 3) & A \cap B = \emptyset \Rightarrow P(A \cup B) = P(A) + P(B) \end{array}$$

Intuitively, the total mass of 1 is distributed throughout the set Ω by the function P. This will assign some particular mass to each subset A of Ω. This mass is the probability of event A, denoted $P(A)$. $A \cap B = \emptyset$ means that A and B are disjoint, i.e., that they have no common element.

Example: For a fair (unbiased) die, where as we recall the sample space is the set $\{\text{One}, \text{Two}, \text{Three}, \text{Four}, \text{Five}, \text{Six}\}$, the mass 1 is evenly and justly divided among the six different singleton sets. Thus $P(\{\text{One}\}) = P(\{\text{Two}\}) = P(\{\text{Three}\}) = P(\{\text{Four}\}) = P(\{\text{Five}\}) = P(\{\text{Six}\}) = \frac{1}{6}$. If A is the event of the outcome being divisible by two, i.e., the subset $\{\text{Two}, \text{Four}, \text{Six}\}$, and if B is the event of the outcome being divisible by three, i.e., the subset $\{\text{Three}, \text{Six}\}$, then $P(A) = \frac{1}{2}$ and $P(B) = \frac{1}{3}$.

A loaded die, used by cheaters, would not have the probability mass as evenly distributed, and could for example assign $P(\{\text{Six}\})$ a substantially larger value than $\frac{1}{6}$.

Some immediate corollaries that are worth remembering fall out from the definition of a probability measure:

$$\begin{array}{ll} a) & P(B \setminus A) = P(B) - P(A \cap B) \\ b) & A \subseteq B \Rightarrow P(A) \leq P(B) \\ c) & P(\bar{A}) = 1 - P(A) \\ d) & P(\emptyset) = 0 \\ e) & P(A \cup B) = P(A) + P(B) - P(A \cap B) \end{array}$$

$B \setminus A$ denotes the difference set B minus A, i.e., the set of elements in B that are not members of A. $\bar{A}$ denotes the complement of A, i.e., $\Omega \setminus A$.

Proofs:

$$\begin{array}{ll} a) & B = (B \setminus A) \cup (A \cap B) \;;\; (B \setminus A) \cap (A \cap B) = \emptyset \Rightarrow \\ & P(B) = P((B \setminus A) \cup (A \cap B)) = P(B \setminus A) + P(A \cap B) \end{array}$$

[2] This is called the *power set* of Ω and is denoted 2^{Ω}.

$$
\begin{array}{ll}
b) & A \subseteq B \;\Rightarrow\; A \cap B \;=\; A \\
 & 0 \;\leq\; P(B \setminus A) \;=\; P(B) - P(A \cap B) \;=\; P(B) - P(A) \\
c) & \bar{A} \;=\; \Omega \setminus A \;;\; A \cap \Omega \;=\; A \;\Rightarrow \\
 & P(\bar{A}) \;=\; P(\Omega \setminus A) \;=\; P(\Omega) - P(A \cap \Omega) \;=\; 1 - P(A) \\
d) & A \;=\; A \cup \emptyset \;;\; A \cap \emptyset \;=\; \emptyset \;\Rightarrow \\
 & P(A) \;=\; P(A \cup \emptyset) \;=\; P(A) + P(\emptyset) \\
e) & A \cup B \;=\; A \cup (B \setminus A) \;;\; A \cap (B \setminus A) \;=\; \emptyset \;\Rightarrow \\
 & P(A \cup B) \;=\; P(A \cup (B \setminus A)) \;=\; P(A) + P(B \setminus A) \;= \\
 & =\; P(A) + P(B) - P(A \cap B)
\end{array}
$$

□

Note how a) follows from 3) and how a) is used to prove b), c) and e). However, d) follows more easily from 3) directly.

1.2.3 Independence

The probability of two events A and B both occurring is the probability of the intersection of the sets A and B, $P(A \cap B)$.

Two events A and B are said to be *independent* iff[3]

$$P(A \cap B) \;=\; P(A) \cdot P(B) \tag{1.1}$$

Intuitively, this means that the probability of A and B occurring simultaneously can be established directly from the individual probabilities of A and B.

> **Example:** To continue the example of the fair die, with the events A of the outcome being divisible by two and B of the outcome being divisible by three, we note that $P(A \cap B) = P(\{\text{Six}\}) = \frac{1}{6}$, and that $P(A) \cdot P(B) = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$. Thus A and B are independent.
>
> Let C be the event of the outcome being divisible by four and note that $P(C) = P(\{\text{Four}\}) = \frac{1}{6}$. Intuitively, the property of being divisible by four is related to the property of being divisible by two, and if we calculate on the one hand $P(A \cap C) = P(\{\text{Four}\}) = \frac{1}{6}$, and on the other hand $P(A) \cdot P(C) = \frac{1}{2} \cdot \frac{1}{6} = \frac{1}{12}$, we see that A and C are not independent.

1.2.4 Conditional Probabilities

$P(A \mid B)$ is a so-called *conditional probability*, namely the probability of event A given that event B has occurred, and is defined as

$$P(A \mid B) \;=\; \frac{P(A \cap B)}{P(B)} \tag{1.2}$$

This is the updated probability of A once we have learned that B has occurred. $P(A)$ is often called the *prior probability* of A since we have no

[3] "Iff" means if and only if.

prior information, while $P(A \mid B)$ is called the *posterior probability* of A (knowing B), since it is defined posterior to that, that event B occurred.

Intuitively, after the event B has occurred, an event A is replaced by the event $A \cap B$, and the sample space Ω is replaced by the event B, and the probabilities are renormalized accordingly, which means dividing by $P(B)$.

The probability of an event A can change drastically when one learns that some other event has occurred. The following example is rather extreme, but illustrates this point:

$$\begin{aligned} P(A \mid A) &= 1 \\ P(A \mid \bar{A}) &= 0 \end{aligned}$$

Now, if A and B are independent, then $P(A \cap B) = P(A) \cdot P(B)$ and thus $P(A \mid B) = \dfrac{P(A \cap B)}{P(B)} = \dfrac{P(A) \cdot P(B)}{P(B)} = P(A)$. This means that no new information is gained about A by knowing that B has occurred.

> **Example:** Again continuing the example of the fair die, with the events A of the outcome being divisible by two, B of the outcome being divisible by three, and C of the outcome being divisible by four, we note that $P(A \mid B) = \dfrac{P(A \cap B)}{P(B)} = \dfrac{\frac{1}{6}}{\frac{1}{3}} = \dfrac{1}{2} = P(A)$, as should be, considering that A and B are independent. On the other hand, $P(A \mid C) = \dfrac{P(A \cap C)}{P(C)} = \dfrac{P(\{\text{Four}\})}{P(\{\text{Four}\})} = 1 \neq P(A) = \dfrac{1}{2}$, since A and C are not independent.

1.2.5 Bayesian Inversion

Either by returning to the definition of conditional probability, Eq. (1.2), or by noting that if both A and B are to occur, then first B must occur, and then A must occur, or the other way around, we can establish that

$$P(B) \cdot P(A \mid B) \;=\; P(A \cap B) \;=\; P(A) \cdot P(B \mid A)$$

This directly gives us the *Bayesian inversion formula*[4]:

$$P(A \mid B) \;=\; \frac{P(A) \cdot P(B \mid A)}{P(B)} \tag{1.3}$$

This formula relates the probability of an event A conditional on another event B, i.e. $P(A \mid B)$, with the probability of event B conditional on event A, $P(B \mid A)$, and is useful if the former quantity is not easily determined, while the latter is.

> **Example:** We turn again to the example of the fair die, with the events A of the outcome being divisible by two, B of the outcome being divisible by three, and C of the outcome being divisible by four. If we wish to calculate $P(C \mid A)$, i.e., the probability that the outcome is divisible by four given that it is divisible by two, we observe that the probability $P(A \mid C)$

[4] The formula is of almost religious importance to so-called Bayesianists, see [Pearl 1988], pp. 29–41.

is 1, since divisibility by four implies divisibility by two, and recalling that $P(A) = \frac{1}{2}$ and that $P(C) = \frac{1}{6}$, we can establish that $P(C \mid A) = \frac{P(C) \cdot P(A \mid C)}{P(A)} = \frac{\frac{1}{6} \cdot 1}{\frac{1}{2}} = \frac{1}{3}$.

1.2.6 Partitions

Assume that we have a collection $\{A_i : i = 1, \ldots, N\}$ of events (sets) such that

$$\begin{aligned} \Omega &= \bigcup_{i=1}^{N} A_i \\ A_i \cap A_j &= \emptyset \quad \text{for } i \neq j. \end{aligned}$$

The first equality means that the sets A_j cover the sample space Ω and the second equality states that all events A_i are disjoint. This situation is illustrated in Figure 1.1. We say that $\{A_i : i = 1, \ldots, N\}$ constitutes a

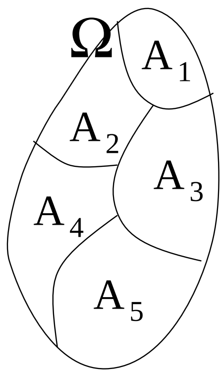

Figure 1.1: The sets A_1 through A_5 constitute a partition of Ω.

partition of Ω and we can establish the following formula:

$$\begin{aligned} P(B) \;=\; P(B \cap \Omega) \;=\; P(B \cap \bigcup_{i=1}^{N} A_i) \;=& \qquad (1.4) \\ =\; P(\bigcup_{i=1}^{N} B \cap A_i) \;=\; \sum_{i=1}^{N} P(B \cap A_i) \;=& \; \sum_{i=1}^{N} P(B \mid A_i) \cdot P(A_i) \end{aligned}$$

This can be seen as follows: Since the sample space Ω must contain the event B, B equals $B \cap \Omega$. Thus the probability $P(B)$ is the same as the probability $P(B \cap \Omega)$. As the sample space consists of the union of the disjoint events A_i, we have $\Omega = \bigcup_{i=1}^{N} A_i$, and we can thus write $P(B \cap \bigcup_{i=1}^{N} A_i)$ instead of $P(B \cap \Omega)$. Now, $B \cap \bigcup_{i=1}^{N} A_i$ is equivalent to $\bigcup_{i=1}^{N} B \cap A_i$, and we can thus rewrite $P(B \cap \bigcup_{i=1}^{N} A_i)$ as $P(\bigcup_{i=1}^{N} B \cap A_i)$. Furthermore, the probability of the union of disjoint events equals the sum of the probability of the individual events. Thus $P(\bigcup_{i=1}^{N} B \cap A_i) = \sum_{i=1}^{N} P(B \cap A_i)$. The last equality of the equation follows from the definition of conditional probabilities, Eq. (1.2).

Equation 1.4 is very useful when there is a natural partition $\{A_i\}$ of Ω, and it is easy to calculate the probabilities $P(A_i)$ and the conditional probabilities $P(B \mid A_i)$.

> **Example:** In the example of the fair die, one way one could determine the probability of event B, that the outcome is divisible by three, is to use the natural partition $A_1 = \{\text{One}\}, \ldots, A_6 = \{\text{Six}\}$ where $P(A_i) = \frac{1}{6}$ for all i, and where $P(B \mid A_i)$ is 1 if i is divisible by 3 and 0 if it is not. Thus $P(B) = 0 \cdot \frac{1}{6} + 0 \cdot \frac{1}{6} + 1 \cdot \frac{1}{6} + 0 \cdot \frac{1}{6} + 0 \cdot \frac{1}{6} + 1 \cdot \frac{1}{6} = \frac{1}{3}$.

1.2.7 Combinatorics

And now for something completely different: This section contains some results from elementary combinatorics that will be needed later, e.g., in Section 1.5.1.

The following table summarizes the number of different ways k elements can be selected from n elements. This is in fact an instance the much dreaded "urn and balls" scenario, which has plagued statistics as long as anyone can remember. This particular incarnation can be viewed as the even more feared "Lotto" scenario where the urn contains n balls numbered from 1 to n, and where we will at random draw a ball k times. In doing this, we may or may not replace selected elements, i.e., we may or may not put the drawn ball back into the urn, and we may or may not be interested in the order in which they are selected. This gives us four different cases:

	Without replacement	With replacement
Ordered	$(n)_k$	n^k
Unordered	$\binom{n}{k}$	$\binom{n+k-1}{k}$

Here

$$
\begin{aligned}
n! &= n \cdot (n-1) \cdot \ldots \cdot 1 \\
(n)_k &= n \cdot (n-1) \cdot \ldots \cdot (n-k+1) = \frac{n!}{(n-k)!} \\
n^k &= \overbrace{n \cdot \ldots \cdot n}^{k \text{ factors}} \\
\binom{n}{k} &= \frac{(n)_k}{k!} = \frac{n!}{(n-k)!k!}
\end{aligned}
$$

$n!$ is read out "n factorial". By definition $0! = 1$. In particular, the number of *permutations* of n elements is $(n)_n = n!$. $\binom{n}{k}$ is read out "n over k".

We devote the rest of the section to proofs of these formulas:

Ordered, without replacement:

We wish to select k items from n items, where selected items are not replaced, and where we are interested in the order in which they are selected. First we have n possible choices, then we have $n-1$ choices left, since we have removed one item, etc., down to the kth choice, where we have $n-(k-1) = n-k+1$ alternatives. Thus $n \cdot (n-1) \cdot \ldots \cdot (n-k+1) = (n)_k$ possibilities in total. $\square$

Ordered with replacement:

We wish to select k items from n items, where selected items are replaced, and where we are interested in the order in which they are selected. First we have n possible choices, then we have again n possible choices, since we put back the selected item, etc., down to the kth choice, where we still have n alternatives. Thus n^k possibilities in total. $\square$

Unordered without replacement:

We wish to select k items from n items, where selected items are not replaced, and where we are not interested in the order in which they are selected. We will first select k items paying attention to the order. Then there are $(n)_k$ possibilities. For each *set* of k different items, there will be $k!$ permutations of them. Thus among these $(n)_k$ sequences of k items there are for each such set $k!$ sequences that should be considered to be the same, since we are not interested in the order in which items are selected. We will then factor out this by dividing $(n)_k$ by $k!$. In short:

$$\#\text{unordered without replacement} = \frac{\#\text{ordered without replacement}}{\#\text{permutations}} =$$

$\dfrac{(n)_k}{k!} = \begin{pmatrix} n \\ k \end{pmatrix}$. Thus $\begin{pmatrix} n \\ k \end{pmatrix}$ possibilities in total. $\square$

Unordered with replacement:

We wish to select k items from n items, where selected items are replaced, and where we are not interested in the order in which they are selected. To this end, we make a list of the n items, and make a tick beside "i" each time we select item i, $i = 1, \ldots, n$:

k	1	...	i	j	...	n
n	$\vert$	...	$\vert\vert$	$\vert$	...	$\vert$

We can represent this by a sequence of n zeros and k ones, where the number of ones immediately preceding the ith zero indicates the number of times item i is selected:

$$\begin{array}{c} \overbrace{0\ldots00\ldots0}^{n} \\ \underbrace{10\ldots11010\ldots10}_{n+k} \end{array}$$

Thus, this reduces to selecting k elements (ones) from $n+k-1$ elements without replacement, and where the order is not significant. (The "minus one" is since the last digit cannot be a one.) Thus $\begin{pmatrix} n+k-1 \\ k \end{pmatrix}$ possibilities in total. $\square$

The last two proofs used the well-known tactics of mathematicians of reducing to Case One.

> **Story time:** A mathematician is faced with the following scenario: A bucket, a well and a house on fire. He immediately finds the correct solution of using the bucket to fetch water from the

well and extinguish the fire. Next, the mathematician is faced with the scenario of a bucket, a well and a house that is not on fire. He rapidly sets fire to the house and reduces the problem to Case One.

1.3 Stochastic Variables

A *stochastic*, or *random*, *variable* ξ is a function from a sample space Ω to R, the set of real numbers. Thus, if $u \in \Omega$, then $\xi(u) \in R$. Figure 1.2 illustrates this. It may seem strange to call a function a variable, but this convention will die hard, if ever. There are two dominant conventions for denoting stochastic variables — one using Greek letters like $\xi, \eta, \zeta, \ldots$ and the other using Roman capital letters $X, Y, Z, \ldots$ We will use the former convention.

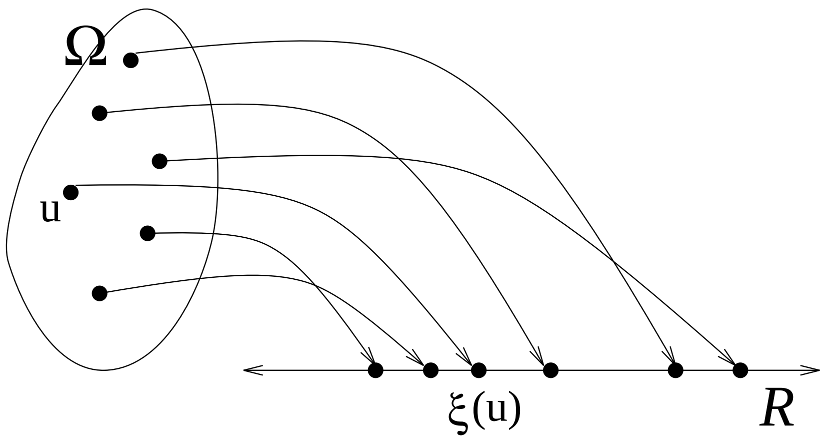

Figure 1.2: A random variable is a function from Ω to R.

Example: The running example of the fair die does not illustrate this very well, since there is a mapping form the sample space $\Omega = \{\text{One}, \text{Two}, \text{Three}, \text{Four}, \text{Five}, \text{Six}\}$ to the subset $\{1, 2, 3, 4, 5, 6\}$ of R that is so obvious, that it seems artificial to distinguish between the two. However, let us define the stochastic variable ζ as the function from Ω to R such that $\zeta(\text{One}) = 1, \zeta(\text{Two}) = 2, \ldots, \zeta(\text{Six}) = 6$.

1.3.1 Distribution Function

Let A be a subset of R, and consider the inverse image of A under ξ, i.e., $\xi^{-1}(A) = \{u : \xi(u) \in A\} \subseteq \Omega$. We will let $P(\xi \in A)$ denote the probability of this set, i.e., $P(\xi^{-1}(A)) = P(\{u : \xi(u) \in A\}) = P(\xi \in A)$. See Figure 1.3.

If A is the interval $(-\infty, x]$,[5] then the real-valued function F defined by

$$F(x) \;=\; P(\{u : \xi(u) \leq x\}) \;=\; P(\xi \leq x) \quad \text{for all } x \in R$$

is called the *distribution function* of the random variable ξ. Sometimes F is denoted F_ξ to indicate that it is the distribution function of the particular random variable ξ.

[5] (a, b) denotes an open interval, while $[a, b]$ denotes a closed interval, i.e., in the former case a and b do not belong to the interval, while in the latter case they do.

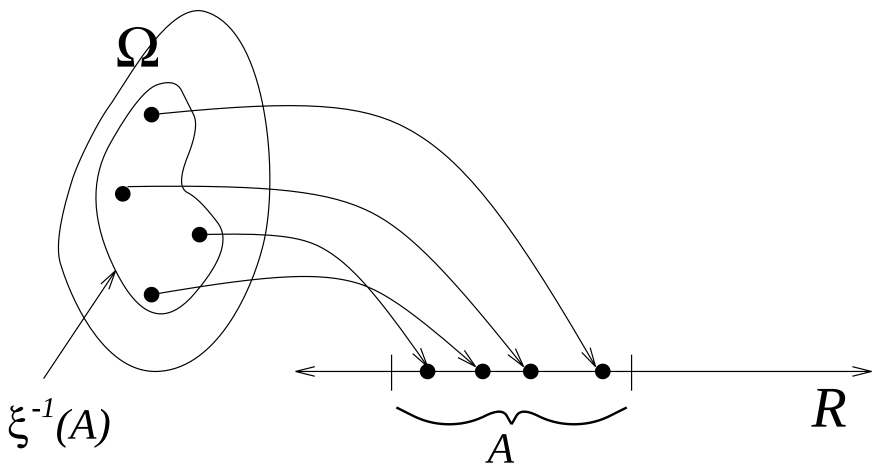

Figure 1.3: $P(\xi \in A)$ is defined through $P(\xi^{-1}(A)) = P(\{u : \xi(u) \in A\})$.

Example: In the reappearing example of the fair die, the distribution function F_ζ is defined by

$$F_\zeta(x) = \begin{cases} \frac{0}{6} = 0 & \text{for} \quad x < 1, \\ \frac{1}{6} & \text{for} \quad 1 \leq x < 2, \\ \frac{2}{6} = \frac{1}{3} & \text{for} \quad 2 \leq x < 3, \\ \frac{3}{6} = \frac{1}{2} & \text{for} \quad 3 \leq x < 4, \\ \frac{4}{6} = \frac{2}{3} & \text{for} \quad 4 \leq x < 5, \\ \frac{5}{6} & \text{for} \quad 5 \leq x < 6, \\ \frac{6}{6} = 1 & \text{for} \quad 6 \leq x. \end{cases}$$

The graph of this function is depicted in Figure 1.4.

Some rather useful corollaries that are worth remembering include:

a) $P(\xi > x) \;=\; 1 - F(x)$

b) $P(a < \xi \leq b) \;=\; F(b) - F(a)$

c) F is a nondecreasing function, i.e. if $x_1 < x_2$, then $F(x_1) \leq F(x_2)$.

d) $\lim_{x \to -\infty} F(x) \;=\; 0$

e) $\lim_{x \to \infty} F(x) \;=\; 1$

The proofs of a–c) are left as an exercise.

1.3.2 Discrete and Continuous Stochastic Variables

The image of the sample space Ω in R under the random variable ξ, i.e., the range of ξ, is called the *sample space of the stochastic variable* ξ and is

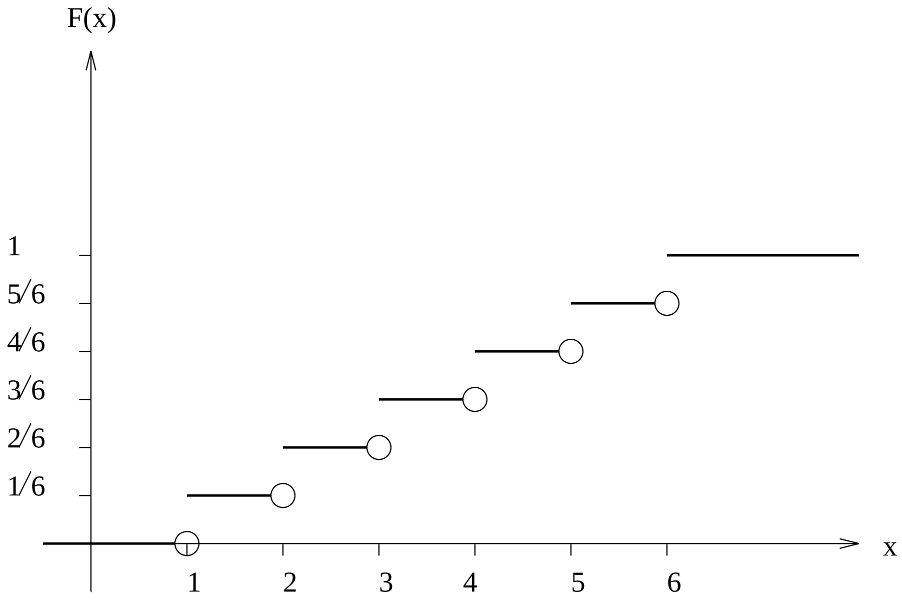

Figure 1.4: Fair die: Graph of the distribution function.

denoted Ω_ξ. In short $\Omega_\xi = \xi(\Omega)$.

Stochastic variables come in several varieties. Two common types are discrete and continuous stochastic variables:

- A random variable is *discrete* iff Ω_ξ is finite or countable.
- A random variable is *continuous* iff
  1. F is continuous, and
  2. F is differentiable with a continuous derivative except in at most a finite number of points.

The attentive reader may have noticed that Calculus is entering into this presentation through the back door. This is really necessary for a good understanding of the subject, in particular of continuous random variables, and we refer those interested in this to some introductory book on calculus until we have had time to include an appendix on the topic.

> **Example:** In the persistent example of the fair die, $\Omega_\zeta = \{1, 2, 3, 4, 5, 6\}$, and ζ is thus a discrete random variable.

1.3.3 Frequency Function

Another convenient way of characterizing a random variable is by its *frequency function* f:

- For a discrete random variable, $f(x) = P(\xi = x)$.
- For a continuous random variable, $f(x) = F'(x) = \frac{d}{dx}F(x)$. [6]

[6] $F'(x) = \frac{d}{dx}F(x) = \lim_{h\to 0} \dfrac{F(x+h) - F(x)}{h}$ is the *derivative* of the function F at point x. Intuitively, this is the slope of the curve at point x when $F(x)$ is plotted against x.

Sometimes f is denoted f_ξ to indicate that it is the frequency function of the particular random variable ξ. The frequency function of a discrete stochastic variable is often referred to as the *probability function*, and the frequency function of a continuous stochastic variable is often referred to as the *probability density function*.

> **Example:** Continuing the perannial example of the fair die, we find that the frequency function f_ζ is
>
> $$f_\zeta(1) = f_\zeta(2) = f_\zeta(3) = f_\zeta(4) = f_\zeta(5) = f_\zeta(6) = \frac{1}{6}$$
>
> This is an example of a uniform distribution, i.e., $f(x)$ has the same value for all elements in Ω_ζ. Note that a uniform distribution can only be assigned to a finite or bounded sample space. Figure 1.5 shows the graphical representation of the frequency function f_ζ.

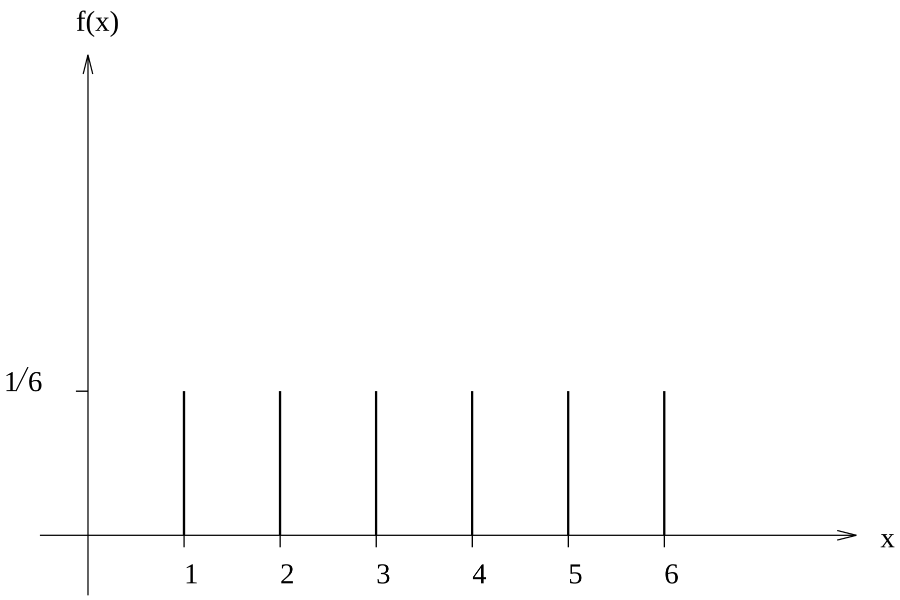

Figure 1.5: Fair die: Graph of the frequency function

The probabilities of events can be calculated from the frequency function:

- For a discrete random variable

  $$P(\xi \in A) \;=\; \sum_{x \in A} f_\xi(x)$$

  and in particular

  $$P(\xi \leq x) \;=\; F_\xi(x) \;=\; \sum_{i:x_i \leq x} f_\xi(x_i)$$

- For a continuous random variable,

  $$P(\xi \in A) \;=\; \int_A f_\xi(x)\;\; dx$$

and in particular

$$P(\xi \leq x) \quad = \quad F_\xi(x) \quad = \quad \int_{-\infty}^{x} f_\xi(t) \;\; dt$$

Since all probabilities must sum to one, we have:

- For a discrete random variable

$$P(\Omega_\xi) \quad = \quad \sum_{x \in \Omega_\xi} f_\xi(x) \quad = \quad 1$$

- For a continuous random variable

$$P(\Omega_\xi) \quad = \quad \int_{-\infty}^{\infty} f_\xi(x) \;\; dx \quad = \quad 1$$

1.3.4 Expectation Value

The *expectation value* or *statistical mean* of a stochastic variable ξ, denoted $\mathrm{E}[\xi]$, is defined as follows:

- For a discrete random variable

$$\mathrm{E}[\xi] \quad = \quad \sum_{x \in \Omega_\xi} x \cdot f(x) \quad = \quad \sum_i x_i \cdot f(x_i)$$

- For a continuous random variable

$$\mathrm{E}[\xi] \quad = \quad \int_{-\infty}^{\infty} x \cdot f(x) \;\; dx$$

Expectation values are often denoted μ. The expectation value is the average value of the outcome of the random variable weighted by probability, indicating the center of gravity of ξ.

Example: Continuing the notorious example of the fair die,

$$\mathrm{E}[\zeta] \quad = \quad \sum_{i=1}^{6} i \cdot \frac{1}{6} \quad = \quad \frac{6 \cdot 7}{2} \cdot \frac{1}{6} \quad = \quad \frac{7}{2}$$

Note that this is not a possible value for ζ.

A random variable can be a function of another random variable, i.e., $\eta = g(\xi)$. The expectation value of the random variable η can be calculated from the frequency function f_ξ of ξ:

- For a discrete random variable

$$\mathrm{E}[\eta] \quad = \quad \mathrm{E}[g(\xi)] \quad = \quad \sum_{x \in \Omega_\xi} g(x) \cdot f_\xi(x) \quad = \quad \sum_i g(x_i) \cdot f_\xi(x_i)$$

- For a continuous random variable,

$$\mathrm{E}[\eta] \quad = \quad \mathrm{E}[g(\xi)] \quad = \quad \int_{-\infty}^{\infty} g(x) \cdot f_\xi(x) \;\; dx$$

As we shall soon see, this trick can come in quite handy.

Example: Continuing the long-lasting example of the fair die, if $\eta = \zeta^2$, then

$$\mathrm{E}[\eta] \quad = \quad \mathrm{E}[\zeta^2] \quad = \quad \sum_{i=1}^{6} i^2 \cdot \frac{1}{6} \quad = \quad \frac{91}{6}$$

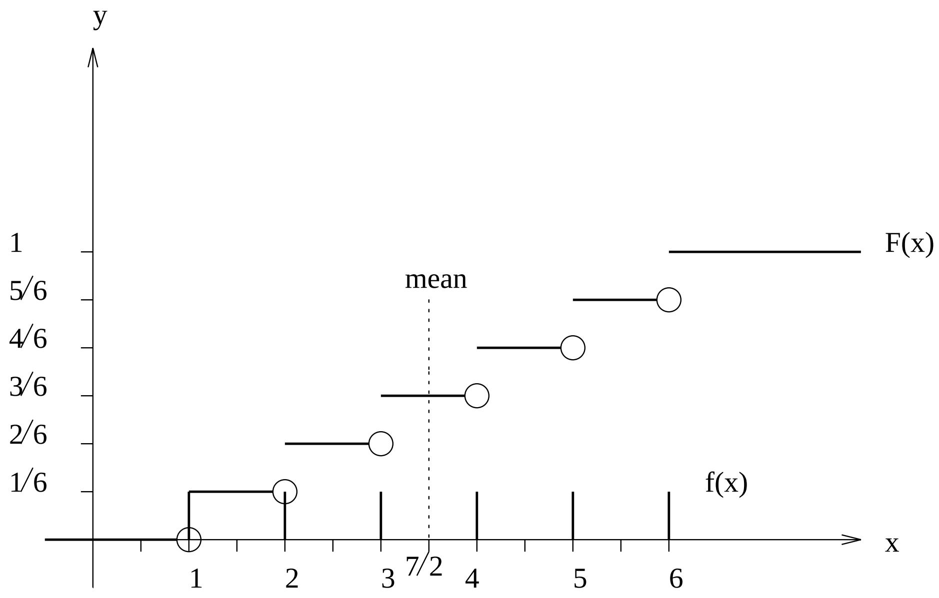

Figure 1.6: Fair die: Expectation value (mean)

1.3.5 Variance

The *variance* of a stochastic variable ξ, denoted $\mathrm{Var}[\xi]$, is defined as the expectation value of $(\xi - \mu)^2$, where $\mu = \mathrm{E}[\xi]$, i.e., $\mathrm{Var}[\xi] = \mathrm{E}[(\xi - \mu)^2]$.

- For a discrete random variable, this means that

$$\mathrm{Var}[\xi] \quad = \quad \sum_{x \in \Omega_\xi} (x - \mu)^2 \cdot f(x)$$

- For a continuous random variable, the corresponding expression is

$$\mathrm{Var}[\xi] \quad = \quad \int_{-\infty}^{\infty} (x - \mu)^2 \cdot f(x) \;\; dx$$

Variances are often denoted σ^2. The variance is a measure of how spread out the probability mass is from the center of gravity μ. σ itself is referred to as the *standard deviation*.

> **Example:** Continuing the inextinguishable example of the fair die,
>
> $$\mathrm{Var}[\zeta] \quad = \quad \sum_{i=1}^{6} (i - \frac{7}{2})^2 \cdot \frac{1}{6} \quad = \quad \frac{35}{12}$$

From the point of view of elementary Mechanics, the variance is simply the quadratic moment about the center of gravity, and Steiner's Theorem from Mechanics can sometimes be useful for calculating $\mathrm{Var}[\xi]$:

$$\begin{aligned} \mathrm{Var}[\xi] &= \mathrm{E}[\xi^2] - (\mathrm{E}[\xi])^2 \text{ or} \\ \sigma^2 &= \mathrm{E}[\xi^2] - \mu^2 \end{aligned} \tag{1.5}$$

Example: Continuing the eternal example of the fair die,

$$\mathrm{Var}[\zeta] \;=\; \mathrm{E}[\zeta^2] - (\mathrm{E}[\zeta])^2 \;=\; \frac{91}{6} - \frac{49}{4} \;=\; \frac{182-147}{12} \;=\; \frac{35}{12}$$

which is the same result as when using the definition directly.

1.3.6 Moments and Moment-Generating Functions

In general, the expectation value of ξ^r, i.e., $\mathrm{E}[\xi^r]$, is known as the *rth moment of* ξ, denoted μ_r. This means that $\mathrm{E}[\xi] = \mu_\xi = \mu_1$ and that the expectation value of the square of ξ, which just figured in Steiner's Theorem, is the second moment of ξ, i.e., $\mathrm{E}[\xi^2] = \mu_2$. If we instead calculate the expectation value of $\mathrm{E}[(\xi - \mu)^r]$, we have the *rth central moment of* ξ *about* μ. We see that the variance is the second central moment of ξ about μ.

The moment-generating function $m(t)$ of ξ is defined as

$$m(t) \;=\; \mathrm{E}[e^{t\xi}] \;=\; \sum_{x\in\Omega_\xi} e^{xt} f_\xi(x)$$

for a discrete random variable and for a continuous random variable as

$$m(t) \;=\; \mathrm{E}[e^{t\xi}] \;=\; \int_{-\infty}^{\infty} e^{xt} f_\xi(x) \; dx$$

This sum or integral doesn't necessarily converge for any value of t, but if it does so for all t around zero, i.e., in $-h < t < h$ for some $h > 0$, we are in business. We then have

$$\frac{d^r}{dt^r} m(t) \;=\; \int_{-\infty}^{\infty} x^r e^{xt} f_\xi(x) \; dx$$

and if we let t tend to zero, we find that

$$\frac{d^r}{dt^r} m(0) \;=\; \int_{-\infty}^{\infty} x^r f_\xi(x) \; dx \;=\; \mathrm{E}[\xi^r] \;=\; \mu_r$$

So in this sense, $m(t)$ generates all moments of ξ. In view of the series expansion of the function e^x discussed in Appendix C.3, this is not altogether too surprising.

1.4 Two-dimensional Stochastic Variables

Let ξ and η be two random variables defined on the same sample space Ω. Then (ξ, η) is a two-dimensional random variable from Ω to $\Omega_{(\xi,\eta)} = \{(\xi(u), \eta(u)) : u \in \Omega\} \subseteq R^2$. Here, $R^2 = R \times R$ is the Cartesian product of the set of real numbers R with itself. This is illustrated in Figure 1.7. We can pull off the same sort of stunts with two-dimensional stochastic variables as with one-dimensional stochastic variables.

1.4.1 Distribution Function

The distribution function of the two-dimensional random variable (ξ, η) is defined as follows:

$$F(x, y) \;=\; P(\xi \le x, \eta \le y) \;=\; P(\{u : \xi(u) \le x, \eta(u) \le y\}) \text{ for all } (x, y) \in R^2$$

This is called the *joint*, or *bivariate*, *distribution* of ξ and η and $P(\xi \in A, \eta \in B)$ is called the *joint probability* of $\xi \in A$ and $\eta \in B$.

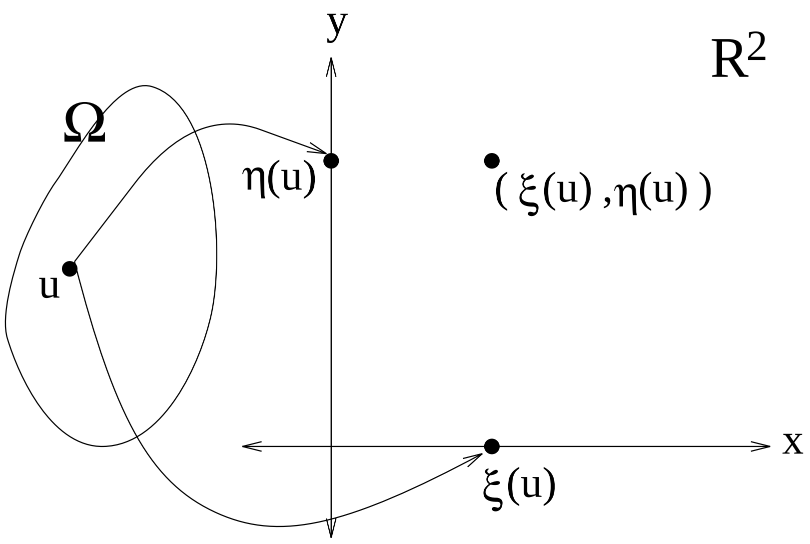

Figure 1.7: A two-dimensional random variable is a function from Ω to R^2.

1.4.2 Frequency Function

There are discrete and continuous versions of two-dimensional random variables, and they all have frequency functions:

- A two-dimensional random variable (ξ, η) is discrete iff $\Omega_{(\xi,\eta)}$ is finite or countable. The frequency function f of (ξ, η) is then defined by

  $$f(x, y) \;=\; P(\xi = x, \eta = y) \;=\; P((\xi, \eta) = (x, y)) \text{ for } (x, y) \in R^2$$

- A two-dimensional random variable is continuous iff the distribution function $F(x, y)$ can be written as

  $$F(x, y) \;=\; \int_{-\infty}^{x} \int_{-\infty}^{y} f(u, v) \; dv \; du$$

  for some non-negative integrable function f. The function f is then called the frequency function of (ξ, η).

Thus, for all two-dimensional stochastic variables we have $f(x, y) \geq 0$. For a continuous variable we also have[7]

$$\begin{aligned} f(x, y) &= \frac{\partial^2}{\partial x \partial y} F(x, y) \quad \text{and} \\ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(u, v) \; dv \; du &= 1. \end{aligned}$$

Furthermore, for $A \subseteq \Omega_{(\xi,\eta)}$

$$\begin{aligned} P((\xi, \eta) \in A) &= \sum_{(x,y) \in A} f(x, y) \text{ if } (\xi, \eta) \text{ is discrete, and} \\ P((\xi, \eta) \in A) &= \int_A f(x, y) \; dx \; dy \text{ if } (\xi, \eta) \text{ is continuous.} \end{aligned}$$

[7] The curly derivative $\frac{\partial}{\partial x}$ is the *partial derivative* w.r.t. x.

Intuitively, a two-dimensional stochastic variable defines a mass distribution in the real plane. For a discrete variable, $f(x,y)$ is the mass at point (x,y); for a continuous variable, $f(x,y)$ is the density at point (x,y).

We can recover the frequency functions of either of the individual variables by summing, or integrating, over the other:
If (ξ,η) is discrete, we have

$$f_\xi(x) = \sum_{y\in\Omega_\eta} f(x,y) \quad \text{and} \quad f_\eta(y) = \sum_{x\in\Omega_\xi} f(x,y)$$

and if (ξ,η) is continuous, we have

$$f_\xi(x) = \int_{-\infty}^{\infty} f(x,y)\ dy \quad \text{and} \quad f_\eta(y) = \int_{-\infty}^{\infty} f(x,y)\ dx$$

In this context, f_ξ is often referred to as the *marginal distribution* of ξ, and similarly for f_η.

1.4.3 Independence

Two stochastic variables ξ and η, defined on the same sample space, are said to be independent iff

$$P(\xi\in A, \eta\in B) = P(\xi\in A)\cdot P(\eta\in B) \tag{1.6}$$

for all subsets A and B of R.

Not very surprisingly, ξ and η being independent is equivalent to

- $F_{(\xi,\eta)}(x,y) = F_\xi(x)\cdot F_\eta(y)$ for all $(x,y)\in R^2$; or
- $f_{(\xi,\eta)}(x,y) = f_\xi(x)\cdot f_\eta(y)$ for all $(x,y)\in R^2$.

1.4.4 Functions of Stochastic Variables

Finally, we look at functions of two random variables, $\phi(u) = g(\xi(u),\eta(u))$. For the expectation value of $g(\xi,\eta)$ we have

$$\begin{aligned}
\mathrm{E}[g(\xi,\eta)] &= \sum_{(x,y)\in\Omega_{(\xi,\eta)}} g(x,y)\cdot f_{(\xi,\eta)}(x,y) \text{ if } (\xi,\eta) \text{ is discrete}\\
\mathrm{E}[g(\xi,\eta)] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)\cdot f_{(\xi,\eta)}(x,y)\ dy\ dx \text{ if } (\xi,\eta) \text{ is continuous.}
\end{aligned}$$

Let ξ,η and $\xi_1,\ldots,\xi_n$ be random variables defined on the same sample space Ω, and let a be a real number. The following formulas are worth memorizing:

$$\begin{aligned}
a)\quad \mathrm{E}[a\cdot\xi] &= a\cdot\mathrm{E}[\xi]\\
b)\quad \mathrm{E}[\xi+\eta] &= \mathrm{E}[\xi]+\mathrm{E}[\eta]\\
c)\quad \mathrm{E}[\xi_1+\ldots+\xi_n] &= \mathrm{E}[\xi_1]+\ldots+\mathrm{E}[\xi_n]\\
d)\quad \mathrm{Var}[a\cdot\xi] &= a^2\cdot\mathrm{Var}[\xi]
\end{aligned}$$

If we in addition to this require that ξ,η and $\xi_1,\ldots,\xi_n$ are independent, we get the following useful formulas:

$$\begin{aligned}
a)\quad \mathrm{E}[\xi\cdot\eta] &= \mathrm{E}[\xi]\cdot\mathrm{E}[\eta]\\
b)\quad \mathrm{E}[\xi_1\cdot\ldots\cdot\xi_n] &= \mathrm{E}[\xi_1]\cdot\ldots\cdot\mathrm{E}[\xi_n]\\
c)\quad \mathrm{Var}[\xi+\eta] &= \mathrm{Var}[\xi]+\mathrm{Var}[\eta]\\
d)\quad \mathrm{Var}[\xi_1+\ldots+\xi_n] &= \mathrm{Var}[\xi_1]+\ldots+\mathrm{Var}[\xi_n]
\end{aligned}$$

We will later have reason to pay attention to sums of stochastic variables. The frequency function $f_{(\xi+\eta)}(x)$ of the sum $\xi+\eta$ can be derived from the joint frequency function $f_{(\xi,\eta)}(x,y)$. In the continuous case we have

$$\begin{aligned} F_{(\xi+\eta)}(x) &= P(\xi+\eta \leq x) = \int_{u+v\leq x} f_{(\xi,\eta)}(u,v)\ du\ dv = \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{x-u} f_{(\xi,\eta)}(u,v)\ \mathrm{dv}\ \mathrm{du} \end{aligned}$$

Now

$$\begin{aligned} f_{(\xi+\eta)}(x) &= \frac{d}{dx}F_{(\xi+\eta)}(x) = \frac{d}{dx}\int_{-\infty}^{\infty}\int_{-\infty}^{x-u} f_{(\xi,\eta)}(u,v)\ dv\ du = \\ &= \int_{-\infty}^{\infty}\frac{d}{dx}\int_{-\infty}^{x-u} f_{(\xi,\eta)}(u,v)\ dv\ du = \int_{-\infty}^{\infty} f_{(\xi,\eta)}(u,x-u)\ du \end{aligned}$$

In particular, if ξ and η are independent, we have $f_{(\xi,\eta)}(x,y) = f_\xi(x)f_\eta(y)$ and thus

$$f_{(\xi+\eta)}(x) = \int_{-\infty}^{\infty} f_\xi(u)f_\eta(x-u)\ du$$

Example: Let $\xi_i : i = 1, 2, \ldots$ be independent random variables with frequency function $\lambda e^{-\lambda x}$. (This particular frequency function will be discussed in Section 1.5.3.) Then

$$\begin{aligned} f_{(\xi_1+\xi_2)}(x) &= \int_{-\infty}^{\infty} f_{\xi_1}(u)f_{\xi_2}(x-u)\ du = \\ &= \int_0^x \lambda e^{-\lambda u}\lambda e^{-\lambda(x-u)}\ du = \int_0^x \lambda^2 e^{-\lambda x}\ du = \\ &= \lambda^2 e^{-\lambda x}\int_0^x du = \lambda^2 x e^{-\lambda x} \end{aligned}$$

If we have the sum of three variables, we can establish that

$$\begin{aligned} f_{(\xi_1+\xi_2+\xi_3)}(x) &= \int_{-\infty}^{\infty} f_{(\xi_1+\xi_2)}(u)f_{\xi_3}(x-u)\ du = \\ &= \int_0^x \lambda^2 u e^{-\lambda u}\lambda e^{-\lambda(x-u)}\ du = \lambda^3 e^{-\lambda x}\int_0^x u\ du = \\ &= \lambda^3\frac{x^2}{2}e^{-\lambda x} \end{aligned}$$

In general we find that $f_{(\xi_1+\cdots+\xi_n)}(x) = \lambda^n\dfrac{x^{n-1}}{(n-1)!}e^{-\lambda x}$

So what is the expectation value of the *maximum* of n independent observations of the same random variable ξ?

$$F_{\max(\xi_1,\ldots,\xi_n)}(x) = P(\max(\xi_1,\ldots,\xi_n)\leq x) = \prod_{i=1}^n P(\xi_i\leq x) = (F_\xi(x))^n$$

Derivation immediately yields:

$$f_{\max(\xi_1,\ldots,\xi_n)}(x) = F'_{\max(\xi_1,\ldots,\xi_n)}(x) = n(F_\xi(x))^{n-1}f_\xi(x)$$

Thus we have

$$\mathrm{E}[\max(\xi_1,\ldots,\xi_n)] = \int_\Omega x f_{\max(\xi_1,\ldots,\xi_n)}(x)dx = \int_\Omega xn(F_\xi(x))^{n-1}f_\xi(x)dx$$

If for example ξ has a uniform distribution on the interval (a,b), i.e., if $\xi \sim U(a,b)$, see Section 1.5.3, then

$$\begin{aligned}
\mathrm{E}[\max(\xi_1,\ldots,\xi_n)] &= \int_\Omega xn(F_\xi(x))^{n-1}f_\xi(x)dx = \int_a^b xn(\frac{x-a}{b-a})^{n-1}\frac{dx}{b-a} = \\
&= \int_a^b n(\frac{x-a}{b-a})^n dx + \int_a^b n(\frac{x-a}{b-a})^{n-1}\frac{a}{b-a}dx = \\
&= \frac{n(b-a)}{n+1}[(\frac{x-a}{b-a})^{n+1}]_a^b + \frac{na}{n}[(\frac{x-a}{b-a})^n]_a^b = \frac{n}{n+1}(b-a)+a
\end{aligned}$$

1.4.5 Higher Dimensions

All this can easily be generalized to several variables.

1.5 Selected Probability Distributions

Some particular types of stochastic variables pop up quite often, and their distributions are given special names. In the following, we will discuss in detail some probability distributions, namely the Binomial and Normal (or Gaussian) distributions, and present more briefly a number of other ones.

1.5.1 Binomial Distribution

A common situation is when we repeat an experiment a number of times and see how many times some particular outcome occurs. For example, assume that we flip a coin a number of times, and count the number of times "heads" comes up.

Let $\Omega = \{u, v\}$ be a basic sample space (where in our example say u is "heads") and let ξ be the number of times u occurs in n independent trials. The stochastic variable ξ is then defined on the sample space

$$\overbrace{\{u,v\} \times \{u,v\} \times \ldots \times \{u,v\}}^{n \text{ factors}}$$

and an outcome in this sample space can be described as a string of length n over the alphabet $\{u, v\}$, e.g., if $n = 5$, $uvvuu$. This is mapped to a real number, in this case 3, since this string contains three us. Thus, the value of ξ is a natural number between 0 and n, i.e. $\Omega_\xi = \{0, 1, \ldots, n\}$, and ξ is a discrete random variable, since this set is finite.

In general, let p be the probability of an event u. If ξ is the number of times u occurs in n independent trials, then ξ has a *Binomial distribution* with parameters n and p. This is often denoted $\xi \sim bin\{n,p\}$. We will now try to establish the frequency function of a stochastic variable with a Binomial distribution. Assume that event u has probability p and event v has probability $q = 1 - p$. Then for example the sequence $uvvuu$ has probability $pqqpp = p^3q^2$. In fact, any sequence with three us and two vs has probability p^3q^2. Now, there are ten different such strings, i.e., there are ten outcomes in $\{u,v\}\times\{u,v\}\times\{u,v\}\times\{u,v\}\times\{u,v\}$ that map to 3, namely $uuuvv$, $uuvuv$, $uuvvu$, $uvuuv$, $uvuvu$, $uvvuu$, $vuuuv$, $vuuvu$, $vuvuu$ and $vvuuu$.

Each of these outcomes has probability p^3q^2. Thus the probability of getting exactly three "heads" in five flips is $10 \cdot (\frac{1}{2})^3 \cdot (\frac{1}{2})^2 = \frac{10}{32} = 0.3125$.

Similarly, we can compute the probability of getting exactly zero, one, two, four or five heads in five independent trials of the coin-flipping experiment. Figure 1.8 shows the resulting probabilities, and Figure 1.9 gives a graphical representation of the distribution.

k	0	1	2	3	4	5	$\sum$
$P(\xi = k)$	$\frac{1}{32}$	$\frac{5}{32}$	$\frac{10}{32}$	$\frac{10}{32}$	$\frac{5}{32}$	$\frac{1}{32}$	1

Figure 1.8: The probability of getting $0, 1, 2, 3, 4, 5$ heads in 5 independent trials.

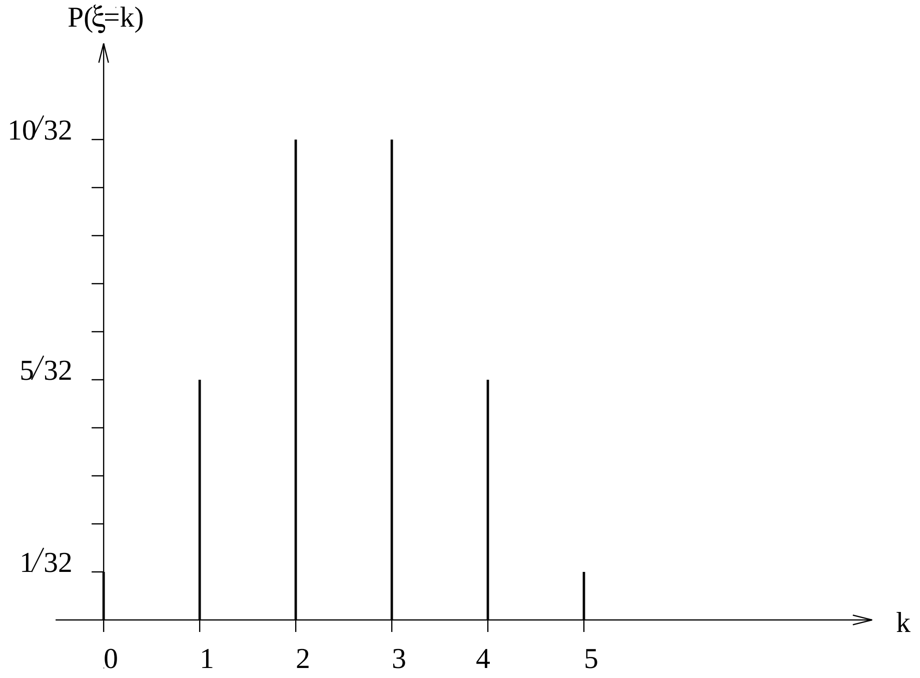

Figure 1.9: The probability distribution when flipping a coin five times.

In general, any string of k us and $n-k$ vs will have probability $p^k q^{n-k}$. Now, there are $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ distinct such strings, as explained in detail in Section 1.2.7. Thus, the probability $P(\xi = k) = f_\xi(k) = \binom{n}{k} p^k q^{n-k}$.

In words, the probability $P(\xi = k)$, defining the frequency function $f_\xi(k)$, is the product of the number of possible ways of selecting k items from n items, $\binom{n}{k}$, and the probability of each such possibility, $p^k q^{n-k}$. This is valid for $k = 0, \ldots, n$.

This gives us the frequency function of a binomially distributed variable:

$$f_\xi(k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k \in \{0, 1, \ldots, n\}, \quad 0 < p < 1 \tag{1.7}$$

This distribution is parameterized by the probability p of the particular outcome u, and n, the number of trials.

It is easy to establish that if a random variable $\xi \sim$ *bin(n,p)*, then:

$$
\begin{aligned}
P(\xi \in \Omega_\xi) &= \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} = 1 \\
F_\xi(x) &= P(\xi \le x) = \sum_{k=0}^{x} f_\xi(k) = \sum_{k=0}^{x} \binom{n}{k} p^k (1-p)^{n-k} \\
\mathrm{E}[\xi] &= \sum_{k=0}^{n} k \cdot \binom{n}{k} p^k (1-p)^{n-k} = np \\
\mathrm{Var}[\xi] &= \sum_{k=0}^{n} (k-np)^2 \cdot \binom{n}{k} p^k (1-p)^{n-k} = np(1-p)
\end{aligned}
$$

The first three equations should come as a surprise to no one. The first equation simply states that the probability of u happening either zero, one, two, ... or n times in n trials is one, i.e., that the probability of the entire sample space is one. The second equation expresses the distribution function $F_\xi(x)$ in terms of the frequency function $f_\xi(x)$. The third equation states that the average number of times u will happen in n trials is $n \cdot p$, where p is the probability of u.

If f_n is the relative frequency of event u, i.e., $f_n = \dfrac{\xi}{n}$, we have:

$$
\begin{aligned}
\mathrm{E}[f_n] &= \mathrm{E}[\frac{\xi}{n}] = \frac{1}{n}\mathrm{E}[\xi] = \frac{1}{n} np = p \\
\mathrm{Var}[f_n] &= \mathrm{Var}[\frac{\xi}{n}] = \frac{1}{n^2}\mathrm{Var}[\xi] = \frac{1}{n^2} np(1-p) = \frac{p(1-p)}{n}
\end{aligned}
$$

1.5.2 Normal Distribution

The *Normal* or *Gaussian distribution*, named after the German mathematician Carl Friedrich Gauss (1777–1855), famous for figuring on the 10 Mark bank notes together with the distribution function below, is probably the most important distribution around. This is due to the Central Limit Theorem presented in Section 1.6. Due to its importance, a number of other probability distributions have been derived from it, e.g., the χ^2 and t distributions, see Section 1.5.3.

We will not beat around the bush, but give the hard-core definition directly: Let ξ be a normally distributed random variable with expectation value μ and variance σ^2, denoted $\xi \sim N(\mu, \sigma)$. Then the frequency function $f_\xi(x)$ is

$$
f_\xi(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in R
$$

Nice figure needed here!

Since

$$
\begin{aligned}
f_\xi(x) &> 0, \quad x \in R, \text{ and} \\
\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\, dx &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi}} e^{-t^2}\, dt = 1
\end{aligned}
$$

this is a proper frequency function[8], and since the distribution function is

$$F_\xi(x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \, dt$$

the associated random variable is continuous.

See Appendix C.3 for a discussion of the number e, the exponential function e^x, and the natural logarithm function $\ln x$. $e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ is simply e^x with $-\dfrac{(x-\mu)^2}{2\sigma^2}$ instead of x.

Next, we will prove that the parameters μ and σ^2 are indeed the expectation value and the variance of the distribution.

Proofs:

$$\begin{aligned} \mathrm{E}[\xi] &= \int_{-\infty}^{\infty} \frac{x}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \, dx = \\ &= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{x-\mu}{\sigma\sqrt{2}} e^{-(\frac{x-\mu}{\sigma\sqrt{2}})^2} \, dx + \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{\mu}{\sigma\sqrt{2}} e^{-(\frac{x-\mu}{\sigma\sqrt{2}})^2} \, dx = \\ &= 0 + \frac{1}{\sqrt{\pi}} \frac{\mu}{\sigma\sqrt{2}} \int_{-\infty}^{\infty} e^{-t^2} \sigma\sqrt{2} \, dt = \frac{\mu}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-t^2} \, dt = \mu \end{aligned}$$

The first equality is obtained by subtracting and adding the same thing. The first integral is zero, since the integrand $g(x)$ is antisymmetric around μ, i.e., $g(\mu-y) = -g(\mu+y)$, which can be taken as the midpoint of the integration interval $(-\infty, \infty)$, and the contributions cancel out. The second integral is rewritten by moving out a bunch of constants from the integral and using the variable substitution $t = \dfrac{x-\mu}{\sigma\sqrt{2}}$; $dt = \dfrac{dx}{\sigma\sqrt{2}}$. □

$$\begin{aligned} \mathrm{Var}[\xi] &= \int_{-\infty}^{\infty} \frac{(x-\mu)^2}{\sigma\sqrt{2\pi}} e^{-(\frac{x-\mu}{\sigma\sqrt{2}})^2} \, dx = \frac{\sigma^2}{\sqrt{\pi}} \int_{-\infty}^{\infty} 2t^2 e^{-t^2} \, dt = \\ &= \frac{\sigma^2}{\sqrt{\pi}} [-te^{-t^2}]_{-\infty}^{\infty} + \frac{\sigma^2}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-t^2} \, dt = 0 + \sigma^2 = \sigma^2 \end{aligned}$$

First, the integral is rewritten using the same variable substitution as in the previous proof. Then the integral is partially integrated, reducing it to an integral with a known value. □

The distribution defined above is parameterized by the expectation value μ and the standard deviation σ. An important special case is when $\mu = 0$ and $\sigma = 1$. This is called the *Standard Normal distribution*. The frequency function is denoted $\phi(x)$ and the distribution function $\Phi(x)$. In this case the random variable is said to be $N(0,1)$. From any normally distributed random variable $\xi \sim N(\mu, \sigma)$ we can derive another random variable $\eta = \dfrac{\xi - \mu}{\sigma} \sim N(0,1)$, i.e., that has a Standard Normal distribution. We will prove this shortly. We can thus recover the Normal distribution function for any mean and standard deviation using a table of the Standard Normal distribution. An example of such a table is given in Section 1.5.4.

If $\xi \sim N(\mu, \sigma), \xi_1 \sim N(\mu_1, \sigma_1)$ and $\xi_2 \sim N(\mu_2, \sigma_2)$ are independent and a is a real constant, then

---

8 $\int_{-\infty}^{\infty} e^{-t^2} \, dt = \sqrt{\pi}$, see Appendix C.3.

1. $a\xi \sim N(a\mu, a\sigma)$,
2. $\xi + a \sim N(\mu + a, \sigma)$ and
3. $\xi_1 + \xi_2 \sim N(\mu_1 + \mu_2, \sqrt{\sigma_1^2 + \sigma_2^2})$.

Proofs: We first note that the expectation values and variances are what they are claimed to be by the formulas at the end of Section 1.4. These results will however fall out from the proofs that the new variables are normally distributed.

1.

$$\begin{aligned}
F_{a\xi}(x) &= P(a\xi \leq x) = P(\xi \leq x/a) = F_\xi(x/a) = \\
&= \int_{-\infty}^{x/a} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}\, dt = \left\{ \begin{array}{ccc} u & = & at \\ du & = & a\,dt \end{array} \right\} = \\
&= \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\frac{u}{a}-\mu)^2}{2\sigma^2}} \frac{du}{a} = \int_{-\infty}^{x} \frac{1}{a\sigma\sqrt{2\pi}} e^{-\frac{(u-a\mu)^2}{2(a\sigma)^2}}\, du = \\
&= \int_{-\infty}^{x} \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{(u-\mu')^2}{2\sigma'^2}}\, du
\end{aligned}$$

with $\mu' = a\mu$ and $\sigma' = a\sigma$. □

2.

$$\begin{aligned}
F_{\xi+a}(x) &= P(\xi + a \leq x) = P(\xi \leq x - a) = F_\xi(x - a) = \\
&= \int_{-\infty}^{x-a} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}\, dt = \left\{ \begin{array}{ccc} u & = & t + a \\ du & = & dt \end{array} \right\} = \\
&= \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(u-(\mu+a))^2}{2\sigma^2}}\, du = \\
&= \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(u-\mu')^2}{2\sigma^2}}\, du
\end{aligned}$$

with $\mu' = \mu + a$. □

3.

$$\begin{aligned}
f_{\xi_1+\xi_2}(x) &= \\
&= \int_{-\infty}^{\infty} \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(u-\mu_1)^2}{2\sigma_1^2}} \cdot \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x-u-\mu_2)^2}{2\sigma_2^2}}\, du = \ldots = \\
&= \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{(x-\mu')^2}{2\sigma'^2}}
\end{aligned}$$

with $\mu' = \mu_1 + \mu_2$ and $\sigma'^2 = \sigma_1^2 + \sigma_2^2$. □

This in turn means that if $\xi_i \sim N(\mu_i, \sigma_i), i = 1, \ldots, n$ are independent and $a_i, i = 0, \ldots, n$ are real constants, then

$$\begin{aligned}
a_0 + \sum_{i=1}^{n} a_i \xi_i &\sim N(\mu', \sigma') \\
\mu' &= a_0 + \sum_{i=1}^{n} a_i \mu_i \\
\sigma'^2 &= \sum_{i=1}^{n} a_i^2 \sigma_i^2
\end{aligned}$$

In words: Any linear combination of independent normally distributed random variables has a Normal distribution. In particular, this proves that if $\xi \sim N(\mu, \sigma)$, then $\frac{\xi - \mu}{\sigma} \sim N(0, 1)$.

1.5.3 Other Distributions

This section lists a number of other common distributions. The Uniform distribution on a finite set and on an interval pop up quite often. The Poisson and Exponential distributions are included mainly for reference, and will only feature occasionally in the following. The χ^2 and t distributions are used for estimating the parameters of normally distributed stochastic variables, as described in Section 1.7.

- If the probability mass of a discrete random variable is evenly divided on a finite set $\{x_i : i = 1, \ldots, M\}$, we have a *Uniform distribution* on a finite set; cf. our example of the fair die. This is obviously a discrete distribution.

$$\begin{aligned} f(x) &= \frac{1}{M}, \qquad x \in \{x_i : i = 1, \ldots, M\} \\ \mathrm{E}[\xi] &= \frac{1}{M} \sum_i x_i \\ \mathrm{Var}[\xi] &= \frac{1}{M} (\sum_i x_i^2 - (\sum_i x_i)^2) \end{aligned}$$

- If the probability mass of a continuous random variable is evenly spread out on a finite interval (a, b), we have a *Uniform distribution* on an interval, denoted $\xi \sim U(a, b)$. This is obviously a continuous distribution.

$$\begin{aligned} f(x) &= \frac{1}{b - a} \qquad x \in (a, b) \\ \mathrm{E}[\xi] &= \frac{a + b}{2} \\ \mathrm{Var}[\xi] &= \frac{(b - a)^2}{12} \end{aligned}$$

- The *Poisson distribution* models a stream of impulses where the impulse probability is differentially proportional to the interval length, but where impulses in disjoint intervals are independent, and where we have translational invariance. [9] ξ is the number of impulses in an interval of length t where c is the (differential) proportionality factor, and $\lambda = ct$. Thus, this is a discrete distribution.

$$\begin{aligned} f_\xi(x) &= \frac{\lambda^x}{x!} e^{-\lambda x} \qquad x \in N,\ \lambda > 0 \\ \mathrm{E}[\xi] &= \lambda \\ \mathrm{Var}[\xi] &= \lambda \end{aligned}$$

- The *Exponential distribution* models first order decay, e.g., radioactive decay. Here, ξ is the life span of an atom. Thus, this is a continuous

[9] insert what translat. invar. means

distribution. Incidentally, the time between two impulses in a Poisson process is exponentially distributed.

$$\begin{aligned} f_\xi(x) &= \lambda e^{-\lambda x} \qquad x \in R^+,\ \lambda > 0 \\ \mathrm{E}[\xi] &= \frac{1}{\lambda} \\ \mathrm{Var}[\xi] &= \frac{1}{\lambda^2} \end{aligned}$$

This distribution, parameterized by λ, is denoted $\xi \sim exp(\lambda)$.

- The discrete counterpart of the Exponential distribution is the *Geometric* or *Pascal distribution* over N^+:

$$\begin{aligned} p_\xi(n) &= p(1-p)^{n-1} \qquad n \in N^+,\ 0 < p < 1 \\ \mathrm{E}[\xi] &= \frac{1}{p} \\ \mathrm{Var}[\xi] &= \frac{1}{p^2} \end{aligned}$$

This distribution is parameterized by p, the probability of ξ taking the value 1. Since this is a probability, we require $0 < p < 1$.

- Let $\xi_i \sim N(0,1), i = 1, \ldots, r$ be independent. Then $\eta = \sum_{i=1}^{r} \xi_i^2$ has a χ^2 *distribution*, read out "chi-square distribution", with r degrees of freedom, denoted $\eta \sim \chi^2(r)$. This is a continuous distribution.

$$\begin{aligned} f_\eta(x) &= \frac{1}{2^{\frac{r}{2}}\Gamma(\frac{r}{2})} x^{\frac{r}{2}-1} e^{-\frac{x}{2}} \qquad x \in R^+ \\ \mathrm{E}[\eta] &= r \\ \mathrm{Var}[\eta] &= 2r \end{aligned}$$

where $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}\,dt,\ \ x \in R^+$. Forget about the frequency function and just remember that a χ^2-distributed variable with r degrees of freedom has the same distribution as the sum of the squares of r independent variables with a Standard Normal distribution.

- Let $\xi \sim N(0,1)$ and $\eta \sim \chi^2(r)$ be independent. Then $\zeta = \dfrac{\xi}{\sqrt{\eta/r}}$ has a *t distribution* with r degrees of freedom, denoted $\zeta \sim t(r)$. This is a continuous distribution.

$$\begin{aligned} \mathrm{E}[\zeta] &= 0 \qquad \text{for } r > 1 \\ \mathrm{Var}[\zeta] &= \frac{r}{r-2} \qquad \text{for } r > 2 \end{aligned}$$

You don't want to know the frequency function. For large rs, ζ is approximately $\sim N(0,1)$.

1.5.4 Distribution Tables

It is cumbersome to use the frequency functions of the above distributions directly for calculations, so tables are compiled that list the solution to the equation $F(x_p) = p$ for a number of interesting probabilities p. x_p is called the $p \cdot 100\%$ fractile, which means that $p \cdot 100\%$ of the probability mass is associated with values of x that are less than x_p. This is done for a variety of different distribution functions $F(x)$, some of which will be listed in a future appendix.

The following is a short example of a table showing the value of the distribution function $\Phi(x)$ of the Standard Normal distribution $N(0,1)$ for some positive values of x. For negative values, we can utilize the fact that $\phi(x) = \frac{d}{dx}\Phi(x)$ is symmetric around zero, i.e., that $\phi(-x) = \phi(x)$. This means that $\Phi(-x) = 1 - \Phi(x)$ since

$$\begin{aligned} \Phi(-x) &= \int_{-\infty}^{-x} \phi(t)\ dt = 1 - \int_{-x}^{\infty} \phi(t)\ dt = \\ &= 1 - \int_{-\infty}^{x} \phi(-t)\ dt = 1 - \int_{-\infty}^{x} \phi(t)\ dt = 1 - \Phi(x) \end{aligned}$$

x	$\Phi(x)$	x	$\Phi(x)$	x	$\Phi(x)$
0.00	0.500	1.10	0.864	2.10	0.977
0.10	0.540	1.20	0.885	2.20	0.986
⋮		⋮		⋮	
0.90	0.816	1.90	0.971	2.90	0.9981
1.00	0.841	2.00	0.977	3.00	0.9987

Of special importance are the value of x when $\Phi(x)$ is 0.90, 0.95 and 0.975:

x	$\Phi(x)$
1.282	0.900
1.645	0.950
1.960	0.975

These are often used to create confidence intervals, see Section 1.7.5.

1.6 Some Theoretical Results

In this section we will discuss some important theoretical results, namely Chebyshev's Inequality, Bernoulli's Theorem and the Central Limit Theorem.

We will start with *Chebyshev's Inequality*, which states that for any random variable ξ with expectation value μ and variance σ^2 and any $t > 0$ we have

$$P(|\xi - \mu| \geq t) \leq \frac{\sigma^2}{t^2}$$

The probability mass $P(|\xi - \mu| \geq t)$ is depicted in Figure 1.10

Proof: Assuming for convenience that ξ is discrete, we have

$$\sigma^2 = \sum_{x \in \Omega_\xi} (x-\mu)^2 \cdot f(x) \geq \sum_{|x-\mu| \geq t} (x-\mu)^2 \cdot f(x)$$

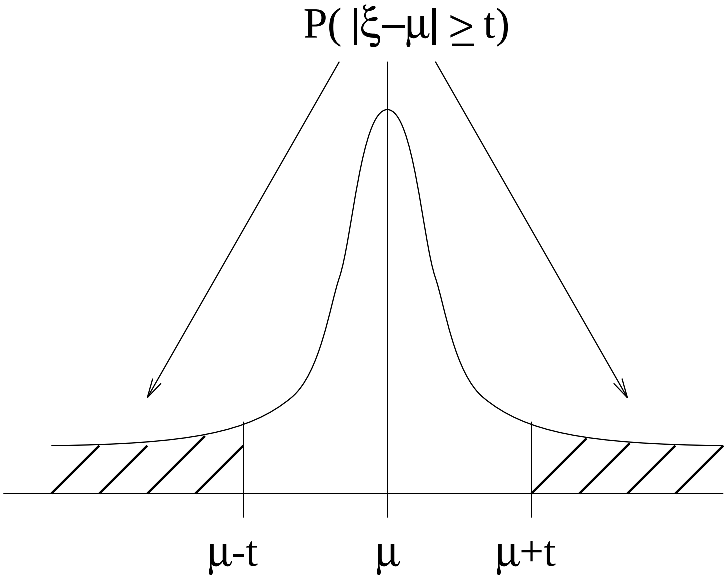

Figure 1.10: The probability mass $P(|\xi - \mu| \geq t)$.

$$\geq \; t^2 \sum_{|x-\mu|\geq t} f(x) \;\; = \;\; t^2 \, P(|x-\mu| \geq t)$$

The first inequality comes from summing a positive function over a subset of Ω_ξ, and the second since $(x-\mu)^2 \geq t^2$ on this subset. The final equality follows from the definition of the frequency function $f(x)$. $\Box$

Equipped with this tool, we will now continue by discussing more formally the stability of the relative frequence presented in the introduction. Let $\xi \sim bin(n,p)$ and let f_n be $\frac{\xi}{n}$. Recalling that $\mathrm{E}[f_n] = p$ and $\mathrm{Var}[f_n] = \frac{p(1-p)}{n}$, and using Chebyshev's Inequality with $t = \epsilon > 0$, we can establish that

$$P(|f_n - p| \geq \epsilon) \;\; \leq \;\; \frac{p(1-p)}{n\epsilon^2}$$

We will now use a regular trick in mathematical analysis: Having trapped some quantity, in our case $P(|f_n - p| \geq \epsilon)$, between 0 and something that approaches 0 when some parameter, in this case n, approaches ∞, we can conclude that the trapped quantity also approaches 0 as the parameter approaches ∞. This means that

$$\lim_{n\to\infty} P(|f_n - p| \geq \epsilon) \;\; = \;\; 0 \text{ for all } \epsilon > 0$$

We can thus establish that the probability of the relative frequence deviating the slightest (i.e., being more than ϵ away) from the probability of the event tends to zero as the number of trials tends to infinity. This is known as *Bernoulli's Theorem*.

The practical importance of the normal distribution is mainly due to the *Central Limit Theorem*, which states that if $\xi_1, \xi_2, \ldots$ is a sequence of independent and identically distributed random variables with expectation value μ and variance σ^2, then

$$\lim_{n\to\infty} P(\frac{\sum_{i=1}^{n}\xi_i - n\mu}{\sigma\sqrt{n}} \le x) \quad = \quad \Phi(x)$$

This means that $\bar{\xi}_n = \frac{1}{n}\sum_{i=1}^{n}\xi_i$ is approximately $\sim N(\mu, \sigma/\sqrt{n})$ for large ns.[10] Or to put it more informally: If you take a very large random sample from a distribution, then the distribution of the sum is approximately normal. We will refrain from proving this theorem.

The conditions that ξ_i be independent and identically distributed can be relaxed somewhat, the important thing is that they are not to strongly dependent, that each makes some small contribution to the sum, and that μ and σ^2 exist. Physical measurements are often disturbed by small, reasonably independent fluctuations in temperature, humidity, electric fields, concentrations of various substances, etc., or random shocks, vibrations and the like, and can therefore in view of the Central Limit Theorem often be assumed to be normally distributed.

In speech recognition there is a strong evidence from corpora that phonetic features are best described acoustically by normal distributions. Thus phonemes are modeled by multidimensional normal distributions.[11]

ξ_i can be either discrete or continuous. In the former case, using the normal distribution to approximate the original one is called a *continuum approximation*. In particular, if $\xi \sim bin(n,p)$, then ξ is approximately $\sim N(np, \sqrt{np(1-p)})$. This is known as the *Moivre-Laplace Limit Theorem*:

$$\lim_{n\to\infty} P(\frac{\xi - np}{\sqrt{np(1-p)}} \le x) \quad = \quad \Phi(x)$$

Since it is cumbersome to work out the binomial coefficients $\binom{n}{k}$ for large values of n and k, this is quite useful. Even more useful is replacing p with the relative frequency f_n almost everywhere, since

$$\frac{(f_n - p)\sqrt{n}}{\sqrt{f_n(1-f_n)}}$$

is in fact also approximately $\sim N(0,1)$.

1.7 Estimation

Statistical estimation is usually reduced to the problem of estimating some parameter θ of a probability distribution, where the rest of the distribution is already known. Also, θ is known to belong to some parameter space T.

1.7.1 Random Samples

Let $\xi_i, i = 1, \ldots, n$ be independent stochastic variables with the same distribution as the variable ξ. Then $(\xi_1, \ldots, \xi_n)$ is said to be a *random sample*

[10] $\bar{\xi}_n$ will be discussed in more detail in the following section.

[11] Cf. [Dalsgaard 1992] for an approach based on neural nets, and [Young 1993] for a HMM-based approach.

of ξ. The set of observations of the outcome of the variables in some trial, $(x_1, \ldots, x_n)$, is called a *statistical material*.

From the random sample, we can create new random variables that are functions of the random sample, which we will call *sample variables*. In particular, the two variables $\bar{\xi}_n$ and s_n^2 are called the *sample mean* and *sample variance* respectively:

$$\begin{aligned} \bar{\xi}_n &= \frac{1}{n}\sum_{i=1}^{n}\xi_i \\ s_n^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(\xi_i - \bar{\xi}_n)^2 \end{aligned}$$

The reason for dividing by $n-1$ rather than n in the latter case will become apparent in Section 1.7.5.

We will here digress slightly to prove the *Law of Large Numbers*:

$$\lim_{n\to\infty} P(|\bar{\xi}_n - \mu| \geq \epsilon) \;=\; 0 \text{ for all } \epsilon > 0.$$

Here it is essential that $\mathrm{E}[\xi] = \mu$ and $\mathrm{Var}[\xi] = \sigma^2$ both exist.

> **Proof:** The proof is very similar to the proof of Bernoulli's Theorem. By Chebyshev's Inequality we have that
>
> $$P(|\bar{\xi}_n - \mu| \geq \epsilon) \;\leq\; \frac{\sigma^2}{n\epsilon^2} \text{ for all } \epsilon > 0.$$
>
> By letting n tend to infinity, we can establish the claim.
>
> To put it in more words: The Law of Large Numbers states that the sample mean $\bar{\xi}_n$ of a random sample $(\xi_1, \ldots, \xi_n)$ converges in probability to the mean μ of the distribution from which the random sample is taken, as n increases. □

This is good news! It tells us that we can estimate μ with any accuracy we wish by simply making enough observations.

1.7.2 Estimators

A sample variable $g(\xi_1, \ldots, \xi_n)$ that is used to estimate some real parameter γ is called an *estimator*. Here γ will in general depend on the unknown parameter θ of the frequency function, and should really be written $\gamma(\theta)$. The value assigned to γ by the statistical material is called an *estimate*. The distinction between estimators and estimates is rather convenient to make.

We want the estimator $g(\xi_1, \ldots, \xi_n)$ to in some way be related to the parameter γ it is supposed to estimate. It is said to be *unbiased* iff

$$\mathrm{E}[g(\xi_1, \ldots, \xi_n)] \;=\; \gamma$$

Also, we desire that the larger the random sample, the better the estimator. Let g_n be real-valued functions from R^n to R for n = 1,2,... The sequences $\{g(\xi_1, \ldots, \xi_n)\}_{n=1}^{\infty}$ of estimators is said to be *consistent* iff

$$\lim_{n\to\infty} P(|g(\xi_1, \ldots, \xi_n) - \gamma| \geq \epsilon) \;=\; 0$$

for each $\epsilon > 0$.

We have that the sample mean $\bar{\xi}_n$ is a unbiased estimator of the expectation value μ since

$$\mathrm{E}[\bar{\xi}_n] \;=\; \mathrm{E}[\frac{1}{n}\sum_{i=1}^{n}\xi_i] \;=\; \frac{1}{n}\sum_{i=1}^{n}\mathrm{E}[\xi_i] \;=\; \frac{1}{n}\sum_{i=1}^{n}\mu \;=\; \mu$$

and that $\bar{\xi}_1, \ldots, \bar{\xi}_n$ form a consistent sequence of estimators in view of the Law of Large Numbers. One can further prove that the sample variance s_n^2 is an unbiased estimator of the variance σ^2, and that $s_1^2, s_2^2, \ldots$ is a consistent sequence of estimators of σ^2 if the fourth central moment of ξ about μ exists, i.e., if $\mathrm{E}[(\xi-\mu)^4] < \infty$.

If for a sequence of unbiased estimators $\eta_n = g(\xi_1, \ldots, \xi_n)$ we have that $\lim_{n\to\infty} \mathrm{Var}[\eta_n] = 0$, then η_n is a consistent sequence of estimators.

Proof: By Chebyshev's Inequality we have that

$$P(|\eta_n - \gamma| \geq \epsilon) \;\leq\; \frac{1}{\epsilon^2}\mathrm{Var}[\eta_n]$$

The claim follows immediately by letting n tend to infinity. $\Box$

If, in general, we have two unbiased estimators η_1 and η_2, then η_1 is said to be *more efficient* than η_2 if $\mathrm{Var}[\eta_1] < \mathrm{Var}[\eta_2]$.

1.7.3 Maximum-Likelihood Estimators

So how do we find good estimators? One method is to use *Maximum-Likelihood Estimators*, MLEs. This method is based on a simple, but important idea:

> *Choose the alternative that maximizes the probability of the observed outcome.*

To this end, we proceed as follows: Let $(\xi_1, \ldots, \xi_n)$ be a random sample of a stochastic variable ξ with frequency function $f_\xi(x)$, and let $(x_1, \ldots, x_n)$ be the corresponding statistical material. We then define the *likelihood function* $L(x_1, \ldots, x_n, \theta)$ as the joint frequency function of $\xi_1, \ldots, \xi_n$

$$L(x_1, \ldots, x_n, \theta) \;=\; f_{(\xi_1,\ldots,\xi_n),\theta}(x_1, \ldots, x_n) \;=\; \prod_{i=1}^{n} f_\theta(x_i)$$

Since the sample is assumed to be random, the joint frequency function is simply the product of the individual frequency functions, which in turn are the same function, since this is a sample. For a discrete variable, this is the probability of the outcome $(x_1, \ldots, x_n)$; for a continuous variable, this is the probability density in the point $(x_1, \ldots, x_n)$.

We will simply choose the value $\hat{\theta}$ that maximizes likelihood function L:

$$\max_\theta \;\; L(x_1, \ldots, x_n, \theta)$$

Seeing that the logarithm function ln (see Appendix C.3 for a discussion of this function) is monotonically increasing on R^+, we can equally well maximize $\ln L$:

$$\max_\theta \;\; \ln L(x_1, \ldots, x_n, \theta) \;=\; \max_\theta \;\; \sum_{i=1}^{n} \ln f_\theta(x_i)$$

This is convenient as L is a product, which means that $\ln L$ is a sum.

The most important example of a maximum-likelihood estimator is the relative frequency f_n of an outcome, which is the maximum-likelihood estimator of the probability p of that outcome.

A regular trick for finding the maximum is to inspect points where $\frac{\partial}{\partial\theta} \ln L = 0$. If we for example happen to know that L is a convex function of θ, this is in fact equivalent to a global maximum for interior points of the parameter space T in which θ is allowed to vary.

Nice figure needed here!

Example: Let $(\xi_1, \ldots, \xi_n)$ be a random sample of $\xi \sim exp(\lambda)$, see Section 1.5.3 for the definition of an exponential distribution, and let $(x_1, \ldots, x_n)$ be the corresponding statistical material. Let θ, the unknown parameter of the exponential distribution, and γ the quantity that we wish to estimate, both be λ. Then

$$L(x_1, \ldots, x_n, \lambda) = \prod_{i=1}^{n} f_{\xi,\lambda}(x_i) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i} = \lambda^n e^{-\lambda \sum_{i=1}^{n} x_i}$$

and

$$\ln L(x_1, \ldots, x_n, \lambda) = n \ln \lambda - \lambda \sum_{i=1}^{n} x_i$$

Thus

$$\frac{\partial}{\partial\lambda} \ln L(x_1, \ldots, x_n, \lambda) = \frac{n}{\lambda} - \sum_{i=1}^{n} x_i$$

which is 0 iff $\frac{n}{\lambda} = \sum_{i=1}^{n} x_i$. Let us introduce

$$\bar{x}_n = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Then the derivative is zero iff $\lambda = \frac{1}{\bar{x}_n}$.

Since L is a convex function of λ, this is a global maximum. Thus the maximum-likelihood estimator of the λ parameter of the exponential distribution is $1/\bar{\xi}_n$.

Maximum-likelihood estimators can be used to estimate several parameters simultaneously. For example, if we want maximum-likelihood estimators of both μ and σ^2 for a normally distributed variable ξ, we can construct $L(x_1, \ldots, x_n, \mu, \sigma)$ as usual (using the frequency function for the normal distribution), take the logarithm, and find the values $\hat{\mu}$ and $\hat{\sigma^2}$ for which simultaneously $\frac{\partial}{\partial\mu} \ln L(x_1, \ldots, x_n, \mu, \sigma)$ and $\frac{\partial}{\partial\sigma} \ln L(x_1, \ldots, x_n, \mu, \sigma)$ are zero. This turns out to be when

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i = \bar{x}_n \quad \text{and} \quad \hat{\sigma^2} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x}_n)^2$$

This corresponds to the estimators

$$\frac{1}{n} \sum_{i=1}^{n} \xi_i \quad \text{and} \quad \frac{1}{n} \sum_{i=1}^{n} (\xi_i - \bar{\xi}_n)^2$$

The first one is simply the sample mean $\bar{\xi}_n$, which as we recall is an unbiased estimator of μ. However, recalling that the sample variance s_n^2 is an unbiased estimator of σ^2, i.e., that

$$s_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\xi_i - \bar{\xi}_n)^2 \text{ and } \mathrm{E}[s_n^2] = \sigma^2$$

we have that

$$\mathrm{E}[\frac{1}{n}\sum_{i=1}^{n}(\xi_i - \bar{\xi}_n)^2] = \mathrm{E}[\frac{n-1}{n}\frac{1}{n-1}\sum_{i=1}^{n}(\xi_i - \bar{\xi}_n)^2] = \frac{n-1}{n}\mathrm{E}[s_n^2] = \frac{n-1}{n}\sigma^2$$

This means that the maximum-likelihood estimator for σ^2 in the normal distribution, $\frac{1}{n}\sum_{i=1}^{n}(\xi_i - \bar{\xi}_n)^2$, is not quite unbiased.

This is an example of the fact that maximum-likelihood estimators are not necessarily unbiased. If we return to our previous example, we find that the expectation value of the maximum-likelihood estimator $1/\bar{\xi}_n$ of the λ parameter of the exponential distribution does not equal λ. In fact, it equals $\frac{n}{n-1}\lambda$, so it is also slightly off the mark:

$$\begin{aligned}
\mathrm{E}[1/\bar{\xi}_n] &= \int_{-\infty}^{\infty}\frac{n}{x}f_{(\xi_1+\ldots+\xi_n)}(x)\;dx \;=\; \int_{0}^{\infty}\frac{n}{x}\lambda^n\frac{x^{n-1}}{(n-1)!}e^{-\lambda x}\;dx \;= \\
&= \frac{n}{n-1}\lambda\int_{0}^{\infty}\lambda^{n-1}\frac{x^{n-2}}{(n-2)!}e^{-\lambda x}\;dx \;=\; \{\text{Partial integration } n-1 \text{ times}\} \\
&= \ldots = \frac{n}{n-1}\lambda[-e^{-\lambda x}]_0^{\infty} \;=\; \frac{n}{n-1}\lambda
\end{aligned}$$

In both cases, though, we can construct an unbiased estimator from the maximum-likelihood estimator by multiplying with a scaling factor. This is actually quite often the case.

1.7.4 Sufficient Statistic

A sample variable $g(\xi_1,\ldots,\xi_n)$ is said to be a *statistic* if it does not depend on any unknown parameter. For example, if $\xi \sim N(\mu,\sigma)$, where μ is not known, then the sample mean $\bar{\xi}_n$ is a statistic, whereas $\bar{\xi}_n - \mu$ isn't, since the latter depends on the unknown parameter μ.

A statistic $g(\xi_1,\ldots,\xi_n)$ is a *sufficient statistic* for the unknown parameter(s) θ iff the conditional distribution given $g(\xi_1,\ldots,\xi_n) = s$ does not depend on θ. Intuitively, this means that as soon as we've fixed the value of the sample variable (in this case to s), the distribution of the sample does no longer depend on θ and there is thus no further information to be extracted about θ from the sample.

A practically more useful definition of sufficient statistic is that the frequency function of the random sample, which is identical to the likelihood function of the previous section, can be written as a product of two functions h_1 and h_2, one that depends only on the statistical material, and one that depends on θ, but only on the statistical material through the function g of the sample variable:

$$\begin{aligned}
L(x_1,\ldots,x_n,\theta) &= \prod_{i=1}^{n} f_\theta(x_i) \;= \\
&= f_{(\xi_1,\ldots,\xi_n)}(x_1,\ldots,x_n,\theta) \;=\; h_1(g(x_1,\ldots,x_n),\theta)\cdot h_2(x_1,\ldots,x_n)
\end{aligned}$$

The entire point is that the equation $g(x_1, \ldots, x_n) = s$ partitions the range of the random sample (i.e., Ω_ξ^n) into subsets, one for each value of s, that can be viewed as equivalent w.r.t. the parameter θ, thus condensing the statistical material without losing information about θ. Thus, we have revealed the innermost secret of sample variables: They condense data.

For example, if n_u is the number of times some outcome u occurs in n trials, then n_u is a sufficient statistic for the purpose of determining the unknown parameter p, the probability of the outcome. Thus no information about p is contained in the particular sequence of outcomes of the trials, only in the number of times u occurred. So whereas the size of the range of the random sample is 2^n, the size of the range of the sample variable n_u is $n+1$, which nonetheless contains the same amount of information about p — a tremendous reduction!

There does not necessarily exist a sufficient statistic, but there is always a set of jointly sufficient statistics, which is a set of statistics:

> **Definition:** Let $(\xi_1, \ldots, \xi_n)$ be a random sample and let θ be the unknown parameter(s). The sample variables $g_1(\xi_1, \ldots, \xi_n)$, $\ldots, g_r(\xi_1, \ldots, \xi_n)$ are said to constitute a set of jointly sufficient statistics iff they do not depend on any unknown parameters and the conditional distribution given $g_k(\xi_1, \ldots, \xi_n) = s_k$, for $k = 1, \ldots, r$, does not depend on θ.

Why there always is one? Set $g_k(\xi_1, \ldots, \xi_n) = \xi_k$ for $k = 1, \ldots, n$. Although this won't condense the data, the random sample itself is trivially jointly sufficient, since once we've specified the outcome, its probability is 1, and thus does not depend on θ.

A set of jointly sufficient statistics is said to be *minimal sufficient* if no other set of sufficient statistics condenses the data more.

> **Example:** The sample mean $\bar{\xi}_n$ and the sample variance s_n^2,
>
> $$\begin{aligned} \bar{\xi}_n &= \frac{1}{n}\sum_{i=1}^{n} \xi_i \\ s_n^2 &= \frac{1}{n-1}\sum_{i=1}^{n} (\xi_i - \bar{\xi}_n)^2 \end{aligned}$$
>
> constitute a minimal sufficient statistics for the parameters $\theta = \langle \mu, \sigma \rangle$ of the normal distribution. This means that we can't condense the statistical material more without losing information about μ or σ.

1.7.5 Confidence Intervals

We have until now only spoken of *point estimates* of parameters. With Chebyshev's Inequality, where we are in essence estimating the probability of a random variable taking a value that is "far out", cf. Figure 1.10, we are nibbling at the idea of interval estimates. The basic idea with a confidence interval is that we can give a lower or an upper bound, or both, for the parameter γ, and we can indicate how confident we are that γ is in fact contained in this interval.

Let $\eta_1 = g_1(\xi_1, \ldots, \xi_n)$ and $\eta_2 = g_2(\xi_1, \ldots, \xi_n)$ be two sample variables such that:

$$P(\eta_1 < \gamma < \eta_2) \;=\; p$$

We then call (η_1, η_2) a *confidence interval* with *confidence degree* p. If $P(\eta_1 > \gamma) = P(\gamma > \eta_2)$, the interval is said to be *symmetric*. This means that we have cut off equally much of the probability mass on the left side as on the right side.

> **Yoga time:** The parameter γ is a (real) number, and associated with no uncertainty or stochastic behavior. γ does not take a value inside the interval with probability p. It is either inside the interval or outside. If it is inside, our method for determining the bounds on it worked. If it isn't, we goofed. The confidence degree p only tells us how often we will in average succeed in establishing a correct interval using this method. That's all. For example, if we conduct measurements on c_0, the speed of light in vacuum, and establish that with 99 % percent confidence degree $299,792,457$ m/s $< c_0 < 299,792,459$ m/s, this is our guess at what c_0 is. In most contemporary theories of Physics, c_0 has some fixed value, and is not subject to random variation.

In the rest of this section, we will assume that $\xi \sim N(\mu, \sigma)$ and construct confidence intervals for μ and σ.

Estimating μ with known σ

As established in Section 1.5.2, $\sum_{i=1}^{n} \xi_i \sim N(n\mu, \sqrt{n}\sigma)$. Thus $\bar{\xi}_n = \frac{1}{n}\sum_{i=1}^{n} \xi_i \sim N(\mu, \sigma/\sqrt{n})$, or $\frac{\bar{\xi}_n - \mu}{\sigma}\sqrt{n} \sim N(0,1)$. Utilizing this fact we have:

$$P(x_1 \;<\; \frac{\bar{\xi}_n - \mu}{\sigma}\sqrt{n} \;<\; x_2) \;\;=\;\; \Phi(x_2) - \Phi(x_1)$$

Rearranging this gives us

$$P(\bar{\xi}_n - x_2\frac{\sigma}{\sqrt{n}} \;<\; \mu \;<\; \bar{\xi}_n - x_1\frac{\sigma}{\sqrt{n}}) \;\;=\;\; \Phi(x_2) - \Phi(x_1)$$

This is usually written

$$\bar{\xi}_n - x_2\frac{\sigma}{\sqrt{n}} \;\;<\;\; \mu \;\;<\;\; \bar{\xi}_n - x_1\frac{\sigma}{\sqrt{n}} \qquad\qquad (p)$$

where $p = \Phi(x_2) - \Phi(x_1)$. In particular, if we want a symmetric interval with confidence degree p, we choose $x = x_2 = -x_1$ such that $\Phi(x) = 1 - (1-p)/2 = (1+p)/2$, resulting in

$$\bar{\xi}_n - x\frac{\sigma}{\sqrt{n}} \;\;<\;\; \mu \;\;<\;\; \bar{\xi}_n + x\frac{\sigma}{\sqrt{n}} \qquad\qquad (p)$$

Estimating σ^2 with known μ

Let $v_n^2 = \frac{1}{n}\sum_{i=1}^{n}(\xi_i - \mu)^2$. Then $\frac{nv_n^2}{\sigma^2} = \sum_{i=1}^{n}(\frac{\xi_i - \mu}{\sigma})^2$ is the sum of n squares of independent normally distributed random variables with expectation value 0 and variance 1, i.e., $\frac{nv_n^2}{\sigma^2} = \sum u_i^2$, where $u_i = \frac{\xi_i - \mu}{\sigma} \sim N(0,1)$. According to Section 1.5.3, such a random variable has a χ^2 distribution with n degrees of freedom, i.e., we have that $\frac{nv_n^2}{\sigma^2} \sim \chi^2(n)$, and thus

$$P(x_1 < \frac{nv_n^2}{\sigma^2} < x_2) = F(x_2) - F(x_1) = p_2 - p_1$$

where $F(x)$ is the distribution function of the χ^2 distribution with n degrees of freedom, and $F(x_1) = p_1; F(x_2) = p_2$. This gives the following confidence interval with $p = p_2 - p_1$:

$$\frac{nv_n^2}{x_2} \quad < \quad \sigma^2 \quad < \quad \frac{nv_n^2}{x_1} \qquad\qquad (p)$$

Estimating σ^2 with unknown μ

If we do not know μ, we instead use the sample variance $s_n^2 = \dfrac{1}{n-1}\sum_{i=1}^{n}(\xi_i - \bar{\xi}_n)^2$ to construct $\dfrac{(n-1)s_n^2}{\sigma^2} = \sum_{i=1}^{n}(\dfrac{\xi_i - \bar{\xi}_n}{\sigma})^2$, which can also be written as the sum of n squares of normally distributed random variables with expectation value 0 and variance 1, i.e., $\dfrac{(n-1)s_n^2}{\sigma^2} = \sum u_i^2$, where $u_i = \dfrac{\xi_i - \bar{\xi}_n}{\sigma} \sim N(0,1)$. Unfortunately, these u_is are not independent since, for example, they sum to zero:

$$\sum_{i=1}^{n} u_i = \sum_{i=1}^{n} \frac{\xi_i - \bar{\xi}_n}{\sigma} = \sum_{i=1}^{n} \frac{\xi_i}{\sigma} - \sum_{i=1}^{n} \frac{\bar{\xi}_n}{\sigma} = n\frac{\bar{\xi}_n}{\sigma} - n\frac{\bar{\xi}_n}{\sigma} = 0$$

One can however prove that $\sum_{i=1}^{n} \dfrac{\xi_i - \bar{\xi}_n}{\sigma}$ can be rewritten as $\sum_{i=1}^{n-1} w_i^2$ where w_i are indeed independent and $\sim N(0,1)$, and we then have that $\dfrac{(n-1)s_n^2}{\sigma^2} \sim \chi^2(n-1)$.

We can thus establish that

$$P(x_1 < \frac{(n-1)s_n^2}{\sigma^2} < x_2) = F(x_2) - F(x_1) = p_2 - p_1$$

where $F(x)$ is the distribution function of the χ^2 distribution with $n-1$ degrees of freedom, and $F(x_1) = p_1; F(x_2) = p_2$. This gives the following confidence interval with $p = p_2 - p_1$:

$$\frac{(n-1)s_n^2}{x_2} \quad < \quad \sigma^2 \quad < \quad \frac{(n-1)s_n^2}{x_1} \qquad\qquad (p)$$

Estimating μ with unknown σ

Of course, it is not generally the case that we know σ but not μ. When both are unknown, we can instead use the fact that $\dfrac{(n-1)s_n^2}{\sigma^2} \sim \chi^2(n-1)$ and that $\dfrac{\bar{\xi}_n - \mu}{\sigma}\sqrt{n} \sim N(0,1)$. Now let

$$s_n \quad = \quad \sqrt{s_n^2}$$

Then $\dfrac{\bar{\xi}_n - \mu}{s_n}\sqrt{n} \sim t(n-1)$ and we can establish the analogous result

$$\bar{\xi}_n - t\frac{s_n}{\sqrt{n}} \quad < \quad \mu \quad < \quad \bar{\xi}_n + t\frac{s_n}{\sqrt{n}} \qquad\qquad (p)$$

where F is the t distribution function with $n-1$ degrees of freedom, and $F(t) = 1 - (1-p)/2 = (1+p)/2$.

Worked Example: Given the following temperature measurements,

x_1	x_2	x_3	x_4	x_5	x_6
20.1 °C	19.7 °C	19.9 °C	20.2 °C	19.8 °C	20.0 °C

estimate a symmetric confidence interval for the temperature.

We will assume a normal distribution, i.e., $\xi_i \sim N(\mu, \sigma)$ for $i = 1, \ldots, 6$. We introduce the sample variables $\bar{\xi}_6$ and s_6^2:

$$\begin{aligned}\bar{\xi}_6 &= \frac{1}{6}\sum_{i=1}^{6}\xi_i \\ s_6^2 &= \frac{1}{5}\sum_{i=1}^{6}(\xi_i - \bar{\xi}_6)^2\end{aligned}$$

From this with can construct a new sample variable with a t distribution with five degrees of freedom:

$$\frac{\bar{\xi}_6 - \mu}{\sqrt{s_6^2/6}} \sim t(5)$$

Thus, the following is a symmetric confidence interval

$$\mu = \bar{x}_6 \pm t\frac{\sqrt{\hat{s}_6^2}}{\sqrt{6}} \qquad (p)$$

where t is determined from the t-distribution function with 5 degrees of freedom and the desired confidence degree p. Let us assume that we want p to be 0.95:

$$F(t) = 1 - (1-p)/2 = 1/2 + p/2 = 1/2 + 0.95/2 = 0.975$$

From the t-distribution table we find $t = 2.571$ and from the data we calculate

$$\begin{aligned}\bar{x}_6 &= \frac{1}{6}\sum_{i=1}^{6}x_i = 19.95\ ^\circ\mathrm{C} \\ \hat{s}_6^2 &= \frac{1}{5}\sum_{i=1}^{6}(x_i - \bar{x}_6)^2 = 0.035\ (^\circ\mathrm{C})^2 \\ t\frac{\sqrt{\hat{s}_6^2}}{\sqrt{6}} &= 0.196\ ^\circ C < 0.20\ ^\circ\mathrm{C}\end{aligned}$$

This yields us

$$\mu = 19.95\ ^\circ\mathrm{C} \pm 0.20\ ^\circ\mathrm{C} \qquad (0.95)$$

1.7.6 Hypothesis Testing and Significance

Assume that we wish to prove something, e.g. that some gambler is using a loaded die with a distinct bias towards (or against) the outcome Six. We then throw the die a large number of times and observe the relative frequency of the outcome Six. If this deviates considerably from $\frac{1}{6}$, we conclude that the die is loaded.

How can we get this more mathematically well-founded? Well, having just learned a lot about confidence intervals, we will use similar techniques.

We create an estimator with a known distribution that depends on p, the probability of Six, in much the same way that we would when constructing a confidence interval. In this case, we know that $nf_n \sim bin(n, np)$ and that thus $P(x_1 < nf_n < x_2) = F(x_2) - F(x_1)$, where $F(x)$ is the binomial distribution function. To simplify the calculations for large values of n, we can instead use that $\frac{(f_n - p)\sqrt{n}}{\sqrt{p(1-p)}}$ is approximately $\sim N(0,1)$ to establish that

$$P(p - 1.96\sqrt{\frac{p(1-p)}{n}} < f_n < p + 1.96\sqrt{\frac{p(1-p)}{n}}) \quad \approx \quad 0.95$$

As we recall from Section 1.5.4, 1.96 is the 97.5-percent fractile of the Standard Normal distribution. Since the confidence interval is symmetric, this means that we cut off 2.5 percent of the probability mass in both ends, leaving 95 percent.

Under the assumption that p is in fact $\frac{1}{6}$, this means that

$$P(\frac{1}{6} - \frac{0.73}{\sqrt{n}} < f_n < \frac{1}{6} + \frac{0.73}{\sqrt{n}}) \quad \approx \quad 0.95$$

If the observed relative frequence is not in this interval, we argue that the observed outcome is so unlikely if p were in fact $\frac{1}{6}$, that we can conclude that p is not $\frac{1}{6}$. If for example we throw the die 100 times and it shows Six in 25 of them, the observed relative frequency 0.25 is not contained in this interval, and we conclude that the die is in fact loaded, and our gambler has had it.

This is an example of a *hypothesis test*. The basic line of reasoning to extract from this is that if the thing we want to prove false were really true, then the observed data would be very unlikely. More formally, we assume that a *null hypothesis* H_0, that we want to prove false, is really true. We then calculate the probability of *as extreme* a statistical material as the one observed, given that H_0 is indeed true. Now, if this probability is less then some predefined value p, we can discard the null hypothesis *at significance level* p. In the above example, the significance level is 5%. The significance level is related to the confidence degree in that a method for constructing a confidence interval with confidence degree $1 - p$ can be used to construct a hypothesis test with significance level p.

Note that if we cannot discard the null hypothesis, this doesn't mean that we may conclude that it is true. It may very well be false or highly unlikely, but the available statistical material does not allow us to discard it at the desired significance level.

> **Yoga time:** Again, the null hypothesis H_0 is associated with no uncertainty or stochastic behavior. It is either true or false. If we decide to discard H_0 and it is indeed false, the method worked. If it is in fact true, we goofed. If we do not decide to discard H_0, we have not made any mistake, but we have on the other hand not gained any new knowledge about the world. The significance level p only gives us an indication of how often we will wrongly conclude that the null hypothesis is false. That's all.

Note that the significance level and the confidence degree are not symmetrical: A confidence interval with 0 % confidence degree will never contain the estimated parameter, but while a hypothesis test with significance level 100 % would be a very crappy test indeed, this does not exclude the possibility that the null hypothesis may in fact be wrong.

1.8 Further Reading

For other literature covering the material presented in this chapter, as well as additional material, we strongly recommend [DeGroot 1975]. Another good text book is [Mood *et al* 1974]. We plan to extend the section on estimation considerably. Estimation and hypothesis testing are related to Multivariate Analysis, see Section 2.3 when this has been written.

Chapter 2

Applied Statistics

In this chapter we will apply the theory of stochastic variables presented in the previous chapter to stochastic processes and information theory. There is however a slight complication stemming from the fact that the theoretical apparatus of the previous chapter assumes that the range of the random variables are numbers. Here, they will be finite sets, but not necessarily subsets of the natural numbers. This can be dealt with by constructing a mapping between the set of outcomes of the random variable and the natural numbers. In particular, since the range of any random variable is finite, it will always be of the form $\{x_1, \ldots, x_N\}$. We will thus tacitly assume the mapping $x_i \rightsquigarrow i$ and merrily apply our high-powered theoretical machinery.

2.1 Markov Models

This section is partly due to Thorsten Brants. Here we will briefly discuss stochastic processes in general, before turning to Markov chains, Markov models and hidden Markov models. We will then proceed to discuss how to efficiently calculate the involved probabilities, how to determine the most likely state sequence using the Viterbi algorithm, and how to find an optimal parameter setting using Baum-Welch reestimation.

2.1.1 Stochastic Processes

A *stochastic* or *random process* is a sequence $\xi_1, \xi_2, \ldots$ of random variables based on the same sample space Ω. The possible outcomes of the random variables are called the set of possible states of the process. The process will be said to be in state ξ_t at time t. Note that the random variables are in general not independent. In fact, the interesting thing about stochastic processes is the dependence between the random variables.

> **Example:** Returning to the imperishable example of the unbiased die, the sequence of outcomes when repeatedly casting the die is a stochastic process with discrete random variables and a discrete time parameter.

> **Example:** Let us consider another simple example — the telephone example. Here we have three telephone lines, and at any given moment 0, 1, 2 or 3 of them can be busy. Once every minute we will observe how many of them are busy. This will be

> a random variable with $\Omega_\xi = \{0, 1, 2, 3\}$. Let ξ_1 be the number of busy lines at the first observation time, ξ_2 the number busy lines at the second observation time, etc. The sequence of the number of busy lines then forms a random process with discrete random variables and a discrete time parameter.

We will here only consider stochastic processes with a discrete time parameter and a finite sample space. Measuring the voltage in one of the telephone lines every minute, instead of counting the number of busy lines, in the previous example would be an example of a random process with continuous random variables and a discrete time parameter. Continuously monitoring the number of busy lines at every point in time would be an example of a random process with a continuous time parameter and a finite sample space.

To fully characterize a random process with a discrete time parameter t, we need to specify

1. the probability $P(\xi_1 = x_j)$ of each outcome x_j for the first observation, i.e., the initial state ξ_1 and
2. for each subsequent observation (state) $\xi_{t+1} : t = 1, 2, \ldots$ the conditional probabilities $P(\xi_{t+1} = x_{i_{t+1}} \mid \xi_1 = x_{i_1}, \ldots, \xi_t = x_{i_t})$.

To avoid getting into a lot of mathematical trouble with defining distributions over infinite sequences of random variables, we can terminate after some finite number of steps T, and instead use our knowledge of multidimensional stochastic variables.

2.1.2 Markov Chains and the Markov Property

A *Markov chain* is a special type of stochastic process where the probability of the next state conditional on the entire sequence of previous states up to the current state is in fact only dependent on the current state. This is called the *Markov property* and can be stated as:

$$P(\xi_{t+1} = x_{i_{t+1}} \mid \xi_1 = x_{i_1}, \ldots, \xi_t = x_{i_t}) \quad = \quad P(\xi_{t+1} = x_{i_{t+1}} \mid \xi_t = x_{i_t})$$

This means that the probability of a Markov chain $\xi_1, \xi_2, \ldots$ can be calculated as:

$$\begin{aligned} P(\xi_1 = x_{i_1}, \ldots, \xi_t = x_{i_t}) &= \\ &= P(\xi_1 = x_{i_1}) \cdot P(\xi_2 = x_{i_2} \mid \xi_1 = x_{i_1}) \cdot \ldots \cdot P(\xi_t = x_{i_t} \mid \xi_{t-1} = x_{i_{t-1}}) \end{aligned}$$

The conditional probabilities $P(\xi_{t+1} = x_{i_{t+1}} \mid \xi_t = x_{i_t})$ are called the transition probabilities of the Markov chain. A *finite Markov chain* must at each time be in one of a finite number of states.

> **Example:** We can turn the telephone example with 0, 1, 2 or 3 busy lines into a (finite) Markov chain by assuming that the number of busy lines will depend only on the number of lines that were busy the last time we observed them, and not on the previous history.

Consider a Markov chain with n states $s_1, \ldots, s_n$. Let p_{ij} denote the transition probability from State s_i to State s_j, i.e., $P(\xi_{t+1} = s_j \mid \xi_t = s_i)$. The transition matrix for this Markov process is then defined as

$$\mathbf{P} = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \cdots & \cdots & \cdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix}, \qquad p_{ij} \geq 0, \qquad \sum_{j=1}^{n} p_{ij} = 1, i = 1, \ldots, n$$

In general, a matrix with these properties is called a *stochastic matrix*.

Example: The stochastic process derived from the immortal example of the unbiased die, can be described by a six-by-six matrix where each element is $1/6$.

Example: In the telephone example, a possible transition matrix could for example be:

$$\mathbf{P} = \begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cccc} s_0 & s_1 & s_2 & s_3 \end{array} \\ \left[\begin{array}{cccc} 0.2 & 0.5 & 0.2 & 0.1 \\ 0.3 & 0.4 & 0.2 & 0.1 \\ 0.1 & 0.3 & 0.4 & 0.2 \\ 0.1 & 0.1 & 0.3 & 0.4 \end{array}\right] \end{array}$$

Assume that currently, all three lines are busy. What is then the probability that at the next point in time exactly one line is busy? The element in Row 3, Column 1 (p_{31}) is 0.1, and thus $p(1|3) = 0.1$. (Note that we have numbered the rows and columns 0 through 3.)

We can now determine the transition matrix for several steps. The following is the transition matrix for two steps:

$$\begin{aligned} p_{ij}^{(2)} &= P(\xi_{t+2} = s_j \mid \xi_t = s_i) \\ &= \sum_{r=1}^{n} P(\xi_{t+1} = s_r \cap \xi_{t+2} = s_j \mid \xi_t = s_i) \\ &= \sum_{r=1}^{n} P(\xi_{t+1} = s_r \mid \xi_t = s_i) \cdot P(\xi_{t+2} = s_j \mid \xi_{t+1} = s_r) \\ &= \sum_{r=1}^{n} p_{ir} \cdot p_{rj} \end{aligned}$$

The element $p_{ij}^{(2)}$ can be determined by matrix multiplication. This is the element in Row i, Column j of the matrix $\mathbf{P}^2 = \mathbf{PP}$. In the same way, we can determine the transition probabilities for several steps. In general, the transition matrix for t steps is $\mathbf{P}^t$.

Example: Returning to the telephone example: Assuming that currently all three lines are busy, what is the probability of exactly one line being busy after two steps in time?

$$\mathbf{P}^2 = \mathbf{PP} = \begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cccc} s_0 & s_1 & s_2 & s_3 \end{array} \\ \left[\begin{array}{cccc} 0.22 & 0.37 & 0.25 & 0.15 \\ 0.21 & 0.38 & 0.25 & 0.15 \\ 0.17 & 0.31 & 0.30 & 0.20 \\ 0.12 & 0.22 & 0.28 & 0.24 \end{array}\right] \end{array}$$

$\Rightarrow p_{31}^{(2)} = 0.22$

A vector $\mathbf{v} = [v_1, \ldots, v_n]$ with $v_i \geq 0, i = 1, \ldots, n$ and $\sum_{i=1}^{n} v_i = 1$ is called a *probability vector*. The probability vector that determines the state probabilities of the observations of the first element (state) of a Markov chain, i.e., where $v_i = P(\xi_1 = s_i)$, is called an *initial probability vector*. The initial probability vector and the transition matrix together determine the probability of the chain being in a particular state at a particular point in time. This probability can be calculated by multiplying the vector $\mathbf{v}$ with the matrix $\mathbf{P}$ the appropriate number of times. If $p^{(t)}(s_i)$ is the probability of the chain being in state s_i at time t, i.e., after $t-1$ steps, then

$$[p^{(t)}(s_1), \ldots, p^{(t)}(s_n)] \;=\; \mathbf{v}\mathbf{P}^{t-1}$$

Example: Returning to the telephone example: Let $v = [0.5, 0.3, 0.2, 0.0]$. What is then the probability that after two steps exactly two lines are busy?

$$\begin{array}{cc} & \begin{array}{cccc} s_0 & s_1 & s_2 & s_3 \end{array} \\ \mathbf{vP}^2 = \mathbf{vPP} = & \left[\begin{array}{cccc} 0.21 & 0.43 & 0.24 & 0.12 \end{array} \right] \end{array}$$

$\Rightarrow p(\xi_2 = s_2) = 0.24$.

2.1.3 Markov Models

Let us attribute each state in a Markov chain with a finite set of signals. After each transition, one of the signals associated with the current state is emitted. Thus, we introduce a new sequence of random variables $\eta_t, t = 1, ..., T$ representing the signal emitted at time t. This determines a *Markov model*.

A Markov model consists of:

- a finite set of states $\Omega = \{s_1, \ldots, s_n\}$;
- an signal alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$;
- an $n \times n$ state transition matrix $\mathbf{P} = [p_{ij}]$ where $p_{ij} = P(\xi_{t+1} = s_j \mid \xi_t = s_i)$;
- an $n \times m$ signal matrix $\mathbf{A} = [a_{ij}]$, which for each state-signal pair determines the probability $a_{ij} = p(\eta_t = \sigma_j \mid \xi_t = s_i)$ that signal σ_j will be emitted given that the current state is s_i;
- and an initial vector $\mathbf{v} = [v_1, \ldots, v_n]$ where $v_i = P(\xi_1 = s_i)$.

The probability of reaching some particular state at some particular time is determined just as in the case of the Markov chain. The probability that signal σ_j will be emitted in the current state s_i is thus precisely the element a_{ij} of the signal matrix. We have thus made a second Markov assumption, namely that the probability of a particular signal being emitted does only depend on the current state, and not on the sequence of pervious ones.

From this it follows that the probability of the model being in state s_i and emitting signal σ_j at time t is

$$p^{(t)}(s_i, \sigma_j) = p^{(t)}(s_i) \cdot p(\eta_t = \sigma_j \mid \xi_t = s_i)$$

where $p^{(t)}(s_i)$ is the ith element of the vector $\mathbf{vP}^{t-1}$. The probability that signal σ_j will be emitted at time t is then:

$$p^{(t)}(\sigma_j) = \sum_{i=1}^{n} p^{(t)}(s_i, \sigma_j) = \sum_{i=1}^{n} p^{(t)}(s_i) \cdot p(\eta_t = \sigma_j \mid \xi_t = s_i)$$

Thus if $p^{(t)}(\sigma_j)$ is the probability of the model emitting signal σ_j at time t, i.e., after $t - 1$ steps, then

$$[p^{(t)}(\sigma_1), \ldots, p^{(t)}(\sigma_m)] \quad = \quad \mathbf{vP}^{t-1}\mathbf{A}$$

The Markov models described this far are of first order, i.e., the probability of the next state depends only on the current state. In an nth order Markov model, this transition probability depends on the way the current state was reached, and in particular on the $n - 1$ previous states. This means that we need to specify the probabilities $P(s_{i_t} \mid s_{i_{t-n}}, \ldots, s_{i_{t-1}})$. A Markov model of higher (nth) order can always be reduced to an equivalent first-order Markov model by expanding out each possible sequence of n states in the old Markov model to a state in the new Markov model.

2.1.4 Hidden Markov Models

If it is not possible to observe the sequence of states $\xi_1, \ldots, \xi_T$ of a Markov model, but only the sequence $\eta_1, \ldots, \eta_T$ of emitted signals, the model is called a *hidden Markov model* (an HMM).

Let $\mathbf{O} \in \Sigma^*$ be a known sequence of observed signals and let $\mathbf{S} \in \Omega^*$ be the unknown sequence of states in which $\mathbf{O}$ was emitted. Our best guess at $\mathbf{S}$ is the sequence maximizing

$$\max_{\mathbf{S}} P(\mathbf{S} \mid \mathbf{O})$$

From Bayes' inversion formula (Eq. 1.3), redisplayed here for your convenience,

$$P(\mathbf{S} \mid \mathbf{O}) \;=\; \frac{P(\mathbf{O} \mid \mathbf{S}) \cdot P(\mathbf{S})}{P(\mathbf{O})}$$

we see that this $\mathbf{S}$ will also maximize

$$\max_{\mathbf{S}} P(\mathbf{O} \mid \mathbf{S}) \cdot P(\mathbf{S})$$

since $P(\mathbf{O})$ does not depend on the choice of $\mathbf{S}$. Here $P(\mathbf{O} \mid \mathbf{S})$ is called the signal model and $P(\mathbf{S})$ is called the language model. Maximizing $P(\mathbf{O} \mid \mathbf{S})$ alone would be the maximum-likelihood estimate of $\mathbf{S}$, see Section 1.7.3.

Prototypical tasks to which hidden Markov models are applied include the following. Given a sequence of signals $\mathbf{O} = (\sigma_{i_1}, \ldots, \sigma_{i_T})$:

- Estimate the probability of observing this particular signal sequence.
- Determine the most probable state sequence that can give rise to this signal sequence.
- Determine the set of model parameters $\lambda = (\mathbf{P}, \mathbf{A}, \mathbf{v})$ maximizing the probability of this signal sequence.

Part-of-speech tagging and speech recognition are examples of the second. Language identification would be an example of the first. The third problem is of interest when creating a hidden Markov model.

> **Example:** Part-of-speech tagging. The set of observable signals are the words of an input text. The states are the set of tags that are to be assigned to the words of input text. The task consists in finding the most probable sequence of states that explains the observed words. This will assign a particular state to each signal, i.e., a tag to each word.

> **Example:** Speech recognition. The set of observable signals are (some representation of the) acoustic signals. The states are the possible words that these signals could arise from. The task consists in finding the most probable sequence of words that explains the observed acoustic signals. This is a slightly more complicated situation, since the acoustic signals do not stand in a one-to-one correspondence with the words.

We can easily specify HMMs at different levels. For example, if we have an HMM for each phoneme defined in terms of the acoustic signals, an HMM for each word defined in terms of the phonemes, and an HMM for word pairs (using bigram statistics), we can expand this to a single HMM where each acoustic signal in each phoneme in each word pair corresponds to a unique state. Of course, this increases the number of states drastically, and thus also the search space.

2.1.5 Calculating $P(\mathbf{O})$

We now turn to the problem of actually calculating the probability $P(\mathbf{O})$ of the observed signal sequence. Let $\mathbf{O} = (\sigma_{k_1}, \ldots, \sigma_{k_T})$ and let $\mathbf{S} = (s_{i_1}, \ldots, s_{i_T})$. Then

$$\begin{aligned}
P(\mathbf{O} \mid \mathbf{S}) &= \prod_{t=1}^{T} P(\eta_t = \sigma_{k_t} \mid \xi_t = s_{i_t}) \;=\; \prod_{t=1}^{T} a_{i_t k_t} \\
P(\mathbf{S}) &= v_{i_1} \cdot \prod_{t=2}^{T} p_{i_{t-1} i_t} \\
P(\mathbf{O} \cap \mathbf{S}) &= P(\mathbf{O} \mid \mathbf{S}) \cdot P(\mathbf{S}) \;=\; (\prod_{t=1}^{T} a_{i_t k_t}) \cdot (v_{i_1} \cdot \prod_{t=2}^{T} p_{i_{t-1} i_t}) \;= \\
&= (a_{i_1 k_1} \cdot v_{i_1}) \cdot \prod_{t=2}^{T} p_{i_{t-1} i_t} \cdot a_{i_t k_t}
\end{aligned}$$

Putting this together yields us

$$P(\mathbf{O}) \;=\; \sum_{\mathbf{S}} P(\mathbf{O} \cap \mathbf{S}) \;=\; \sum_{s_{i_1}, \ldots, s_{i_T}} (a_{i_1 k_1} \cdot v_{i_1}) \cdot \prod_{t=2}^{T} p_{i_{t-1} i_t} \cdot a_{i_t k_t}$$

Using this formula directly requires $O(2Tn^T)$ calculations. Obviously, this is not a good idea computationally.

The forward algorithm

Instead, we reshuffle the expression and introduce a set of accumulators, so-called *forward variables*, one for each time t and state i:

$$\alpha_t(i) \;=\; P(\mathbf{O}_{\leq t}; \xi_t = s_i) \;=\; P(\eta_1 = \sigma_{k_1}, \ldots, \eta_t = \sigma_{k_t}; \xi_t = s_i)$$

This is the joint probability of being in state s_i at time t and the observed signal sequence $\mathbf{O}_{\leq t} = \sigma_{k_1}, \ldots, \sigma_{k_t}$ from time 1 to time t.

Note that

$$\begin{aligned}
P(\mathbf{O}) &= P(\eta_1 = \sigma_{k_1}, \ldots, \eta_T = \sigma_{k_T}) \;= \\
&= \sum_{i=1}^{n} P(\eta_1 = \sigma_{k_1}, \ldots, \eta_T = \sigma_{k_T}; \xi_T = s_i) \;=\; \sum_{i=1}^{n} \alpha_T(i)
\end{aligned}$$

that

$$\alpha_1(i) \;=\; P(\eta_1 = \sigma_{k_1}; \xi_1 = s_i) \;=\; a_{i k_1} \cdot v_i \qquad \text{for } i = 1, \ldots, n$$

and that

$$\alpha_{t+1}(j) \;=\; [\sum_{i=1}^{n} \alpha_t(i) \cdot p_{ij}] \cdot a_{j k_{t+1}} \qquad \text{for } t = 1, \ldots, T-1 \;;\; j = 1, \ldots, n$$

since

$$
\begin{aligned}
P(\mathbf{O}_{\leq t+1}; \xi_{t+1} = s_j) &= \\
&= \sum_{i=1}^{n} p(\mathbf{O}_{\leq t}; \xi_t = s_i) \cdot P(\eta_{t+1} = \sigma_{k_{t+1}}; \xi_{t+1} = s_j \mid \mathbf{O}_{\leq t}; \xi_t = s_i) = \\
&= \sum_{i=1}^{n} P(\mathbf{O}_{\leq t}; \xi_t = s_i) \cdot P(\eta_{t+1} = \sigma_{k_{t+1}} \mid \xi_{t+1} = s_j) \cdot P(\xi_{t+1} = s_j \mid \xi_t = s_i)
\end{aligned}
$$

The first equality is exact, while the second one uses the Markov assumptions. Calculating $P(\mathbf{O})$ this way is known as the *forward algorithm* and requires $O(n^2T)$ calculations, which is a considerable savings.

The trellis

An efficient way of representing this is by the use of a *trellis*, i.e., a graph with a node for each state-time pair, where each node at time t is connected only to the nodes at times $t-1$ and $t+1$. Figure 2.1 shows an example of a trellis. Each node in the trellis corresponds to being in a particular state at a particular time, and can be attributed some appropriate accumulator, for example a forward variable. Using the appropriate recurrence equation, we can calculate the values of the accumulators for the next time $t+1$ from those of the current time t, or possibly the other way around.

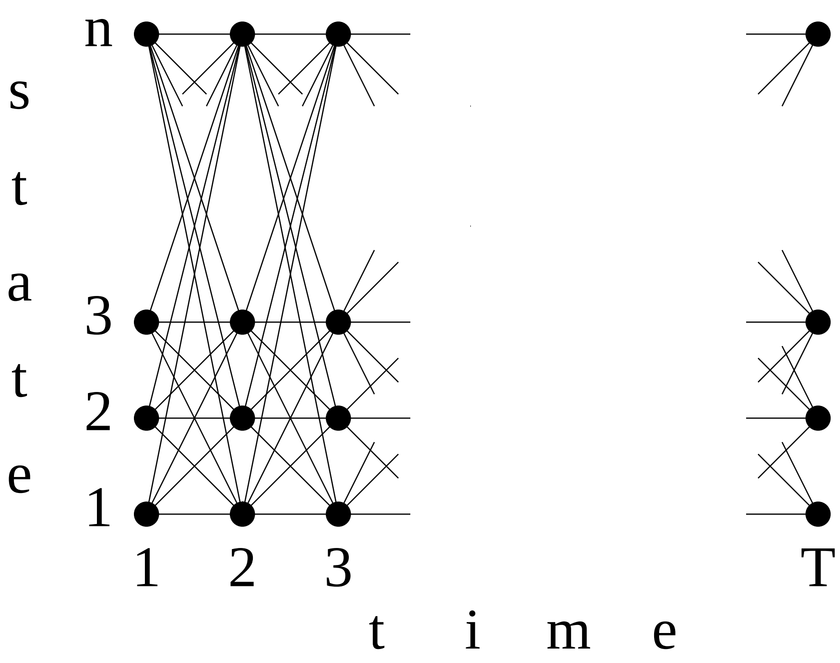

Figure 2.1: Trellis

The backward algorithm

Alternatively, we can define the set of *backward variables*:

$$
\beta_t(i) = P(\mathbf{O}_{>t} \mid \xi_t = s_i) = P(\eta_{t+1} = \sigma_{k_{t+1}}, \ldots, \eta_T = \sigma_{k_T} \mid \xi_t = s_i)
$$

This is the probability of the observed signal sequence $\mathbf{O}_{>t} = \sigma_{k_{t+1}}, \ldots, \sigma_{k_T}$ from time $t+1$ to time T conditional on being in state s_i at time t.

Note that

$$\begin{aligned} P(\mathbf{O}) &= \sum_{i=1}^{n} P(\eta_1 = \sigma_{k_1}, \xi_1 = s_i) \cdot P(\eta_2 = \sigma_{k_2}, \ldots, \eta_T = \sigma_{k_T} \mid \xi_1 = s_i) \;= \\ &= \sum_{i=1}^{n} a_{ik_1} \cdot v_i \cdot \beta_1(i) \end{aligned}$$

Let us define

$$\beta_T(i) \;=\; 1 \quad \text{for } i = 1, \ldots, n$$

and note that

$$\beta_t(i) \;=\; \sum_{j=1}^{n} p_{ij} \cdot a_{jk_{t+1}} \cdot \beta_{t+1}(j) \quad \text{for } t = 1, \ldots, T-1 \;;\; i = 1, \ldots, n$$

since

$$\begin{aligned} P(\mathbf{O}_{>t} \mid \xi_t = s_i) &= \sum_{j=1}^{n} P(\mathbf{O}_{>t}; \xi_{t+1} = s_j \mid \xi_t = s_i) \;= \\ &= \sum_{j=1}^{n} P(\mathbf{O}_{>t} \mid \xi_t = s_i, \xi_{t+1} = s_j) \cdot P(\xi_{t+1} = s_j \mid \xi_t = s_i) \;= \\ &= \sum_{j=1}^{n} p(\eta_{t+1} = \sigma_{k_{t+1}} \mid \xi_{t+1} = s_j) \cdot P(\mathbf{O}_{>t+1} \mid \xi_{t+1} = s_j) \cdot P(\xi_{t+1} = s_j \mid \xi_t = s_i) \end{aligned}$$

Again, the first equality is exact, while the second one uses the Markov assumptions. Calculating $P(\mathbf{O})$ using the backward variables is called the *backward algorithm* and also requires $O(n^2 T)$ calculations.

The forward-backward algorithm

We can also use a mixture of the forward and backward variables since

$$\begin{aligned} P(\mathbf{O}) &= \sum_{i=1}^{n} P(\mathbf{O}; \xi_t = s_i) \;=\; \sum_{i=1}^{n} P(\mathbf{O}_{\leq t}; \xi_t = s_i) \cdot P(\mathbf{O}_{>t} \mid \mathbf{O}_{\leq t}; \xi_t = s_i) \;= \\ &= \sum_{i=1}^{n} P(\mathbf{O}_{\leq t}; \xi_t = s_i) \cdot P(\mathbf{O}_{>t} \mid \xi_t = s_i) \;=\; \sum_{i=1}^{n} \alpha_t(i) \cdot \beta_t(i) \end{aligned}$$

Thus we need the forward variables $\alpha_t(i)$ and backward variables $\beta_t(i)$ at time t. These can be calculated recursively from the forward variables for time $1, \ldots, t$ and the backward variables for time $t, \ldots, T$. This way of calculating $P(\mathbf{O})$ is called the *forward-backward algorithm* and also requires $O(n^2 T)$ calculations.

In fact, we can define a set of *forward-backward variables*:

$$\gamma_t(i) \;=\; P(\xi_t = s_i \mid \mathbf{O}) \;=\; \frac{P(\mathbf{O}; \xi_t = s_i)}{P(\mathbf{O})} \;=\; \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{i=1}^{n} \alpha_t(i) \cdot \beta_t(i)}$$

This is the probability of being in state s_i at time t conditional on the entire observed signal sequence $\mathbf{O}$ from time 1 to time T.

2.1.6 Finding the Optimal State Sequence

A first attempt at finding an optimal sequence of states could be the one maximizing the γ-factors individually:

$$s_{i_t}^* = \arg \max_{1 \leq i \leq n} \gamma_t(i)$$

This does however not take the dependencies between the random variables into account. In particular, the constructed sequence may have zero probability, since some transitions may be impossible. In fact, due to the Markov assumption(s), this turns out to be the maximum-likelihood estimate $\max_{\mathbf{S}} P(\mathbf{O} \mid \mathbf{S})$. Instead we want to find the state sequence that maximizes $P(\mathbf{O} \mid \mathbf{S}) \cdot P(\mathbf{S})$. To this end we employ a dynamic programming technique — the Viterbi algorithm.

The Viterbi algorithm

We will instead define the set of variables

$$\begin{aligned}
\delta_t(i) &= \max_{\mathbf{S}_{\leq t-1}} P(\mathbf{S}_{\leq t-1}, \xi_t = s_i; \mathbf{O}_{\leq t}) = \\
&= \max_{s_{i_1},\ldots,s_{i_{t-1}}} P(\xi_1 = s_{i_1}, \ldots, \xi_{t-1} = s_{i_{t-1}}, \xi_t = s_i, \mathbf{O}_{\leq t})
\end{aligned}$$

This is the joint probability of the most likely state sequence from time 1 to time t ending in state s_i and the observed signal sequence $\mathbf{O}_{\leq t}$ up to time t. We also introduce the vector $\psi_t(i)$ which indicates the predecessor of state s_i in the path corresponding to $\delta_t(i)$.

Note that

$$\delta_1(i) = v_i \cdot a_{ik_1} \quad \text{for } i = 1, \ldots, n$$

that

$$\begin{aligned}
\delta_t(j) &= [\max_i \delta_{t-1}(i) \cdot p_{ij}] \cdot a_{jk_t} \quad \text{for } t = 2, \ldots, T \ ; \ j = 1, \ldots, n \\
\psi_t(j) &= \arg \max_{1 \leq i \leq n} (\delta_{t-1}(i) \cdot p_{ij}) \quad \text{for } t = 2, \ldots, T \ ; \ j = 1, \ldots, n
\end{aligned}$$

that

$$\begin{aligned}
P^* &= \max_i \delta_T(i) \\
s_{k_T}^* &= \arg \max_{1 \leq i \leq n} \delta_T(i)
\end{aligned}$$

and that

$$s_{k_t}^* = \psi_{t+1}(s_{k_{t+1}}^*) \quad \text{for } t = 1, \ldots, T-1$$

The recurrence equation can be established as follows:

$$\begin{aligned}
\delta_t(j) &= \max_{\mathbf{S}_{\leq t-1}} P(\mathbf{S}_{\leq t-1}, \xi_t = s_j; \mathbf{O}_{\leq t}) = \\
&= \max_i \max_{\mathbf{S}_{\leq t-2}} P(\mathbf{S}_{\leq t-2}, \xi_{t-1} = s_i, \xi_t = s_j; \mathbf{O}_{\leq t-1}, \eta_t = \sigma_{k_t}) = \\
&= \max_i \max_{\mathbf{S}_{\leq t-2}} P(\xi_t = s_j; \eta_t = \sigma_{k_t} \mid \mathbf{S}_{\leq t-2}, \xi_{t-1} = s_i; \mathbf{O}_{\leq t-1}) \cdot \\
&\qquad \cdot P(\mathbf{S}_{\leq t-2}, \xi_{t-1} = s_i; \mathbf{O}_{\leq t-1}) =
\end{aligned}$$

$$
\begin{aligned}
&= \max_i \max_{\mathbf{S}_{\leq t-2}} P(\xi_t = s_j \mid \xi_{t-1} = s_i) \cdot P(\eta_t = \sigma_{k_t} \mid \xi_t = s_j) \cdot \\
&\qquad \cdot P(\mathbf{S}_{\leq t-2}, \xi_{t-1} = s_i; \mathbf{O}_{\leq t-1}) \;=\\
&= [\max_i P(\xi_t = s_j \mid \xi_{t-1} = s_i) \cdot \max_{\mathbf{S}_{\leq t-2}} P(\mathbf{S}_{\leq t-2}, \xi_{t-1} = s_i; \mathbf{O}_{\leq t-1})] \cdot \\
&\quad \cdot P(\eta_t = \sigma_{k_t} \mid \xi_t = s_j) \;=\\
&= [\max_i p_{ij} \cdot \delta_{t-1}(i)] \cdot a_{j\,k_t}
\end{aligned}
$$

The first three equalities are exact, while the fourth one uses the Markov assumptions.

The Viterbi algorithm can be implemented efficiently using a trellis, and the main difference to the forward algorithm for calculating $P(\mathbf{O})$ is that here the accumulators represent maximums, rather than sums. In fact, the number of required calculations is the same, namely $O(n^2T)$.

The Viterbi algorithm is discussed in more detail in [Forney 1973], while general dynamic programming is well-described in [Bellman 1957].

2.1.7 Parameter Estimation for HMMs

We now turn to the problem of estimating the parameters $\mathbf{v}$, $\mathbf{P}$ and $\mathbf{A}$ defining the HMM. If we have access to annotated training data, i.e., data where we know both the sequence of signals and the underlying sequence of states, we can calculate the parameters directly from the observed relative frequencies. It is however in this case necessary to deal appropriately with the problem of sparse data as discussed in Section 2.4.

If we only have training data consisting of signal sequences where the underlying state sequences are unknown, we have to resort to other methods. Here we will define a set of recurrence equations that can be used to iteratively improve our estimates of the parameters. By doing this right, we can get some guarantees that each iteration will yield us a better set of estimates.

To this end, we first define the set of probabilities

$$
\epsilon_t(i,j) \;=\; P(\xi_t = s_i, \xi_{t+1} = s_j \mid \mathbf{O}) \;=\; \frac{P(\mathbf{O}; \xi_t = s_i, \xi_{t+1} = s_j)}{\mathbf{O}}
$$

This is the joint probability of being in state s_i at time t and of being in state s_j at time $t+1$ conditional on the entire observed signal sequence $\mathbf{O}$ from time 1 to time T.

Note that

$$
\epsilon_t(i,j) \;=\; \frac{\alpha_t(i) \cdot p_{ij} \cdot a_{j\,k_{t+1}} \cdot \beta_{t+1}(j)}{P(\mathbf{O})} \;=\; \frac{\alpha_t(i) \cdot p_{ij} \cdot a_{j\,k_{t+1}} \cdot \beta_{t+1}(j)}{\sum_{i,j} \alpha_t(i) \cdot p_{ij} \cdot a_{j\,k_{t+1}} \cdot \beta_{t+1}(j)}
$$

since

$$
\begin{aligned}
&P(\mathbf{O}; \xi_t = s_i, \xi_{t+1} = s_j) \;=\; P(\mathbf{O}_{\leq t+1}, \mathbf{O}_{>t+1}; \xi_t = s_i, \xi_{t+1} = s_j) \;=\\
&\quad = P(\mathbf{O}_{\leq t+1}; \xi_t = s_i, \xi_{t+1} = s_j) \cdot P(\mathbf{O}_{>t+1} \mid \mathbf{O}_{\leq t+1}; \xi_t = s_i, \xi_{t+1} = s_j) \;=\\
&\quad = P(\mathbf{O}_{\leq t}; \xi_t = s_i) \cdot P(\eta_{t+1} = \sigma_{k_{t+1}}; \xi_{t+1} = s_j \mid \mathbf{O}_{\leq t}; \xi_t = s_i) \cdot \\
&\qquad \cdot P(\mathbf{O}_{>t+1} \mid \xi_{t+1} = s_j) \;=\\
&\quad = P(\mathbf{O}_{\leq t}; \xi_t = s_i) \cdot P(\xi_{t+1} = s_j \mid \xi_t = s_i) \cdot P(\eta_{t+1} = \sigma_{k_{t+1}} \mid \xi_{t+1} = s_j) \cdot \\
&\qquad \cdot P(\mathbf{O}_{>t+1} \mid \xi_{t+1} = s_j) \;=\\
&\quad = \alpha_t(i) \cdot p_{ij} \cdot a_{j\,k_{t+1}} \cdot \beta_{t+1}(j)
\end{aligned}
$$

For this reason, the probabilities $\epsilon_t(i,j)$ are also referred to as forward-backward variables.

We immediately see that

$$\gamma_t(i) \;=\; P(\xi_t = s_i \mid \mathbf{O}) \;=\; \sum_{j=1}^{n} P(\xi_t = s_i, \xi_{t+1} = s_j \mid \mathbf{O}) \;=\; \sum_{j=1}^{n} \epsilon_t(i,j)$$

We now observe that

$$\begin{aligned}
\gamma_1(i) &= \text{probability of starting in state } s_i \\
\sum_{t=1}^{T-1} \epsilon_t(i,j) &= \text{expected number of transitions from state } s_i \text{ to state } s_j \\
\sum_{t=1}^{T-1} \gamma_t(i) &= \text{expected number of transitions from state } s_i \\
\sum_{t=1:\sigma_{k_t}=\sigma_j}^{T} \gamma_t(i) &= \text{expected number of times signal } \sigma_j \text{ is emitted in state } s_i
\end{aligned}$$

We can now establish the recurrence equations:

$$\begin{aligned}
\bar{v}_i &= \gamma_1(i) \\
\bar{p}_{ij} &= \frac{\sum_{t=1}^{T-1} \epsilon_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\
\bar{a}_{ij} &= \frac{\sum_{t=1:\sigma_{k_t}=\sigma_j}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}
\end{aligned}$$

This is known as the *Baum-Welch* or *forward-backward reestimation algorithm*, but can also be interpreted as an incarnation of the EM (expectation-modification) algorithm or as an application of Lagrange multipliers to enforce the constraints

$$\begin{aligned}
\sum_{i=1}^{n} v_i &= 1 \\
\sum_{j=1}^{n} p_{ij} &= 1 \quad \text{for } i = 1, \ldots, n \\
\sum_{j=1}^{m} a_{ij} &= 1 \quad \text{for } i = 1, \ldots, n
\end{aligned}$$

when maximizing $P(\mathbf{O})$ w.r.t. $\lambda = (\mathbf{v}, \mathbf{P}, \mathbf{A})$. The resulting equations are

$$\begin{aligned}
v_i &= \frac{v_i \frac{\partial P}{\partial v_i}}{\sum_{k=1}^{n} v_k \frac{\partial P}{\partial v_k}} \quad \text{for } i = 1, \ldots, n \\
p_{ij} &= \frac{p_{ij} \frac{\partial P}{\partial p_{ij}}}{\sum_{k=1}^{n} p_{ik} \frac{\partial P}{\partial p_{ik}}} \quad \text{for } i = 1, \ldots, n \;;\; j = 1, \ldots, n \\
a_{ij} &= \frac{a_{ij} \frac{\partial P}{\partial a_{ij}}}{\sum_{k=1}^{m} a_{ik} \frac{\partial P}{\partial a_{ik}}} \quad \text{for } i = 1, \ldots, n \;;\; j = 1, \ldots, m
\end{aligned}$$

which turn out to be exactly the reestimation equations given above after some symbolic manipulation, see [Rabiner 1989].

Let $P(\mathbf{O} \mid \lambda)$ denote the probability of the observation sequence $\mathbf{O}$ calculated using the old parameters $\mathbf{v}$, $\mathbf{P}$ and $\mathbf{A}$, and let $P(\mathbf{O} \mid \bar{\lambda})$ denote this probability calculated using the new parameters $\bar{\mathbf{v}}$, $\bar{\mathbf{P}}$ and $\bar{\mathbf{A}}$. Some nice people have shown that either $\bar{\lambda} = \lambda$ or $P(\mathbf{O} \mid \bar{\lambda}) > P(\mathbf{O} \mid \lambda)$, see [Baum 1972]. This means that we either get a better set of parameters, or the same one, in the next iteration.

2.1.8 Further reading

A comprehensive tutorial on hidden Markov models for speech recognition is given in [Rabiner 1989].

2.2 Elementary Information Theory

2.2.1 Entropy

Let ξ be a discrete stochastic variable ξ with a finite range $\Omega_\xi = \{x_1, \ldots, x_M\}$ and let $p_i = p(x_i)$ be the corresponding probabilities. How much information is there in knowing the outcome of ξ? Or equivalently: How much uncertainty is introduced by not knowing the outcome of ξ? This is the information needed to specify which of the x_i has occurred. We could do this simply by writing "x_i", but let us assume further that we only have a small set of symbols $A = \{a_k : k = 1, \ldots, D\}$ — a *coding alphabet* — at our disposal to do this. Thus each x_i will be represented by a string over A.

To get this further grounded, assume that ξ is in fact uniformly distributed, i.e., that $p_i = \frac{1}{M}$ for $i = 1, \ldots M$, and that the coding alphabet is $\{0, 1\}$. Thus, each x_i will be represented by a binary number. We will then need $N : 2^{N-1} < M \leq 2^N$ digits to specify which x_i actually occurred. Thus we need $\log_2 M$ digits[1].

So what if the distribution is nonuniform, i.e., if the p_is are not all equal? How much uncertainty does a possible outcome with probability p_i introduce? The basic assumption is that it will introduce equally much uncertainty regardless of the rest of the probabilities $p_j : j \neq i$. We can thus reduce the problem to the case where all outcomes have probability p_i. In such a case, there are $\frac{1}{p_i} = M_{p_i}$ possible outcomes. The fact that this is not in general an integer can be handled with Mathematical Yoga. We thus need $\log_2 M_{p_i} = \log_2 \frac{1}{p_i} = -\log_2 p_i$ binary digits to specify that the outcome was x_i. Thus, the uncertainty introduced by p_i is in the general case $-\log_2 p_i$. The uncertainty introduced by the random variable ξ will be taken to be the expectation value of the number of digits required to specify the outcome. This is the expectation value of $-\log_2 P(\xi)$, i.e., $\mathrm{E}[-\log_2 P(\xi)] = \sum_i -p_i \log_2 p_i$.

Now, we let D be an arbitrary base $D > 0; D \neq 1$, and not necessarily 2. Since $\log_2 X = \log_2 D \cdot \log_D X$, this does not change anything material, only introduces a scaling factor $\log_2 D$, so the uncertainty could equally well be taken to be $\sum_i -p_i \log_D p_i$, for any base $D > 0; D \neq 1$. We will use the particularly good base $e \approx 2.72$, which is not an integer, nor a rational number, and where "$\log_e$" is usually denoted "ln". [2] Having already dealt with sets of noninteger size, to have a coding alphabet with a noninteger number of code symbols should be only slightly unsettling. This allows us

[1] Or really, the smallest integer $N \geq \log_2 M$.

[2] See Appendix C.3 for a discussion of the number e and the function $\ln x$.

to define the quantity $\mathrm{H}[\xi]$, called the *entropy* of ξ:

$$\mathrm{H}[\xi] \quad = \quad - \sum_{x_i \in \Omega_\xi} p(x_i) \ln p(x_i) \quad = \quad - \sum_{i=1}^{M} p_i \ln p_i$$

If q_i is another set of probabilities on Ω_ξ, then

$$- \sum_{i=1}^{M} p_i \ln p_i \quad \leq \quad - \sum_{i=1}^{M} p_i \ln q_i \tag{2.1}$$

Equality is equivalent to $\forall_i \ \ p_i = q_i$.

> **Proof:** The logarithm is a convex function on the real line, i.e., $f(x) \leq f(x_0) + f'(x_0) \cdot (x - x_0)$ for all $x \in R$. Since $\frac{d}{dx} \ln x = \frac{1}{x}$, choosing $x_0 = 1$ and thus $f(x_0) = 0$ and $f'(x_0) = 1$, we have $\ln x \leq x - 1$ for all $x \in R$. In fact, if $x \neq 1$, then $\ln x < x - 1$. In particular
>
> $$\ln \frac{q_i}{p_i} \quad \leq \quad \frac{q_i}{p_i} - 1$$
>
> By multiplying by p_i and summing over i we obtain
>
> $$\sum_{i=1}^{M} p_i \ln \frac{q_i}{p_i} \quad \leq \quad \sum_{i=1}^{M} p_i (\frac{q_i}{p_i} - 1) \quad = \quad \sum_{i=1}^{M} q_i \quad - \quad \sum_{i=1}^{M} p_i \quad = \quad 1 - 1 \quad = \quad 0$$
>
> Thus
>
> $$\sum_{i=1}^{M} p_i \ln \frac{q_i}{p_i} \quad = \quad \sum_{i=1}^{M} p_i \ln q_i \quad - \quad \sum_{i=1}^{M} p_i \ln p_i \quad \leq \quad 0$$
>
> and the proposition follows. We note that if for any i, $0 \neq p_i \neq q_i$, we have
>
> $$p_i \ln \frac{q_i}{p_i} \quad < \quad q_i - p_i$$
>
> and we obtain strict inequality through a similar summation. $\square$

The *joint entropy* of ξ and η is defined as:

$$\mathrm{H}[\xi, \eta] \quad = \quad - \sum_{i=1}^{M} \sum_{j=1}^{L} p(x_i, y_j) \ln p(x_i, y_j)$$

The *conditional entropy* of ξ given η is defined as:

$$\begin{aligned} \mathrm{H}[\xi \mid \eta] \quad &= \\ = \quad - \sum_{j=1}^{L} p(y_j) \sum_{i=1}^{M} p(x_i \mid y_j) \ln p(x_i \mid y_j) \quad &= \quad - \sum_{j=1}^{L} \sum_{i=1}^{M} p(x_i, y_j) \ln p(x_i \mid y_j) \end{aligned}$$

The conditional and joint entropies are related just like the conditional and joint probabilities:

$$\mathrm{H}[\xi, \eta] \quad = \quad \mathrm{H}[\eta] + \mathrm{H}[\xi \mid \eta]$$

This can easily be established from the above definitions by utilizing the fact that $p(x_i, y_j) = p(y_j) \cdot p(x_i \mid y_j)$ and rearranging the summation orders.

The following two inequalities hold

$$\begin{aligned} \mathrm{H}[\xi, \eta] &\leq \mathrm{H}[\xi] + \mathrm{H}[\eta] \\ \mathrm{H}[\xi \mid \eta] &\leq \mathrm{H}[\xi] \end{aligned}$$

where in both cases equality is equivalent to ξ and η being independent. The first one is an immediate consequence of the second one, which in turn can be established much in the same way as the previous equation using Eq. 2.1.

The *information conveyed by* η, denoted $\mathrm{I}[\xi \mid \eta]$, is the reduction in entropy of ξ by finding out the outcome of η. This is defined by

$$\mathrm{I}[\xi \mid \eta] = \mathrm{H}[\xi] - \mathrm{H}[\xi \mid \eta]$$

Note that:

$$\begin{aligned} \mathrm{I}[\xi \mid \eta] &= \mathrm{H}[\xi] - \mathrm{H}[\xi \mid \eta] = \mathrm{H}[\xi] - (\mathrm{H}[\xi, \eta] - \mathrm{H}[\eta]) = \\ &= \mathrm{H}[\xi] + \mathrm{H}[\eta] - \mathrm{H}[\xi, \eta] = \mathrm{H}[\eta] + \mathrm{H}[\xi] - \mathrm{H}[\eta, \xi] = \mathrm{I}[\eta \mid \xi] \end{aligned}$$

This means that the information about ξ conveyed by η is necessarily equal to the information about η conveyed by ξ.

Finally, we note that:

$$\begin{aligned} \mathrm{H}[\xi] &= \mathrm{E}[-\ln P(\xi)] \\ \mathrm{H}[\xi \mid \eta] &= \mathrm{E}[-\ln P(\xi \mid \eta)] \\ \mathrm{I}[\xi \mid \eta] &= \mathrm{H}[\xi] - \mathrm{H}[\xi \mid \eta] = \mathrm{E}[-\ln P(\xi)] - \mathrm{E}[-\ln P(\xi \mid \eta)] = \\ &= \mathrm{E}[-\ln P(\xi) + \ln P(\xi \mid \eta)] = \mathrm{E}[-\ln \frac{P(\xi)}{P(\xi \mid \eta)}] \end{aligned}$$

All this this can readily be generalized to several variables by instead letting ξ and η denote sets of variables, and letting i and j be multiindices, i.e., ordered tuples of indices.

2.2.2 Related Information Measures

Mutual Information

A very popular quantity in the Computational Linguistics literature around 1990 was Mutual Information Statistics, denoted MI. If ξ and η are two discrete random variables, we can define the *mutual information* as:

$$\mathrm{MI}[\xi, \eta] = \mathrm{E}[\ln \frac{P(\xi, \eta)}{P(\xi) \cdot P(\eta)}]$$

Note that this is symmectic in ξ and η, i.e., that $\mathrm{MI}[\xi, \eta] = \mathrm{MI}[\eta, \xi]$. Actually, the quantity presented as the mutual information in most of these publications was the corresponding expression for the outcomes x_i, and y_j:

$$\mathrm{MI}[x_i, y_j] = \ln \frac{P(x_i, y_j)}{P(x_i) \cdot P(y_j)}$$

If $\mathrm{MI}[x_i, y_j] \gg 0$, there is a strong correlation between x_i and y_j; if $\mathrm{MI}[x_i, y_j] \ll 0$, there is a strong negative correlation. We will however let mutual information refer to the quantity with the expectation value.

Now

$$\mathrm{MI}[\xi,\eta] \;=\; \mathrm{E}[\ln \frac{P(\xi,\eta)}{P(\xi)\cdot P(\eta)}] \;=\; \mathrm{E}[\ln \frac{P(\xi \mid \eta)}{P(\xi)}] \;=\; \mathrm{E}[-\ln \frac{P(\xi)}{P(\xi \mid \eta)}]$$

Comparing this with the expression for the conveyed information given at the end of Section 2.2.1, we find

$$\mathrm{I}[\xi \mid \eta] \;=\; \mathrm{E}[-\ln \frac{P(\xi)}{P(\xi \mid \eta)}] \;=\; \mathrm{MI}[\xi,\eta]$$

Thus, the mutual information is identical to the conveyed information. This incidentally allows us to see the symmetry of the conveyed information:

$$\mathrm{I}[\xi \mid \eta] \;=\; \mathrm{MI}[\xi,\eta] \;=\; \mathrm{MI}[\eta,\xi] \;=\; \mathrm{I}[\eta \mid \xi]$$

Perplexity

The perplexity of a random variable ξ is the exponential of its entropy, i.e.

$$Perp[\xi] \;=\; e^{\mathrm{H}[\xi]}$$

To take a linguistic example, assume that we are trying to predict the next word ξ_t in a word string from the previous $t-1$ ones $\xi_1,...,\xi_{t-1}$. The entropy is then a measure of how hard this prediction problem is. Let $P(w_t^i \mid w_1,...,w_{t-1})$ be an abbreviation of $P(\xi_t = w_i \mid \xi_1 = w_{k_1},...,\xi_{t-1} = w_{k_{t-1}})$.

$$\mathrm{H}[\xi_t \mid w_1,...,w_{t-1}] \;=\; \sum_{i=1}^{N} - P(w_t^i \mid w_1,...,w_{t-1}) \cdot \ln P(w_t^i \mid w_1,...,w_{t-1})$$

If all words have equal probability, i.e., $P(w_t^i \mid w_1,...,w_{t-1}) = \frac{1}{N}$, the entropy is the logarithm of the branching factor N at this point in the input string:

$$\mathrm{H}[\xi_t \mid w_1,...,w_{t-1}] \;=\; \sum_{i=1}^{N} -\frac{1}{N} \cdot \ln \frac{1}{N} = \ln N$$

This means that the perplexity $e^{\ln N}$ is the branching factor N itself. If the probabilities differ, the perplexity can be viewed as a generalized branching factor that takes this into account.

In [Jelinek 1990] we find the following definition of perplexity:

$$\ln Perp_E \;=\; \lim_{t\to\infty} -\frac{1}{t} \ln P(w_1,...,w_t)$$

where we let $Perp_E$ denote the empirical perplexity. Here $P(w_1,...,w_t)$ denotes the probability of the word string $w_1,...,w_t$, and should be read as an abbreviation of $P(\xi_1 = w_{k_1},...,\xi_t = w_{k_t})$.

Since we cannot experimentally measure infinite limits, we terminate after a finite test string $w_1,...,w_T$, arriving at the measured perplexity $Perp_M$:

$$\ln Perp_M \;=\; -\frac{1}{T} \ln P(w_1,...,w_T)$$

Rewriting $P(w_1, ..., w_t)$ as $P(w_t \mid w_1, ..., w_{t-1}) \cdot P(w_1, ..., w_{t-1})$ gives us

$$\ln Perp_M = \frac{1}{T}\sum_{t=1}^{T} - \ln P(w_t \mid w_1, ..., w_{t-1})$$

What is the expectation value of $\ln Perp_M$? Let's find out!

$$\begin{aligned}
\mathrm{E}[\ln Perp_M] &= \mathrm{E}[\frac{1}{T}\sum_{t=1}^{T} - \ln P(w_t \mid w_1, ..., w_{t-1})] = \\
&= \frac{1}{T}\sum_{t=1}^{T} \mathrm{E}[- \ln P(w_t \mid w_1, ..., w_{t-1})] = \\
&= \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} - P(w_t^i \mid w_1, ..., w_{t-1}) \cdot \ln P(w_t^i \mid w_1, ..., w_{t-1}) = \\
&= \frac{1}{T}\sum_{t=1}^{T} \mathrm{H}[\xi_t \mid w_1, ..., w_{t-1}]
\end{aligned}$$

This is the average entropy at each string position of any string of length T, corresponding to the geometric mean of the string perplexity of any string of length T. In view of the Law of Large Numbers, we realize that the empirical preplexity as defined above, will approach this quantity as t approches ∞, motivating the definition.

Cross Entropy, Relative Entropy and Metrics

Let us return to the Equation 2.1 of Section 2.2.1: If we have two distributions (collections of probabilities) $p(x)$ and $q(x)$ on Ω_ξ, then

$$-\sum_x p(x) \ln p(x) \leq -\sum_x p(x) \ln q(x) \tag{2.2}$$

Equality is equivalent to $\forall_x \;\; p(x) = q(x)$, which means that the distributions are identical. The latter quantity,

$$\mathrm{H}_p[q] = -\sum_x p(x) \ln q(x)$$

is often referred to as the *cross entropy* of q w.r.t. p. The above inequality means that the cross entropy of a distribution q w.r.t. another distribution p is minimal when q is identical to p.

We can utilize this to create a distance measure for distributions by measuring the distance between the probability collections $p(x)$ and $q(x)$. First we note using Equation 2.1 that:

$$0 \leq \sum_x p(x) \ln p(x) - p(x) \ln q(x) = \sum_x p(x) \ln \frac{p(x)}{q(x)}$$

with equality only when $\forall_x \;\; p(x) = q(x)$. We now define

$$\mathrm{D}[p \,||\, q] = \sum_x p(x) \ln \frac{p(x)}{q(x)} = \mathrm{H}_p[q] - \mathrm{H}[p]$$

This is known as the *relative entropy* or *Kullback-Leibler (KL) distance*.

A *metric* is a real-valued function that measures distances. Any metric $m(X, Y)$ must have the following three properties:

1. $m(X,Y) \geq 0$ and $m(X,Y) = 0 \Leftrightarrow X = Y$.
2. $m(Y,X) = m(X,Y)$.
3. $m(X,Y) \leq m(X,Z) + m(Z,Y)$.

The first condition states that the metric is a positive function unless the two elements are identical, in which case it takes the value zero. The second requirement is that the metric is symmetric, i.e., that the distance from Y to X is the same as the distance from X to Y. The third one is called the triangle inequality, which means that the distance from X to Y is not greater than the distance from X to some third point Z plus the distance from Z to Y.

Although we have established that the relative entropy has the first of these properties, it lacks the second two. If for example p(1) = 1/4, p(2) = 3/4; r(1) = r(2) = 1/2; and q(1) = 3/4, q(2) = 1/4, then

$$\begin{aligned} \mathrm{D}[p \,||\, r] &\neq \mathrm{D}[r \,||\, p] \quad \text{and} \\ \mathrm{D}[p \,||\, q] &> \mathrm{D}[p \,||\, r] + \mathrm{D}[r \,||\, q] \end{aligned}$$

A true metric for probability distributions is the *Euclidian distance*, which is the natural measure of distance used in the real world:

$$d(p,q) = \sqrt{\sum_x (p(x) - q(x))^2}$$

Ever since Pythagoras, we sum the squares of the Cartesian coordinates and take the square root to get the distance between two points in space. This doesn't have anything to do with entropy, though.

The relative entropy, and the cross entropy, can also be used for evaluation. Assume that p is the true probability distribution, and that we have various estimates q^k of p. The best estimate is of course p itself which will, accoring to Equation 2.2, minimize the cross entropy $\mathrm{H}_p[q]$ and thereby also the relative entropy $\mathrm{D}[p \,||\, q] = \mathrm{H}_p[q] - \mathrm{H}[p]$. We can thus claim that the smaller the cross entropy w.r.t. p, the better the estimate of p.

A problem with this approach is that we may not know the real distribution p. However, if we take a random sample X from the distribution p, and calculate $\frac{1}{|X|} \sum_{x \in X} - \ln q^k(x)$, we have an estimate of $\mathrm{H}_p[q]$. So we can argue that if q^k yields a lower value of this than $q^{k'}$, then the former is a better estimate of p than the latter. This technique is known as *cross-entropy evaluation*.

2.2.3 Noiseless Coding

Let ξ continue to be a discrete stochastic variable ξ with a finite range $\Omega_\xi = \{x_1, \ldots, x_M\}$ and corresponding probabilities $p_i = p(x_i), i = 1, \ldots, p_M$. Let us make a sequence of independent observations of ξ and call the generated string over $\{x_1, \ldots, x_M\}$ a *message*. [3] Further, let $A = \{a_k : k = 1, \ldots, D\}$ be a code alphabet. We will associate with each element x_i a string over A which we will call the *code word* for x_i. We will call the set of code words a *code*. Let n_i denote the length of the code word for x_i, and let $\bar{n} = \sum_{i=1}^M p_i n_i$ be the expectation value of the code-word length.

[3] This is an example of a discrete memoryless stochastic process, see Section 2.1.1.

The Noiseless Coding Problem: Given ξ and A defined as above, construct a code that minimizes $\bar{n}$.

A code where any string in A^* corresponds to at most one message is said to be *uniquely decipherable*.[4] This means that it is impossible to come up with a sequence of code words that can be interpreted as another sequence of code words. This is obviously a highly diserable property.

Still, we can do even better than this by requiring that it should be possible to determine the code word without looking further into the coded message. Such a code is said to be *instantaneous*. One way of guaranteeing this is to require that no code word is a prefix of another code word. A little afterthought yields that this is actually a necessity for an instantaneous code. Note that an instantaneous code is also uniquely decipherable.

Let us temporarily assume that in addition to ξ and A, the code word lengths $n_i : i = 1, \ldots, M$ are also prescribed. How can we determine whether or not it is possible to construct a uniquely decipherable, or instantaneous code? The answer is simply that the existence of both an uniquely decipherable code and an instantaneous code is equivalent to the requirement $\sum_{i=1}^{M} D^{-n_i} \leq 1$.

Proof: <To be written> □

Amongst other things, this means that if we can construct an uniquely decipherable code, we can also construct an instantaneous code.

Returning to the noiseless coding problem, where we again are in command of the code-word lengths, we can establish the following lower bound on the mean code-word length $\bar{n}$:

$$\bar{n} \quad = \quad \sum p_i n_i \quad \geq \quad \frac{\mathrm{H}[\xi]}{\ln D}$$

Equality in the above $\geq$ is equivalent to $\forall_{i \in \{1,\ldots,M\}} p_i \;=\; D^{-n_i}$. This is known as the *Noiseless Coding Theorem*. Here, the size D of the code alphabet is often referred to as the *base of the code*.

Proof: <To be written> □

In general, it is impossible to find integers $n_i : p_i = D^{-n_i}$. To construct an optimal code, we will do the second best thing and choose n_i such that

$$n_i - 1 \quad < \quad \frac{-\ln p_i}{\ln D} \leq n_i$$

We can construct an instantaneous code with these code-word lengths. Firstly, note that the second inequality implies that $p_i \geq D^{-n_i}$. Thus, there exists an instantaneous base D code with these code-word lengths, since

$$\sum_{i=1}^{M} D^{-n_i} \leq \sum_{i=1}^{M} p_i \quad = \quad 1$$

So what is the mean code-word length of this code? From the first inequality we have

$$\bar{n} \quad = \quad \sum_{i=1}^{M} p_i n_i \quad \leq \quad \sum_{i=1}^{M} p_i(\frac{-\ln p_i}{\ln D} + 1) \quad = \quad \frac{-\sum_{i=1}^{M} p_i \ln p_i}{\ln D} + \sum_{i=1}^{M} p_i \quad = \quad \frac{\mathrm{H}[\xi]}{\ln D} + 1$$

[4] A^* denotes all possible strings over the Alphabet A including the empty string.

and similarly from the second inequality we have

$$\bar{n} \;=\; \sum_{i=1}^{M} p_i n_i \;\geq\; \sum_{i=1}^{M} p_i \frac{-\ln p_i}{\ln D} \;=\; \frac{-\sum_{i=1}^{M} p_i \ln p_i}{\ln D} \;=\; \frac{\mathrm{H}[\xi]}{\ln D}$$

Thus we have proved that

> Given a discrete random variable ξ with finite Ω_ξ and with entropy $\mathrm{H}[\xi]$, there exists an base D instantaneous code whose mean code-word length $\bar{n}$ satisfies
>
> $$\frac{\mathrm{H}[\xi]}{\ln D} \;\leq\; \bar{n} \;\leq\; \frac{\mathrm{H}[\xi]}{\ln D} + 1$$

Now assume that we instead of inventing code words for individual symbols x_i, invent code words for blocks of length s, i.e., strings over $\Omega_\xi = \{x_1, \ldots, x_M\}$ of length s. This is known as *block coding*. This will correspond to the outcome of s independent observations $\xi_i : i = 1, \ldots, s$ of ξ. We can then cut down the interval to

$$\frac{\mathrm{H}[\xi]}{\ln D} \;\leq\; \bar{n}_s \;\leq\; \frac{\mathrm{H}[\xi]}{\ln D} + \frac{1}{s}$$

Here $\bar{n}_s$ is the mean code word length per symbol in Ω_ξ.

To actually construct an optimal code, we will turn to the special case of $D = 2$. The procedure is much the same for $D > 2$. We can without further loss of generality assume that the coding alphabet A is $\{0, 1\}$. We will describe the *Huffmann Procedure*, which works by recursion over the cardinality M of Ω_ξ.

If $M = 2$, then we assign x_1 the code word 0, and x_2 the code word 1.

If $M > 2$, we proceed as follows: Firstly, we may without loss of generality assume that $i < j \Rightarrow p_i \leq p_j$, and that within each group of symbols with equal probability, the symbols are ordered after code word length. Thus x_{M-1} and x_M have the smallest probabilities. For any such optimal instantaneous code of size M we have the following:

1. $i < j \Rightarrow n_i \leq n_j$.

2. $n_{M-1} = n_M$.

3. At least two code words of length n_M have a common prefix of length $n_M - 1$.

Proofs:

1. If this were not the case we could simply swap code words for x_i and x_j, reducing $\bar{n}$ and contradicting the optimality. $\square$

2. If this were not the case we could simply strip the last digit from W_M, reducing $\bar{n}$ and contradicting the optimality. $\square$

3. If no two code words of length n_M had a common prefix of length $n_M - 1$, we could strip the last from digit all code words of length n_M, reducing $\bar{n}$ and contradicting the optimality. $\square$

> The reason that we can strip the last digit from a code word without the result being another code word is the fact that the code is assumed to be an instantaneous code, were no code word is the prefix of another.

We construct a problem that is one step smaller by dropping x_M, letting $p'_i = p_i$ for $i = 1, \dots, M-2$, and setting $p'_{M-1} = p_{M-1} + p_M$. We solve the problem of size $M-1$ by recursion (most likely after having reordered the symbols, since p'_{M-1} is most likely no longer the smallest probability), yielding an optimal instantaneous code $C' = \{W'_i : i = 1, \dots, M-1\}$. We now construct from this a code $C = \{W_i : i = 1, \dots, M\}$ for the problem of size M. This is done by setting $W_i = W'_i$ for $i = 1, \dots, M-2$ and setting $W_{M-1} = W'_{M-1}0$ and $W_M = W'_{M-1}1$.

The Huffman procedure results in an optimal instantaneous code C.

> **Proof:** We prove this by induction over the size M. For the base case $M = 2$, we have the code $x_1 \to 0$ and $x_2 \to 1$, which is clearly both optimal and instantaneous.
>
> If C' is instantaneous, then C will also be instantaneous, since due to the way the code C' is extended, no code word will be a prefix of another. We now need to prove that given that the procedure yields an optimal code for size $M-1$, it will also yield an optimal code for size M. Assume the opposite[5], i.e., that there is a better code $C'' = \{W''_i : i = 1, \dots, M\}$, and thus that $\sum_{i=1}^{M} p_i n''_i < \sum_{i=1}^{M} p_i n_i$. We will use this code to create a new code $C''' = \{W'''_i : i = 1, \dots, M-1\}$ for $M-1$ that is better than C', contradicting the induction assumption and thus establishing the induction step.
>
> Again we remove W''_M and set $W'''_i = W''_i$ for $i = 1, \dots, M-2$, strip the last digit from W''_{M-1}, and assign $p'''_{M-1} = p_{M-1} + p_M$. Before we deliver the final blow, we will summerize what we have got this far:

$$
\begin{aligned}
n'_i &= n_i \text{ for } i = 1, \dots, M-2 \\
n'''_i &= n''_i \text{ for } i = 1, \dots, M-2 \\
n_{M-1} &= n_M \\
n'_{M-1} &= n_{M-1} - 1 \\
n''_{M-1} &= n''_M \\
n'''_{M-1} &= n''_{M-1} - 1 \\
p'''_i &= p''_i \;=\; p'_i \;=\; p_i \text{ for } i = 1, \dots, M-2 \\
p''_{M-1} &= p_{M-1} \\
p'''_{M-1} &= p'_{M-1} \;=\; p_{M-1} + p_M \\
p''_M &= p_M \\
\sum_{i=1}^{M} p_i n''_i &< \sum_{i=1}^{M} p_i n_i
\end{aligned}
$$

> Now we go for the kill:

$$
\sum_{i=1}^{M-1} p'''_i n'''_i - \sum_{i=1}^{M-1} p'_i n'_i \;=
$$

[5] This is a regular trick in Mathematics: To prove P under the assumptions Γ, assume $\neg P$ and prove that $\neg P \wedge \Gamma \Rightarrow \perp$. Of course, Γ is required to be consistent.

$$
\begin{aligned}
&= \sum_{i=1}^{M-2} p_i''' n_i''' + p_{M-1}''' n_{M-1}''' - \sum_{i=1}^{M-2} p_i' n_i' - p_{M-1}' n_{M-1}' \quad = \\
&= \sum_{i=1}^{M-2} p_i n_i'' + (p_{M-1} + p_M)(n_{M-1}'' - 1) \; - \\
&\quad - \sum_{i=1}^{M-2} p_i n_i - (p_{M-1} + p_{M-1})(n_{M-1} - 1) \quad = \\
&= \sum_{i=1}^{M-2} p_i n_i'' + p_{M-1} n_{M-1}'' + p_M n_{M-1}'' - p_{M-1} - p_M \; - \\
&\quad - \sum_{i=1}^{M-2} p_i n_i - p_{M-1} n_{M-1} + p_M n_{M-1} + p_{M-1} + p_M \quad = \\
&= \sum_{i=1}^{M} p_i n_i'' \; - \; \sum_{i=1}^{M} p_i n_i < 0
\end{aligned}
$$

Thus C''' is a better code than C', and we have established a contradiction. The induction step must therefore be valid, and thus the entire claim. □

2.2.4 More on Information Theory

The plan is to extend the section on Information Theory considerably to include among other things the Noisy-Channel Model. In the mean while, we refer the reader to [Ash 1965].

2.3 Multivariate Analysis

The plan is to include a rather long section on Multivariate Analysis. In the mean while, we refer the reader to [Tatsuoka 1988].

2.4 Sparse Data

The plan is to include a section on Sparse Data covering at least the following:

- Overtraining.
- Back-off Smoothing: Use the distributions in more general contexts to improve on the MLE of the probs (ie, rel freqs).
- Good-Turing Reestimation: General consideration on how population distributions behave. $r^* = (r+1) \cdot \frac{N_{r+1}}{N_r}$ and $\tilde{P}_r = \frac{r^*}{N}$. Controversial intuition: Population size for each rank invariant, $N_r \cdot r = C$ or $N_r = \frac{C}{r}$. Cf. Zipf's law, [Zipf 1935].
- Deleted Interpolation: Maximize the probability of held out data w.r.t. the backoff weights.
- Successive Abstraction.

In the mean while, we refer the reader to [Jelinek & Mercer 1980] for Deleted Interpolation, to [Good 1953] and [Katz 1987] for Good-Turing reestimation, and to [Samuelsson 1996] for Successive Abstraction.

Chapter 3

Basic Corpus Linguistics

3.1 Empirical Evaluation

Chinchor et al. write in Computational Linguistics 19(3): "One of the common problems with evaluations is that the statistical significance of the results is unknown", [Chinchor *et al* 1993], p. 409.

Empirical evaluation of natural language processing systems in general is very young. The attempt to put evaluation of the efficiency of speech recognition, and natural language processing systems on solid grounds started at the end 80ies in the US within the (D)ARPA[1] speech and natural language workshops as well as the (D)ARPA message understanding conferences. First evaluation settings were so called **blackbox evaluations**, i.e. the systems under evaluation are tested as a whole, no information about the performance of single system components can be made. In order to cope with this drawback a different type of evaluation setting, called **glassbox evaluation**, has been created. The task is to provide guidelines and measures for detailed evaluation of single system components. Crucial preconditions for efficiency evaluation are: (1) thoroughly designed test material, **corpora** annotated according to the tasks tested, and (2) suiteable **evaluation models** and **measures**.

The theoretical background for measuring system efficiency stems from information retrieval research. In the following major characteristics of a basic efficiency evaluation model, and its adaptation to natural language processing tasks will be described.

3.1.1 Contingency Tables

Contingency tables are matrices to represent classifications of observations. The use of contingency tables in natural language system evaluation was influenced by work in information retrieval research. In a simple setting contingency tables summarize results of binary decisions (yes/no decisions) on categories which are defined to classify a text. Considering a single category n binary decisions have to be made to characterize n units of text, see figure 3.1. The elements of a in figure 3.1 are called **true positives** or hits more colloquially speaking, i.e. elements that are correctly detected as positives by the system. The elements of b are named **false positives** or false alarms, i.e. elements that are wrongly considered as positives.

[1] (Defence) Advanced Research Projects Agency

Element classes c and d are called **false positives** (misses) or **true negatives** (correct rejections) respectively. False negatives are positives that are rejected by the system, true negatives on the contrary are correctly detected negatives. In terms of probabilities the fields of a contingency table are filled with conditional probabilities, see figure 3.2, with A representing the number of positive decisions made by the system, N representing the number of rejections made by the system, a and n representing the total number of positives and negatives in the test set. Note that the probabilities of true positives and false negatives, and false positives and true negatives respectively sum up to one.

	Yes is Correct	No is Correct	
Decides Yes	a	b	a+b
Decides No	c	d	c+d
	a+c	b+d	a+b+c+d=n

Figure 3.1: contingency table for binary decisions, cf. Chinchor et al. 93

	Yes is Correct	No is Correct
Decides Yes	$P(A\|a)$	$P(A\|n)$
Decides No	$P(N\|a)$	$P(N\|n)$
	1.0	1.0

Figure 3.2: contingency table with probabilities

3.1.2 Important Measures on Contingency Tables

The most important measures defined on contingency tables are the completeness measures **recall**, also known as true positive rate, **fallout**, the false positive rate, and **precision**, a measure for accuracy. Recall and fallout were first used in signal detection theory [Swets 64] to measure a system's ability to detect signals within noise. In information retrieval fallout is often replaced by precision. Another measure interdependent with precision is **error rate**.

In terms of distinguishing signals from noise, recall characterizes the number of correctly detected signals wrt. all signals presented. Fallout determines the number of correctly detected non-signals relative to the total number of non-signals, and precision determines the number of correctly detected signals relativ to the total number of items considered as signals by the system.

According to figure 3.1 recall, fallout and precision can be defined as following:

$$recall = \frac{a}{a+c}$$

$$fallout = \frac{b}{b+d}$$

$$precision = \frac{a}{a+b} = \frac{true\_positives}{true\_positives + false\_positives} = accuracy$$

$$error\_rate = 100\% - precision$$

3.1.3 Extended Contingency Table Model

In evaluation of message understanding systems the contingency table model is adapted according to the needs of natural language systems' evaluation for more fine-grained and distinctive scoring. The binary decision between correct and incorrect system output as characteristic for contingency tables is replaced by a dichotomy of answers between correct and incorrect. This is achieved by modelling the system as making binary decisions between generation of correct and incorrect fillers for database templates. For each message in the test set appropriate fills for the database templates are predefined. These template fills are called answer keys.[2] Thus the number of YES decisions made by the system is regarded as the number of fillers generated by the system, while the number of positive decisions to be made at all is the number of fillers in the answer key. Figure 3.3 gives an overview of scoring criteria developed within MUC-3 ([MUC-3 1991]). Partial matches (PAR) between system answers and answer keys are scored as half correct.

Category	Criterion
Correct COR	system response = key
Partial PAR	response $\cong$ key
Incorrect INC	response $\neq$ key
Spurious SPU	blank key vs non-blank response
Missing	blank response vs non-blank key
Noncommittal NON	blank response and blank key

Figure 3.3: scoring criteria, cf. Chinchor et al. 93

3.1.4 Measures, Extended

According to the above established criteria the already familiar evaluation metrics of recall, precision and fallout can be computed. Recall refers to the completeness of fills attempted by the system. Precison refers to the accuracy of attemted fills, and fallout refers to the degree of incorrect fills produced by the system relative to the number of possible incorrect fills (POSINC). A new measure, called **overgeneration**, is introduced. Overgeneration refers to the degree of spurious generation, i.e. the number of false decisions (not correct items are judged as correct) wrt. the total number of decisions made by the system.

In order to provide non-trivial information on system effectiveness at least two measures have to be considered. Perfect system performance for instance could be characterized in terms of 100% precision and 100% recall or 100% recall and 0% overgeneration.

$$recall = \frac{COR + 0.5 * PAR}{POS}$$

$$precision = COR + \frac{0.5 * PAR}{ACT}$$

$$fallout = \frac{INC + SPU}{POSINC}$$

[2]For a detailed description of generation of answer keys see [Chinchor *et al* 1993].

$$overgeneration = \frac{SPU}{ACT}$$

$$\#POS = \#COR + \#PAR + \#INC + \#MIS$$

$$\#ACT = \#COR + \#PAR + \#INC + \#SPU$$

Another type of measure is the so called "+1-2–scoring", which originates from evaluation of speech recognition systems. The measure allows for the option not to answer without penalty. Correct answers are scored +1, incorrect ones are scored -2, and no system responses are scored -1.

3.1.5 Loose Attempts on Measuring System Efficiency

Apart from systematic attemts to evaluate the performance of systems, a number of other, more or less loose, criteria are mentioned in the literature: time and space complexity (O-notation), measures for the correctness of the output of part-of-speech taggers, robustness, and ease of adaptability to different domains or languages. The effectiveness of part-of-speech taggers is measured in terms of accuracy, i.e. the percentage of correctly assigned parts-of-speech in running text. The notion of robustness addresses a system's ability to cope with corrupted or ungrammatical input. This feature is extremely relevant for real world applications, where ungrammatical[3] and corrupted input is likely to appear, e.g. speech signals corrupted by noise, repairs in speech, headlines and various kinds of insertions in newspaper text, etc. Even though robustness is a major factor for a system's quality, there are no objective criteria for measuring robustness. The same is true for adaptability. Adaptability addresses the question how easily a system can be adapted to new domains and languages.

In the following we examine criteria for empirical evaluation of part-of-speech taggers.

3.1.6 Empirical Evaluation of Part-of-Speech Taggers: A Case Study

The most prominent feature in the evaluation of part-of-speech taggers is accuracy. There might be at least two accuracy measures, one measuring the percentage of correct output on word level, the other measuring the percentage of correct output on sentence level, cf. [Merialdo 1994]. It is obvious that accuracy on sentence level will always be below accuracy on word level. The performance of state of the art taggers is exclusively measured by word level accuracy. Word level accuracy of state of the art taggers ranges from 95% [Church 1988], 96% [DeRose 1988], [Cutting *et al* 1992] to the extreme of 99% [Church 1988]. For a discussion of comparability problems of accuracy data cf. also [Abney 1997]. Accuracy is coupled with different conditions the systems operate on. Thus comparability of the quality of the output of systems depends on comparability of conditions such as:

[3] Ungrammaticality as opposed to a strong notion of grammaticality such as Chomsky's notion of competence grammar.

- Size of the tagset used: It is more likely to achieve modest tagging results with a small tagset.

- Size of training corpus and size of tagset: The size of the training set wrt. the size of the tagset used affects accuracy. Taggers working on small tagsets achieve relatively good results when trained on small corpora (some 10 to 100 000 words) while increasing the size of the tagset reduces accuracy in case the size of the training set is not increased accordingly.

- Type of training and test corpus: The type of corpora used for training and testing affects the quality of a tagger's output in several ways. Accuracy differs whether training and testing material are derived from a similar type of corpus or not, and whether the systems compared train and test on comparable corpora. As an example for differences in training and test corpus see experiments with VOLSUNGA [DeRose 1988]. The experiments revieled accuracy differences due to the type of test corpus.

- Complete or incomplete vocabulary: Accuracy is higher in case all words in the test corpus are known (complete vocabulary) than in case the test text comprises unknown words (incomplete vocabulary).

- Type of vocabulary: Different systems make use of different types of vocabulary, such as vocabulary derived from test and/or training corpus including frequency information relativ to the corpus in question, external lexica with or without a morphology component, extra lists of collocations, etc. An example for the effect of the change in accuracy due to change in vocabulary is given in [DeRose 1988]. The addition of an idiom list to the vocabulary improved the accuracy of the tagger described from 94% to at least 96%.

Accuracy also differs wrt. the trade-off between training method, size of training corpus, and number of iterations in training. For a detailed analysis see [Merialdo 1994].

Summing up, questions typically kept quiet about are:

How successful is a tagging method reaching 96% accuracy when 90% ?

On the basis of which data is accuracy computed? Is it either a measure of precision or simply the percentage of positives found?

Are the figures reported for the accuracy of different taggers comparable at all?

3.2 Corpora

The number of corpora (text and speech), and lexical databases available is constantly increasing; see for example figure 3.2. Various initiatives for data collection and specification of annotation guidelines exist, cf. figure 3.1. Especially, LDC has become a major site for the distribution of corpora, lexical databases, and corpus processing tools.

Initiative	Acronym	Location
Data Collection Initiative of the Association of Computational Linguisitcs	ACL/DCI	University of Pennsylvania, USA
European Corpus Initiative	ECI	University of Edinburgh, Scotland
(Defense) Advanced Research and Projects Agency	(D)ARPA/LDC	
Linguistic Data Consortium	LDC	
Text Incoding Initiative	TEI	University of Pennsylvania, USA
International Computer Archive of Modern English	ICAME	Bergen, Norway
Centre for Lexical Information	CELEX	Nijmegen, NL

Table 3.1: data collection and coding initiatives

3.2.1 Types of Corpora

Corpora are used for **training** and **testing** of statistic models of natural language and speech, and for **evaluation** of components of natural language systems; cf. sections 3.1 and 3.2.2. We distinguish **raw** and **annotated** corpora. Annotations are formal representations of partial linguistic descriptions. Due to generalization properties (find local maxima from parameters set) of statistical models, learning from annotated data (supervised learning) leads to better results than learning from raw data (unsupervised learning). Corpora differ according to the selection of material collected. We distinguish **balanced, pyramidal** and **opportunistic corpora**.

The most prominent types of corpora are **part-of-speech tagged** corpora, and corpora annotated with **syntactic structure**. They are used for various kinds of natural language statistics such as part-of-speech tagging, stochastic grammar induction, stochastic parsing, disambiguation of structural or semantic ambiguities, machine translation, lexical knowledge acquisition, etc. In the case of machine translation, **parallel corpora** are required, i.e. corpora that consist of texts and their translations. For training and testing of speech recognition systems, **time aligned, phonlogically transcribed speech corpora** are required. For the time being, corpora anotated with **discourse structure** and **semantically annotated** corpora are rare. (Cf. table 3.2.)

Corpora According to Text Type

Balanced corpora per definition consist of different text genres of size proportional to the distribution (relevance) of a certain text type within the language in question. Which of course is tricky to operationalize. See the Brown Corpus as an attempt to construct a balanced corpus.

Pyramidal corpora range from very large samples of a few representative genres to small samples of a wide variety of genres.

Opportunistic corpora: Most corpora available can be characterized as opportunistic which means "take what you can get!"

Corpora According to Annotation Type

Raw: Text is tokenized and cleaned, e.g. control characters are eliminated. Text type, headlines, and paragraphs are possibly marked. Cf. the ECI MCI.

PoS-tagged: Raw text is annotated with syntactic category at word level (part-of-speech PoS).

Treebanks: PoS-tagged text is annotaded with sceletal syntactic structure. Typically, a parse grammar is defined. Corpora are automatically parsed. Parse trees are selected and if necessary corrected by human annotators. Word strings for which no parse tree is found by the grammar are either omitted or manually annotated. Cf. the Penn Treebank I and II, [Marcus *et al.* 1994].

Linguistically interpreted corpora: In contrast to treebanks, where only syntactic category and phrase structure is annotated, linguistically interpreted corpora aim at deliberate annotation of various kinds of linguistic information; cf. [Black *et al.* 1996], [Skut *et al.* 1997].

Corpora According to Use

Training: Statistical models for NL and speech processing are trained on, we may also say learn from, large, typically annotated corpora.

Testing: Test corpora are used for evaluation of statistical models after training.

Evaluation: For system component evaluation, particularly annotated corpora (cf. [MUC-3 1991]) and test suites (cf. [Lehmann *et al.* 1996]) are used.

3.2.2 Test Suites versus Corpora

Test suites are deliberately constructed linguistic data, specified according to particular features to be tested. Thus, the annotation schemes are designed such that the characteristics of particular linguistic phenomena are optimally captured. Test suites are:

- Artificial: Selected linguistic phenomena are considered in isolation. Example data are constructed such that particular phenomena are exhaustively described. The lexicon is kept as small as possible.

- Restricted with respect to the number of phenomena treated: Only phenomena central to the particular evaluation task are described.

- Descriptive: Linguistic properties of isolated phenomena are described. Theory specific assumptions are avoided.

- Competence driven: The examples are constructed by linguists according to the above specified criteria.

- Collections of positive and negative examples: The linguistic phenomenon of interest is described by systematically varied, annotated grammatically correct and false examples.

In contrast to test suites, corpora consist of real world data, i.e. language data as occurring in running text and speech. Thus, major characteristics of corpus data are:

- Linguistic phenomena occur in context.
- Combination of performance and competence aspects of language: Corpus data reflect language usage. Thus, all data, even data that are false in terms of competence grammar must be considered as positive data.
- Frequency information on a variety of phenomena instead of full coverage of selected phenomena.
- All data appearing must be captured by the annotation scheme.

3.2.3 Tokenization

Tokenization of text, i.e. splitting up a text into tokens (words, numbers, abbreviations, acronymes, dates, etc.) is not as simple as it might seem at first glance. To get a flavour of what is meant, see the following examples.

Full Stop versus Period: Sentence boundaries are not obvious. Periods are tokens on their own (full stop), or they are part of tokens, e.g. in the case of abbreviations, dates, ordinals, enumerations.

Dashes: might be token internal (*needle-like, corpus-based*), but they are also used for punctuation (*..., will contain 16 processors - twice as many as ...*). The distinction seems to be easy, but what if blanks are missing or wrongly inserted? Words are usually delimited by blanks, or a blank and a punctuation mark. Most English compound nouns in contrast to German ones are graphically not distinct from non-compounds (*fruit cake, sandwich maker*). There are even worse cases: how to deal for instance with *New York-based*? Dashes also appear in clitics such as in French verb subject inversions, e.g. *a-t-il* (has-t-he). Here the question arises: How many tokens are there? One – *a-t-il*, two – the verb *a* (has) and the pronoun *il* (he), three – verb, pronoun and clitic *t*, or even five – verb, hyphen, clitic t, hyphen, pronoun? A standard solution is to delete the clitic t and the hyphens during preprocessing, thus information on the clitic is lost. A solution to preserve information given by the clitic is either to consider the whole string as one cliticised token, or as complex expression consisting of three tokes, namely the verb *a*, the clitic *t*, and the pronoun *il*.

Acronyms: appear in various shapes:

- capital letters only e.g. IBM (International Business Machines), BBC (British Broadcasting Company);
- uppercase and lowercase letters mixed with or without periods, e.g. Ges.m.b.H. (Gesellschaft mit beschränkter Haftung, private limited liability company);

Thus, acronymes must be distinguished on the one hand from ordinary words in capital letters such as headlines, and on the other hand from abbreviations. A clear distinction between abbreviations and acronyms is problematic as acronyms are abbreviations that function as names or have become names.

Apostrophes: There is a number of words occurring with a single quote, such as engl.: *can't, he'll, he's, father's,* or *'z* in the French string *les 'z oreilles* (the 'z ears). *Can't* is really tricky as it can be rewritten as *cannot* or *can not*. While the first looks pretty like a single word (one token), the second consists obviously of two tokens, a verb and an adverb. *He'll* can be rewritten as *he will*, thus a two token analysis seems to be justified. As a consequence *'ll* must either be considered as a word, or reinterpreted as *will*. From a linguistic point of view the second solution would be the preferred one. In the case of *he's* and *father's*, we have a similar surface form wordstring+apostrophe+s[4], but different interpretations of *s*, with the former a shorthand for the verb *is*, and the later a morphological marker for case genitive. Thus, a two token analysis of the former, but a one token anaysis for the later is justified. In the case of *les 'z oreilles*, *'z* represents the phonological liaison between the phonemes [s] and [o]. Thus, *z* neither belongs to *les* nor to *oreilles* and therefore should be treated as token on its own.

3.2.4 Training and Testing

There are different ways to train and test stochastic models for natural language and speech processing:

Training and testing on one corpus type: A corpus is divided into two parts comprising similar text genres. One part (the larger one) is used for training and thus is called **training corpus**, the other part is used for testing, obviously the **test corpus**.

Training and testing on different corpus types: If a statistical model is trained on one corpus type and tested on another, the result is likely to be worse than the result obtained from training and testing on the same corpus type. In order to get a clearer picture of a model's performance it is trained and tested on various subcorpora.

Testing on the training set: This is what you better don't! Testing on the training set means that the same corpus is used for training and testing, which is a major sin in corpus linguistics. Due to the fact that the model is already optimized on the test corpus, the outcome of testing is much better than it would be for any another test set. Thus, no valid statement about the performance of the statistical model is possible.

3.2.5 Tagsets

Tagsets vary in size, contents, complexity, and level of annotation.

The size of tagsets varies from approx. 40 to 200 different tags, e.g. 36 PoS-tags and 20 tags for non-terminals in the Penn Treebank, or 197 PoS- tags in the London-Lund Corpus.

Tagsets differ with respect to linguistic classification , see for instance German participles. They can be classified as verbal and adjectival or verbal forms. In the later case, participles used as adjectives and adjectives proper fall into one class. Thus, depending on the decisions

[4] + is here used as concatenation operator.

made in the annotation scheme, tagsets are not necessarily compatible.

Tagsets vary in granularity of the information conveyed. The LOB and Brown tagsets for instance distinguish conjunctions and prepositions, whereas this information is conflated into a single tag in the Penn Treebank. In contrast to tagsets with different classifications, tagsets with different granularity but the same partition of linguistic information can be easily transformed into each other.

Tagsets differ with respect to the complexity of tags, i.e. a tag could convey simplex or complex information. Part-of-speech tags for instance represent syntactic category only or syntactic category and inflection . The former is sufficient for languages with little inflection such as English. The later is perferable for highly inflecting languages like German and French.

Summing up, current tagsets represent mainly syntactic information, i.e. syntactic category (such as noun, verb, adverb, adjective, pronoun, article, etc.), and phrase structure (non-terminals, such as NP, VP, S, ...). Even in the case of highly inflecting languages, morphological information is largely omitted in the annotion.

In the majority of tagsets different classes of information are mingled into a single tag; e.g. syntactic category and inflection, word level and phrase level information. A typical example for the former are verbs as they usually are annotated according to their syntactic category (main, auxiliary, modal) and their inflectional properties, such as finite, infinite, past participle, present participle.

Examples for the later are adjectives and pronouns. Adjectives are often categorized as attributive or predicative. Thus, phrase level information is annotated at word level. Some German pronouns for instance can be annotated according to their syntactic context as attributive (e.g. *viele Kinder kamen*, many children came), substituting (e.g. *viele kamen*, many came), or even as accompanied by a determiner (*die vielen Kinder kamen*, the many children came). Cf. the tagset presented in [Thielen and Schiller 1995].

Apart from syntactic category and inflection, tagsets also comprise tags for foreign words, unknown words, multi-word lexemes, symbols, etc. Tagsets for semantic or pragmatic annotation are still rare.

For examples of various tagsets see appendix D.

Corpus Name	Appearance	Availability
Brown Corpus	text PoS-tagged	freely available contact: Henry_Kucera@brown.edu
	retagged and parsed by Penn Treebank project	not freely available contact: LDC ldc@linc.cis.upenn.edu http://www.ldc.upenn.edu
LOB (Lancester-Oslo/Bergen Corpus and London/Lund Corpus)	text PoS-tagged	available through ICAME contact: icame@hd.uib.no
British National Corpus	text and speech PoS-tagged	freely available within Europe contact: natcorp@oucs.ox.ac.uk
United Nations Parallel Text Corpus	parallel text English, French, Spanish	available through LDC
Language Bank of Modern Swedish	text partially with concordances	non-commercial use only contact: Goteborg University Dept. of Cl
Wall Street Journal	text	available on ACL/DCI CD-ROM I
DSO Corpus	sense-tagged English nouns, verbs	available through LDC
Collins English Dictionary '79 edition	lexicon	available on ACL/DCI CD-ROM I
CELEX-2	lexicon English, German, Dutch graphemic, phonemic transcription morpheme and syllable structure word stress	available through LDC
Word Net	conceptual word hierarchy	
Penn Treebank II	text PoS-tagged phrase structure dependency structure	available through LDC contact: ldc@linc.cis.upenn.edu
ECI MCI European Corpus Initiative's Multilingual Corpus I	raw text	available through ELSNET contact: elsnet@let.ruu.nl
MARSEC Machine-Readable Spoken English Corpus	speech orthographic transcription time-aligned	available through Uni Lancester Dept. of Ling.
Groningen Speech Corpus	speech orthographic transcription	available through SPEX contact: spex@spex.nl
TIMIT	speech time-aligned phonetic and orthographic transcription, speech waveforms	available through LDC
ATIS	speech graphemic transcription classification of utterances speech waveforms	available through LDC
VERBMOBIL	speech dialogues orthographic transcription dialogue structure	available

Table 3.2: corpora and lexical databases

Chapter 4

Stochastic Grammars

The basic idea behind stochastic grammars is to augment a grammatical theory with probabilities, thereby assigning probabilities to strings, parse trees and derivations. Or as Rens Bod puts it ([Bod 1995b], p. 14):

> "What does a statistical extension of a linguistic theory, often referred to as a "stochastic grammar", look like? In the literature, we can observe the following recurrent theme: (1) take your favorite linguistic theory (a competence model), (2) attach application probabilities to the productive units of this theory."

There are several reason for wanting to do this. One reason is to be able to rank various word strings according to probability to select the most probable one from a set of string hypotheses, for example generated by a speech-recognition system. In this case, the stochastic grammar is used as a language model. Another reason is to be able to select a preferred analysis, amongst those assigned to a given word string by the linguistic theory, by selecting the most probable one. In this case, the stochastic grammar is used for disambiguation. A third reason, although related to the second one, is to speed up parsing and/or limit the search space by pruning away suboptimal or improbable search barnches, so-called preference-based parsing. [Kimball 1973] and [Marcus 1980] constitute highly interesting reading on rule-based versions of this, and their psycholinguistic motivations.

We will in the following consider various stochastic extenstions of context-free grammars, and we will start by recapitulating some formal language theory.

4.1 Some Formal Language Theory

A formal grammar is a system for rewriting strings over some alphabet V.

4.1.1 Context-free Grammars

A context-free grammar G is a quadruple $\langle V_N, V_T, S, R\rangle$ where:

V_N	is a finite set of nonterminal symbols.
V_T	is a finite set of terminal symbols. Let V denote $V_N \cup V_T$.
$S \in V_N$	is a distinguished start symbol, or axiom.
R	is a finite set of productions $X \rightarrow \beta$ where $X \in V_N$ and $\beta \in V^*$.

The string $\alpha X \gamma \in V^*$ can be rewritten in one step as $\alpha\beta\gamma \in V^*$ iff $X \rightarrow \beta$ is in R. This means that one occurrence of the nonterminal symbol

X in a string is replaced by the string β. This is denoted $\alpha X \gamma \Rightarrow \alpha\beta\gamma$. In general, if a string $\phi \in V^*$ can be rewritten as the string ψ in a finite number of steps, this is denoted $\phi \Rightarrow^* \psi$. Thus $\Rightarrow^*$ is the transitive closure of the relation $\Rightarrow$. The number of rewrite steps can be zero, thus allowing $\phi \Rightarrow^* \phi$, i.e., $\Rightarrow^*$ is reflexive. If $S \Rightarrow^* \phi$, i.e., if ϕ can be derived from the start symbol S, then ϕ is said to be a *sentential form*.

$L(G)$ denotes the set of strings over V_T generated by G and is defined as $\{\omega \in V_T^* : S \Rightarrow^* \omega\}$.

A context-free grammar is said to be in *Chomsky Normal Form (CNF)*, if the productions are either of the form $X_i \rightarrow X_j X_k$, a binary production rewriting a nonterminal as two nonterminals, or of the form $X_i \rightarrow w$ where $w \in V_T$, a unary production rewriting a nonterminal as a single terminal symbol. Also allowing productions of the form $X_i \rightarrow X_j$, i.e., unary productions rewriting a nonterminal as another single nonterminal, results in grammars in *Extended Chomsky Normal Form (ECNF)*.

4.1.2 Derivations

A derivation of a string of terminal symbols ω is a sequence of string rewrites

$$S = \phi_0 \Rightarrow \ldots \Rightarrow \phi_M = \omega$$

over V where the first sentential form, ϕ_0, consists of the axiom S only and the last one, ϕ_M, is ω. This is the *leftmost derivation* iff in each step, the leftmost nonterminal symbol of the sentential form is rewritten. A leftmost derivation step is denoted $\Rightarrow_l$ and the transitive closure is denoted $\Rightarrow_l^*$.

This means that for a leftmost derivation

$$\begin{array}{rcl} \phi_m & = & \alpha X \gamma \\ \alpha & \in & V_T^* \\ X & \in & V_N \\ \gamma & \in & V^* \\ X \rightarrow \beta & \in & R \\ \phi_{m+1} & = & \alpha\beta\gamma \end{array}$$

Note that ϕ_m is uniquely decomposed into an initial (possibly empty) string of terminals, followed by a nonterminal, followed by the rest of the string. Since this specifies in each rewrite step which nonterminal of the string will be rewritten, we can represent a leftmost derivation as the sequence $r_1, \ldots, r_M$ of rules applied in each step.

We will now proceed to establish the correspondences between derivations and parse trees.

4.1.3 Trees

A tree τ is a connected directed acyclic graph. Let the set of nodes be denoted $\mathcal{N}$. There is a function ℓ that maps the nodes of the tree into some set of symbols, the so-called labelling function. The arcs in the tree depict the immediate-dominance relation $\mathcal{ID}$. If n immediately dominates n', then n is called the mother of n' and n' is called a daughter of n. The immediate-dominance relation is intransitive and irreflexive. Its transitive closure is called the dominance relation, denoted $\mathcal{D}$. This relation is transitive and antisymmetric, and thus reflexive.

There is also a partial order indicated by the horizontal position of each node — the precedence relation $\mathcal{P}$. This relation is transitive and asymmetric (and thus irreflexive). If one node dominates another, neither one of them precedes the other. Thus a pair of nodes $\langle n, n' \rangle$ can be a member of $\mathcal{P}$ only if neither $\langle n, n' \rangle$ nor $\langle n', n \rangle$ is a member of $\mathcal{D}$.

Arcs in the tree are not allowed to cross. This means that if some node precedes another node, all nodes that the former dominates precede the latter, and that the former node precedes all nodes the latter node dominates. This can be stated formally as

$$\forall n, n', n'' \;\; \mathcal{P}(n, n') \land \mathcal{D}(n, n'') \Rightarrow \mathcal{P}(n'', n')$$

and

$$\forall n, n', n'' \;\; \mathcal{P}(n, n') \land \mathcal{D}(n', n'') \Rightarrow \mathcal{P}(n, n'')$$

In addition to this, we will require that for any two distinct nodes n and n', they are related either by $\mathcal{D}$ or $\mathcal{P}$, i.e., exactly one of the following holds:

$$\mathcal{D}(n, n'),\ \mathcal{D}(n', n),\ \mathcal{P}(n, n') \text{ or } \mathcal{P}(n', n)$$

A tree has a single root, i.e., there is a unique minimal element w.r.t. $\mathcal{D}$. If τ is a tree, then $\mathcal{R}(\tau)$ denotes the root of τ. The maximal elements w.r.t. $\mathcal{D}$ are called the leaves of the tree. These must be ordered by $\mathcal{P}$, and the ordered sequence of the labels of the leaves is called the yield of the tree. $\mathcal{Y}(\tau)$ denotes the yield of the tree τ. The non-maximal elements w.r.t. $\mathcal{D}$ are called internal nodes.

4.1.4 Parse Trees

A parse tree (or derivation tree) of any string of terminal symbols ω generated by a context-free grammar G as defined above must obey the following:

- The root of the tree is labelled with the axiom S.
- All leaves of the tree are labelled with elements in V_T. More specificly, the yield of the tree must be ω.
- All internal nodes are labelled with elements in V_N.
- If there is a node labelled X in the parse tree that immediately dominates nodes $n_1, \ldots, n_K$ labelled $X_1, \ldots, X_K$ (where n_i precedes n_j for $i < j$, i.e., $\mathcal{P}(n_i, n_j)$), then there is a production in G of the form $X \rightarrow X_1, \ldots, X_K$.

$T(G)$ denotes the set of parse trees generated by the grammar G and is defined as $\{\tau : \mathcal{Y}(\tau) \in L(G)\}$.

A grammar G is said to be *finitely ambiguous* iff there is only a finite number of parse trees for any finite string in $L(G)$, i.e., if

$$\forall \omega \in L(G) \quad |\omega| < \infty \rightarrow |\{\tau : \mathcal{Y}(\tau) = \omega\}| < \infty$$

This is equivalent to requiring that no (nonterminal) symbol can be rewritten as itself in one or more rewrite step, i.e., that $X \Rightarrow^{+} X$ is impossible. We will in the following assume that the grammars are finitely ambiguous. Even so, the number of parse trees in general grows exponetially in the string length.

A partial parse tree is a parse tree where we have dropped the requirements that the root must be labelled with the axiom S and the leaves must be labelled with terminals. Partial parse trees can be combined through tree substitution. We will here define leftmost tree substitution.

> **Leftmost tree substitution:**
> Let τ and τ' be partial parse trees. Let $\tau \circ \tau'$ denote τ extended by identifying the leftmost (i.e., minimal w.r.t. $\mathcal{P}$) leaf of τ labelled with a nonterminal symbol with the root of τ'. The selected leaf node of τ is called the ***substitution site***. The label of the substitution site much match the label of the root of τ'. The dominance and precedence relations $\mathcal{D}$ and $\mathcal{P}$ are extended accordingly. Although the tree-substitution operation is not associative, let $\tau \circ \tau' \circ \tau''$ denote $((\tau \circ \tau') \circ \tau'')$.

⟨Packed parse forests will be defined in a later release.⟩

4.2 Stochastic Context-free Grammars

The simplest example of how a grammar can be augmented with a probabilistic theory is a *stochastic context-free grammar (SCFG)*; we simply add a probability distribution P on the set of productions R. A stochastic context-free grammar is thus a quintuple $\langle V_N, V_T, S, R, P\rangle$ where:

V_N	is a finite set of nonterminal symbols.
V_T	is a finite set of terminal symbols. Again V denotes $V_N \cup V_T$.
$S \in V_N$	is a distinguished start symbol, or axiom.
R	is a finite set of productions $X \rightarrow \beta$ where $X \in V_N$ and $\beta \in V^*$.
P	is a function from R to $[0,1]$ such that: $\forall X \in V_N \quad \sum_{\beta \in V^*} P(X \rightarrow \beta) = 1$

Note the conditioning on the LHS symbol of the production r_i. The following is a simple example of a SCFG:

$$\begin{array}{llll}
S & \rightarrow & NP\ VP & (1.0) \\
VP & \rightarrow & V & (0.5) \\
VP & \rightarrow & VP\ PP & (0.5) \\
NP & \rightarrow & John & (1.0) \\
V & \rightarrow & sleeps & (1.0) \\
PP & \rightarrow & outside & (1.0)
\end{array}$$

We will define string probabilities in terms of tree probabilities, which in turn will be defined in terms of derivations probabilities. One could alternatively define string probabilities dirctly in terms of derivation probabilities.

- The probability of a string is defined as the sum of the probabilities of the parse trees that have this string as a yield.
- The probability of a parse tree is defined as the probability of the corresponding leftmost derivation.

Let the leftmost derivation be represented by the sequence of productions used in each rewrite step, and let the random variable ξ_m be the production used in rewrite step m. We can thus view a leftmost derivation as a stochastic process $\xi_1, \ldots, \xi_M$, see Section 2.1.1, where the set of states are the set of productions of the grammar. This allows us to define the probability of a leftmost derivation recursively:

$$\begin{aligned}
P(\xi_1 = r_{i_1}, \ldots, \xi_{M-1} = r_{i_{M-1}}, \xi_M = r_{i_M}) \quad &= \\
= \quad & P(\xi_M = r_{i_M} \mid \xi_1 = r_{i_1}, \ldots, \xi_{M-1} = r_{i_{M-1}}) \cdot \\
& \cdot P(\xi_1 = r_{i_1}, \ldots, \xi_{M-1} = r_{i_{M-1}}) \quad = \\
= \quad & \prod_{m=1}^{M} P(\xi_m = r_{i_m} \mid \xi_1 = r_{i_1}, \ldots, \xi_{m-1} = r_{i_{m-1}})
\end{aligned}$$

To fully characterize this stochastic process, we need to specify the probabilities $P(\xi_m = r_{i_m} \mid \xi_1 = r_{i_1}, \ldots, \xi_{m-1} = r_{i_{m-1}})$ for all possible values of $m = 1, \ldots, M$ and $i_1, \ldots, i_M$. The independence assumption $P(\xi_m = r_{i_m} \mid \xi_1 = r_{i_1}, \ldots, \xi_{m-1} = r_{i_{m-1}}) = P(r_{i_m})$ is characteristic for stochastic context-free grammars. Here $P(r_i)$ is the probability of production r_i given by the stochastic grammar. This means that the probability of rewriting a nonterminal X with a production in R is independent of the previous sequence of rewrites. Thus we have

$$P(\xi_1 = r_{i_1}, \ldots, \xi_M = r_{i_M}) \quad = \quad \prod_{m=1}^{M} P(r_{i_m})$$

However mathematically sound, this approach obscures the rather intricate relationship between the sequence of derivation steps and the current leftmost nonterminal node in the partial parse trees resulting from each derivation step. Each production $X \to X_1, \ldots, X_K$ in R corresponds to a partial parse tree where the root, labelled X, immediately dominates a sequence of nodes labelled $X_1, \ldots, X_K$, and where there are no other nodes. Now, let τ_m be the partial parse tree corresponding to r_m in a leftmost derivation $r_1, \ldots, r_M$. Then $\tau_1 \circ \ldots \circ \tau_M$ is the parse tree corresponding to this leftmost derivation.

Consider the sequence of partial parse trees $\mathbf{t}_m = \tau_1 \circ \ldots \circ \tau_m$ resulting from the m first rewrite steps for $m = 1, \ldots, M$. Note that the final parse tree is simply the last element $\mathbf{t}_M$ of this sequence. It is more natural to discuss the probability of the resulting parse tree in terms of this sequence of partial parse trees, rather than in terms of the sequence of rewrite steps, although they are isomorphic. This yields the following formulation of the probability of a parse tree τ:

$$\begin{aligned}
P(\tau) \; = \; P(\mathbf{t}_M) \quad &= \quad \prod_{m=1}^{M} P(\tau_m \mid \mathbf{t}_{m-1}) \qquad (4.1) \\
\mathbf{t}_m \quad &= \quad \tau_1 \circ \ldots \circ \tau_m \qquad 1 \leq m \leq M \\
\mathbf{t}_0 \quad &= \quad \begin{bmatrix} \mathcal{N} = \{n_0\} \\ \mathcal{D} = \{\langle n_0, n_0 \rangle\} \\ \mathcal{P} = \emptyset \\ \ell(n_0) = S \end{bmatrix} \\
P(\tau_m \mid \tau_1 \circ \ldots \circ \tau_{m-1}) \quad &= \quad P(r_m \mid r_1, \ldots, r_{m-1})
\end{aligned}$$

We will use some extractor function $g(\mathbf{t}_m)$ to extract the relevant information from the partial parse tree $\mathbf{t}_m$ when estimating the probabilities $P(\tau_{m+1} \mid \mathbf{t}_m)$. This means that we will only look at the portions of the tree that we think are relevant for estimating this probability:

$$P(\tau_{k+1} \mid \mathbf{t}_m) \quad \approx \quad P(\tau_{k+1} \mid g(\mathbf{t}_m)) \qquad (4.2)$$

This means that the set of possible partial parse trees is partitioned into a set of equivalence classes, each of which is associated with a probability distribution over the set of elementary trees that constitute the set of productions.

For stochastic context-free grammars, the interesting information is the label of the leftmost nonterminal of the yield of the tree, which is required to match the LHS of the production used in the next rewrite step. Thus, the function $g(\tau)$ is $\mathcal{L}(\mathcal{Y}(\tau))$, where $\mathcal{Y}(\tau)$ is the yield of τ and $\mathcal{L}(\phi)$ returns the leftmost nonterminal symbol of the string ϕ.

The probabilities assigned to analyses are preserved under transformation to and from CNF:

- Productions of the form $A \rightarrow BCDE \quad (p)$ with $A, B, C, D, E \in V_N$ are replaced with a set of productions of the form $A \rightarrow BC' \quad (p)$, $C' \rightarrow CD' \quad (1.0)$ and $D' \rightarrow DE \quad (1.0)$. This introduces new nonterminals C', D', E' which do not figure elsewhere in the transformed grammar.
- Productions of the form $A \rightarrow B \quad (p_1)$ are removed. (Alternatively Extended Chomsky Normal Form is employed.) For each production of the form $X \rightarrow \alpha A \beta \quad (p_2)$ with $\alpha, \beta \in V^*$, a new production $X \rightarrow \alpha B \beta \ (p_1 \cdot p_2)$ is introduced.
- Productions of the form $A \rightarrow \epsilon \ (p_1)$, where ϵ is the empty string, are removed. For each production of the form $X \rightarrow \alpha A \beta \quad (p_2)$, a new production $X \rightarrow \alpha\beta \ (p_1 \cdot p_2)$ is introduced.
- Productions of the form $X \rightarrow \alpha a \beta \ (p)$ with $a \in V_T$ are replaced with the production pair $X \rightarrow \alpha A' \beta \ (p)$ and $A' \rightarrow a \ (1.0)$.

4.3 A Parser for SCFGs

We will next adapt the Viterbi algorithm discussed in Section 2.1.6 to parsing with stochastic context-free grammars. This presentation is inspired by [Wu 1995] and we will use a variant of the Cocke-Younger-Kasami (CYK) parsing algorithm, see [Aho & Ullman 1972], pp. 314–320. The parser described in this section finds the most probable parse (MPP) of a given input string ω in $O(N^3T^3)$ time and $O(NT^2)$ space, where $N = |V_N|$ is the size of the nonterminal alphabet and T is the string length.

Assume that $V_N = \{X_1, \ldots, X_N\}$, where $S = X_1$, and that $\omega = w_1, \ldots, w_T$. Assume further that the grammar is in Chomsky Normal Form, see Section 4.1.1. We will maintain an accumulator $\delta_n(X_i)$ for each nonterminal symbol X_i in the grammar and each node n in the tree.

In any parse tree, a node n uniquely determines a substring, which in turn specifies a pair of string positions (s, t). Let $\mathbf{w}_{st}$ denote $w_{s+1}, \ldots, w_t$, and let $\mathbf{w}_{st}$ be the yield of the subtree rooted in n. This function is not necessarily a surjunction, i.e., there may be string positions that do not correspond to any node in the parse tree. Nor is it necessarily an injunction, i.e., several nodes may be mapped to the same pair of string postitions. We can however define a partial inverse function by choosing the node closest to the root. Thus τ is the partial parse tree whose yield is $\mathbf{w}_{st}$ and whose root node n is minimal w.r.t. $\mathcal{D}$. More formally, $\mathcal{Y}(\tau) = \mathbf{w}_{st} \ \wedge \ \mathcal{R}(\tau) = n \ \wedge \ (\mathcal{Y}(\tau') = \mathbf{w}_{st} \ \wedge \ \mathcal{R}(\tau') = n' \ \rightarrow \ \mathcal{D}(n, n'))$. The nodes that do not correspond to any pair of string positions will thus be the daughters of unary productions. For a grammar in CNF, these nodes are the leaves of the tree, which will not figure directly in the algorithm below.

We thus have the set of accumulators

$$\delta_{s,t}(X_i) \qquad 1 \leq i \leq N, 1 \leq s < t \leq T$$

These are defined as the maximum probability of any partial parse tree τ spanning the substring $\mathbf{w}_{st} = w_{s+1}, \ldots, w_t$ given that the root of the parse tree is labelled X_i:

$$\delta_{s,t}(X_i) \quad = \quad \max_{\tau:\mathcal{Y}(\tau)=\mathbf{w}_{st}} P(\tau \mid \ell(\mathcal{R}(\tau)) = X_i)$$

The probability of the most probable parse tree with $\omega = \mathbf{w}_{0T}$ as a yield is thus

$$\delta_{0,T}(S) \quad = \quad \max_{\tau:\mathcal{Y}(\tau)=\omega} P(\tau \mid \ell(\mathcal{R}(\tau)) = S)$$

and the parse tree itself is

$$\operatorname*{argmax}_{\tau \,:\, \mathcal{Y}(\tau) = \omega} P(\tau \mid \ell(\mathcal{R}(\tau)) = S)$$

Let $p_{i \to jk}$ denote $P(X_i \to X_j X_k \mid X_i)$ and $p_{i \to w}$ denote $P(X_i \to w \mid X_i)$. We can construct this parse tree bottom-up:

1. **Initialization**
   $\forall i, t : 1 \leq i \leq N, 1 \leq t \leq T$

$$\delta_{t-1,t}(X_i) \quad = \quad p_{i \to w_t}$$

2. **Recursion**
   $\forall i, r, t : 1 \leq i \leq N, 1 \leq r < t \leq T$

$$\delta_{r,t}(X_i) \quad = \quad \max_{\substack{1 \leq j \leq N \\ 1 \leq k \leq N \\ r < s < t}} \quad p_{i \to jk}\, \delta_{r,s}(X_j)\, \delta_{s,t}(X_k)$$

$$\begin{bmatrix} \iota_{r,t}(X_i) \\ \kappa_{r,t}(X_i) \\ \sigma_{r,t}(X_i) \end{bmatrix} \quad = \quad \operatorname*{argmax}_{\substack{1 \leq j \leq N \\ 1 \leq k \leq N \\ r < s < t}} \quad p_{i \to jk}\, \delta_{r,s}(X_j)\, \delta_{s,t}(X_k)$$

3. **Reconstruction**
   $n = (s, t)$

$$\begin{aligned}
\text{Left}(n) \quad &= \quad \begin{cases} \text{Nil} & \text{if } t - s \leq 2 \\ (s, \sigma_{s,t}(\ell(n))) & \text{otherwise} \end{cases} \\
\text{Right}(n) \quad &= \quad \begin{cases} \text{Nil} & \text{if } t - s \leq 2 \\ (\sigma_{s,t}(\ell(n)), t) & \text{otherwise} \end{cases} \\
\ell(\text{Left}(n)) \quad &= \quad \iota_{s,t}(\ell(n)) \\
\ell(\text{Right}(n)) \quad &= \quad \kappa_{s,t}(\ell(n))
\end{aligned}$$

In the initialization step, the probabilities of the lexical productions are assigned to the nonterminals at the nodes immediately dominating the input string. In the recursion step, we are interested in calculating

$$\delta_{r,t}(X_i) \quad = \quad \max_{\tau:\mathcal{Y}(\tau)=\mathbf{w}_{rt}} P(\tau \mid \ell(\mathcal{R}(\tau)) = X_i)$$

from

$$\delta_{r,s}(X_j) = \max_{\tau':\mathcal{Y}(\tau')=\mathbf{w}_{rs}} P(\tau' \mid \ell(\mathcal{R}(\tau')) = X_j)$$

and

$$\delta_{s,t}(X_k) = \max_{\tau'':\mathcal{Y}(\tau'')=\mathbf{w}_{st}} P(\tau'' \mid \ell(\mathcal{R}(\tau'')) = X_k)$$

for all possible intermediate points s in the string $\mathbf{w}_{rt}$ and all possible productions $X_i \rightarrow X_j X_k$. We note that

$$\begin{aligned}
\max_{\tau:\mathcal{Y}(\tau)=\mathbf{w}_{rt}} P(\tau \mid \ell(\mathcal{R}(\tau)) = X_i) &= \\
&= \max_{j,k,s}[P(X_i \rightarrow X_j X_k \mid X_i) \cdot \max_{\tau':\mathcal{Y}(\tau')=\mathbf{w}_{rs}} P(\tau' \mid \ell(\mathcal{R}(\tau')) = X_j) \cdot \\
&\quad \cdot \max_{\tau'':\mathcal{Y}(\tau'')=\mathbf{w}_{st}} P(\tau'' \mid \ell(\mathcal{R}(\tau'')) = X_k)]
\end{aligned}$$

which gives the recursion formula.

The complexity of this probabilistic parsing scheme is $O(N^3T^3)$ in time and $O(NT^2)$ in space. It can easily be generalized to the case of a general stochastic context-free grammar, not necessarily in Chomsky Normal Form. $\langle$**Ref?**$\rangle$

$\langle$**The relationship between the HMM trellis, the δ-variables and a packed parse forest will be revealed in a later release.**$\rangle$

This section described a probabilistic version of the CYK parsing algorithm. For a very nice presentation of a probabilistic Earley parsing scheme, we are happy to be able to refer to [Stolcke 1995].

4.4 Parameter Estimation for SCFGs

We now turn to the problem of estimating the production probabilities $p_{i \rightarrow jk} = P(X_i \rightarrow X_j X_k \mid X_i)$ and $p_{i \rightarrow w} = P(X_i \rightarrow w \mid X_i)$ of a SCFG. If we have access to an annotated corpus of parse trees, we can calculate these probabilities directly from the observed relative frequencies. It is however in this case necessary to deal appropriately with the problem of sparse data, as discussed in Section 2.4.

If we only have a corpus of unannotated text, we have to resort to other methods. Here we will define a set of recurrence equations that can be used to iteratively improve our estimates of these probabilities. By doing this right, we can get some guarantees that each iteration will yield us a better set of estimates.

One method is to first find all vaild parse trees for each sentence in the corpus. In the first iteration step, we assign a uniform distribution to the set of parse trees resulting from each sentence. We then perform a frequency count of the productions, conditioned on the LHS symbol, each time weighting with the probability of the parse tree. This will determine a new probability distribution for the productions. Using this, we can estimate a new probabilities for the parse trees conditional on the corresponding sentence. This in turn allows us to estimate the production probabilities from the frequency counts conditioned on the LHS symbol, again weighting with the previous estimate of the probability of the parse tree. This in turn determines a new estimate of the tree probabilities, and we can iterate this procedure until we tire.

A problem with this method is that the number of possible parse trees of an input string is exponential in the string length. Another method for reestimating the production probabilities that avoids this problem is called the *inside-outside algorithm*. The basic idea of the inside-outside algorithm is to use the current estimates of the production probabilities to estimate from the training corpus the expected frequencies of certain other derived quantities that depend on the production probabilities, and then recompute new estimates of the production probability using these derived quantities. The goal is to find a set of production probabilities that (locally) maximizes the likelihood of generating the training corpus.

4.4.1 The Inside and Outside Variables

We will use two sets of accumulators, namely the inside probabilities and the outside probabilities: [1]

- The inside probability $I_i^\omega(s,t)$ estimates the probability

$$P(X_i \Rightarrow^* \mathbf{w}_{st} \mid X_i)$$

  of a given nonterminal X_i deriving the substring $\mathbf{w}_{st} = w_{s+1}, \ldots, w_t$.

- The outside probability $O_i^\omega(s,t)$ estimates the probability

$$P(S \Rightarrow^* \mathbf{w}_{0s} X_i \mathbf{w}_{tT} \mid S)$$

  of deriving the string $\mathbf{w}_{0s} X_i \mathbf{w}_{tT} = w_1, \ldots, w_s X_i w_{t+1}, \ldots, w_T$ from the axiom S. We will in the following omit the conditioning on S to aid readability.

Nice figure needed here!

We have the following relationships between the inside, outside and production probabilities:

1. **Inside-variable initialization**
   $\forall i,t : 1 \le i \le N, 1 \le t \le T$

$$I_i^\omega(t-1,t) \quad = \quad p_{i \to w_t}$$

2. **Inside-variable recursion**
   $\forall i,r,t : 1 \le i \le N, 1 \le r < t \le T$

$$I_i^\omega(r,t) \quad = \quad \sum_{j,k=1}^{N} \sum_{r<s<t} p_{i \to jk} \cdot I_j^\omega(r,s) \cdot I_k^\omega(s,t)$$

3. **Outside-variable recursion**
   $\forall i,r,t : 1 \le i \le N, 1 \le r < t \le T$

$$\begin{aligned} O_i^\omega(r,t) \quad = \quad \sum_{j,k=1}^{N} & \sum_{s=0}^{r-1} p_{j \to ki} \cdot O_j^\omega(s,t) \cdot I_k^\omega(s,r) \quad + \\ & + \quad \sum_{s=t+1}^{T} p_{j \to ik} \cdot O_j^\omega(r,s) \cdot I_k^\omega(t,s) \end{aligned}$$

[1] In an extension to this scheme, Larson introduces the so-called far-side probabilities.

Comparing this with the parsing algorithm of the previous section, we see that the only difference between the $\delta_{r,t}(i)$ and $I_i^\omega(r,t)$ accumulators, apart from the fact that the argument and subscripts have been swapped, is that the latter are summed in the recursion step, whereas the former are maximized, in both cases over the same set of quantities.

The recursion formula for the inside variables follows from

$$P(X_i \Rightarrow^* \mathbf{w}_{rt} \mid X_i) \;=\; \sum_{j,k=1}^{N} P(X_i \Rightarrow X_j X_k \Rightarrow^* \mathbf{w}_{rt} \mid X_i) \;=$$

$$\sum_{j,k=1}^{N} \sum_{r<s<t} P(X_i \Rightarrow X_j X_k \mid X_i) \cdot P(X_j \Rightarrow^* \mathbf{w}_{rs} \mid X_j) \cdot P(X_k \Rightarrow^* \mathbf{w}_{st} \mid X_k)$$

The recursion formula for the outside variables follows from

$$\begin{aligned}
& P(S \Rightarrow^* \mathbf{w}_{0r} X_i \mathbf{w}_{tT}) \;= \\
\sum_{j,k=1}^{N} \quad & \sum_{s=0}^{r-1} P(S \Rightarrow^* \mathbf{w}_{0s} X_j \mathbf{w}_{tT} \Rightarrow^* \mathbf{w}_{0s} X_k X_i \mathbf{w}_{tT} \Rightarrow^* \mathbf{w}_{0s} \mathbf{w}_{sr} X_i \mathbf{w}_{tT}) + \\
& + \sum_{s=t+1}^{T} P(S \Rightarrow^* \mathbf{w}_{0r} X_j \mathbf{w}_{sT} \Rightarrow^* \mathbf{w}_{0r} X_i X_k \mathbf{w}_{sT} \Rightarrow^* \mathbf{w}_{0r} X_i \mathbf{w}_{ts} \mathbf{w}_{sT}) \;= \\
\sum_{j,k=1}^{N} \quad & \sum_{s=0}^{r-1} P(S \Rightarrow^* \mathbf{w}_{0s} X_j \mathbf{w}_{tT}) \cdot P(X_j \Rightarrow X_k X_i \mid X_j) \cdot P(X_k \Rightarrow^* \mathbf{w}_{sr} \mid X_k) + \\
& + \sum_{s=t+1}^{T} P(S \Rightarrow^* \mathbf{w}_{0r} X_j \mathbf{w}_{sT}) \cdot P(X_j \Rightarrow X_i X_k \mid X_j) \cdot P(X_k \Rightarrow^* \mathbf{w}_{ts} \mid X_k)
\end{aligned}$$

We see that the complexity is $O(N^3 T^3)$ for calculating both all inside variables and all outside variables.

4.4.2 Deriving the Reestimation Equations

The basic idea is to estimate $p_{i\to w}$ from $\dfrac{\hat{P}(X_i \to w)}{\hat{P}(X_i)}$ and $p_{i\to jk}$ from $\dfrac{\hat{P}(X_i \to X_j X_k)}{\hat{P}(X_i)}$:

$$\begin{aligned}
\bar{p}_{i\to w} &= \hat{P}(X_i \to w \mid X_i) = \frac{\hat{P}(X_i \to w)}{\hat{P}(X_i)} \\
\bar{p}_{i\to jk} &= \hat{P}(X_i \to X_j X_k \mid X_i) = \frac{\hat{P}(X_i \to X_j X_k)}{\hat{P}(X_i)}
\end{aligned}$$

We will thus need the quantities $\hat{P}(X_i)$, $\hat{P}(X_i \to w)$ and $\hat{P}(X_i \to X_j X_k)$.

Deriving $\hat{P}(X_i)$

The string probability $P(\omega)$ is the probability of deriving ω from the axiom $S = X_1$:

$$P^\omega = P(\omega) = P(S \Rightarrow^* \omega) = I_1^\omega(0,T)$$

The joint probability of deriving ω and the nonterminal X_i figuring (at least once) in some derivation of ω is

$$P_i^\omega \;=\; P(S \Rightarrow^* \phi_1 X_i \phi_2 \Rightarrow^* \omega) \;=$$

$$
\begin{aligned}
&= \sum_{0 \leq r < s \leq T} P(S \Rightarrow^* \mathbf{w}_{0r} X_i \mathbf{w}_{sT} \Rightarrow^* \omega) \quad = \\
&= \sum_{0 \leq r < s \leq T} P(S \Rightarrow^* \mathbf{w}_{0r} X_i \mathbf{w}_{sT}) \cdot P(X_i \Rightarrow^* \mathbf{w}_{rs} \mid X_i) \quad = \\
&= \sum_{0 \leq r < s \leq T} O_i^\omega(r,s) \cdot I_i^\omega(r,s)
\end{aligned}
$$

So the probability $P(X_i)$ of X_i figuring in some derivation (of any string) can be estimated by normalizing with the string probability P^ω and averaging this quantity over the corpus W:

$$
\hat{P}(X_i) \quad = \quad \frac{1}{|W|} \sum_{\omega \in W} \frac{P_i^\omega}{P^\omega}
$$

where $|W|$ is the corpus size, i.e., the number of sentences. The complexity of calculating this is $\sum_{\omega \in W} O(N \cdot T_\omega^2)$.

Deriving $\hat{P}(X_i \rightarrow w)$

Likewise, the joint probability of deriving ω and the (lexical) production $X_i \rightarrow w$ figuring in some derivation of ω is

$$
\begin{aligned}
&P(S \Rightarrow^* \phi_1 X_i \phi_2 \Rightarrow \phi_1 w \phi_2 \Rightarrow^* \omega) \quad = \\
&= \sum_{1 \leq t \leq T, w_t = w} P(S \Rightarrow^* \mathbf{w}_{0t-1} X_i \mathbf{w}_{tT} \Rightarrow^* \omega) \quad = \\
&= \sum_{1 \leq t \leq T, w_t = w} P(S \Rightarrow^* \mathbf{w}_{0t-1} X_i \mathbf{w}_{tT}) \cdot P(X_i \Rightarrow^* w_t \mid X_i) \quad = \\
&= \sum_{1 \leq t \leq T, w_t = w} O_i^\omega(t-1,t) \cdot p_{i \rightarrow w}
\end{aligned}
$$

The probability $P(X_i \rightarrow w)$ of applying the production $X_i \rightarrow w$ in some derivation can similarily be estimated by

$$
\hat{P}(X_i \rightarrow w) \quad = \quad \frac{1}{|W|} \sum_{\omega \in W} \frac{\displaystyle\sum_{1 \leq t \leq T, w_t = w} O_i^\omega(t-1,t) \cdot p_{i \rightarrow w}}{P^\omega}
$$

The complexity of calculating this is $\sum_{\omega \in W} O(N \cdot T_\omega)$ if the lexical entries can be accessed in constant time.

Deriving $\hat{P}(X_i \rightarrow X_j X_k)$

Finally, the joint probability of deriving ω and the production $X_i \rightarrow X_j X_k$ figuring in some derivation of ω is

$$
\begin{aligned}
&P(S \Rightarrow^* \phi_1 X_i \phi_2 \Rightarrow \phi_1 X_j X_k \phi_2 \Rightarrow^* \omega) \quad = \\
&= \sum_{0 \leq r < t \leq T} P(S \Rightarrow^* \mathbf{w}_{0r} X_i \mathbf{w}_{tT} \Rightarrow \mathbf{w}_{0r} X_j X_k \mathbf{w}_{tT} \Rightarrow^* \omega) \quad = \\
&= \sum_{0 \leq r < s < t \leq T} P(S \Rightarrow^* \mathbf{w}_{0r} X_i \mathbf{w}_{tT}) \cdot P(X_i \rightarrow X_j X_k \mid X_i) \cdot \\
&\qquad \cdot P(X_j \Rightarrow^* \mathbf{w}_{rs} \mid X_j) \cdot P(X_k \Rightarrow^* \mathbf{w}_{st} \mid X_k) \quad = \\
&= \sum_{0 \leq r < s < t \leq T} O_i^\omega(r,t) \cdot p_{i \rightarrow jk} \cdot I_j^\omega(r,s) \cdot I_k^\omega(s,t)
\end{aligned}
$$

Analogously, the probability $P(X_i \to X_j X_k)$ of applying the production $X_i \to X_j X_k$ in some derivation can be estimated by

$$\hat{P}(X_i \to X_j X_k) = \frac{1}{|W|} \sum_{\omega \in W} \frac{\sum_{0 \leq r < s < t \leq T} O_i^\omega(r,t) \cdot p_{i \to jk} \cdot I_j^\omega(r,s) \cdot I_k^\omega(s,t)}{P^\omega}$$

The complexity of calculating this is $\sum_{\omega \in W} O(N^3 \cdot T_\omega^3)$.

The Reestimation Equations

Assembling all this will yield us the final set of recurrence equations for the probabilities $p_{i \to w}$ and $p_{i \to jk}$ just as in the case of estimating the model parameters of an HMM, cp. Section 2.1.7:

$$\bar{p}_{i \to w} = \hat{P}(X_i \to w \mid X_i) = \frac{\hat{P}(X_i \to w)}{\hat{P}(X_i)} = $$

$$= \frac{\sum_{\omega \in W} \frac{\sum_{1 \leq t \leq T, w_t = w} O_i^\omega(r,s) \cdot p_{i \to w}}{P^\omega}}{\sum_{\omega \in W} \frac{P_i^\omega}{P^\omega}}$$

$$\bar{p}_{i \to jk} = \hat{P}(X_i \to X_j X_k \mid X_i) = \frac{\hat{P}(X_i \to X_j X_k)}{\hat{P}(X_i)} = $$

$$= \frac{\sum_{\omega \in W} \frac{\sum_{0 \leq r < s < t \leq T} O_i^\omega(r,t) \cdot p_{i \to jk} \cdot I_j^\omega(r,s) \cdot I_k^\omega(s,t)}{P^\omega}}{\sum_{\omega \in W} \frac{P_i^\omega}{P^\omega}}$$

Each iteration step requires $\sum_{\omega \in W} O(N^3 T_\omega^3)$ calculations, which is clearly polynomial in corpus size.

The method is extended to general context-free grammars, not necessarily in Chomsky Normal Form, in [Kupiec 1992] and to partially bracketed training text in [Pereira & Schabes 1992]

As we have seen, the string probability $P(S \Rightarrow^* \omega) = I_1^\omega(0,T)$ can also be calculated in $O(N^3 T^3)$ time and $O(NT^2)$ space. Unfortunately, finding the most probable sentence (MPS) of a word graph, which is the normal output from a speech recognizer, is NP hard in string length, see [Sima'an 1996].

4.5 Adding Probabilistic Context to SCFGs

Consider the following SCFG

$$
\begin{array}{lcll}
S & \rightarrow & NP\ VP & (0.70) \\
 & \vdots & & \\
VP & \rightarrow & V\ NP\ NP & (0.10) \\
 & \vdots & & \\
NP & \rightarrow & Pron & (0.20) \\
NP & \rightarrow & Det\ \bar{N} & (0.20) \\
NP & \rightarrow & NP\ PP & (0.30) \\
 & \vdots & &
\end{array}
$$

It will assign

> "(Show) (me) (all fligths to Boston from Atlanta leaving before ten AM that serve breakfast and stop over in Baltimore ...)"

the same probability as

> "(Show) (all fligths to Boston from Atlanta leaving before ten AM that serve breakfast and stop over in Baltimore ...) (me)"

So instead of

$$P(NP \rightarrow Pron \mid NP)$$

it would be better to use

$$P(NP \rightarrow Pron \mid VP \rightarrow V \cdot NP\ NP)$$

Likewise, the stochastic model as it stands is too blunt to be of any use when disambiguating for example noun-noun compounds.

It is, theoretically, very simple to add more contextual information to the probability distributions by looking further up the tree at the substitution site. Assume for convenience that the grammar is in CNF. To take not only the LHS label into account, but the entire mother dotted item:

- Replace each occurrence of a RHS nonterminal with a new unique nonterminal.

- Multiply out the rules by creating a new rule for each matching old LHS.

This creates a new SCFG G_1 that effectively yields the probabilities of the old productions of grammar G_0 conditional on the mother production and the expanded nonterminal of its RHS. This can be repeated ad infinum to yield stochastic models G_p with increasingly more contextual information. Note again the similarity to HMMs, this time how Nth-order HMMs can be represented as first-order HMMs by expanding out sequences of states into distinct new states.

The number of nonterminals $|G_1|$ in the new grammar G_1 equals the number of distinct occurrences of nonterminals in the RHSs of the productions of the original grammar G_0, which is exactly $2|R_0|$, two times the number of original productions. This is maximally $2N^3$, where $N = |G_0|$ is the number of nonterminals in G_0. Thus, the most probable parse can be found in $O(N^9T^3)$ time and $O(N^3T^2)$ space. The algorithm is thus still polynomial both in N and T.

In the general case, the parsing complexity is $O(N^{6p+3}T^3)$ in time and $O(N^{2p+1}T^2)$ in space for p generalization steps. The two key general reccurence equations are:

$$\begin{array}{rcl} |G_{p+1}| & = & 2|R_p| \\ |R_{p+1}| & \leq & 2N^2|R_p| \;\; \leq \;\; (2N^2)^p|R_0| \end{array}$$

The first equality can be established as follows: Each RHS occurrence in the grammar G_p of the previous step yields a new symbol in the resulting grammar G_{p+1}. This is simply the number of rules in the previous step, $|R_p|$, times two.

For the following inequalities, we first observe that the number of productions in G_{p+1} that each production in G_p gives rise to equals the number of occurrences of the particular LHS symbol in the RHSs of the productions of G_p.

Assume that no symbol occurs more than B times in the RHSs of G_p. Then this is true also for G_{p+1}. We establish this claim as follows: When we construct G_{p+1} from G_p we replace each occurrence of a grammar symbol in the RHSs of the productions with a new symbol. This means that before LHS expansion, each RHS occurrence is a unique symbol. The number of occurences of any symbol will thus be the number of times the RHS containing it is duplicated by LHS expansion. We need exactly one version of each production for each occurence of the LHS symbol in a RHS of G_p. But this is bounded by B.

So if there is an upper bound B to the number of occurences of any grammar symbol in G_0, this upper bound is valid also for $G_p : p = 1, 2\ldots$ by induction. So each production in G_p is expanded into maximally B productions in G_{p+1}. No symbol can occur more than $2N^2$ times in the RHSs of G_0, so $B = 2N^2$ yields the inequalities.

Starting with a maximally dense grammar G_0 we have $|R_0| = N^3$. We thus have $|G_p| = 2^pN^{2p-2}N^3 = 2^pN^{2p+1}$. This means that the parsing complexity is $O(|G_p|^3T^3)$ in time and $O(|G_p|T^2)$ in space, which establishes the claim. Note however that the complexity is exponential in the number of generalization steps.

4.6 Theoretical Probability Losses

A somewhat surprising fact is that the string probabilities assigned by an SCFG do not necessarily sum to one over the set of finite strings. The reason for this is that each string probability is defined as the sum of the derivation probabilities of the string. Now, the SCFG will assign probabilities also to nonterminating derivations. For example, the probability of a terminating derivation is zero in the following SCFG

$$S \rightarrow aS \quad (1)$$

since there are none.

This is in some sense a degenerate case, since the probability of the one nonterminating derivation $S \Rightarrow aS \Rightarrow aaS\ldots$ is nonzero. But even for grammars where the probability of each nonterminating derivation is zero, the sum of these probabilities need not be, as in the following case:

Proposition

Consider the SCFG

$$\begin{array}{ll} S \to a & (p) \\ S \to SS & (q) \end{array}$$

where $p, q \geq 0$ and $p + q = 1$.

Then

$$\sum_{n=1}^{\infty} P(a^n) = \min\left(1, \frac{p}{q}\right)$$

Proof

The total number of nonterminal nodes in any parse tree for a^n is $2n - 1$. There are n preterminal nodes, i.e., S rewritten using $S \to a$ and thus $n - 1$ Ss that are rewritten using $S \to SS$. This means that the string probability of a^n is $p^n \cdot q^{n-1}$ times the number of different parse trees for a^n. The number of different parse trees equals the number of different binary bracketings of a string with n terminals. The recurrence equation is

$$f(n) = \sum_{k=1}^{n-1} f(k) \cdot f(n-k)$$

With initial value $f(1) = 1$, the solution is the sequence of Catalan numbers

$$\frac{\binom{2n-2}{n-1}}{n}$$

Thus, the string probability of a^n is

$$P(a^n) = f(n) \cdot p^n \cdot q^{n-1}$$

and the sum of the string probabilities is

$$x = \sum_{n=1}^{\infty} P(a^n) = \sum_{n=1}^{\infty} f(n) \cdot p^n \cdot q^{n-1} = p \cdot \sum_{n=1}^{\infty} f(n) \cdot r^{n-1} = p \cdot g(r)$$

where $r = p \cdot q = p \cdot (1-p) = (1-q) \cdot q$.

From the grammar we can establish that

$$x = p + q \cdot x^2$$

with solutions

$$x = \begin{cases} 1 \\ \frac{p}{q} \end{cases}$$

and thus

$$g(r) = \begin{cases} \frac{1}{q} \\ \frac{1}{p} \end{cases}$$

Let y be defined by

$$\begin{aligned} g(r) &= \frac{1}{y} \\ r &= y \cdot (1-y) \geq 0 \end{aligned}$$

Then

$$y = \frac{1}{2} \pm \sqrt{\frac{1}{4} - r}$$

Since $g(r)$ is continuous on $[0, \frac{1}{4}]$, we have

$$g(0) = \lim_{\epsilon \to 0^+} \sum_{n=1}^{\infty} f(n) \cdot \epsilon^{n-1} = f(1) = 1$$

and thus for all $r : 0 \leq r \leq \frac{1}{4}$

$$\begin{aligned} y &= \frac{1}{2} + \sqrt{\frac{1}{4} - r} = \frac{1}{2} + \sqrt{\frac{1}{4} - y + y^2} = \\ &= \frac{1}{2} + \sqrt{(\frac{1}{2} - y)^2} = \frac{1}{2} + \left|\frac{1}{2} - y\right| = \max(1 - y, y) = \max(p, q) \end{aligned}$$

This means that

$$g(r) = \min\left(\frac{1}{p}, \frac{1}{q}\right)$$

and thus

$$x = p \cdot g(r) = \min\left(1, \frac{p}{q}\right)$$

In the general case, let $C = c_i$ be the vector of the probability of each nonterminal X_i not terminating. Let A be the matrix of the quantities

$$a_{ij} = \sum_k P(R_k \mid X_i) \cdot n_j(R_k)$$

where $n_j(R)$ is the number of times the symbol X_j occurs in the RHS of R.

Then $C \leq AC$ and for any c_i to be non-zero, i.e., $c_i > 0$, we see by diagonalizing with T that in order for $C \leq AC$ to be true, we need $TC \leq TAT^{-1}TC$, which in turn means that for some eigenvalue λ_k of $\Lambda = TAT^{-1}$, we must have $\lambda_k > 1$. Some nice spectral theorem guarantees that, under suitable conditions, such a matrix T exists, and that the eigenvalues of A are the same as those of TAT^{-1}. Using a continuity argument on c_i as functions of the model parameters $P(R_k \mid X_j)$, we find that a non-zero c_i requires an eigenvalue greater than one.

In the example above, $A = [2q]$, and the requirement is $2q > 1$, or $q > \frac{1}{2}$.

Check out: Wetherell 1980, Booth and Thompson 1973, Fu's "Syntactic Pattern Recognition".

4.7 Stochastic Tree-Substitution Grammars

We will consider another way in which a context-free grammar can be augmented with a probabilistic theory. Reconsider the definition of the probability of a leftmost derivation, Eq. (4.1). Assume that we allow the set of elementary parse trees used here to consist of trees of depth greater than one. We will thus replace the set of trees corresponding to the set of productions R with a more general set of elementary trees. Let us further assume that the extractor function $g(\tau)$ of Eq. 4.2 is the same as for SCFGs, i.e., $\mathcal{L}(\mathcal{Y}(\tau))$. This means that the probability of an elementary tree being substituted at the leftmost nonterminal leaf is conditional only on the label of this node.

Since we are working with a general set of elementary trees, we must take into account that we no longer have a one-to-one correspondence between leftmost derivations and parse trees — Now there may exist several leftmost derivations of a parse tree. In order to get the probability of a parse tree, we will sum over the set of possible leftmost derivations of it. The probability of a string will still be the sum of the probabilities of the parse trees that have this string as a yield.

A *stochastic tree-substitution grammar (STSG)* thus consists of a quintuple $\langle V_N, V_T, S, R, P\rangle$ where

V_N	is a finite set of nonterminal symbols.
V_T	is a finite set of terminal symbols. As usual, $V = V_N \cup V_T$.
$S \in V_N$	is a distinguished start symbol, or axiom.
R	is a finite set of partial parse trees, so-called elementary trees.
P	is a function from R to $[0,1]$ such that: $\forall X \in V_N \;\; \sum_{\tau:\ell(\mathcal{R}(\tau))=X} P(\tau) = 1$

For these grammars, finding the most probable parse (MPP) is NP hard, [Sima'an 1996]. One can however find the most probable derivation (MPD) in $O(N^3|R|)$, time see [Sima'an 1995]. This is done by first using a CYK parser to construct a packed parse forest from the underlying CFG, and then performing Viterbi search, with a few clever optimizations, on the resulting packed parse forest, now taking the probabilities of the STSG into account. Similar techniques allow using Monte-Carlo techniques to estimate the MPP by sampling from the set of derivations, see [Bod 1995b].

For a nice description of STSGs and a related paradigm referred to as data-oriented parsing (DOP), originally due to Remko Scha, see [Bod 1995a] and [Bod 1995b]. Tree-substitution grammars are a special case of tree-adjoining grammars, see [Joshi *et al* 1975]. In addition to tree substitution, the latter also allow the adjoining operation. Stochastic lexicalized tree-adjoining grammars are described in [Schabes 1992].

$\langle$**To be continued.**$\rangle$

4.8 Stochastic History-Based Grammars

Again, reconsider the definition of the probability of a leftmost derivation, Eq. (4.1). Assume that we in Eq. 4.2 instead use a more general extractor function $g(\tau)$ than $\mathcal{L}(\mathcal{Y}(\tau))$, the one used for SCFGs and STSG. The typical type of function $g(\tau)$ will be a decision tree, which is used to arrive at a probability distribution by asking a sequence of questions about the parse tree τ, or equivalently, the about sequence of derivation steps in the leftmost derivation, see [Black *et al* 1993]. The decision-tree techniques are described in [Bahl *et al* 1989], which is well worth reading.

We realize that we can in general view any derivation of a parse tree as a sequence of disambiguation decisions, and that we can define the derivation probability based on the decision history. If there is a ono-to-one correspondence between the derivation and the restulting parse tree, we can use this to define the parse probability. We can again use a decision tree to "ask 20 (binary) questions" about the derivation history, i.e., about the previous disambiguation decisions made, and thus determine a probability distribution on the set of disambiguation actions currently available.

This is exactly what is done in the work reported in [Magerman 1995], one of the most important, and most readable, articles on probabilistic parsing to date. One of the nice characteristics of the scheme is that it allows lexical information to be used for structural disambiguation. Due to the complex conditionings of the probabilistic language model, it uses a disambiguation scheme distinct from Viterbi search: It first finds a reasonably high-probability complete parse, and then switches to breadth-first search, pruning search branches with an accumulated probability below that of the complete parse found.

⟨**To be continued.**⟩

4.9 Lexicalization of Stochastic Grammars

Lexicalization of stochastic grammars has become a major research topic. For the time being, approaches modeling a combination of phrase structure and lexical dependency structure show the best results, cf. [Eisner 1996] and [Collins 1997]. There are other approaches like stochastic lexicalized tree adjoining grammar ([Schabes 1992]), and history-based grammar using lexical information in the derivation history ([Black *et al* 1993], [Magerman 1995]).

An early arpproach on stochastic dependency grammar is probabilistic Link Grammar ([Lafferty *et al.* 1992]), a grammar that models n-gram lexical dependencies. An examole for a more recent approach on n-gram lexical dependencies is presented in [Collins 1996].

4.9.1 Stochastic Dependency Grammar and Related Approaches

In classical dependency grammar [Tesnière 1959], syntactic structure is determined on the basis of dependency (head-modifier) relations between words. Sentences and phrases are thus represented by the set of dependency relations that hold between their lexical elements.

Dependency structures can be derived from context-free grammars by mapping trees into dependency structures. This is an advantage as the statistics already well known from stochastic context-free grammar can also be applied to dependency-based approaches. In order to map trees into dependency structures a head-daughter is defined for each phrase and its lexical head is perculated up the tree. The other daughters of the phrase are considered as modifiers. Accordingly a dependency relation for each head-modifier pair can be defined, for example by a triple representing the modifier non-terminal, the mother non-terminal and the head non-terminal. $< NP, S, VP >$, for instance, expresses a subject-verb dependency, cf. [Collins 1996]. For illustration, we present three recent approaches in more detail.

A Model for Bigram Lexical Dependencies

The probability of a parse tree T given a particular word string S is defined by the probability of the set of head words of NPs B (called base NPs in [Collins 1996]) conditioned on S, and the probability of the set of dependencies D given S and B, cf. [Collins 1996]. It is

$$P(T|S) = P(B|S) * P(D|B, S)$$

$P(D|B, S)$ is defined by the conditional probabilities of all binary modifer-head relations given S and B. The set D of links from a modifier to a head is defined by the product of the Maximum Likelihood estimates of the syntactic relations R_j that hold between the modifer word-tag pairs $< w_j, t_j >$ and the head word-tag pairs $< w_{h_j}, t_{h_j} >$ in S. As an extra condition, a distance measure d_{j,h_j} between a modifier and its head is added. The distance measure is based on information like the number of words and the syntactic category of phrases between the head and its modifier. We have

$$P(D|S, B) = \prod_{j=1}^{m} F(R_j| < w_j, t_j >, < w_{h_j}, t_{h_j} >, d_{j,h_j})$$

Two Generative Models

Another possibility to represent the dependency structure of a sentence is to generate for each word i in a sentence the sequence of its right and left children, cf. [Eisner 1996], [Collins 1997].

In [Eisner 1996] for each word-tag pair $tword(i)$ of a sentence the sequence of left and right children $(left\_kids(i), right\_kids(i))$ is generated. The probability that a child is generated is conditioned on the tag of the previously generated left child of i (kid_{c+1}) or the previously generated right child of i (kid_{c-1}) and the word-tag pair $tword(i)$ itself. The formula that generates the sequence of right children is given below.

$$P(T, S) = \prod_{i=1}^{n} \left(\prod_{c=-(1+\#left\_kids(i)),c\neq 0}^{1+\#right\_kids(i)} P(tword(kid_c(i))|tag(kid_{c-1}(i)), tword(i) \right)$$

A linguistically more refined model is presented in [Collins 1997]. In addition to the lexical head, information on the argument structure defined by the head is given. The probability of a parse tree T for a sentence S is defined by the product of the probabilities of its productions. Productions are defined by the conditional probability of the right-hand sides RHS_i given the left-hand sides LHS_i. Thus we have

$$P(T, S) = \prod_{i=1}^{n} P(RHS_i|LHS_i)$$

Right-hand sides are further decomposed into the probability P_H that a head category H is generated conditioned on a parent category P and a head word h, and the probabilities of the left and right contexts of $H(h)$ under P. The contexts are defined by the probabilities of the left and right constituents (P_l, P_r), and the probability of the left and right complements (P_{LC}, P_{RC}). Constituents are calculated from the probability of the head category (L_i or R_i) and the probability of the head word (l_i or r_i) conditioned on the parent category P, the parent's head category H and the head word h, as well as a distance measure d between the head and the edge of the context constituent, and a set representing the left or the right arguments (LC, LR) of $H(h)$. The probability of left and right arguments is conditioned by P, H and h. Thus we have the following formula:

$$\prod_{i=1}^{n} P_H(H|P,h) * P_l(L_i,l_i|P,H,h,d_r,LC) * P_r(R_i,r_i|P,H,h,d_r,RC) * P_{LC}(LC|P,H,h) * P_{RC}(RC|P,H,h)$$

4.10 Probabilistic LR Parsing

In Section 4.2, we defined the parse probability of any stochastic extension of a CFG as the probability of its leftmost derivation.[2] Conceptually, the stochastic version of CFG was viewed as a top-down process, accentuated by the fact that the conditioning of the production probabilities is on their LHS. In contrast to this, the parser for SCFG described in Section 4.3 worked in a bottom-up fashion. We could equally well instead define the parse probability as that of the rightmost derivation, and then specify this sequence backwards. This is exactly what an LR parser does — it constructs the rightmost derivation in reverse. In fact, this is what the "R" in "LR" stands for. The "L" stands for left-to-right scanning of the input string.

4.10.1 Basic LR Parsing

An LR parser is a type of shift-reduce parser that was originally devised for programming languages, [Knuth 1965], and is well described in e.g. [Aho *et al* 1986]. Various aspects of it relevant to natural-language parsing are descussed in [Tomita (ed.) 1991]. The success of LR parsing lies in handling a number of production rules simultaneously by the use of prefix merging, rather than attempting one rule at a time.

An LR parser is basically a pushdown automaton, i.e., it has a pushdown stack in addition to a finite set of internal states and a reader head for scanning the input string from left to right one symbol at a time. The stack is used in a characteristic way: The items on the stack consist of alternating grammar symbols and states. The current state is simply the state on top of the stack. The most distinguishing feature of an LR parser is however the form of the transition relation — the action and goto tables. A non-deterministic LR parser can in each step perform one of four basic actions. In state `S` with lookahead symbol[3] `Sym` it can:

1. `accept(S,Sym)`: Halt and signal success.

2. `error(S,Sym)`: Fail and backtrack.

3. `shift(S,Sym,S2)`: Consume the input symbol `Sym`, push it onto the stack, and transit to state `S2` by pushing it onto the stack.

4. `reduce(S,Sym,R)`: Pop off two items from the stack for each phrase in the RHS of grammar rule `R`, inspect the stack for the old state `S1` now on top of the stack, push the `LHS` of rule `R` onto the stack, and transit to state `S2` determined by `goto(S1,LHS,S2)` by pushing `S2` onto the stack.

[2] If the leftmost derivation was not unique, as in Section 4.7, we defined it as the sum of the probabilities of the leftmost derivations.

[3] The lookahead symbol is the next symbol in the input string i.e. the symbol under the reader head.

$$
\begin{array}{lcll}
S & \rightarrow & NP\ VP & (1) \\
VP & \rightarrow & V & (2) \\
VP & \rightarrow & V\ NP & (3) \\
VP & \rightarrow & V\ NP\ NP & (4) \\
VP & \rightarrow & VP\ PP & (5) \\
NP & \rightarrow & Det\ N & (6) \\
NP & \rightarrow & Pron & (7) \\
NP & \rightarrow & NP\ PP & (8) \\
PP & \rightarrow & Prep\ NP & (9)
\end{array}
$$

Figure 4.1: Sample Grammar

Prefix merging is accomplished by each internal state corresponding to a set of partially processed grammar rules, so-called "dotted items" containing a dot ($\cdot$) to mark the current position. For example, if the grammar contains the following three rules,

$$
\begin{array}{lcl}
VP & \rightarrow & V \\
VP & \rightarrow & V\ NP \\
VP & \rightarrow & V\ NP\ NP
\end{array}
$$

there will be a state containing the dotted items

$$
\begin{array}{lcl}
VP & \rightarrow & V\ \cdot \\
VP & \rightarrow & V \cdot NP \\
VP & \rightarrow & V \cdot NP\ NP
\end{array}
$$

This state corresponds to just having found a verb (V). Which of the three rules to apply in the end will be determined by the rest of the input string; at this point no commitment has been made to any of them.

4.10.2 LR-Parsed Example

The example grammar of Figure 4.1 will generate the internal states of Figure 4.2. These in turn give rise to the parsing tables of Figure 4.3. The entry "s2" in the action table, for example, should be interpreted as "shift the lookahead symbol onto the stack and transit to State 2". The action entry "r7" should be interpreted as "reduce by Rule 7". The goto entries simply indicate what state to transit to once a phrase of that type has been constructed. Note the two possibilities in States 11, 12 and 13 for lookahead symbol preposition, "Prep", both of which must be tried. We can either shift it onto the stack or perform a reduction. This is called a *shift-reduce conflict* and is the source to the ambiguity in the sentence *John sees a man with a telescope.*

Using these tables we can parse the sentence *John reads a book* as follows:

State 0

$S' \Rightarrow \cdot\, S$
$S \Rightarrow \cdot\, NP\ VP$
$NP \Rightarrow \cdot\, Det\ N$
$NP \Rightarrow \cdot\, Pron$
$NP \Rightarrow \cdot\, NP\ PP$

State 1

$S \Rightarrow NP \cdot VP$
$NP \Rightarrow NP \cdot PP$
$VP \Rightarrow \cdot\, V$
$VP \Rightarrow \cdot\, V\ NP$
$VP \Rightarrow \cdot\, V\ NP\ NP$
$VP \Rightarrow \cdot\, VP\ PP$
$PP \Rightarrow \cdot\, Prep\ NP$

State 2

$NP \Rightarrow Det \cdot N$

State 3

$NP \Rightarrow Pron\, \cdot$

State 4

$S' \Rightarrow S\, \cdot$

State 5

$S \Rightarrow NP\ VP\, \cdot$
$VP \Rightarrow VP \cdot PP$
$PP \Rightarrow \cdot\, Prep\ NP$

State 6

$VP \Rightarrow V\, \cdot$
$VP \Rightarrow V \cdot NP$
$VP \Rightarrow V \cdot NP\ NP$
$NP \Rightarrow \cdot\, Det\ N$
$NP \Rightarrow \cdot\, Pron$
$NP \Rightarrow \cdot\, NP\ PP$

State 7

$NP \Rightarrow NP\ PP\, \cdot$

State 8

$PP \Rightarrow Prep \cdot NP$
$NP \Rightarrow \cdot\, Det\ N$
$NP \Rightarrow \cdot\, Pron$
$NP \Rightarrow \cdot\, NP\ PP$

State 9

$VP \Rightarrow VP\ PP\, \cdot$

State 10

$NP \Rightarrow Det\ N\, \cdot$

State 11

$VP \Rightarrow V\ NP\, \cdot$
$VP \Rightarrow V\ NP \cdot NP$
$NP \Rightarrow NP \cdot PP$
$NP \Rightarrow \cdot\, Det\ N$
$NP \Rightarrow \cdot\, Pron$
$NP \Rightarrow \cdot\, NP\ PP$
$PP \Rightarrow \cdot\, Prep\ NP$

State 12

$VP \Rightarrow V\ NP\ NP\, \cdot$
$NP \Rightarrow NP \cdot PP$
$PP \Rightarrow \cdot\, Prep\ NP$

State 13

$PP \Rightarrow Prep\ NP\, \cdot$
$NP \Rightarrow NP \cdot PP$
$PP \Rightarrow \cdot\, Prep\ NP$

Figure 4.2: The resulting internal states

State	Action							Goto			
	Det	N	NP	Prep	Pron	V	eos	NP	PP	S	VP
0	s2		s1		s3			1		4	
1				s8		s6			7		5
2		s10									
3	r7		r7	r7	r7	r7	r7				
4							acc				
5				s8			r1		9		
6	s2		s11	r2	s3		r2	11			
7	r8		r8	r8	r8	r8	r8				
8	s2		s13		s3			13			
9				r5			r5				
10	r6		r6	r6	r6	r6	r6				
11	s2		s12	s8/r3	s3		r3	12	7		
12				s8/r4			r4		7		
13	r9		r9	s8/r9	r9	r9	r9		7		

Figure 4.3: The corresponding LR parsing tables

Action	Stack	String
init	[0]	*John reads a book*
s1	[1, NP, 0]	*reads a book*
s6	[6, V, 1, NP, 0]	*a book*
s2	[2, Det, 6, V, 1, NP, 0]	*book*
s10	[10, N, 2, Det, 6, V, 1, NP, 0]	ϵ
r6	[11, NP, 6, V, 1, NP, 0]	ϵ
r3	[5, VP, 1, NP, 0]	ϵ
r1	[4, S, 0]	ϵ
accept	[4, S,0]	ϵ

Initially State 0 is pushed onto the empty stack. The noun phrase (*NP*) corresponding to the word "John" is shifted onto the stack and the parser transits to State 1 by pushing it onto the stack. Next, the verb (*V*) corresponding to the word "reads" is shifted onto the stack and the parser transits to State 6 by pushing it onto the stack. Then the determiner (*Det*) corresponding to the word "a" is shifted onto the stack and the parser transits to State 2. The noun (*N*) corresponding to the word "book" is shifted onto the stack and the parser transits to State 10. At this point, the noun and the determiner on top of the stack are reduced to a noun phrase using Rule 6 ($NP \rightarrow Det\ N$) by popping State 10, the noun, State 2 and the determiner from the stack. The noun phrase is then pushed onto the stack, and the parser transits to State 11 by pushing it onto the stack. Next, the noun phrase and the verb on top of the stack are reduced to a verb phrase (*VP*) using Rule 3 ($VP \rightarrow V\ NP$), which is pushed onto the stack, and the parser transits to State 5. Then the verb phrase and the noun phrase on top of the stack are reduced to a sentence (*S*) using Rule 1 ($S \rightarrow NP\ VP$), which is pushed onto the stack, and the parser transits to State 4. Finally, the input string is accepted.

4.10.3 LR-Table Compilation

Compiling LR parsing tables consists of constructing the internal states (i.e. sets of dotted items) and from these deriving the shift, reduce, accept and goto entries of the transition relation.

New states can be induced from previous ones; given a state `S1`, another state `S2` reachable from it by `goto(S1,Sym,S2)` (or `shift(S1,Sym,S2)` if `Sym` is a terminal symbol) can be constructed as follows:

1. Select all items in state `S1` where a particular symbol `Sym` follows immediately after the dot and move the dot to after this symbol. This yields the kernel items of state `S2`.

2. Construct the non-kernel closure by repeatedly adding a so-called non-kernel item (with the dot at the beginning of the RHS) for each grammar rule whose LHS matches a symbol following the dot of some item in `S2`.

For example State 1 of Figure 4.2 can be constructed from State 0 by advancing the dot of the items $S \rightarrow \cdot\ NP\ VP$ and $NP \rightarrow \cdot\ NP\ PP$ to form the items $S \rightarrow NP\ \cdot\ VP$ and $NP \rightarrow NP\ \cdot\ PP$ which constitute the kernel of State 1. The remaining non-kernel items are generated by the grammar rules for VPs and PPs (the categories following the dots in the new kernel items), namely Rules 2, 3, 4, 5 and 9.

Using this method, the set of all parsing states can be induced from an initial state whose single kernel item has the top symbol of the grammar preceded by the dot as its RHS. In Figure 4.2 this is the item $S' \to \cdot\, S$ of State 0.

The shift, goto and accept entries fall out automatically from this procedure. Any dotted item where the dot is at the end of the RHS gives rise to a reduction by the corresponding grammar rule. Thus it remains to determine the lookahead symbols of the reduce entries.

In *Simple LR (SLR)* the lookahead is any terminal symbol that can follow immediately after a symbol of the same type as the LHS of the rule. In *LookAhead LR (LALR)* it is any terminal symbol that can immediately follow the LHS given that it was constructed using this rule in this state. In general, LALR gives considerably fewer reduce entries than SLR, and thus results in faster parsing.

4.10.4 Generalized LR Parsing

Generalized LR (GLR) parsing extends basic LR parsing with two concepts; a graph-structured stack (GSS) and a packed parse forest ([Tomita (ed.) 1991]). It also differs in being a breadth first, accumulative search, synchronizing on shift actions, rather than a depth-first backtracking algorithm. Conceptually, a GLR parser works as follows:

1. Shift the next input symbol onto the stack(s). Don't keep the old stack(s), only the new one(s).

2. Perform all possible reduce actions on the stack(s). This will give rise to a new stack for each reduction. Keep the old stacks (before the reductions). Repeatedly perform all possible reduce actions on the new set of stacks, accumulating new stacks, until no further reduce actions are possible.

3. Goto 1.

A graph-structured stack is used instead of a set stacks. This means that the new portions of the GSS constructed by shifting in Step 1 are merged with the stack portions resulting from subsequent (repeated) reductions in Step 2.

After a shift action, two stack portions are merged if the new top states are equal. If we perform a reduction and the corresponding goto entry would imply creating a stack continuation that already exists, we simply use the existing one. The two partial parse trees corresponding to the two merged LHS symbols will then dominate the same substring, but be structually different: Since all surviving stack portions will have just shifted the word prior to the current string position (and possibly have been subjected to subsequent reductions), the dominted string will end at the the current string position. Since the same node was reached by the reduction prior to pushing the LHS symbol onto the GSS, the dominated string will begin at the same string position. Since the action seqeunces producing the two derivations are different, the partial parse trees will also be different.

The partial parse trees associated with any nonterminal symbol will however not be recoverable from the GSS. For this reason, we will store the structure of each partial parse tree associated with each node in the GSS in a parse forest. By doing this in a clever way, we can avoid multiplying out the potentially exponentially many different parse trees that can be associated with the parsed word string — This is where the packed parse forest enters into the story.

The LHS symbol of any reduction is constructed from a sequence of RHS symbols, each associated with a (set of) partial parse tree(s). The latter will be recorded in the packed parse forest, and the only information we need to specify for the node of the packed parse forest associated with the LHS of a reduciton is the set of nodes in the packed parse forest associated with the RHS symbols. This means that the internal structure of each RHS symbol is encapsulated in its node in the packed parse forest. So, regardless of in how many different ways these RHS symbols can in turn have been constructed, this ambiguity is contained locally at the corresponding nodes of the packed parse forest, and not multiplied out at the node corresponding to the current LHS of the production. This is known as *local ambiguity packing*.

In the case of a normal CFG parsing algorithm, we need one node for each nonterminal symbol and each string position, see 4.1.4. Here we need one for each node that ever figured in the graph-structured stack. ⟨**John Carroll writes one for each internal state.**⟩

4.10.5 Adding Probabilities

As pointed out above, we can calculate the parse probability $P(\tau)$ from the probability of the sequence of productions used in the rightmost derivation of the parse viewed in reverse order. This gives us the very same recurrence equation as when using the sequence of productions used in the leftmost derivation, but the interpretation is now somewhat different:

$$\begin{aligned} P(\tau) &= P(\xi_1 = r_{i_1}, \ldots, \xi_{M-1} = r_{i_{M-1}}, \xi_M = r_{i_M}) \quad = \\ &= \prod_{m=1}^{M} P(\xi_m = r_{i_m} \mid \xi_1 = r_{i_1}, \ldots, \xi_{m-1} = r_{i_{m-1}}) \end{aligned}$$

Since there is a one-to-one correspondence between this sequence and the sequence of actions of the LR parser, we can equally well use the action sequence to specify the rightmost derivation in reverse, and thus the parse:

$$\begin{aligned} P(\tau) &= P(\xi_1 = a_{i_1}, \ldots, \xi_{L-1} = a_{i_{L-1}}, \xi_L = a_{i_L}) \quad = \\ &= \prod_{l=1}^{L} P(\xi_l = a_{i_l} \mid \xi_1 = a_{i_1}, \ldots, \xi_{l-1} = a_{i_{l-1}}) \end{aligned}$$

Here a_l is the lth action performed by the LR parser.

We will approximate the action probabilities conditional on the previous action sequence, $P(\xi_l = a_{i_l} \mid \xi_1 = a_{i_1}, \ldots, \xi_{l-1} = a_{i_{l-1}})$, with the action probabilities conditional on the parsing state S_{k_l} after $l-1$ actions, just prior to the lth action, and further approximate these probabilities by using an extractor function $g(S)$ to select relevant portions of the parsing states:

$$P(\xi_l = a_{i_l} \mid \xi_1 = a_{i_1}, \ldots, \xi_{l-1} = a_{i_{l-1}}) \quad \approx \quad P(\xi_l = a_{i_l} \mid g(S_{k_l}))$$

For example, this function might select the current internal state and the current lookahead symbol, and discard the rest of the stack content, and the remaining input string. The main point is that during parsing, these approximations can be determined locally, and thus multiplied together:

$$P(\tau) \quad \approx \quad \prod_{l=1}^{L} P(\xi_l = a_{i_l} \mid g(S_{k_l}))$$

This effectively constructs equivalence classes for the various histories of action sequences.

Let us return to the parse tables of 4.3. We will here assign a probability distribution to each pair of state and lookahead symbol over the set of possible actions. As we see, the only situations where we have nondeterminism in the parsing tables are in States 11,12 and 13 with a preposition as a lookahead. In State 11, for example, we have the shift-reduce conflict s8/r3, which corresponds to a potential PP-attachment ambiguity in the input string. We will here assign some probability p_{11} to s8 and $1 - p_{11}$ to r3 in State 11 with lookahead symbol preposition. Similarly, we assign p_{12} to s8 and $1 - p_{12}$ to r4 in State 12, and p_{13} to s8 and $1 - p_{13}$ to r9 in State 13, in both cases with lookahead symbol preposition. The rest of the actions are uniquely determined by the state and lookahead symbol and thus have probability 1.

4.10.6 Probabilistic GLR Parsing

One way of doing probabilitisc LR parsing is to first construct a packed parse forest by normal GLR parsing, but in the process attribute a probability to each different local decision made at each node. These probabilities are calculated as the product of the action sequences required to construct the mother symbol from its daughter symbols. However, the probabilities of the latter have not yet been propagated to the former to determine the probability of each possible analysis; the representeation is thus packed also probabilistically. The probabilities can be made conditional on arbitrary portions of the state of the LR parser, i.e., on the stack content and the remaining input string.

The parse probability is defined as the product of each action probability conditional on the previous sequence of actions. Since these action probabilites are approximated by the probabilites conditional on the state of the LR parser, *the probability of a parse is simply the product of the probability of each decision in the packed parse forest*. So in the disambiguation step, a Viterbi-like search of the packed parse forest can be used to find the most probable parse. This means that the time complexity of the disambiguation step is linear in the size of the packed parse forest, as is calculating the string probability by summing instead of maximizing. ⟨**Check!**⟩

In this way, probabilistic LR parsing is broken down into two distinct steps: Normal GLR parsing, with an extra element of attributing each decision in the packed parse forest with a probability. Each such probability is simply the product of a sequence of action probabilities, and calculating them introduces only a constant overhead, and does not add to the complexity of GLR parsing. The worst-case time complexity of the original GLR parsing algorithm is exponential, both in string length and grammar size. This is since there can theoretically be an exponential number of different internal states in the size of the grammar, and for certain grammars there may be inputs that force a parser to visit all states [Johnson 1991]. It can however be brough down to polynomial in string length by an optimization due to Kipps, see [Kipps 1991]. It essentially involves avoiding to search the graph-structured stack for the return state when reducing by instead employing a dynamically built table. Using large portions of the stack for probabilistic conditioning counteracts this effect, and increases parsing complexity.

The actual probabilistic disambiguation is done on the output of the

GLR parsing step, on the packed parse forest. This can be done using essentially Viterbi search in time linear in the size of the packed parse forest. Although the size of the parse forest is polynomial in string length, it is worst-case exponential on the size of the grammar for the same reasons that parsing complexity is exponential in grammar size.

So although probabilistic LR parsing is theoretically exponential in grammar size, in practice, it doesn't seem to get near these bounds [Carroll 1994]. For a discussion on how to extend probabilistic LR parsing to unification grammars, and for interesting reading on probabilistic LR parsiong in general, we can recommend [Briscoe & Carroll 1993].

4.11 Scoring

A practical problem is that due to various probability losses, i.e., the fact that the estimated probabilities tend to be smaller than they ought to be, parses involving longer derivations tend to be penalized. This is because they involve a greater number of derivation steps, involving a greater number of multiplications with too small quantities. One popular remedy, [Magerman & Marcus 1991], [Carroll 1994], is to instead use the geometric mean of the probabilities instead of their product. This means that we leave the realm of probability theory, and enter into the heuristic scoring business. For example, this complicates using the Viterbi algorithm for finding the most probable parse.

Due to the conditionings of the probabilites, SCFGs are rather insensitive to lexical information, and in particular to lexical co-occurrences. One method for trying to compenstate for this is to include amongst other things lexical information in more or less ad hoc ways by devising a score accepting contributions from various information sources. This has been explored in a number of different contexts: For Earley parsing with Scoring, see [Magerman & Marcus 1991], for CYK parsing with scoring, see [Magerman & Weir 1992], for LR parsing with scoring, see [Su *et al* 1991], and for parsing with unification grammars with scoring, see [Alshawi & Carter 1994].

⟨To be extented.⟩

Chapter 5

Selected Topics in Statistical NLP

In the following, we will distinguish three types of literatur: Main articles, additional articles which are of major interest in a basic course on statistical methods in NLP (see "Also" item), and further readings.

5.1 Statistical Part-of-Speech Tagging

5.1.1 In short

$$\max_{Tags} P(\mathit{Tags} \mid \mathit{Word\ String})$$

5.1.2 Linguistic Background

Part-of-speech (PoS) tagging consists in assigning to each word of an input text a (set of) tag(s) from a finite set of possible tags, a tag palette or a tag set. The reason that this is a research issue is that a word can in general be assigned different tags depending on context. This assignment can be done in a number of different ways. One of these is statistical tagging, which is advocated in [Church 1988], [Cutting *et al* 1992] and many other articles. Here, the relevant information is extracted from large sets of often hand-tagged training data and fitted into a statistical language model, which is then used to assign the most likely tag to each word in the input text.

5.1.3 Basic Statistical PoS tagging

We will describe a generic, but somewhat vanilla-flavoured statistical PoS tagger. Statistical PoS taggers generally distinguish between lexical probabilities, i.e., the probability of a particular tag conditional on the particular word, and contextual probabilities, which describe the probability of a particular tag conditional on the surrounding tags. The latter conditioning is usually on the tags of the neighbouring words, and very often on the $n-1$ previous tags.

Thus we in general have the following two information sources:

- **Lexical probabilities:**
  The probability of each tag T^i conditional on the word W that is to

be tagged, $P(T^i \mid W)$. Often the converse probability $P(W \mid T^i)$ is given instead.

- **Tag N-grams:**
The probability of tag T^i at position k in the input string, denoted T_k^i, given that tags $T_{k-n+1} \ldots T_{k-1}$ have been assigned to the previous $n-1$ words. Often n is set to two or three, and thus bigrams or trigrams are employed. When using trigram statistics, this quantity is $P(T_k^i \mid T_{k-2}, T_{k-1})$.

These probabilities can be estimated either from a pretagged training corpus or from untagged text, a lexicon and an initial bias, see Section 2.1.7. The training data is often divided into a training set, used to estimate the statistical parameters, and a set of held back data used to cope with sparse data by way of backoff smoothing. For example, tag trigram probabilities can be estimated as follows:

$$P(T_k^i \mid T_{k-2}, T_{k-1}) \quad \approx \quad \lambda_3 f(T_k^i \mid T_{k-2}, T_{k-1}) + \lambda_2 f(T_k^i \mid T_{k-1}) + \lambda_1 f(T_k^i)$$

Here f is the relative frequence in the training set. The weights $\lambda_j = \lambda_j(T_{k-2}, T_{k-1})$ may depend on the particular contextual tags, but are required to be nonnegative and to sum to one over j. Appropriate values for these weights can be estimated using the held-out portion of the training corpus by employing any of a number of techniques; two ones much used today are deleted interpolation, [Jelinek & Mercer 1980], and modified Good-Turing estimation, [Church & Gale 1991]. Another possibility is to use all data for training and employ successive abstraction to perform the backoff smoothing, see [Brants & Samuelsson 1995].

In general, the information sources $S_1, \ldots, S_n$ are combined by multiplying the scaled probabilities:

$$\frac{P(T \mid S_1, \ldots, S_n)}{P(T)} \quad \approx \quad \prod_{i=1}^{n} \frac{P(T \mid S_i)}{P(T)}$$

This formula can be established by Bayesian inversion, then performing the independence assumptions, and renewed Bayesian inversion: Assume that we have information sources $S_1, \ldots, S_n$ and we wish to estimate $P(T \mid S_1, \ldots, S_n)$, the probability of tag T given this information.

$$\begin{aligned}
P(T \mid S_1, \ldots, S_n) \quad &= \\
= \quad \frac{P(T) \cdot P(S_1, \ldots, S_n \mid T)}{P(S_1, \ldots, S_n)} \quad &\approx \quad P(T) \cdot \prod_{i=1}^{n} \frac{P(S_i \mid T)}{P(S_i)} \quad = \\
= \quad P(T) \cdot \prod_{i=1}^{n} \frac{P(T) \cdot P(S_i \mid T)}{P(T) \cdot P(S_i)} \quad &= \quad P(T) \cdot \prod_{i=1}^{n} \frac{P(T \mid S_i)}{P(T)}
\end{aligned}$$

In particular, using lexical statistics and trigram probabilities, we get

$$\begin{aligned}
&P(T_k \mid T_1, \ldots, T_{k-1}; W_1, \ldots, W_n) \quad \approx \\
&\approx \quad \frac{P(T_k \mid T_{k-2}, T_{k-1}) \cdot P(T_k \mid W_k)}{P(T_k)} \quad = \quad \frac{P(T_k \mid T_{k-2}, T_{k-1}) \cdot P(W_k \mid T_k)}{P(W_k)}
\end{aligned}$$

The tagger works as follows: First, each word is assigned the set of all possible tags according to the lexicon. This will create a lattice. A dynamic programming technique is then used to find the sequence of tags $T_1, \ldots, T_n$ that maximizes

$$\begin{aligned}
P(T_1, \ldots, T_n \mid W_1, \ldots, W_n) &= \\
&= \prod_{k=1}^{n} P(T_k \mid T_1, \ldots, T_{k-1}; W_1, \ldots, W_n) \approx \\
&\approx \prod_{k=1}^{n} P(T_k \mid T_{k-2}, T_{k-1}; W_k) \approx \\
&\approx \prod_{k=1}^{n} \frac{P(T_k \mid T_{k-2}, T_{k-1}) \cdot P(T_k \mid W_k)}{P(T_k)} = \\
&= \prod_{k=1}^{n} \frac{P(T_k \mid T_{k-2}, T_{k-1}) \cdot P(W_k \mid T_k)}{P(W_k)}
\end{aligned}$$

Since the maximum does not depend on the factors $P(W_k)$, these can be omitted, yielding the standard statistical PoS tagging task:

$$\max_{T_1, \ldots, T_n} \prod_{k=1}^{n} P(T_k \mid T_{k-2}, T_{k-1}) \cdot P(W_k \mid T_k)$$

This is well-described in for example [DeRose 1988].

5.1.4 Suggested Reading

- Main: [DeRose 1988], [Cutting *et al* 1992]
- Also: [Church 1988], [Weischedel *et al* 1993]
- Further: [Black *et al* 1992], [Schuetze 1994], [Merialdo 1994]

The first chapter of [Karlsson *et al* 1995] contains a nice overview of part-of-speech tagging in general. [Church 1988] is a classical reference on basic statistical part-of-speech taggging. Dynamic programming, in particular the Viterbi algorithm, as applied to part-of-speech tagging is very well described in [DeRose 1988]. Other sections, in particular that on the CLAWS system, are less perspicuous. The decisive reference on dynamic programming in general is [Bellman 1957]. [Cutting *et al* 1992] describes the use of an HMM-based tagger were the parameters are estimated from unannotated text using the Baum-Welch algorithm.

5.2 Statistical Machine Translation

Currently under construction. Due mid 1996.

5.2.1 In short

$$\max_{\mathit{Target\ Text}} P(\mathit{Target\ Text} \mid \mathit{Source\ Text})$$

5.2.2 Suggested Reading

- Main: [Brown *et al* 1993],
- Also: [Church 1993], [Wu 1995]
- Further: [Brown *et al* 1990], [Kay & Röscheisen 1993], [Dagan *et al* 1993], [Gale & Church 1993], [Kupiec 1993]

5.3 Statistical Language Learning

Currently under construction. Due mid 1996.

- Main: [Brill 1993], [Magerman & Marcus 1990]
- Also: [Osborne & Bridge 1994], [Daelemans 1994].

5.4 Structural Ambiguity and Semantic Classes

5.4.1 Linguistic Background

In natural language analysis we have to deal with a number of structural ambiguous constructions, i.e. syntactic structures that are equally well licensed by different derivations. Thus we get more than one correct parse tree for such a construction. Depending on the context, one of the syntactically possible structures is the preferred one. In linguistics the most famous class of structural ambiguities are prepositional phrase attachments. For those who have never heard about such a thing, go on reading. Those who are familiar with the term could skip this subsection.

The syntactic structure of a the famous sentence *I saw the man with the telescope* can be considered as either $[_S$ I $[_{VP}$ saw $[_{NP}$ the man $[_{PP}$ with the telescope$]]]]$ or $[_S$ I $[_{VP}$ saw $[_{NP}$ the man$]$ $[_{PP}$ with the telescope$]]]$. That means the with_phrase can either be attached to the object noun phrase or to the verb phrase. In the first case, the interpretation is that the man had the telescope, in the second the telescope modifies the seeing event. Without any extra knowledge (semantic or pragmatic context) there is no way to decide which of these two readings will be more appropriate. If we consider the sentence *I saw the deer with the telescope* our knowledge about the world tells us that attachment of the PP to the object NP leads to an odd interpretation, as in our world a deer rarely comes with a telescope. Nevertheless the structure is absolutely correct.

To deal with this kind of ambiguities a number of different approches have been proposed, such as:

- The Discourse Model Approach

  Altman and Steedman 1988 ([Altmann & Steedmann 1988]) claim that pp-attachment can only be resolved by considering discourse information. In terms of computational linguistics this means modelling of discourse information which is fairly complex and not yet well understood.

- The Structure-Based Approach

  A number of psycholinguists tried to explain attachment preferences by means of general principles that are supposed to govern human language processing. Two famous but controversal priciples are right association ([Kimball 1973]), and minimal attachment ([Frazier 1978]). Right association states that a constituent tends to attach to another constituent immediately to its right. Whereas minimal attachment says that a constituent tends to attach so that a minimal number of nodes in a derivation tree is required. But these two principles are not accepted without argument. Interested readers may find further discussion in [Konieczny *et al.* 1997].

- The Lexical Association Approach

  This approach basically assumes that attachment ambiguities can be resolved by lexical information, such as information related to the dominating verb, the object head noun the preposition, and the head noun of the NP dominated by the preposition. We distinguish two major branches, one influenced from psycholinguistics[1], the other based on statistics. The later we will elaborate on in the following.

5.4.2 Association Models

The main assumption is that resolution of structural ambiguity is often possible with limited lexical information, and this information can be learned from either a corpus ([Hindle & Rooth 1993]), or a corpus and a semantic hierarchy ([Resnik 1993]). While Resnik advocates a class-based model which makes use of conceptual relationships such as those represented in WordNet[2], Hindle and Rooth adopt lexical associations discovered from text for structural disambiguation. Hindle and Rooth integrate lexical information on the preceding verb, the object head noun, and the following preposition into their mathematical model. Resnik incorporates information on the head verb, the preposition, the semantic class of the head noun of the NP governed by the preposition, and the semantic class of the object head noun.

Preparation of the Corpus

In order to access the relevant head nouns, the verbs, and the prepositions the corpus must be PoS-tagged, and rudimentary parsed.

elaborate on standard taggers and synt. bracketing elsewhere

Estimation of Attachment Preferences by Lexical Information

In the training phase noun-preposition and verb-preposition bigrams as well as noun and verb unigrams are derived from the corpus. The preposition is either assigned to the noun or the verb. Assignment decisions are made according to:

[1] See for example [Wittemore *et al* 1992], [Ford *et al.* 1982], [Taraban & McClelland 1988], [Marcus 1980].

[2] WordNet is a conceptual taxonomy for English words. Conceptual relations are represented as is-a hierarchies. You find more about WordNet from ftp clarity.princeton.edu [128.112.144.1] (in the US), and ftp.ims.uni-stuttgart.de [141.58.127.61] (in Europe).

- purely linguistic criteria, such as PPs do not attach to pronouns, PPs that follow a subject NP or a NP in preverbal position attach to the NP, at least in English,
- thresholds derived from t-scores or
- defaults

  If no clear attachment decision can be made, neither by linguistics, nor by t-scores, the prepositions are equally attachted to the head nouns of the object NPs, or to the verbs.

In order to guess attachment preferences, one is interested in the contrast or difference between the conditional probalility of seeing a certain preposition given a particular noun, and the conditional probalility of seeing a certain preposition given a particular verb, i.e.

$$P(prep|noun) - P(prep|verb)$$

These conditional probabilies correspond to the preposition-noun or preposition-verb bigram frequences respectively which have been derived from the training corpus. The bigram frequencies are normalized by the global verb or noun frequency respectively. Thus the conditional probabilities are defined as follows:

$$P(prep|noun) \equiv \frac{f(noun\_preposition\_bigram)}{f(noun)}$$

, and

$$P(prep|verb) \equiv \frac{f(verb\_preposition\_bigram)}{f(verb)}$$

For a statistics-based determination of pp-attachment preferences for a specific sentence a t-score is derived from the conditional probabilities;

$$t \equiv \frac{P(prep|noun) - P(prep|verb)}{\sqrt[2]{\sigma^2(P(prep|noun)) + \sigma^2(P(prep|verb))}}$$

To make sure that the significance of the result is at a certain (at least a 95%) level a threshold is defined accordingly.[3] In all cases of attachment ambiguity where the results are below the defined threshold, pp-attachment is resolved according to the default case.

Sparse Data

Sparseness of training data is a common problem in statistics-based approaches. In case of estimation of lexical associations on the basis of word forms a fairly large ammount of training data is required, because all inflected forms of a word will be considered as different words. This can be a major problem in highly inflecting languages like German. A better result by the same amount of data can be achieved by looking at lemmata (inflectional differences are ignored) or even at semantic classes. In the later case different words/lemmata are grouped together according to their semantic or conceptual characteristics.

[3] For definition of significance levels see chapter ??.

Estimation of Class Frequencies

In a training corpus the head nouns of the object NPs, and the head nouns of the NPs subcategorized from a preposition are related to conceptual classes in WordNet. Class frequencies are estimated from lexical frequencies such that the frequency of a specific class C is determined by the lexical frequencies of the members n of that class and its subclasses c, i.e.

$$f(C) = \sum_{n \in c \subseteq C} f(n)$$

Estimation of Conceptual Relationships

Conceptual relationships are discribed by means of selectional preference, selectional association, and semantic similarity.

The term **selectional preference** originates from linguistics. It is used to capture semantic differences such as those given in *Mary drank some wine / gasoline / pencils / sadness.* While drinking and wine semantically correspond well, drinking of gasoline is quite odd, pencils are impossible to drink, and the semantic interpretation of drinking sadness is subject to metaphorisation. A widely used technique in statistics-based NLP to represent conceptual (dis)similarity is entropy (cf. ??). Thus the selectional preference of a word w for C, a semantic class and its subclasses, is defined as the relative entropy (also known as Kullback-Leibler distance) between two sets of probabilities, the prior probability $P(C)$ and the conditional probability $P(C|w)$, which can be written as

$$H[P(C|w), P(C)] = \sum_{c|subconcept\_of\_C} P(c|w) log \frac{P(c|w)}{P(c)}$$

The **selectional association** between a word w and a subconcept c, $A(w,c)$, is the contribution c makes to the selectional preference of w for C, i.e. the relative entropy between a wird w, and a concept c normalized by the relative entropy between the word w and the larger concept C.

$$A(w,c) = \frac{P(c|w) log \frac{P(c|w)}{P(c)}}{H[P(C|w), P(C)}$$

The selectional association between two words w_1 and w_2, $A(w_1, w_2)$ is the maximum of $A(w_1, c)$ over all classes c to which w_2 belongs.

$$A(w_1, w_2) = max \sum_{c|w_2 \in c} A(w_1, c)$$

The **semantic similarity** of two nouns n_1, n_2, $sim(n_1, n_2)$, is defined by the most specific class c both n_1 and n_2 belong to, i.e. the class c which is most informative for n_1 and n_2.

$$sim(n_1, n_2) = max[-log P(c)]$$

Class-Based Evaluation of Attachment Sites

The following trigrams are derived from the corpus and the noun taxonomy:

- verb_preposition_$class_j$

- $class_i$_preposition_$class_j$

Where $class_i$ is the conceptual class of the object NPs head noun and $class_j$ is the conceptual class of the head noun of the NP dominated by the preposition.

Verb- and noun-scores are calculated as the relative entropy of the verb_preposition_$class_j$ trigram frequencies, and the $class_i$_preposition_$class_j$ trigram frequencies, such that

$$vscore = f(verb\_prep\_class_j) log \frac{P(verb\_prep\_class_j)}{P((verb)}$$

, and

$$nscore = f(class_i\_prep\_class_j) log \frac{P(class_i\_prep\_class_j)}{P((class_i)}$$

.

To reduce the load of computation, first all classes of the preposition's nominal object are considered, and then the class of the verb's nominal object is maximized. A paired t-test is calculated on the v- and nscores. In case of a positive result, the PP is attached to the object NP. In case of a negative result, the PP is attached to the verb.

Estimators

Hindle and Rooth work with ELE, Resnik suggested MLE or Good-Turing. For a discussion of estimation techniques see ??.

5.4.3 Suggested Reading

- Main: [Hindle & Rooth 1993]
- Also: [Resnik 1993]
- Further [Alshawi & Carter 1994]

5.5 Word Sense Disambiguation

Word sense disambiguation is a rather widespread task in statistical NLP. The approaches vary wrt. the parameters they estimate disambiguatuation information from. The techniques applied are rather similar. Parameters are either estimated from monolingual ([], []) or bilingual[4] corpora ([Gale *et al* 1992], [Brown *et al* 1990]). In case of monolingual training material, words are either clustered according to their distribution in structural context ([Pereira *et al* 1993], [Dagan *et al* 1994], []) or their distribution wrt. semantic category ([Gale *et al* 1992], [Yarowsky 1992]).

5.5.1 Phenomena

5.5.2 Parameter Estimation

5.5.3 Disambiguation Model

Bayesian discrimination

[4] The standard bilingual is the Hansards corpus which comprises the french-english translations of Canadian Parliament speeches.

relative entropy
mutual information
?singular value decomposition
?linear regression
???

5.5.4 Suggested Reading

- Main: [Gale *et al* 1992], [Yarowsky 1992]
- Also: [Yarowsky 1992], [Gale *et al* 1992]
- Further: [Pereira *et al* 1993], [Schuetze 1992], [Dagan *et al* 1993], [Dagan *et al* 1994]

5.6 Lexical Knowledge Acquisition

$$\max_{\mathit{Lexical\ Entry}} P(\mathit{Lexical\ Entry} \mid \mathit{Text})$$

- Main:[Manning 1993], [Utsuro *et al* 1992], [Smadja 1993]

Appendix A

Desiderata

The follwing appendices are planned:

- More on Calculus
- Some Numerical Analysis
- Some Statistical Tables (Distributions, etc.)
- Corpus Linguisitc Tools

Appendix B

Tools

B.1 Simple Unix Commands

Before we can do all the fancy statistics on texts, we need to access the raw data. Unix provides a number of built-in commands that are very useful to do so. In the following a list of elementary commands valuable for text preprocessing is given. The online man pages on your Unix machine will give you more information on the specific commands.

Command	Usage
grep	search a file for a pattern
egrep	search a file for disjunctive patterns
sort	sort and/or merge files
uniq	report repeated lines in a file
tr	translate characters
wc	display a count of lines,
	words and characters in a file
cut	cut out selected fields of each line of a file
paste	merge same lines of several files or
	subsequent lines of one file
join	merges those lines of a file that have the same key
comm	select or reject lines common to two sorted files
cat	concatenate files
tail	writes the last part of a file to standard output
split	split a line by indicated split characters
	and particularly useful:
split -n	split a file into pieces of n lines
sed	stream editor

Unix also comes with a number of programming languages of which the following are popular for text handling:

Awk is a pattern scanning and processing language that is preferably used for database construction.

Lex or its successor **flex** is a language for lexical analysis. Typically tokenizer are written in lex or flex. Lex programmes can be interleaved with C code. They are compiled into C, and thus are rather efficient.

Perl is a language for easy manipulation of text, files, and processes.

It borrows capabilities from shells and C, and subsumes awk and sed. Perl is compared to C easy to learn, but not very efficient.

From the following examples[1] you might get an idea about what you could do with the above commands. A good guide for novices is Ken Church's *Unix for poets* [Church 1994].

For those who are not used to Unix, the following is to understand the syntax of the command lines occurring in the examples below:

```
<                  read from input file
>                  write to output file
|                  pipe
| more             pipe to more;
                   more displays standard output screenwise
```

The following is a list with the most essential characters to build search patterns with:

```
+          one or more instances of a specified pattern
*          zero or more instances of a specified pattern
?          at most one instance of a specified pattern
.          any single character
[a-z]      any one character from the small alphabet
[A-Z]      any one character from the ?big alphabet
[0-9]      any one number between 0 and 9
^          begin of line
$          end of line
|          disjunction of patterns
\          indicates that the following character is interpreted
           as such
```

B.2 Split up a text using tr

- Split up a text by blanks, and write each word (i.e. string of characters surrounded by blanks) in a single line to a file. The input file is infile, the output file is outfile. \012 is the ASCII code for new line.

  ```
  tr ' ' '\012'  < infile >  outfile
  ```

- Get rid of empty lines.

  ```
  tr -sc 'A-Za-z' '\012' < infile > outfile
  ```

Now you have a file, lets call it wl, containing your input text as word list. With the word list you can do other fancy things.

B.3 Sort word list: sort, uniq

- Sort word list by dichtionary order.

  ```
  sort -d wl
  ```

[1] The material is taken from an introductory course on corpus linguistics held by the authors at Uppsala University.

- Sort word list by dichtionary order, and get rid of duplicates. Write output to outfile.

  ```
  sort -d wl |  uniq > outfile
  or
  sort -du wl > outfile
  ```

- Sort word list by dictionary order, get rid of duplicates, and give a count of how often each word appeared in the word list.

  ```
  sort -d wl | uniq -c > outfile
  ```

- Sort word list according to word frequency (numeric order).

  ```
  uniq -c wl  | sort -d > outfile
  ```

B.4 Merge counts for upper and lower case: tr, sort, uniq

- Sometimes you just want to know how often a specific word occurs in a text. And you do not want to make a distinction between upper and lower case letters.

  ```
  tr 'A-Z' 'a-z' wl | uniq -c | sort -d > outfile
  ```

B.5 Count lines, words, characters: wc

`wc infile` counts lines, words, characters

`wc -l infile` counts lines

`wc -w infile` counts words

`wc -m infile` counts characters

B.6 Display the first n lines of a file: sed

`sed 5q infile` displays the first 5 lines of the file infile

`sed 50q infile` displays the first 50 lines of the file infile

B.7 Find lines: grep, egrep

```
grep gh            find lines containing 'gh'

grep '^gh'         find lines beginning with 'gh'

grep 'hg$'         find lines ending with 'hg'

grep '^gh.*hg$'    find lines beginning with 'gh'
                   and ending with 'hg'

grep -v gh         delete lines containing 'gh'

grep -v '^gh'      delete lines beginning with 'gh'

grep -v 'gh$'      delete lines ending with 'gh'

grep -v '^gh.*hg$' delete lines beginning with 'gh'
                   and ending with 'hg'
```

Note with egrep you can specify disjunctive search patterns; for instance:

```
egrep 'ful$|ly$|ant$'   find lines ending with
                        'ful', 'ly', or 'ant'
```

B.8 n-grams: tail, paste

Suppose we have got a text file with each word on a single line. Now we can easily create all sorts of n-grams simply by using tail and paste. wl contains the original word list, nextwords contains the original word list reduced by the first element, nextnextwords contains the original word list reduced by the first two elements. The files bigrams, and trigrams contain all bigrams or trigrams respectively occurring in the text.

- Create bi-grams: tail, paste

  ```
  tail +2 wl > nextwords
  paste wl nextwords > bigrams
  ```

- Create tri-grams:

  ```
  tail +3 wl > nextnextwords
  paste wl nextwords nextnextwords > trigrams
  ```

- ...

B.9 Manipulation of lines and fields: awk

Despite **awk** is a general purpose programming language it is intended for shorter programs, especially for easy manipulation of lines and fields. Note, awk is fully integrated into Perl. Warning: awk also comes as **nawk**, and **gawk**.

The following is useful to know:

- Fields are addressed by $ fieldnumber

$1	first field
$NF	last field
$(NF-1)	penultimate field

- Some operators

  ```
  >, <, >=, <=, ==, =!, etc.
  ```

  For more information see the man pages.

- With -F you specify the field seperator, e.g. -F: specifies : as current field seperator. Blank is the default field seperator.

- Select fields by position

  The following command line specifies : as the current field seperator, and prints the 4th field of each line (record) to standard output. The input file is infile.

  ```
  awk -F: '{print $4}' infile
  ```

- Filter fields by numerical comparison

  The following command line matches fields where the numerical value of the first field is larger than 2, and prints the contents of the 3rd field of the according record to standard output. The file seperator is set to default (i.e. blank). The input file is infile.

  ```
  awk '$1 > 2 {print $3}' infile
  ```

- Filter fields by string comparison

  The following command line matches lines where the first field is identical to the last one, and prints the contents of the 3rd field to standard output. The file seperator is set to default (i.e. blank). The input file is again infile.

  ```
  awk '$1 == $NF {print $3}' infile
  ```

Appendix C

Some Calculus

C.1 Numbers

C.1.1 Natural Numbers

$$N = \{0, 1, 2, \ldots\}$$

The natural numbers are the natural ones when counting, i.e., for answering questions of the type "How many articles did Brigitte write this year?".

The most important formal property of these numbers is that every one of them has a unique successor, i.e., there are countably infinitely many of them, and there is a linear order defined on them.

Also, one can add and multiply natural numbers, e.g., $2 + 3 = 5$ and $2 \cdot 3 = 6$. Thus, addition ($+$) and multiplication ($\cdot$) are defined for these numbers.

C.1.2 Integers

$$Z = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

$$Z^+ = \{1, 2, \ldots\}$$

If we can add natural numbers, we can also formulate equations over them, e.g., $2+x = 5$. However, to guarantee that we always have a solution to these additive equations, we are forced to extend the set of natural numbers with the set of negative integers to form the set of integers, e.g., when we want to solve the equation $5 + x = 2$.

This means that for each $n \in N$ we introduce the additive inverse $-n$ with the property that $n + (-n) = 0$. $n + (-m)$ is usually written $n - m$ if $m \in Z^+$. Note that 0 is its own inverse, since $0 + 0 = 0$. In fact, $x + 0 = 0 + x = x$ for all numbers x, and 0 is formally referred to as the neutral element, or identity, of addition.

C.1.3 Rational Numbers

The set of rational numbers Q is the set of all numbers of the form $\frac{m}{n}$, with $m, n \in Z, n \neq 0$.

$$Q = \{x : x = \frac{m}{n}, m, n \in Z; n \neq 0\}$$

We can also formulate multiplicative equations over the natural numbers and the integers, e.g., $2 \cdot x = 6$. If we want to guarantee that we always have a solution to these equations, we are forced to extend the set of integers to the set of rational numbers, e.g., when we want to solve the equation $2 \cdot x = 5$.

Analogous to extending the natural numbers with their additive inverses, we will extend the integers with their multiplicative inverses. i.e., for each integer n we add n^{-1} with the property that $n \cdot n^{-1} = 1$. $m \cdot n^{-1}$ is usually written m/n or $\frac{m}{n}$. The only exception is the number 0, which has no inverse, since $0 \cdot x = 0 \neq 1$ for all numbers x. Similarly, 1 is the neutral element, or identity, of multiplication, since $x \cdot 1 = 1 \cdot x = x$ for all numbers x, and 1 is its own multiplicative inverse.

Since we don't want to be able to multiply ourselves out of Q, we need to add a lot of other numbers, apart from n^{-1} for all integers n, namely $m \cdot n^{-1}$ for all pairs of $m, n \in Z$. Note that some pairs correspond to the same rational number, e.g. $\frac{2}{4} = \frac{1}{2}$. This is called constructing the multiplicative closure of $\{0, n, n^{-1} : 0 \neq n \in Z\}$. We should really construct the additive and multiplicative closure, which is known as the *algebraic closure* of this set, but here, the multiplicative closure alone suffices.

Since each rational has an additive inverse and each rational, except 0 has a multiplicative inverse, and since addition and multiplication are both commutative operations ($m + n = n + m$ and $m \cdot n = n \cdot m$), the rational numbers constitute what is in algebra known as a *field*.

There is also a linear order defined on the rationals Q. Note that, as opposed to the case with the intergers, there is always another rational number between any two distinct rational numbers, i.e., if we have $p < q$, then there is an $r \in Q$ such that $p < r < q$.

C.1.4 Real Numbers

We can formulate nonlinear equations over Q, e.g., $x \cdot x = 2$. The fact that there is no rational number that solves this equation came as a shock for Pythagoras and his followers, a shock that they never fully recovered from. The proof of this is rather simple, and runs roughly as follows: The basic idea is that if $x = \frac{m}{n}$ solves this equation, where m and n are integers with no common factors, then both m and n are divisible by 2, and thus *do* have a common factor. Now, any rational number can be written on the form $x = \frac{m}{n}$, where the integers m and n have no common factors, so we must conclude that there is no rational solution to this equation.

There is however another way to escape from the set of rationals using the linear order, rather than addition and multiplication, namely by the use of limits, see Section C.2.6. Assume that $a_n \in Q$ for $n = 1, 2, \ldots$ This does not imply that $\lim_{n\to\infty} a_n \in Q$. We will informally define the set of real numbers as is the set of limits of sequences (a_n) over Q, rather than mucking around with Dedekind cuts, although this definition is circular, since the real numbers are used to define limits.

Like the rational numbers, the real numbers have the property that each of them has an additive inverse and each one of them, except 0, has a multiplicative inverse, and also here, addition and multiplication are both commutative operations. Thus, the real numbers also constitute a field.

There is also a linear order defined on the real numbers, like on the rationals Q. Unlike the rationals, though, if $a_n \in R$ for $n = 1, 2, \ldots$, then the limit $\lim_{n\to\infty} a_n \in R$. Thus the set of real numbers is closed under limits. This property is referred to as constituting a *complete metric space*.

Since we defined R as the set of limits of sequences in Q, there is a rational number arbitrarily close to any real number. This means that Q is *dense* in R.

C.1.5 Complex Numbers

Let x^2 denote $x \cdot x$, see Section C.2.1, and assume that we want to solve the equation $x^2 = -1$. There is no real number that solves this equation, since for real numbers, $x^2 \geq 0$. If we extend the real field with a solution to this equation, denoted i, and take the algebraic closure, we get the complex numbers C, which also constitute a field.

The complex numbers are usually written on the form $z = x + iy$, where $x, y \in R$. x is called the *real part* of z (as you might have guessed), and y is called the *imaginary part* of z.

This extension actually allows us to solve any equation of the form

$$c_n z^n + c_{n-1} z^{n-1} + \cdots + c_1 z + c_0 = 0 \qquad \text{with} \qquad c_i \in C$$

in the sense that

$$\forall z \in C \quad c_n z^n + c_{n-1} z^{n-1} + \cdots + c_1 z + c_0 = (z - z_1) \cdot \ldots \cdot (z - z_n)$$

Note that the roots (i.e., solutions) z_i need not be distinct.

This means that there are no natural algebraic extensions of C, unlike the case with N, Q and R. So from this point of view, we've reached the end of the line.

However, we do not have any linear order defined on the complex numbers. We can nevertheless introduce a distance between any two complex numbers $z_1 = x_1 + iy_1$ and $z_2 = x_2 + iy_2$, namely the Euclidian distance $d(z_1, z_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Just like for the real numbers, it is the case that any sequence $z_1, z_2, \ldots$ in C that converges (using the Euclidian distance), converges to an element in C, so also from this point of view C is complete (i.e., a complete metric space), and no extension to the complex numbers suggests itself.

C.1.6 Algebraic and Transcendental Numbers

Assume that all the cofficients c_i of the equation above are rational numbers (or, equivalently, integers), i.e:

$$a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0 = 0 \qquad \text{with} \qquad a_i \in Q$$

The solutions to these equations are known as the *algebraic numbers over Q*. Note that the rationals and thus the integers and natural numbers are all algebraic over Q. Other examples are $\sqrt{3}$, $\sqrt[12]{2}$ and $\dfrac{1 + \sqrt{5}i}{2}$.

However, not all real numbers are algebraic over Q. It is actually the case that the overwhelming majority of the complex and real numbers are not algebraic over Q. These numbers are called *transcendental numbers* (over Q). The transcendental numbers include such celebrities as π, the ratio of the circumference and the diameter of a circle, and e, the base of the natural logarithm. This was first observed by the German mathematician Cantor, who proved the existence of (a very large number of) transcendental numbers without actually constructing a single one. He did this by comparing the sizes of the set of algebraic numbers, which is countable, and the set of real numbers, which is uncountable. Thus, there must be an awful lot of non-algebraic, i.e., transcendental, numbers.

For further details on the number systems, consult any introductory book on the topic.

C.2 The Very Basics

C.2.1 Exponentials

If not indicated differently, the following holds for $a, b \neq 0$, $n, m \in Z$, $r, s, p, q \in Z^+$

$$a^n = a^{n-1} \cdot a$$

$$a^0 = 1$$

$$a^{-n} = \frac{1}{a^n}$$

$$0^r = 0$$

$$0^0 \quad \text{not defined}$$

$$0^{-n} \quad \text{not defined}$$

$$a^n \cdot a^m = a^{n+m}$$

$$a^{\frac{p}{q}} \cdot a^{\frac{r}{s}} = a^{\frac{p}{q}+\frac{r}{s}} \qquad \text{obviously} \qquad q, s \neq 0$$

$$\frac{a^r}{a^s} = a^{r-s} \qquad \text{for} \qquad r, s, \in Z$$

$$\frac{a^{\frac{p}{q}}}{a^{\frac{r}{s}}} = a^{\frac{p}{q}-\frac{r}{s}} \qquad \text{obviously} \qquad q, s \neq 0$$

$$(a^r)^s = (a^s)^r = a^{rs} \qquad \text{but note that} \qquad (a^s)^r \neq a^{(s^r)} \qquad \text{for} \qquad r, s, \in Z$$

$$(a \cdot b)^m = a^m \cdot b^m$$

$$(\frac{a}{b})^r = (\frac{a^r}{b^r}) \qquad \text{for} \qquad r, s, \in Z$$

$$\frac{a^r}{a^s} = \begin{cases} a^{r-s} & r > s \\ 1 & r = s \\ \frac{1}{a^{s-r}} & r < s \end{cases} \qquad \text{for} \qquad r, s, \in Z$$

$$a > 1: \qquad r < s \qquad \text{iff} \qquad a^r < a^s$$

$$0 < a < 1: \qquad r < s \qquad \text{iff} \qquad a^r > a^s$$

$$(a+b)^2 = a^2 + 2ab + b^2$$

$$(a+b)^3 = a^3 + 3a^2b^2 + b^3$$

or generally:

$$(a+b)^n = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) a^k b^{n-k}$$

$$a^2 - b^2 = (a-b)(a+b)$$

$$a^3 - b^3 = (a-b)(a^2 + ab + b^2)$$

or generally:

$$a^n - b^n = (a-b)(a^{n-1} + ba^{n-2} + \cdots + b^{n-2} + b^{n-1})$$

C.2.2 Roots

The following holds for $a, b \in R_0^+, n \in Z$:

$$\sqrt[n]{a} = b \qquad \text{iff} \qquad b^n = a$$

$$\sqrt[1]{a} = a$$

$$\sqrt[n]{0} = 0$$

$$\sqrt[n]{1} = 1$$

$$(\sqrt[n]{a})^n = \sqrt[n]{a^n} = a$$

$$a^{\frac{1}{n}} = \sqrt[n]{a}$$

$$a^{\frac{p}{q}} = \sqrt[q]{a^p} \qquad \text{for} \qquad p, q \in Z^+$$

$$(a^{\frac{1}{n}})^n = a$$

$$(a^{\frac{p}{q}})^q = a^p \qquad \text{for} \qquad p, q \in Z^+$$

$$\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$$

$$\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$$

$$(\sqrt[n]{a})^r = \sqrt[n]{a^r}$$

$$(\sqrt[n]{a})^{-r} = \sqrt[n]{a^{-r}}$$

$$\sqrt[s]{\sqrt[q]{a}} = \sqrt[q]{\sqrt[s]{a}} = \sqrt[sq]{a}$$

$$\sqrt[q]{a^p} = \sqrt[qk]{a^{pk}}$$

If you did not know it already, you might have realized from the above that exponentials and roots are inverses over R_0^+.

C.2.3 The Exponential Function

The exponential function exp(x) also known as e^x is defined as follows:

$$exp(x) := \sum_{k=0}^{\infty} \frac{x^k}{k!} = e^x \ , x \in R$$

$$exp(1) = \sum_{k=0}^{\infty} \frac{1^k}{k!} = e$$

For the exponential function e^x it holds that:

$$exp(x) \cdot exp(y) = exp(x+y) \qquad \text{,for all} \qquad x, y \in R$$

This is called the **addition theorem** for exponential functions. Now we can derive

$$exp(0) = 1$$

Which followes from

$$exp(x) = exp(x+0) = exp(x) \cdot exp(0)$$

$$exp(x) = exp(x) \cdot exp(0)$$

$$exp(0) = \frac{exp(x)}{exp(x)} = 1$$

From this it followes that

$$exp(-x) = \frac{1}{exp(x)}$$

We can see this from

$$exp(-x) \cdot exp(x) = exp(x - x) = exp(0) = 1$$

For all $x \in R$, the exponential function $exp(x)$ is positive:

$$exp(x) > 0$$

The exponential function $exp(x)$ is continuous, and strictly monotonically increasing.

$$\lim_{x \to -\infty} exp(x) = 0$$

$$\lim_{x \to \infty} exp(x) = \infty$$

For the exponential function $exp(x)$ the following equations hold:

$$e = exp(1) = exp(n \cdot \frac{1}{n}) = exp(\frac{1}{n} + \frac{1}{n} + \cdots + \frac{1}{n}) = (exp\frac{1}{n})^n$$

$$\sqrt[n]{e} = \sqrt[n]{(exp\frac{1}{n})^n} = exp(\frac{1}{n}) = e^{\frac{1}{n}}$$

$$e^{\frac{m}{n}} = exp(\frac{m}{n}) = exp(m \cdot \frac{1}{n}) = exp(\frac{1}{n})^m = \sqrt[n]{e^m} =$$

With the knowlege on exponential functions at hand, the formulas in C.2.1 and C.2.2 should have become much clearer now.

C.2.4 Logarithms for Beginners

We differentiate between the natural logarithm ln with base e, and the general logarithm $\log_a$ with base a, $a > 0$. The natural logarithm ln is the inverse of the exponential function $exp(x)$ in the interval $(0, \infty)$. Thus, the following holds:

$$\ln(exp(x)) = x \quad \text{, for all} \quad x \in R$$

$$exp(\ln(x)) = x \quad \text{, for all} \quad x \in R^+$$

$$y = \ln(x) \leftrightarrow x = e^y \quad \text{, for all} \quad x \in R^+$$

ln is continuous and strictly monotonically increasing.

$$\ln 1 = 0$$

$$\lim_{x \to 0} \ln(x) = -\infty$$

$$\lim_{x \to \infty} \ln(x) = \infty$$

The multiplication theorem states:

$$\ln(x \cdot y) = \ln(x) + \ln(y) \ , x, y > 0$$

From the multiplication theorem we can derive:

$$\ln(x^n) = n \cdot \ln(x)$$

$$\ln(\sqrt[n]{x}) = \frac{1}{n} \cdot \ln(x)$$

From e^x and $\ln(x)$, the general exponential function a^b with $a > 0$, $a \neq 1$, and $b \in R$ is definded as

$$a^b := exp(b \cdot \ln(a)) = e^{b \cdot \ln(a)}$$

Similar to e^x, the inverse of the general exponential function exists, which is called the general logarithm $\log_a$ with base a:

$$\log_a(a^x) = x \qquad \text{, for all} \qquad x \in R$$

From this it is obvious that:

$$\log_a(a) = 1$$

$$\log_a(1) = 0$$

We further have:

$$a^{\log_a(x)} = x \qquad \text{, for all} \qquad a > 0$$

and obviously

$$\log_a a^{\log_a(x)} = \log_a x \qquad \text{, for all} \qquad a > 0$$

$$y = \log_a(x) \leftrightarrow a^y = x \qquad \text{, for} \qquad x > 0$$

From the multiplication theorem we can derive

$$\log_a(x^y) = y \cdot \log_a(x)$$

$$\log_a(x^{-1}) = -\log_a(x)$$

The general logarithm $\log_a(x)$ is continuous, and strictly monotonically increasing for $a > 1$, and strictly monotonically decreasing for $0 < a < 1$.

$$\log_a(x \cdot y) = \log_a(x) + \log_a(y)$$

Of course, if $a = e$ we get the natural logarithm ln.

For the relation between logarithms with different bases see the following equation:

$$\log_a(y) = \frac{\ln(b)}{\ln(a)} \cdot \log_b(y) \qquad \text{, for} \qquad y > 0$$

From the following three equations we easily understand how the minus comes into the formula for entropy (cf. chapter Rfentropy, p. ??).

$$\log 1 = 0$$

$$\log(\frac{a}{b}) = \log(a) - \log(b) \qquad \text{obviously we have}$$

$$(-\log(x)) - (-log(y)) = -\log(\frac{x}{y})$$

Remember, we had

$$\frac{1}{p_i} = M_{p_i}$$

possible outcomes, which we rewrote as

$$log_2 \frac{1}{p_i} = log_2 M_{p_i}$$

which in turn equals

$$log_2 M_{p_i} = log_2 \frac{1}{p_i} = log_2 1 - log_2 p_i = 0 - log_2 p_i = -log_2 p_i$$

In the same chapter we also had

$$-p_i \ln p_i \to 0$$

This becomes clearer by considering that

$$\lim_{x \to 0} x \ln x = 0$$

which followes from the known limit, that x goes faster to zero than $\ln x$ goes to $-\infty$.

C.2.5 Factorial

$$n! = n \cdot (n-1)! = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot n \qquad , \qquad n \geq 1$$

$$0! = 1$$

C.2.6 Sequences

A sequence defines an ordering on the elements of a set. Consider for instance a sequence of real numbers, which is an enumeration of real numbers ordered by natural numbers. Each real number in a set of reals is related to a natural number. Thus, the reals are ordered according to the natural numbers. We have the function $f : N \to R$, $n \in N$, with $f(n)$ usually written as a_n.

$$a_1, a_2, a_3, \cdots, a_n, \cdots$$

The element a_n is called the n-th element of a sequence. The set of all elements a_i, i=1,2,3,...,n is called the range of the sequence (a_n).

For sequences, we say:

A sequence is called strictly monotonically increasing if for all n

$$a_n < a_{n+1}$$

A sequence is called monotonically increasing if for all n

$$a_n \leq a_{n+1}$$

A sequence is called strictly monotonically decreasing if for all n

$$a_n > a_{n+1}$$

A sequence is called monotonically decreasing if for all n

$$a_n \geq a_{n+1}$$

Upper and lower bounds

An upper bound of a sequence is an element k such that

$$a_n \leq k \qquad \text{, for all} \qquad n$$

A lower bound of a sequence is an element k such that

$$a_n \geq k \qquad \text{, for all} \qquad n$$

A sequence that has an upper bound is called upwards bounded.
A sequence that has a lower bound is called downwards bounded.
A sequence that has an upper bound and a lower bound is called upwards and downwards bounded.

The least upper bound is called the supremum. If some element in the sequence takes the value of the supremum, the supremum is called the maximum.
Equivalently, the greatest lower bound is called the infimum. If some element in the sequence takes the value of the infimum, the infimum is called the minimum.

Convergence

A sequence (a_n) is called convergent with a limit a, $a \in R$ if of all $\varepsilon > 0$ there is an index n_ε such that for all indices $n \geq n_\varepsilon$ it holds that $|a - a_n| < \varepsilon$.

In other words, all but a few elements of a convergent sequence (a_n) lie in the open interval $(a-\varepsilon, a+\varepsilon)$, which is called a neighbourhood of a. For the limit a we write:

$$a = lim_{n\to\infty} a_n$$

Characteristics of convergent sequences:

(i) A convergent sequence (a_n) has an unambiguously defined limit.

(ii) Convergent sequences are upwards and downwards bounded.

(iii) Given two convergent sequences $(a_n), (b_n)$ with $lim_{n\to\infty} a_n = a$ and $lim_{n\to\infty} b_n = b$ then the sum, difference and product are convergent, and the following holds:

$$lim_{n\to\infty}(a_n + b_n) = lim_{n\to\infty}(a_n) + lim_{n\to\infty}(b_n)$$

$$lim_{n\to\infty}(a_n - b_n) = lim_{n\to\infty}(a_n) - lim_{n\to\infty}(b_n)$$

$$lim_{n\to\infty}(a_n \cdot b_n) = lim_{n\to\infty}(a_n) \cdot lim_{n\to\infty}(b_n)$$

$$lim_{n\to\infty} c \cdot a_n = c \cdot lim_{n\to\infty}(a_n) \qquad \text{, for all} \qquad c \in R$$

The quotient $\frac{a_n}{b_n}$ with $b_n \neq 0$ is convergent, and the following holds:

$$lim_{n\to\infty} \frac{a_n}{b_n} = \frac{lim_{n\to\infty}(a_n)}{lim_{n\to\infty}(b_n)}$$

In case the limit a is unknown, convergence can be decided by the following criteria:

- the Cauchy criterion, and more practically relevant by
- the following theorem:
  A monotonically increasing, upwards bounded sequence is convergent.
  A monotonically decreasing, downwards bounded sequence is convergent.

The Cauchy criterion for sequences states:
A sequence (a_n) is convergent iff for all $\varepsilon > 0$ there is a n_ε such that for all $p, q \geq n_\varepsilon$ it holds that $|a_p - a_q| < \varepsilon$.

C.2.7 Series

From any sequence (a_n) we can define a series (s_n) by summing up the elements a_i of the the sequence (a_n). Thus, we have

$$a_1 + a_2 + a_3 + \cdots + a_n + \cdots = \sum_{k=1}^{\infty} a_k$$

$$s_n = a_1 + a_2 + a_3 + \cdots + a_n = \sum_{k=1}^{n} a_k$$

We say that the series (s_n) is the n-th partial sum of the sequence (a_n).

Convergence
If the series (s_n) converges to s, the series is said to converge. We write

$$\sum_{k=1}^{\infty} a_k = s$$

The number s is called the sum of the series or point of convergence, which in fact means that s is the limit of the sequence of partial sums of the series.

$$s = lim_{n\to\infty} s_n = lim_{n\to\infty} \sum_{k=1}^{n} a_k$$

The Cauchy criterion for convergence restated for series says:
$\sum_{k=1}^{\infty} a_k$ converges iff for all $\varepsilon > 0$ there is a n_ε such that for all $n \geq n_\varepsilon$ and for all $p \in N, p \geq 1$ it holds that

$$|\sum_{k=n+1}^{n+p} a_k| < \varepsilon$$

In the following, characteristics of convergent series are given:

For two convergent series $\sum_{k=1}^{\infty} a_k$ with sum s, and $\sum_{k=1}^{\infty} b_k$ with sum t, the series $\sum_{k=1}^{\infty} a_k + b_k$ and $\sum_{k=1}^{\infty} a_k - b_k$ are also convergent, and the following holds:

$$\sum_{k=1}^{\infty} a_k + b_k = s + t$$

$$\sum_{k=1}^{\infty} a_k - b_k = s - t$$

Given a convergent series $\sum_{k=1}^{\infty} a_k$ with sum s:

$$\sum_{k=1}^{\infty} c \cdot a_k = c \cdot s \qquad \text{, for} \qquad c \in R \qquad \text{is convergent}$$

For the elements $a_i,\ i = 1, 2, 3, \cdots, n$ of a convergent series (s_n) the following holds:

$$\lim_{n\to\infty} a_n = 0$$

In other words, the elements of a convergent series (s_n) form a sequence (a_n) that rapidly goes to zero.

The series $\sum_{k=1}^{\infty} a_k$ is said to converge absolutely if the series $\sum_{k=1}^{\infty} |a_k|$ converges. From this it follows that for the absolutely convergent series $\sum_{k=1}^{\infty} a_k$ with sum s, and $\sum_{k=1}^{\infty} b_k$ with sum t, any series of the products $a_k \cdot b_l,\ k, l \geq 1$ is absolutely convergent, and

$$\lim_{n\to\infty} \sum_{k=1}^{n} a_k \sum_{l=1}^{n} b_l = s \cdot t$$

By definition, a non-negative convergent series $\sum_{k=1}^{\infty} a_k$, where thus $a_k > 0$, is absolutely convergent.

Theorem: If $\sum_{k=1}^{\infty} a_k$ converges absolutely, then $\sum_{k=1}^{\infty} a_k$ converges.

In the following three sufficient conditions for convergence of series are given:

Comparison test

Given two series $\sum_{k=1}^{n} b_n$ and $\sum_{k=1}^{n} a_n$, and $0 \leq a_n \leq b_n$,
(i) $\sum_{k=1}^{n} a_n$ is convergent if $\sum_{k=1}^{n} b_n$ is convergent,

(ii) $\sum_{k=1}^{\infty} b_k$ is divergent if $\sum_{k=1}^{\infty} a_k$ is divergent.

If

$$\lim_{n \to \infty} \frac{a_n}{b_n} = s \neq 0$$

this can be restated as

$\sum_{k=1}^{n} a_n$ is convergent iff $\sum_{k=1}^{n} b_n$ is convergent.

Root test

If $\sqrt[k]{|a_k|} \leq q < 1$, for all $k \geq 1$, then $\sum_{k=1}^{\infty} a_k$ is absolutely convergent.

If $\sqrt[k]{|a_k|} \leq q > 1$, for all $k \geq 1$, then $\sum_{k=1}^{\infty} a_k$ is divergent.

If $\sqrt[k]{|a_k|} \leq q = 1$, for all $k \geq 1$, no information on convergence can be derived.

Ratio test

If $\frac{|a_{k+1}|}{|a_k|} \leq q < 1$, for $k \geq k_0$, then $\sum_{k=1}^{\infty} a_k$, with $a_k \neq 0$ is absolutely convergent.

If $\frac{|a_{k+1}|}{|a_k|} \leq q \geq 1$, for $k \geq k_0$, then $\sum_{k=1}^{\infty} a_k$, with $a_k \neq 0$ is divergent.

Theorem: Leibniz

Suppose, a series $\sum_{k=1}^{\infty} a_k$ has the following characteristics,

(i) the sequence of its elements is positive and monotonically decreasing

$$|a_1| \geq |a_2| \geq |a_3| \geq \cdots$$

(ii) negative and positive elements alternate

$$a_{2m-1} \geq 0, \ \ , a_{2m} \leq 0 \ , m = 1, 2, 3, \cdots$$

(iii) the sequence converges to 0

$$\lim_{n \to \infty} a_n = 0$$

Then, the series $\sum_{k=1}^{\infty} a_k$ converges.

C.3 On the Number e, the Exponential Function and the Natural Logarithm

The number e has received a lot of attention in Mathematical Analysis for several reasons. One often sees the following as the definition of $e \approx 2.72$:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

e is not an integer, nor a rational number, nor, in fact, an algebraic one. Alternatively one sees the following limit as the definition of e:

$$e = \lim_{n \to \infty} (1 + \frac{1}{n})^n$$

The exponential function is denoted e^x or $\exp(x)$. It is defined on the set of real numbers R as

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

In fact, it is defined in the same way also for complex numbers. The reason that this function has received so much attention is the fact that it is its own derivative, i.e:

$$\frac{d}{dx} e^x = e^x$$

Actually, Ce^x is the only function for which this is the case. Here C is an arbitrary multiplicative constant. This is true not only viewed as a real-valued function of a real variable, but also as a complex-valued function of a complex variable.

The inverse function of e^x, i.e., the logarithm with base e, is called the natural logarithm and "$\log_e x$" is often denoted "$\ln x$". Extending this function to the complex plane is slightly more complicated, and $\ln x$ is only defined for positive x when viewed as a real function.

The following integral keeps popping up when working with the Normal distribution:

$$\int_{-\infty}^{\infty} e^{-t^2}\, dt \;=\; \sqrt{\pi}$$

This can most easily be seen by noting that the integral over the real plane R^2 should be that same in both Cartesian and Polar coordinates:

$$(\int_{-\infty}^{\infty} e^{-t^2} dt)^2 \;=\; \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-(x^2+y^2)} dx\ dy \;=$$
$$\int_{0}^{\infty} \int_{0}^{2\pi} re^{-r^2} dr\ d\theta \;=\; 2\pi[-\frac{1}{2}e^{-r^2}]_0^{\infty} \;=\; \pi$$

For those who are a bit worried about the terms Cartesian and Polar coordinates: Cartesian coordinates are the usual (rectangular) coordinates of the real plane, often referred to as the x and y coordinates. Polar (circular) coordinates describe the real plane by giving the distance to the origin and the angel, in radians, with the x axis. For example, the point $(1, 1)$ in Cartesian coordinates corresponds the the point $(\sqrt{2}, \pi/4)$ in Polar coordinates.

Appendix D

Tagsets

For the annotation of written text we distinguish two classes of tags: word level and phrase level tags. Word level tags represent information related to lexical descriptions, i.e. information related to single word forms or lemmas. Phrase level tags represent information related to phrasal descriptions, i.e. the role a lexical element plays within a phrase, and the relations that hold between phrases.

D.1 Word Level Tagsets

D.1.1 Representation of Word Level Tags

The most wellknown species of tagsets are part-of-speech (PoS) tagsets. They have become popular because of the variety of recent approaches to PoS-tagging. PoS-tagsets ideally represent exhaustive information on syntactic category, morphological features, punctuation, and some other kind of word level information such as symbol, abbreviation, foreign material, brackets, quotes, etc. A tag represents a specific portion of information related to lexical descriptions. This may either be

- a single dimension of linguistic information, see for instance the Constraint Grammar tags for syntactic category such as V, A, ADV, DET, N (verb, adjective, adverb, determiner, noun), p. 146, or
- various combinations of dimensions of linguistic information, see for instance the Susanne wordtags for verbs where syntactic category, information on transitivity and inflection are combined. [Tzoukermann *et al* 1995] propose tags that are composed from syntactic category and morphological features, e.g. VIP3S the tag representing the information: verb, indicative, present, third person, singular.

Another way of combining linguistic information is by conjunction of tags. See for instance the Constraint Grammar tagset, where a small number of PoS-tags, representing syntactic category, is combined with sets of PoS-specific features, p. 146. The word level annotation `$<$SVO$>$ V PRES -SG3 VFIN @+FMAINV` for instance represents the following linguistic information: transitive (<SVO>), verb (V), present (PRES), non-third singular (-SG3), finite (VFIN), matrix verb (@+FMAINV).

In some tagsets, information on word level is combined with phrase level information. In Constraint Grammar, for instance, phrase level information is annotated at word level, cf.@+FMAINV. Another example are the Stuttgart-Tübingen tags ADJA and ADJD representing information on the adjective's grammatical function, namely attributival (ADJA) vs. adverbial or predicative (ADJD), or the distinction between prepositions (APPR) and postpositions (APPO), p. 152.

D.1.2 Mapping between Linguistic Descriptions and Tagsets

Armstrong et al. ([Armstrong *et al.* 1995]) for instance propose mapping between tags and descriptions with manually defined mapping tables. Linguistic descriptions are envisaged to be either represented as attribute value pairs, e.g. cat = prep is mapped to the tag PREP, or as typed feature structures, such as N[num=pl] which is mapped to the tag NPL, or as shortcuts for feature combinations, such as BD3FP (personal pronoun, third person, femininum, plural) which is maped to the less informative tag bfp (personal pronoun, femininum, plural), or V.sg.1.pres (verb, singular, first person, present) which is maped to the less informative tag VPR (verb, present tense). More consequently, Tzoukermann et al. ([Tzoukermann *et al* 1995]) regard tags as abbreviations of hierarchically organized features, with PoS as supertype. Thus, the information related to verbs is organized as followes: verb (PoS) is a supertype of mood, person, number, and tense.

D.1.3 Tagsets and the Representation of Ambiguity

A crucial feature of lexical description is ambiguity. A single word form may relate to various mutually exclusive portions of linguistic information. The English word form *stress* for instance is either a verb or a noun. In the verb reading, it relates to a finite or non-finite form, and the finite reading can be interpreted as either present indicative or imperative. In highly inflecting languages, such as German, inflection introduces a great amount of ambiguity. Therefore morhological information is largely excluded from tagsets, see for instance the small and the large version of Stuttgart-Tübingen tagset for German (STTS) [Thielen and Schiller 1995].

Ambiguity within tagsets is dealt with either by

- omitting portions of highly ambiguous linguistic description,

  cf. the small Stuttgart-Tübingen tagset;

- underspecification,

  see for instance the hierarchical organisation of the tag names, where the initial characters represent the most general information, typically the syntactic category, e.g. VB for verb in general in the Penn Threebank tagset, V in the Susanne tagset, and the STTS;

- disjunctions over tags or tag combinations.

D.1.4 Minimal Criteria for the Development of PoS-Tagsets

- Tags shall be kept as theory neutral as possible, in order to guarantee a high degree of descriptive generality.

- Tags should reflect a clear-cut distinction between different dimensions of linguistic description, in order to allow for a clear specification of the descriptive level, and for flexible and controlled access to various kinds of linguistic information via the tags.

- Tags must be defined and documented with examples in such a way that maximal intercoder consistency is guaranteed. Especilly problematic cases for annotation must be documented extensively.

To achieve compatibility of PoS-tagsets guidelines for standardized lexical description have been proposed, and tools for compatibility checks between tagsets have been developed. For the former see for instance the EAGLES activities on standardization of lexical descriptions (e.g. [Monachini and Calzolari 1994], [Teufel 1995b]). For the later, see for instance [Teufel 1995a], [Armstrong *et al.* 1995].

In the following, a number of tagsets for annotation of text (written language) is given. From the examples, we will see that tagsets provide more than mere PoS, but the information actually used in statistics reduces to a subset of approx. 50 to 100 PoS-tags. This is because when using large tagsets huge training corpora are necessary, which unfortunately do not exist.

D.2 Tagsets for English Text Corpora

English is the language the largest amount of annotated text is available for. Thus, most of the statistics-based approaches to NLP have been developed on English text.

D.2.1 The Susanne Tagset

The Susanne tagset is a classic among tagsets. The Susanne coprus has been derived from the Brown corpus ([Francis & Kucera 1982]). Annotation at word level in the Suzanne corpus is comparable to a data record: Each record of a Susanne file comprises six fields: reference, status, wordtag, word, lemma, and parse field. The Susanne annotation scheme distinguishes tags representing surface grammar (i.e. phrase structure), and tags representing logical grammar (i.e. argument structure).

The **reference field** contains information about which file the annotated word is part of, and where it occurrs in the Brown Corpus.

The **status field** indicates whether a word form is an abbreviation (A), a symbol (S), or a misprint (E). If none of those are applicable the status field contains a hyphen.

The **wordtag field** comprises 353 distinct wordtags (PoS-tags), plus additional tags for idioms. It is based on the "Lancaster" tagset listed in [Johansson *et al.* 1986], and enriched by additional grammatical distinctions which are indicated by lower-case suffixes. In the following we will only give a few examples on verbs, so that you get an idea of how Susanne wordtags look like. The gory details you'll find in [Sampson1995].

Command	Usage
VB0	be
VBDR	were
VBDZ	was
VBG	being
VBM	am
VBN	been
VBR	are
VBZ	is
VD0	do
VDD	did
VDG	doing
VDN	done
VDZ	does
VH0	have
VHD	had (past tense finite)
VHG	having
VHN	had (past participle)
VHZ	has
VMK	ought
VMd	modal (past)
VMo	modal (present)
VV0i	intransitive verb base form
VV0t	transitive verb base form
VV0v	base form with transitive verb or intransitive verb
VVDi	intransitive verb, past tense
VVDt	transitive verb, past tense
VVDv	transitive or intransitive verb, past tense
VVGi	present participle of intransitive verb
VVGt	present participle of transitive verb
VVGv	present participle of verb having (in)transitive uses
VVNi	past participle of intransitive verb
VVNt	past participle of transitive verb
VVNv	past participle of verb having (in)transitive uses
VVZi	intransitive verb, third person singular
VVZt	transitive verb, third person singular
VVZv	transitive or intransitive verb,third person

The **word field** contains what ever is considered to be a word token.

The **lemma field** contains the base form(s) of a word token. Typographical variation is eliminated.

The **parse field** gives information on syntactic structure. Formtags, functiontags, and indices are destinguished.

Formtags are comparable to labels assigned to the nodes in a parse tree. In the annotation, the parse tree is represented by labelled brackets. Wordlevel, phraselevel, clauselevel, and rootlevel (paragraph, heading, title etc.) formtags are distinguished. Wordlevel formtags (= wordtags) represent information related to terminal nodes(= lexical elements). Phrase level formtags give information on phrase types such as NP, VP, AP, AdjP etc. Clause level formtags give information on the clause type, such as main clause, adverbial clause, relative clause etc.

While formtags address surface grammatical properties, functiontags represent properties of logical grammar. Functiontags indicate the grammatical function of complements, such as subject, direct object, indirect object, prepositional object, and adjuncts such as place, direction, time, manner.

Indices specify the relation between surface and deep structure elements.

To give a flavour of the information encoded, the full list of rootlevel, clauselevel, and phraselevel formtags, and functiontags is presented below. Comments of the author are surrounded by brackets.

```
Rootlevel Formtags

O       paragraph
Oh      heading
Ot      title (e.g. of book)
Q       quotation
I       interpolation (material inserted into a grammatically
                       and semantically complete structure)
Iq      tag question
Iu      scientific citation

Clauselevel Formtags

S       main clause
Ss      quoting clause embedded within quotation
Fa      adverbial clause
Fn      nominal clause
Fr      relative clause
Ff      "fused" relative (relative clause lacking an explicit
                          antecedent, and functioning as a nominal
                          element)
Fc      comparative clause
Tg      present participle clause
Ti      infinitival clause
Tn      past participle clause
Tf      "for-to" clause
Tb      "bare" nonfinite clause
Tq      infinitival relative clause
Z       reduced ("whiz-deleted") relative clause (relative clause
              where relative pronoun and finite verb have been omitted)
L       other verbless clause
A       special "as" clause (clause beginning with the subordinating
               conjunction or comparative preposition 'as', and not
               being last part of an as-comparison, and 'as' is not
               interpretavble as 'because', 'while', or 'when'.
W       "with" clause

Phraselevel Formtags

N       noun phrase
V       verb group
J       adjective phrase
R       adverb phrase
P       prepositional phrase
```

```
D       determiner phrase
M       numeral phrase
G       genitive phrase

Vo      operator section of verb group
        (i.e. the first part of a split
              verb group, e.g. in the case of
              subject-auxiliary inversion)
Vr      remainder of verb group from which Vo has been separated (ie.
               the second part of a split verb group)
Vm      V beginning with "am"
Va      V beginning with "are"
Vs      V beginning with "was"
Vz      V beginning with other 3rd-singular verb
Vw      V beginning with "were"
Vj      V beginning with "be"
Vd      V beginning with past tense
Vi      infinitival V
Vg      V beginning with present participle
Vn      V beginning with past participle
Vc      V beginning with modal
Vk      V containing emphatic DO
Ve      negative V
Vf      perfective V
Vu      progressive V
Vp      passive V
Vb      V ending with BE
Vx      V lacking main verb
Vt      catenative V

Nq      "wh-" N
Nv      "wh...ever" N (e.g. 'whoever', 'whichever direction')
Ne      "I/me" head
Ny      "you" head
Ni      "it" head
Nj      adjective head
Nn      proper name
Nu      unit noun head
Na      marked as subject
No      marked as nonsubject
Ns      singular N
Np      plural N

Jq      "wh-" J (e.g. 'how good')
Jv      "wh...ever" J (e.g. 'however bad', 'no matter how high')
Jx      measured absolute J (e.g. '10 cm long')
Jr      measured comparative J (e.g. 'almost two years later than ...')
Jh      postmodified J (e.g. 'long enough', 'analoguous to ...')

Rq      "wh-" R
Rv      "wh...ever" R
Rx      measured absolute R
Rr      measured comparative R
Rs      adverb conducive to asyndeton (a phrasal 'as', 'then',
```

```
                                         'otherwise yet')
Rw      quasi-nominal adverb (phrase headed by one of the adverbs
               'here', 'there', 'now', 'then', 'anywhere',
               'elsewhere' , 'somewhere', 'nowhere')

Po      "of" phrase
Pb      "by" phrase
Pq      "wh-" P (e.g. 'after how many generations', 'of which')
Pv      "wh...ever" P (e.g. 'in whichever direction')

Dq      "wh-" D (e.g. 'how many', 'each of which')
Dv      "wh...ever" D (e.g. 'however much')
Ds      singular D (e.g. 'however much' Dvs)
Dp      plural D (e.g. 'how many' Dqp)

Ms      M headed by a cardinal number (e.g. 'several thousand',
                '1 billion or more')

?       interrogative clause
*       imperative clause
%       subjunctive clause
!       exclamatory clause or item
"       vocative item

+       subordinate conjunct introduced by conjunction
-       subordinate conjunct not introduced by conjunction
@       appositional element
&       co-ordinate structure acting as first conjunct within a
        higher co-ordination (marked in certain cases only)

WT&     co-ordination of words
WT+     conjunct within wordlevel co-ordination that is introduced
        by a conjunction
WT-     conjunct within wordlevel co-ordination not introduced by
        a conjunction


Complement Functiontags

s       logical subject
o       logical direct object
S       surface (and not logical) subject
O       surface (and not logical) direct object
i       indirect object
u       prepositional object
e       predicate complement of subject (e.g. He is not interested in
              being named 'a full-time director'.)
j       predicate complement of object (e.g. he established himself
              'as one of the guiding spirits')
a       agent of passive (i.e. by-object)
n       particle of phrasal verb (e.g. They slled 'up' the dean.)
z       complement of catenative
x       relative clause having higher clause as antecedent (e.g. He
              left early, 'which surprised me'.)
```

```
G       "guest" having no grammatical role within its tagma

Adjunct Functiontags

p       place
q       direction
t       time
h       manner or degree
m       modality
c       contingency
r       respect (phrases beginning with 'in connection with',
                'with respect to', etc.)
w       comitative (with-phrases meaning ''together_with'')
k       benefactive (e.g. she gave me a scarf 'for her son')
b       absolute (verbless adverbial clauses)
```

Susanne Annotation

```
A01:0010a       -       YB      $<$minbrk$>$    -       [Oh.Oh]
A01:0010b       -       AT      The     the     [O[S[Nns:s.
A01:0010c       -       NP1s    Fulton  Fulton  [Nns.
A01:0010d       -       NNL1cb  County  county  .Nns]
A01:0010e       -       JJ      Grand   grand   .
A01:0010f       -       NN1c    Jury    jury    .Nns:s]
A01:0010g       -       VVDv    said    say     [Vd.Vd]
A01:0010h       -       NPD1    Friday  Friday  [Nns:t.Nns:t]
A01:0010i       -       AT1     an      an      [Fn:o[Ns:s.
A01:0010j       -       NN1n    investigation   investigation   .
A01:0020a       -       IO      of      of      [Po.
A01:0020b       -       NP1t    Atlanta Atlanta [Ns[G[Nns.Nns]
A01:0020c       -       GG      +$<$apos$>$s    -       .G]
A01:0020d       -       JJ      recent  recent  .
A01:0020e       -       JJ      primary primary .
A01:0020f       -       NN1n    election        election  .Ns]Po]Ns:s]
A01:0020g       -       VVDv    produced        produce [Vd.Vd]
A01:0020h       -       YIL     $<$ldquo$>$ -       .
A01:0020i       -       ATn     +no     no      [Ns:o.
A01:0020j       -       NN1u    evidence        evidence        .
A01:0020k       -       YIR     +$<$rdquo$>$    -       .
A01:0020m       -       CST     that    that    [Fn.
A01:0030a       -       DDy     any     any     [Np:s.
A01:0030b       -       NN2     irregularities  irregularity    .Np:s]
A01:0030c       -       VVDv    took    take    [Vd.Vd]
A01:0030d       -       NNL1c   place   place   [Ns:o.Ns:o]Fn]Ns:o]Fn:o]S]
A01:0030e       -       YF      +.      -       .O]
```

Literature

An exhaustive description of the Susanne tagset is presented in [Sampson1995]. The Susanne corpus is freely available.

D.2.2 The Penn Treebank

The second version of the PennTreebank (Treebank II) comprises a million words of 1989 Wall Street Journal material. The corpus is annotated with PoS, and a labelled bracket structure that allows for the extraction of predicate-argument structure. The PoS labels , as presented in [Santorini 1995], are listed in section D.2.2. The bracket labels and syntactic functions are presented in section D.2.2. Labeling examples you'll find in section D.2.2. The list of labels, and the examples are taken from [Bies *et al.* 1995].

The Penn Treebank Parts-of-Speech

Command	Usage
CC	coordinating conjunction
CD	cardinal number
DT	determiner
EX	existential there
FW	foreign word
IN	preposition or subordinating conjunction
JJ	adjective
JJR	comparative adjective
JJS	superlative adjective
LS	list item marker
MD	modal
NN	noun, singular or mass
NNS	noun, plural
NNP	proper noun, singular
NNPS	proper noun, plural
PDT	predeterminer
POS	possessive ending
PRP	personal pronoun
PP$	possessive pronoun
RB	adverb
RBR	comparative adverb
RBS	superlative adverb
RP	particle
SYM	symbol
TO	infinitiv marker to
UH	interjection
VB	verb, base form
VBD	verb, past tense
VBG	verb, gerund or present participle
VBN	verb, past participle
VBP	verb, non-3rd person singular, present
VBZ	verb, 3rd person singular, present
WDT	wh-determiner
WP	wh-pronoun
WP$	possessive wh-pronoun
WRB	wh-adverb
#	Pound sign
$	Dollar sign
.	sentence final punctuation
,	comma
:	colon, semi-colon
(	left bracket character
)	right bracket character
”	straight double quote
‘	left open single quote

Command	Usage
“	left open double quote
’	right close single quote
”	right close double quote

The Penn Treebank II Syntactic Structure

Command	Usage
I Bracket Labels	
I.I Clause-Level Labels	
S	simple declarative sentence
SINV	subject-auxiliary inversion (not used with questions)
SBAR	relative or subordinate clause, incl. indirect questions
SBARQ	wh-question
SQ	within SBARQ (SBARQ consists of wh-element and SQ), labels yes/no question, and tag question
S-CLF	it-cleft, e.g. it was Casey who threw the ball
SQ-CLF	interrogative it-cleft, e.g. was it Casey who threw the ball
I.II Phrase-Level Labels	
RRC	reduced relative clause, complementizer and finite verb are missing, e.g. an orangutan 'foaming at the mouth', titles 'not presently in the collection'
FRAG	clause fragment
VP	verb phrase
NP	noun phrase
ADJP	adjective phrase
PP	prepositional phrase
ADVP	adverbial phrase
WHNP	wh-noun phrase, e.g. who
WHADVP	wh-adverbial phrase, e.g. why
WHADJP	wh-adjectival phrase,e .g. how cold
WHPP	wh-prepositional phrase, e.g. on what
QP	quantifier phrase
PRT	particle, i.e. separated verb prefix
UCP	unlike coordinated phrase
PRN	parenthetical
NX	head of a complex noun phrase
NAC	not a constituent; to show scope of certain prenominal modifiers in a noun phrase
INTJ	interjection, e.g. Mike, would you 'please' close the door
CONJP	conjunction phrase, only used with adjacent multi-element conjunctions, e.g. (CONJP not only) ... (CONJP but rather)
X	unknown, uncertain, unbracketable
II Function Tags	
II.I Text Categories	

Command	Usage
-HLN	headlines, datelines
-TTL	titles
-LST	list markers, i.e. mark list items in a text

II.II Grammatical Functions

-CLF	true clefts, see S-CLF, and SQ-CLF above
-NOM	non-NP functioning as NP, e.g. I do not mind about 'your' leaving early, 'what' I really like is chocolat
-ADV	clausal, and nominal adverbials, e.g. a little bit, you can leave 'if you really want to go'
-LGS	logical subjects in passives
-PRD	non-VP predicates, i.e. after copula verbs or in small claus e.g. Mary considers 'Peter a fool' (small clause)
-SBJ	surface subject
-TPC	topicalized, fronted constituent
-CLR	closely related, i.e. constituents between complement and adjunct, such as constituents closely related to the verb, elements in fixed phrases, elements in support verb constructions
-DTV	dative PP-object; only with verbs that undergo dative shif e.g. I baked a cake for John, I baked John a cake

II.III Semantic Roles

-VOC	vocative, 'Mike', would you please close the door
-DIR	direction, trajectory, e.g. 'from' Tokyo 'to' New York
-LOC	location
-MNR	manner
-PRP	purpose, reason
-TMP	temporal phrases
-BNF	benefactive; only with verbs that undergo dative shift
-PUT	locative complement of the verb 'put'
-EXT	extent, spatial extent of an activity, e.g. she walked '5 mile

III Null Elements

T	used for wh-movement, relative clauses, tough movement, parasitic gaps, and topicalization
(NP *)	used in passive constructions, control and raising, reduced relative clauses, participal clauses and gerunds, imperatives, and infinitives
0	the null complementizer, used in SBAR if wh-element or that is missing

Command	Usage
U	unit, marks interpreted position of a unit symbol
?	placeholder for ellipsed material
NOT	anti-placeholder in template gapping, used when template and copy in a coordinated structure are not entirely parallel
Pseudo-Attachment	to show attachment between non-adjacent constituents
ICH	interprete constituent here, used for discontinuous dependency
PPA	permanent predictable ambiguity, e.g. for structurally unresolvable pp-attachment
RNR	right node raising, indicating shared constituents
EXP	expletive

IV Coindexing

identity index	integer following a label tag, e.g. (SBAR-1)
reference index	integer following a null element, e.g. (NP *T*-3) the indices of the null element and the related constituent are identical; null element and related element must appear within the same sentence

Bracketed Structures

```
(S (NP-SBJ Casey)
   (VP will
       (VP throw
           (NP the ball))))


(S (NP-SBJ-1 the guide)
   (VP was
       (VP given
           (NP *-1)
           (PP-DTV to
                   (NP Arthur)
           (PP by
               (NP_LGS Ford))))))


(S (NP-SBJ (NP the person)
           (SBAR (WHNP-1 who)
                 (S (NP-SBJ *T*-1)
                    (VP threw
                        (NP the ball)))))
   (VP is
       (ADJP_PRD very athletic)))
```

Literature

For the description of the predicate-argument structure see [Marcus *et al.* 1994], for the example annotations and all other details see [Bies *et al.* 1995]. More information on the Treebank II can be found via `http:`/www.ldc.upenn.edu

D.2.3 The Constraint Grammar Tagset

The Constraint Grammar tagset comprises PoS-tags , and structural tags. PoS-tags give information on syntactic category and on morphosyntactic features related to specific syntactic categories. Additionally there are some other tags such as $2-NL representing a sequence of two or more newlines, and '*' indicating an upper case in words. Structural tags represent the syntactic function of a word, such as subject, object etc. The crucial thing to know is that in Constraint Grammar all information, even structural one, is expressed at word level.

The Constraint Grammar PoS-Tagset

Command	Usage
A	adjective
ABBR	abbreviation
ADV	adverb
CC	coordinating conjunction
CS	subordinating conjunction
DET	determiner
INFMARK>	infinitive marker such as 'to'
INTERJ	interjection
N	noun
NEG-PART	negative particle such as 'not'
NUM	numeral
PCP1	participial with -ing form
PCP2	participial with -ed/-en form
PREP	preposition
PRON	pronoun
V	verb
Features for abbreviations	
<Title>	title such as '*dr' for 'Doctor'
GEN	genitive
NOM	nominative
PL	plural
SG	singular
SG/PL	singular or plural
Features for adjectives	
<Attr>	attributive
<DER:al>	derived adjective in -al
<DER:ble>	derived adjective in -ble
<DER:ic>	derived adjective in -ic
<DER:ive>	derived adjective in -ive
<DER:less>	derived adjective in -less
<DER:like>	derived adjective in -like
<DER:ward>	derived adjective in -ward
<DER:wise>	derived adjective in -wise
<Nominal>	likely NP head
<Pred>	predicative
ABS	absolute form
CMP	comparative form
SUP	superlative form
Features for adverbs	
<**CLB>	clause boundary as it is introduced by the adverb why
<DER:bly>	derived adverb in -bly
<DER:ed>	derived adverb in -ed
<DER:ing>	derived adverb in -ing
<DER:ly>	derived adverb in -ly
<DER:ward>	derived adverb in -ward
<DER:wards>	derived adverb in -wards

Command	Usage
<DER:wise>	derived adverb in -wise
ABS	absolute form
CMP	comparative form
SUP	superlative form
WH	wh-adverb
Features for determiners	
<**CLB>	clause boundary as it is introduced by which
<Def>	definite
<Genord>	general ordinal such as next
<Indef>	indefinite such as an
<Quant>	quantifier such as some
ABS	absolute form such as much
ART	article
CENTRAL	central determiner
CMP	comparative form
DEM	demonstrative determiner
GEN	genitive
NEG	negative form
PL	plural as required from few
POST	postdeterminer e.g. much
PRE	predeterminer e.g. all
SG	singular
SG/PL	singular or plural
SUP	superlative form
WH	wh-determiner such as whose
Features for nouns	
<DER:bility>	derived noun in -bility
<DER:ble>	derived noun in -ble
<DER:er>	derived noun in -er
<DER:ing>	derived noun in -ing
<DER:ness>	derived noun in -ness
<DER:or>	derived noun in -or
<DER:ship>	derived noun in -ship
<NRare>	word only rarely used as a noun
<Proper>	proper
<-Indef>	noun with no indefinite article such as furniture
<Title>	title e.g. *professor
GEN	genitive case
NOM	nominative case
PL	plural
SG	singular
SG/PL	singular or plural
Features for numerals	
<Fraction>	fraction e.g. two-thirds
CARD	cardinal numeral
ORD	ordinal numeral
SG	singular e.g. one-eighth

Command	Usage
PL	plural e.g. three-eighths
Features for pronouns	
<**CLB>	clause boundary e.g. introduced to who
<Comp-Pron>	compound pronoun e.g. something
<Generic>	generic pronoun e.g. one's
<Interr>	interrogative
<NonMod>	pronoun with no DET or premodifier
<Quant>	quantitative pronoun
<Refl>	reflexive pronoun
<Rel>	relative pronoun
ABS	absolute form
ACC	accusative (objective) case
CMP	comparative form
DEM	demonstrative pronoun
FEM	feminine
GEN	genitive
INDEP	independent genitive form
MASC	masculine
NEG	negative form such as none
NOM	nominative
PERS	personal pronoun
PL	plural
PL1	1st person plural
PL2	2nd person plural
PL3	3rd person plural
RECIPR	reciprocal pronoun e.g. each other
SG	singular
SG/PL	singular or plural
SG1	1st person singular
SG2	2nd person singular
SG2/PL2	2nd person singular or plural
SG3	3rd person singular
SUP	superlative form
WH	wh-pronoun
Features for prepositions	
<CompPP>	multi-word preposition e.g. in spite of
Features for verbs	
<Arch>	archaic form
<DER:ate>	derived verb in -ate
<Rare>	word only rarely used as a verb
<Vcog>	verb that takes a that-clause
<SV>	intransitive
<SVO>	monotransitive
<SVOO>	ditransitive
<SVC/A>	copular with adjective complement
<SVC/N>	copular with noun complement such as become
<down/SVC/A>	copular with A, phrasal verb e.g. fall down

Command	Usage
<out/SVC/A>	copular with A, phrasal verb e.g. turn out
<out/SVC/N>	copular with N, phrasal verb e.g. turn out
<up/SVC/A>	copular with A, phrasal verb e.g. stand up
<up/SVC/N>	copular with N, phrasal verb e.g. end up
<SVOC/A>	complex trans. with adjective complement
<SVOC/N>	complex trans. with noun complement
<as/SVOC/A>	complex trans. with A, prepositional verb
<for/SVOC/A>	complex trans. with A, prepositional verb
<into/SVOC/A>	complex trans. with A, prepositional verb
-SG1,3	other than 1st or 3rd person sg.
-SG3	other than 3rd person sg.
AUXMOD	modal auxiliary
IMP	imperative
INF	infinitive
NEG	negative
PAST	past tense
PRES	present tense
SG1	1st person sg.
SG1,3	1st or 3rd person sg.
SG2	2nd person sg.
SG3	3rd person sg.
SUBJUNCTIVE	subjunctive
VFIN	finite form

Constraint Grammar Syntactic Tags

Command	Usage
@+FAUXV	Finite Auxiliary Predicator
@-FAUXV	Nonfinite Auxiliary Predicator
@+FMAINV	Finite Main Predicator
@-FMAINV	Nonfinite Main Predicator
@NPHR	Stray NP
@SUBJ	Subject
@F-SUBJ	Formal Subject
@OBJ	Object
@I-OBJ	Indirect Object
@PCOMPL-S	Subject Complement
@PCOMPL-O	Object Complement
@ADVL	Adverbial
@APP	Apposition
@N	Title
@DN>	Determiner
@NN>	Premodifying Noun
@AN>	Premodifying Adjective
@QN>	Premodifying Quantifier
@GN>	Premodifying Genitive
@AD-A>	Premodifying Ad-Adjective
@<NOM-OF	Postmodifying Of
@<NOM-FMAINV	Postmodifying Nonfinite Verb
@<AD-A	Postmodifying Ad-Adjective
@<NOM	Other Postmodifier
@INFMARK>	Infinitive Marker
@<P-FMAINV	Nonfinite Verb as Complement of Preposition
@<P	Other Complement of Preposition
@CC	Coordinator
@CS	Subordinator
@O-ADVL	Object Adverbial
@NEG	Negative particle ”not”
@DUMMY	A word without a syntactic function,

*the = The

Sample Constraint Grammar Analysis

```
"<*i>"
        "i" <*> <NonMod> PRON PERS NOM SG1 SUBJ @SUBJ
"<see>"
        "see" <as/SVOC/A> <SVO> <SV> <InfComp> V PRES -SG3 VFIN @+FMAINV
"<a>"
        "a" <Indef> DET CENTRAL ART SG @DN>
"<bird>"
        "bird" N NOM SG  @OBJ
"<$.>"
```

Literature

For a comprehensive introduction to ENGCG see [Karlsson *et al* 1995]. The material can be obtained by sending an emty email to engcgi@ling.helsinki.fi.

D.3 Tagsets for German Text Corpora

D.3.1 The Stuttgart-Tübingen Tagset

The Stuttgart-Tübingen tagset is a mere PoS-tagset. It comes in two versions, a small and a large one. In the large tagset PoS information is augmented by morphlogical information such as features for person, number, gender, case, tense, mode, comparation, and inflection type. In the following the inventary of the small version of the tagset is presented:

Command	Usage
ADJA	attributival adjektiv
ADJD	adverbial or predicative adjective
ADV	adverb
APPR	preposition ; circumposition left
APPRART	preposition with incorporated article
APPO	postposition
APZR	circumposition right
ART	definite or indefinite article
CARD	cardinal
FM	foreign material
ITJ	interjection
KOUI	subordinating conjunction with zu and infinitive
KOUS	subordinating conjunction with sentence
KON	coordinating conjunction
KOKOM	comaration particle, without sentence
NN	common noun
NE	proper noun Hans*Hamburg*HSV
PDS	substitutive demonstrative pronoun
PDAT	attributive demonstrative pronoun
PIS	substitutive indefinit pronoun
PIAT	attributive indefinit pronoun
PIDAT	attributive indefinit pronoun mit Determiner
PPER	irreflexive personal pronoun
PPOSS	substitutive possessive pronoun
PPOSAT	attributive possessive pronoun
PRELS	substitutive relative pronoun
PRELAT	attributive relative pronoun
PRF	reflexive personal pronoun
PWS	substitutive interrogativpronoun
PWAT	attributive interrogativpronoun
PWAV	adverbial interrogative- or relative pronoun
PROAV	pronominal adverb
PTKZU	"zu" before infinitive
PTKNEG	negation partikel
PTKVZ	abgetrennter verb prefix
PTKANT	Antwortpartikel
PTKA	partikel occurring with adjective or adverb
TRUNC	lexical ellipsis
VVFIN	finite verb, main
VVIMP	imperative, main
VVINF	infinitive, main
VVIZU	infinitive with "zu", main
VVPP	past partiziple, main
VAFIN	finite verb, aux
VAIMP	imperative, aux
VAINF	infinitive, aux

Command	Usage
VAPP	past artiziple, aux
VMFIN	finite verb, modal
VMINF	infinitive, modal
VMPP	past partiziple, modal
XY	non-word, Sonderzeichen
$,	comma
$.	sentence final punctation
$(	sentence internal punctuation

Annotation Example

```
M"ogen  VMFIN
Puristen        NN
aller   PIDAT
Musikbereiche   NN
auch    ADV
die     ART
Nase    NN
r"umpfen        VVINF
,       $,
die     ART
Zukunft NN
der     ART
Musik   NN
liegt   VVFIN
f"ur    APPR
viele   PIDAT
junge   ADJA
Komponisten     NN
im      APPRART
Crossover-Stil  NN
.       $.
Sie     PPER
gehen   VVFIN
gewagte ADJA
Verbindungen    NN
und     KON
Risiken NN
ein     PTKVZ
,       $,
versuchen       VVFIN
ihre    PPOSAT
M"oglichkeiten  NN
auszureizen     VVIZU
.       $.
```

Literature

[Thielen and Schiller 1995]

Appendix E

Optimization Theory, Wild-West Style

E.1 Introduction

Here we will discuss constrained optimization using Lagrange multiplyers. This requires that we first go through the following necessary prerequisites:

- R, the set of real numbers: $0, -1, 42, \frac{1}{7}, \sqrt{2}, e, \pi, \ldots$
- Intervals: $[-1, 0], (\frac{1}{7}, \sqrt{2}), [\pi, \infty), \ldots$
- Sequences and limits in R: $e = \lim_{n\to\infty}(1 + \frac{1}{n})^n$.
- $R^n = \overbrace{R \times \cdots \times R}^{n \text{ factors}}$.
- Functions $f(x)$ from R to R ($f : R \mapsto R$): $x + 5, x^2, \sin x, e^x$.
- Sequences and limits of functions in $R \mapsto R$: $\lim_{x\to x_0} f(x)$.
- Derivatives: $\frac{df(x)}{dx} = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$.
- Functions $f(x_1, \ldots, x_n)$ from R^n to R ($f : R^n \mapsto R$): $x^2 + y^2 + z^2$.
- Partial derivatives: $\frac{\partial f(x_1,\ldots,x_n)}{\partial x_i}$. Gradients: $\nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$.
- Optimization in one dimension:

$$\max_{x\in A\subseteq R} f(x)$$

- The solutions: Stationary points and boundary points.
- The zeros of the derivative: $\frac{df}{dx} = 0$.
- Optimization in several dimensions:

$$\max_{(x_1,\ldots,x_n)\in A\subseteq R^n} f(x_1, \ldots, x_n)$$

- The zeros of the partial derivatives: $\forall_i \frac{\partial f}{\partial x_i} = 0$. Or, equivalently, the zeros of the gradient: $\nabla f = \mathbf{0} = (0, \ldots, 0)$.

- Constrained optimization in several dimensions:

$$\begin{aligned} \max_{(x_1,\ldots,x_n)\in A\subseteq R^n} & f(x_1,\ldots,x_n) \\ g_k(x_1,\ldots,x_n) &= 0 \quad \text{for } k=1,\ldots,m \end{aligned}$$

- Lagrange multiplyers: $\nabla[f(x_1,\ldots,x_n)-\sum_{k=1}^m \lambda_k g_k(x_1,\ldots,x_n)] = 0$.

In addition to this, we need to recapitulate numerical analysis to see how to solve the resulting equations. Since this is clearly impossible within the given time, we will start from the end and work our way backwards.

E.2 Constrained Optimization in R^n

The basic problem that we want to solve is the following:

$$\begin{aligned} \max_{(x_1,\ldots,x_n)\in A\subseteq R^n} & f(x_1,\ldots,x_n) \\ g_k(x_1,\ldots,x_n) &= 0 \quad \text{for } k=1,\ldots,m \end{aligned}$$

This means that we want to maximize the real-valued object function f of n real variables $x_1,\ldots,x_n$ where the vector $\mathbf{x} = (x_1,\ldots,x_n)$ is restricted to the subset A of R^n, while observing m constraints of the form $g_k(x_1,\ldots,x_n) = 0$.

Example: A typical example would be estimating the parameters $v_j, p_{jk}, a_{jl} \in [0,1] \; : \; j,k = 1,\ldots,N \; ; \; l = 1,\ldots,M$ of a hidden Markov model with N states and M signals.

$$\begin{aligned} \max_{(v_1,\ldots,v_N,p_{11},\ldots,p_{NN},a_{11},\ldots,a_{NM})\in[0,1]^n} & P(\mathbf{O}) \\ \sum_{j=1}^N v_j &= 1 \\ \sum_{k=1}^N p_{jk} &= 1 \quad \text{for } j=1,\ldots,N \\ \sum_{k=1}^M a_{jl} &= 1 \quad \text{for } j=1,\ldots,N \end{aligned}$$

Here the probability $P(\mathbf{O})$ of the observed training data $\mathbf{O}$ is the object function, and the requirement that the various sets of probabilities sum to 1 constitutes the set of constraints. We have $n = N + N\cdot N + N\cdot M = N(N+M+1)$ variables x_i and $m = 1 + N + N = 2N+1$ constraints.

Let us simplify this problem slightly to the case where we we have no constraints, i.e., where $m = 0$. As we shall soon see, the points of interest would then under suitable conditions be those where $\nabla f(\mathbf{x}) = \mathbf{0}$. Intuitively, this can be understood as follows: $\nabla f(\mathbf{x})$ is a vector in R^n indicating

1. the direction in point $\mathbf{x}$ in which $f(\mathbf{x})$ grows the most,

2. and how fast it grows in this direction.

If $\mathbf{x}$ is a maximum, $f(\mathbf{x})$ doesn't grow at all in any direction, and $\nabla f(\mathbf{x})$ should thus be zero. Unfortunately, this is also true for a minimum, and for other so-called stationary points. In addition to this, $f(\mathbf{x})$ may not be particularly well-behaved, and may not allow us to form $\nabla f(\mathbf{x})$, i.e., this quantity may be undefined. We'll get back to this later. For the time being, maximizing $f(\mathbf{x})$ will involve inspecting the points where $\nabla f(\mathbf{x}) = \mathbf{0}$.

We have however not taken the constraints $g_k(\mathbf{x}) = 0$ into account. A clever way of doing this, due to the famous French mathematician Joseph Louis Lagrange (1736–1813), is to introduce so-called Lagrange multiplyers. Note that

$$\left\{ \begin{array}{lcl} \max_{\mathbf{x}\in A\subseteq R^n} f(\mathbf{x}) & & \\ g_k(\mathbf{x}) & = & 0 \quad \text{for } k = 1, \ldots, m \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{lcl} \max_{\mathbf{x}\in A\subseteq R^n} [f(\mathbf{x}) - \sum_{k=1}^m \lambda_k g_k(\mathbf{x})] & & \\ g_k(\mathbf{x}) & = & 0 \quad \text{for } k = 1, \ldots, m \end{array} \right\}$$

for any choice of real numbers λ_k since $g_k(\mathbf{x}) = 0$. It turns out that the points of interest when solving the constrained optimizations problem are, again under suitable conditions, precisely the points where

$$\left\{ \begin{array}{lcl} \nabla[f(\mathbf{x}) - \sum_{k=1}^m \lambda_k g_k(\mathbf{x})] = \mathbf{0} & & \\ g_k(\mathbf{x}) & = & 0 \quad \text{for } k = 1, \ldots, m \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{lcl} \nabla f(\mathbf{x}) = \sum_{k=1}^m \lambda_k \nabla g_k(\mathbf{x}) & & \\ g_k(\mathbf{x}) & = & 0 \quad \text{for } k = 1, \ldots, m \end{array} \right\}$$

The parameters $\lambda_1, \ldots, \lambda_m$ are determined by the new set of equations. Compared with the unconstrained optimization problem, we have increased the set of unknown variables from n to $n + m$ by introducing these parameters, but we have also added m equations $\{g_k(\mathbf{x}) = 0 : k = 1, \ldots, m\}$. We have thus given the system m extra degrees of freedom, but also added m extra constraints, a net change of zero degrees of freedom.

Example: In the example of estimating the parameters $v_j, p_{jk}, a_{jl} \in [0,1] \ : \ j, k = 1, \ldots, N \ ; \ l = 1, \ldots, M$ of a hidden Markov model with N states and M signals, this amounts to the following set of equations (P stands for $P(\mathbf{O})$):

$$\begin{array}{rcl} \dfrac{\partial P}{\partial v_j} & = & \lambda_1 \quad \text{for } j = 1, \ldots, N \\ \dfrac{\partial P}{\partial p_{jk}} & = & \lambda_{j+1} \quad \text{for } j = 1, \ldots, N \ ; \ k = 1, \ldots, N \\ \dfrac{\partial P}{\partial a_{jl}} & = & \lambda_{N+j+1} \quad \text{for } j = 1, \ldots, N \ ; \ l = 1, \ldots, M \\ \sum_{j=1}^{N} v_j & = & 1 \\ \sum_{k=1}^{N} p_{jk} & = & 1 \quad \text{for } j = 1, \ldots, N \\ \sum_{l=1}^{M} a_{jl} & = & 1 \quad \text{for } j = 1, \ldots, N \end{array}$$

By multiplying each equation by the appropriate parameter, and summing over the appropriate index (j in the first set of equations, k in the second one, and l in the third one), we can find the values of $\lambda_1, \ldots, \lambda_{2n+1}$ that satisfy the constraints:

$$\sum_{j=1}^{N} v_j \frac{\partial P}{\partial v_j} = \lambda_1$$

$$\begin{aligned}
\sum_{k=1}^{N} p_{jk}\frac{\partial P}{\partial p_{jk}} &= \lambda_{j+1} \quad \text{for } j = 1,\ldots,N \\
\sum_{l=1}^{M} a_{jl}\frac{\partial P}{\partial a_{jl}} &= \lambda_{N+j+1} \quad \text{for } j = 1,\ldots,N
\end{aligned}$$

Thus

$$\begin{aligned}
v_j &= \frac{v_j \frac{\partial P}{\partial v_j}}{\sum_{j=1}^{N} v_j \frac{\partial P}{\partial v_j}} \quad \text{for } j = 1,\ldots,N \\
p_{jk} &= \frac{p_{jk} \frac{\partial P}{\partial p_{jk}}}{\sum_{k=1}^{N} p_{jk} \frac{\partial P}{\partial p_{jk}}} \quad \text{for } j = 1,\ldots,N \ ; \ k = 1,\ldots,N \\
a_{jl} &= \frac{a_{jl} \frac{\partial P}{\partial a_{jl}}}{\sum_{l=1}^{M} a_{jl} \frac{\partial P}{\partial a_{jl}}} \quad \text{for } j = 1,\ldots,N \ ; \ l = 1,\ldots,M
\end{aligned}$$

It is not easy to prove that the points satisfying this new set of equations are exactly the set of interesting points for the constrained optimization problem, and we will refrain from attempting the proof. We instead offer the following intuitive explanation, based on reasoning about surfaces and curves in R^n:

$\nabla f(\mathbf{x})$, ∇f for short, is a normal to the surface $\{\mathbf{x} \in R^n : f(\mathbf{x}) = C\}$, $f = C$ for short. Similarly, ∇g_k is a normal to the surface $g_k = 0$. Any vector normal to the intersection curve of the surfaces $g_1 = 0$ and $g_2 = 0$ can be written as $\lambda_1 \nabla g_1 + \lambda_2 \nabla g_2$, where λ_1 and λ_2 are real numbers. Generalizing, any vector normal to the intersection curve of the surfaces $\{g_k = 0 : k = 1,\ldots,m\}$ can be written as $\sum_{k=1}^{m} \lambda_k \nabla g_k$, where $\lambda_1,\ldots,\lambda_m$ are real numbers.

Now comes the crucial bit: As we vary C, the surface $f = C$ will hopefully cut the intersection curve of the surfaces $\{g_k = 0 : k = 1,\ldots,m\}$ at some point(s) for some value(s) of C. Otherwise, the constrained optimization problem has no solutions. We want to maximize C. This will under suitable conditions happen when the surface $f = C$ just barely touches this intersection curve. This in turn will happen when a normal of the surface $f = C$ is also a normal to the intersection curve of the surfaces $\{g_k = 0 : k = 1,\ldots,m\}$. This means that ∇f is a normal both to the surface $f = C$ and to the intersection curve. Since any normal to the intersection curve can be written as $\sum_{k=1}^{m} \lambda_k \nabla g_k$, this will happen when $\nabla f = \sum_{k=1}^{m} \lambda_k \nabla g_k$.

E.3 Numerical Analysis

This is *not* the only thing there is to know about numerical analysis.

Assume that we wish to calculate $\sqrt{2}$, but that some nerd has knicked our belt-strapped scientific calculator, and left us with a useless economic calculator which only lends itself to the four basic arithmetic operations addition, subtraction, multiplication and division ($+, -, \times$, and $\div$), and, most importantly, percentage calculations (%). We then need to calculate a numerical approximation of $\sqrt{2}$ using only these four basic arithmetic operations.

We realize that $x = \sqrt{2}$ must satisfy

$$x^2 \;=\; 2 \quad \text{and} \quad x \;>\; 0. \tag{E.1}$$

and that this uniquely determines x.

The basic idea is to come up with a recurrence equation $x_{k+1} = f(x_k)$ that only involves addition, subtraction, multiplication and division, and that will allow us to calculate a better next estimate x_{k+1} of x from the current one x_k.

A first attempt might be to note that

$$x \;=\; \frac{2}{x}$$

and simply let $f(x) = 2/x$. Thus

$$x_{k+1} \;=\; \frac{2}{x_k}$$

Unfortunately, we see that

$$x_{k+1} \;=\; \frac{2}{x_k} \;=\; \frac{2}{\frac{2}{x_{k-1}}} \;=\; x_{k-1}$$

Thus, we won't get any closer to the value $\sqrt{2}$ this way. By instead adding x to both sides of Equation E.1, we find that

$$x^2 + x = 2 + x \quad \text{or} \quad x(x+1) = 2 + x \quad \text{or} \quad x = \frac{2+x}{x+1} \quad \text{or} \quad x = 1 + \frac{1}{x+1}$$

Thus

$$x_{k+1} \;=\; 1 + \frac{1}{x_k + 1}$$

Let $x_1 = 1$ and note that $\sqrt{2} \approx 1.41421$.

$$\begin{array}{rclcl}
x_1 & = & 1 & = & 1.00000 \\
x_2 & = & \frac{3}{2} & = & 1.50000 \\
x_3 & = & \frac{7}{5} & = & 1.40000 \\
x_4 & = & \frac{17}{12} & \approx & 1.41667 \\
x_5 & = & \frac{41}{29} & \approx & 1.41379 \\
x_6 & = & \frac{99}{70} & \approx & 1.41428
\end{array}$$

We see to our joy that this is rapidly approaching $\sqrt{2}$.

But why is this? We have that

$$\begin{aligned}
\mid x_{k+1}^2 - 2 \mid \;&=\; \mid (\frac{2+x_k}{x_k+1})^2 - 2 \mid \;=\; \mid \frac{4 + 4x_k + x_k^2 - 2 - 4x_k - 2x_k^2}{(x_k+1)^2} \mid \;=\\
&=\; \frac{1}{(x_k+1)^2} \mid x_k^2 - 2 \mid
\end{aligned}$$

We also note that $x_k \geq 1 \Rightarrow x_{k+1} = 1 + \frac{1}{x_k+1} > 1$. So if we start with $x_1 = 1$, then $x_k > 1$ for $k = 2, 3, \ldots$ This means that $\frac{1}{(x_k+1)^2} < \frac{1}{(1+1)^2} = \frac{1}{4}$. Thus

$$\mid x_{k+1}^2 - 2 \mid \;<\; \frac{1}{4} \mid x_k^2 - 2 \mid \;<\; (\frac{1}{4})^2 \mid x_{k-1}^2 - 2 \mid \;<\; (\frac{1}{4})^k \mid x_1^2 - 2 \mid \;=\; (\frac{1}{4})^k$$

Finally noting that $x_{k+1}^2 - 2 = (x_{k+1} - \sqrt{2}) \cdot (x_{k+1} + \sqrt{2})$, we see that

$$| \, x_{k+1} - \sqrt{2} \, | \; < \; (\frac{1}{4})^k \frac{1}{x_{k+1} + \sqrt{2}} \; < \; (\frac{1}{4})^k \frac{1}{1 + \sqrt{2}}$$

and that x_k approaches $\sqrt{2}$ with geometric speed as k tends to infinity.

Example: In the example of estimating the parameters $v_j, p_{jk}, a_{jl} \in [0,1] \, : \, j,k = 1,\ldots,N \; ; \; l = 1,\ldots,M$ of a hidden Markov model with N states and M signals, we arrived at the following set of equations:

$$\begin{aligned}
v_j &= \frac{v_j \frac{\partial P}{\partial v_j}}{\sum_{j=1}^N v_j \frac{\partial P}{\partial v_j}} \qquad \text{for } j = 1,\ldots,N \\
p_{jk} &= \frac{p_{jk} \frac{\partial P}{\partial p_{jk}}}{\sum_{k=1}^N p_{jk} \frac{\partial P}{\partial p_{jk}}} \qquad \text{for } j = 1,\ldots,N \; ; \; k = 1,\ldots,N \\
a_{jl} &= \frac{a_{jl} \frac{\partial P}{\partial a_{jl}}}{\sum_{l=1}^M a_{jl} \frac{\partial P}{\partial a_{jl}}} \qquad \text{for } j = 1,\ldots,N \; ; \; l = 1,\ldots,M
\end{aligned}$$

From this we can construct a set of recurrence equations with the nice property that either the next iteration of parameters will increase the value of $P(\mathbf{O})$ (the probability of the training data) or the value of the parameters will be the same. Since we have a lot of subscript indices, we will let $\hat{x}$ denote x_{k+1} and x denote x_k.

$$\begin{aligned}
\hat{v}_j &= \frac{v_j \frac{\partial P}{\partial v_j}}{\sum_{j=1}^N v_j \frac{\partial P}{\partial v_j}} \qquad \text{for } j = 1,\ldots,N \\
\hat{p}_{jk} &= \frac{p_{jk} \frac{\partial P}{\partial p_{jk}}}{\sum_{k=1}^N p_{jk} \frac{\partial P}{\partial p_{jk}}} \qquad \text{for } j = 1,\ldots,N \; ; \; k = 1,\ldots,N \\
\hat{a}_{jl} &= \frac{a_{jl} \frac{\partial P}{\partial a_{jl}}}{\sum_{l=1}^M a_{jl} \frac{\partial P}{\partial a_{jl}}} \qquad \text{for } j = 1,\ldots,N \; ; \; l = 1,\ldots,M
\end{aligned}$$

Bibliography

[Abney 1997] Steven Abney. Part-of-Speech Tagging and Partial Parsing. In Steve Young and Gerrit Bloothooft (eds.), *Corpus-Based Methods in Language and Speech Processing*. Kluwer, Dordrecht, The Netherlands, 1997.

[Aho & Ullman 1972] Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Computing*. Prentice-Hall, New jersey.

[Aho *et al* 1986] 1986. Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman. *Compilers, Principles, Techniques and Tools*, Addison-Wesley.

[Alshawi & Carter 1994] Hiyan Alshawi and David Carter. 1994. "Training and Scaling Preference Functions for Disambiguation". In *Computional Linguistics 20(4)*, pp. 635–648.

[Altmann & Steedmann 1988] Gerry Altmann and Mark Steedman. 1988. "Interaction with context during human sentence processing". In *Cognition 30*.

[Armstrong *et al.* 1995] Susan Armstrong, Graham Russel, Dominique Petitpierre and Gilbert Robert. An Open Architecture for Multilingual Text Processing. In *Proceedings of the ACL Sigdat Workshop*, pages 30 – 35, Dublin, Ireland, 1995.

[Ash 1965] Robert Ash. 1965. *Information Theory*. John Wiley, New York.

[Bahl *et al* 1989] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza and Robert L. Mercer. 1992. "A Tree-Based Statistical Language Model for Natural Language Speech Recognition". In *IEEE Transactions on Acoustics, Speech, and Signal Processing 36(7)*, pp. 1001–1008.

[Baum 1972] L. E. Baum. 1972. " An Inequality and Assiciated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes. In *Inequalities 3*, pp. 1–8.

[Bellman 1957] R. E. Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, New York.

[Bies *et al.* 1995] Ann Bies, Mark Ferguson, Karen Katz and Robert MacIntyre. Bracketing Guidelines for Treebank II Style Penn Treebank Project. Technical Report, Unniversity of Pennsylvania, 1995.

[Black *et al* 1992] Ezra Black, Fred Jelinek, John Lafferty, Robert Mercer and Salim Roukos. 1992. "Decision Tree Models Applied to the Labeling of Text with Part-of-Speech". In *Proceedings of the DARPA Speech* *and Natural Language Workshop*, pp. 117–121, Morgan Kaufmann, San Mateo, California.

[Black *et al* 1993] Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer and Salim Roukos. 1993. "Towards History-based Grammars: Using Richer Models for Probabilistic Parsing". In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pp. 31–37, ACL.

[Black *et al.* 1996] Ezra Black, Stephen Eubank, Roger Garside, Geoffrey Leech, Hideki Kashioka and David Magerman. Beyond Skeleton Parsing: Producing a Comprehensive Large-Scale General-English Treebank With Full Grammatical Analysis. In *The 16th International Conference on Computational Linguistics*, pages 107 – 113, Kopenhagen, Denmark, 1996.

[Bod 1992] Rens Bod. 1992. "A computational model for language performance: Data Oriented Parsing". In *Proceedings of the 14th International Conference on Computational Linguistics*, pp. 855–859, ICCL.

[Bod 1995a] Rens Bod. 1995. "The Problem of Computing the Most Probable Parse in Data-Oriented Parsing and Stochastic Tree Grammars". In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 104–111, ACL.

[Bod 1995b] Rens Bod. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language.* ILLC Dissertation Series 1995-14, Amsterdam.

[Brants & Samuelsson 1995] Thorsten Brants and Christer Samuelsson. 1995. "Tagging the Teleman Corpus". In *Proceedings of the 10th Scandinavian Conference on Computational Linguistics*, pp. 7–20, University of Helsinki.

[Brill 1992] Eric Brill. 1992. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 112–116, Morgan Kaufmann, San Mateo, California.

[Brill 1993] Eric Brill. 1993. "Automatic grammar induction and parsing free text: A transformation-based approach". In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 259–265, ACL.

[Briscoe & Carroll 1993] Ted Briscoe and John Carroll. 1993. "Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars". In *Computational Linguistics 19(1)*, pp. 25–59.

[Brown *et al* 1990] Brown *et al.* 1990. "A statistical approach to machine translation". In *Computational Linguistics 16(2)*, pp. 79–85.

[Brown *et al* 1991] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra and Robert Mercer. 1991. "Word-sense disambiguation using statistical methods". In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 264–270, ACL.

[Brown *et al* 1993] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra and Robert Mercer. 1993. "The Mathematics of Statistical Machine Translation: Parameter Estimation". In *Computational Linguistics 19(2)*, pp. 263–311.

[Carroll 1994] John Carroll. 1994. "Relating Complexity to Practical Performance in Parsing with Wide-Coverage Unification Grammars". In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 287–294, ACL.

[Charniak 1993] Eugene Charniak. 1993. *Statistical Language Learning*. MIT Press.

[Chinchor *et al* 1993] Nancy Chinchor, Lynette Hirschman, David D. Lewis. 1993. "Evaluating Message Understanding Systems: An Analysis of the Third Message Understanding Conference (MUC-3)". In *Computational Linguistics 19(3)*, pp. 409–449.

[Church 1988] Kenneth W. Church. 1988. "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text". In *Proceedings of the 2nd Conference on Applied Natural Language Processing*, pp. 136–143, ACL.

[Church 1993] Kenneth W. Church. 1993. "Char_align: A Program for Aligning Parallel Texts at the Character Level". In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 1–8, ACL.

[Church 1994] Kenneth Ward Church. Unix for poets. In *European Summer School on Language and Speech Communication. Corpus-Based Methods. Course Notes Vol. 1*, Utrecht University, 1994. Research Institute for Language and Speech.

[Church & Gale 1991] Kenneth W. Church and William A. Gale. 1991. "A Comparison of the Enhanced Good-Turing and Deleted Estimation Methods for Estimating Probabilities of English Bigrams". In *Computer Speech and Language 19(5).*

[Collins 1996] Michael John Collins. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th ACL*, Santa Cruz, CA, 1996.

[Collins 1997] Michael Collins. Three Generative, Lexicalized Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, Madrid, 1997.

[Cutting *et al* 1992] Douglass Cutting, Julian Kupiec, Jan Pedersen and Penelope Sibun. 1992. "A Practical Part-of-Speech Tagger". In *Proceedings of the 3rd Conference on Applied Natural LanguageProcessing*, pp. 133–140, ACL.

[Daelemans 1994] Walter Daelemans. 1994. "Memory-Based Lexical Acquisition and Processing". Lecture Notes in Artificial Intelligence, Machine Translation and the Lexicon.

[Dalsgaard 1992] Paul Dalsgaard. 1992. "Phoneme label alignment using acoustic-phonetic features and Gaussian probability density functions". In *Computer Speech and Language 6*. ⟨**Really?**⟩

[Dagan *et al* 1993] Ido Dagan, Kenneth W. Church and William A. Gale. 1993. "Robust alignment of bilingual texts". In *IEEE 93*.

[Dagan *et al* 1993] Ido Dagan, Shaul Marcus and Shaul Markovitch. 1993. "Contextual Word Similarity and Estimation from Sparse Data". In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 164–171, ACL.

[Dagan *et al* 1994] Ido Dagan, Fernando Perreira and Lillian Lee. 1994. "Similarity-Based Estimation of Word Cooccurrence Probabilities". In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 272–278, ACL.

[Dano 1975] Sven Dano. 1975. *Nonlinear and Dynamic Programming*. Springer, New York.

[DeGroot 1975] Morris H. DeGroot. 1975. *Probability and Statistics*. Addison-Wesley, Reading, Massachusetts.

[DeRose 1988] Steven J. DeRose. 1988. "Grammatical Category Disambiguation by Statistical Optimization". In *Computational Linguistics 14(1)*, pp. 31–39.

[Dreyfuss & Law 1977] Stuart E. Dreyfuss and Averill M. Law. 1977. *The Art and Theory of Dynamic Programming*. Academic Press, New York.

[Eisner 1996] Jason Eisner. Tree New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340 – 345, Copenhagen, Denmark, 1996.

[Ford *et al.* 1982] Marilyn Ford, Joan Bresnan and Ronald M. Kaplan. A competence-based theory of syntactic closure. In Joan Bresnan (ed.), *The mental representation of grammatical relations*. MIT Press, Cambridge, MA., 1982.

[Forney 1973] G. David Forney, Jr. 1973. "The Viterbi Algorithm". In *Proceedings of the IEEE 61(3)*, pp. 268–278.

[Francis & Kucera 1982] N. W. Francis and H. Kucera. *Frequency Analysis of English Usage*. Houghton Mifflin, Boston, 1982.

[Frazier 1978] L. Frazier. 1978. *On comprehending sentences: Syntactic parsing strategies*. Doctoral dissertation, Univ. of Connetticut.

[Furui 1989] Sadaoki Furui. 1989. *Digital Speech Processing, Synthesis, and Recognition*. Marcel Dekker, New York, New York.

[Gale & Church 1993] William A. Gale and Kenneth W. Church. 1993. "A Program for Aligning Sentences in Bilingual Corpora". In *Computational Linguistics 19(1)*, pp. 75–102.

[Gale *et al* 1992] William A. Gale, Kenneth W. Church and David Yarowsky. 1992. "Work on statistical methods for word sense disambiguation". In *Proceedings from the AAAI Fall Symposium: Probabilistic Approaches to Natural Language*, pp. 55–60.

[Good 1953] I. J. Good. 1953. "The Population Frequencies of Species and the Estimation of Population Parameters". In *Biometrika 40*, pp. 237–264.

[Hearst 1991] Marty Hearst. 1991. "Toward noun homonym disambiguation using local context in large corpora". ???.

[Herwijnen 1994] Eric van Herwijnen. 1994. *Practical SGML*, 2nd Ed. Kluwer, Boston/Dordrecht/London.

[Hindle 1992] Donald Hindle. 1992. "An Analogical Parser for Restricted Domanins". In *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 150–154.

[Hindle & Rooth 1993] Donald Hindle and Mats Rooth. 1993. "Structural Ambiguity and Lexical Relations". In *Computational Linguistics 19(1)*, pp. 103–121.

[Jelinek 1990] Fred Jelinek. 1990. "Self-Organizing Language Models for Speech Recognition". In *Readings in Speech Recognition*, pp. 450–506, Morgan Kaufmann, San Mateo, California.

[Jelinek & Mercer 1980] Frederick Jelinek and Robert L. Mercer. 1980. "Interpolated Estimation of Markov Source Paramenters from Sparse Data". In *Pattern Recognition in Practice*, pp. 381–397. North Holland.

[Jelinek *et al* 1992] Fred Jelinek, John Lafferty, Robert Mercer. 1992. "Basic methods of probabilistic context free grammars". In Pietro Laface and Renato De Mori *Speech Recognition and Understanding. Recent Advances, Trends, and Applications,* volume f75 of *NATO Advanced Science Institute Series,* pp ??–??. Springer, Berlin. Proceedings of the NATO Advanced Study Institute, Cetraro, Italy, 1990.

[Johansson *et al.* 1986] Stig Johansson, Eric Atwell, Roger Garside and Geoffrey Leech. The Tagged LOB Corpus: User's Manual. Manual, Norwegian Computing Centre for the Humanities, Bergen, 1986.

[Johnson 1991] Mark Johnson. 1991. "The Computational Complexity of GLR Parsing". In [Tomita (ed.) 1991], pp. 35–42.

[Joshi *et al* 1975] Aravind Joshi, L. Levy and M. Takahashi. 1975. "Tree Adjunct Grammars". In *Journal of the Computer and System Sciences 10(1)*, pp. 136–163.

[Karlsson 1990] Fred Karlsson. 1990. "Constraint Grammar as a Framework for Parsing Running Text". In *Proceedings of the 13th International Conference on Computational Linguistics, Vol. 3*, pp. 168–173, ICCL.

[Karlsson *et al* 1995] Fred Karlsson, Atro Voutilainen, Juha Heikkilä and Arto Anttila (eds). 1995. *Constraint Grammar. A Language-Independent System for Parsing Unrestricted Text*, Mouton de Gruyter, Berlin / New York.

[Katz 1987] Slava M. Katz. 1987. "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer". In *IEEE Transactions on Acoustics, Speech, and Signal Processing 35(3)*, pp. 400–401.

[Kay & Röscheisen 1993] Martin Kay and Martin Röscheisen. 1993. "Text-Translation Alignment". In *Computational Linguistics 19(1)*, pp. 121–142.

[Kimball 1973] J. Kimball. 1973. "Seven principles of surface structure parsing in natural language". In *Cognition 2*.

[Kipps 1991] James R. Kipp. 1991. "GLR Parsing in Time $O(n^3)$". In [Tomita (ed.) 1991], pp. 43–59.

[Knuth 1965] Donald E. Knuth. 1965. "On the translation of languages from left to right". In *Information and Control 8(6)*, pp. 607–639.

[Konieczny *et al.* 1997] Lars Konieczny, Barbara Hemforth, Christioph Scheepers and Gerd Strube. The role of lexical heads in parsing: evidence from German. *Language and Cognitive Porcesses*, 12(2/3), 1997.

[Kupiec 1992] Julian Kupiec. 1992. "An algorithm for estimating the parameters of unrestricted hidden stochastic context-free grammars". In *Proceedings of the 14th International Conference on Computational Linguistics*, pp. 387–393, ICCL.

[Kupiec 1993] Julian Kupiec. 1993. "An Algorithm for Finding Noun Phrase Correspondences in Bilingual Corpora". In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 17–22, ACL.

[Lafferty *et al.* 1992] John Lafferty, Daniel Sleator and Davy Temperley. Grammatical Trigrams: A Probabilistic Model of Link Grarmmar. In *Proceedings of the AAAI Fall Symposion on Probabilistic Approaches to Natural Language*, 1992.

[Lehmann *et al.* 1996] Sabine Lehmann et al. TSNLP – Test Suites for Natural Language Processing. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 711 – 717, Copenhagen, Denmark, 1996.

[Magerman 1995] David M. Magerman. 1995. "Statistical Decistion-Tree Models for Parsing". In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 276–283, ACL.

[Magerman & Marcus 1990] David M. Magerman and Mitchell P. Marcus. 1990. "Parsing a Natural Language Using Mutual Information Statistics". In *Proceedings of the 9th National Conference on Artifical Intelligence*, pp. 984–989, AAAI/MIT Press.

[Magerman & Marcus 1991] David M. Magerman and Mitchell P. Marcus. 1991. "$\mathcal{P}$earl: A probabilistic chart parser". In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 15–20, ACL.

[Magerman & Weir 1992] David M. Magerman and Carl Weir. 1992. "Probabilistic Prediction and $\mathcal{P}$icky Chart Parsing". In *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 128–133, Morgan Kaufmann, San Mateo, California.

[Manning 1993] Christopher D. Manning. 1993. "Automatic Acquisition of a Large Subcategorization Dictionary from Corpora". In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 235–242, ACL.

[Marcus 1980] Mitchell P. Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press.

[Marcus *et al.* 1994] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz and Britta Schasberger. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the Human Language Technology Workshop*, San Francisco, March, 1994. Morgan Kaufmann.

[Mark *et al* 1992] Kevin Mark, Michael Miller, Ulf Grenander and Steve Abney. 1992. "Parameter Estimation for Constrained Context Free Language Models". In *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 146–149, Morgan Kaufmann, San Mateo, California.

[Merialdo 1994] Bernard Merialdo. 1994. "Tagging English Text with a Probabilistic Model". In *Computational Linguistics 20(2)*, pp. 155–171.

[Monachini and Calzolari 1994] Monica Monachini and Nicoletta Calzolari. Synopsis and Comparison of Morphosyntactic Phenomena Encoded in Lexicon and Corpora. Technical Report, EAGLES Lexicon Group, 1994.

[Mood *et al* 1974] A. Mood, F. Greybill and D. Boes. 1974. *Introduction to the Theory of Statistics 3rd Ed.*. McGraw Hill.

[MUC-3 1991] 1991. *Proceedings of the Third Message Understanding Conference (MUC-3)*. Morgan Kaufmann, San Mateo, California.

[Nádas 1985] Arthur Nádas. 1985. "On Turing's Formula for Word Probabilities". In *IEEE Transactions on Aucostics, Speech, and Signal Processing 33(6)*, pp. 1414–1416.

[Osborne & Bridge 1994] Miles Osborne and Derek Bridge. 1994. "Learning unification-based grammars using the Spoken English Corpus". In *Proceedings of the 2nd International Grammatical Inference Colloquim*, Alicante, Spain.

[Pearl 1988] Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, California.

[Pereira & Schabes 1992] Fernando Pereira and Yves Schabes. 1992. "Inside-outside reestimation from partially bracketed corpora". In *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 122–127, Morgan Kaufmann, San Mateo, California.

[Pereira *et al* 1993] Fernando Pereira, Naftali Tishby and Lillian Lee. 1993. "Distributional Clustering of English Words". In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 183–190, ACL.

[Pustejovsky 1992] James Pustejovsky. 1992. "The Acquisition of Lexical Semantic Knowledge from Large Corpora". In *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 243–248, Morgan Kaufmann, San Mateo, California.

[Råde & Rudemo 1975] Lennart Råde and Mats Rudemo. 1975. *Sannolikhetslära och statistik för teknisk högskola* (In Swedish). Biblioteksförlaget, Stockholm, Sweden.

[Rabiner 1989] Lawrence R. Rabiner. 1989. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". In *Proceedings of the IEEE 77(2)*, pp. 257–295.

[Resnik 1993] Philip Resnik. 1993. "Semantic classes and syntactic ambiguity". In *Proceedings of the ARPA Workshop on Human Language Technology*, Morgan Kaufmann, San Mateo, California.

[Sampson1995] Geoffrey Sampson. *English for the Computer. The SUSANNE Corpus and Analytic Scheme.* Clarendon Press, Oxford, 1995.

[Samuelsson 1994] Christer Samuelsson. 1994. "Grammar Specialization through Entropy Thresholds". In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 88–195, ACL.

[Samuelsson 1996] Christer Samuelsson. 1996. "Handling Sparse Data by Successive Abstraction". In *Procs. 16th International Conference on Computational Linguistics*, pp. 895–900, ICCL, 1996.

[Santorini 1995] Beatrice Santorini. Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision, 2nd printing). Technical Report, University of Pennsylvania, 1995.

[Schabes 1992] Yves Schabes. 1992. "Stochastic lexicalized tree-adjoining grammars". In *Proceedings of the 14th International Conference on Computational Linguistics*, pp. 426–432, ICCL.

[Schuetze 1992] Hinrich Schuetze. 1992. "Word sense disambiguation with sublexical representations". In *Proceedings of the 11th National Conference on Artifical Intelligence.* ⟨**Really?**⟩

[Schuetze 1993] Hinrich Schuetze. 1993. "Part-of-Speech Induction from Scratch". In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 251–257, ACL.

[Schuetze 1994] Hinrich Schuetze and Yoram Singer. 1994. "Part-of Speech Tagging Using a Variable Memory Markov Model". In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 181–187, ACL.

[Sima'an 1995] Khalil Sima'an. 1995. "An optimized algorithm for Data Oriented Parsing". In *Proceedings of the 1st International Conference on Recent Advances in Natural Language Processing*, pp. 45–54.

[Sima'an 1996] Khalil Sima'an. 1996. "Computational Complexity of Probabilistic Disambiguations by means of Tree-Grammars". Forthcoming.

[Skut *et al.* 1997] Wojciech Skut, Brigitte Krenn, Thorsten Brants and Hans Uszkoreit. An Annotation Scheme for Free Word Order Languages. In *ANLP*, Washington, 1997.

[Smadja 1993] Frank Smadja. 1993. "Retrieving Collocations from Text: Xtract". ???.

[Stolcke & Segal 1994] Andreas Stolke and Jonathan Segal. 1994. "Precise N-Gram Probabilities from Stochastic Context-Free Grammars". In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 74–79, ACL.

[Stolcke 1995] Andreas Stolcke. 1995. "An Efficient Probabilistic Context-free Parsing Algorithm that Computes Prefix Probabilities". In *Computational Linguistics 21(2)*, pp. 165–202.

[Su *et al* 1991] Keh-Yih Su, Jong-Nae Wang, Mei-Hui Su and Jing-Shin Chang. 1991. "GLR Parsing with Scoring". In [Tomita (ed.) 1991], pp. 93–102.

[Taraban & McClelland 1988] Roman Taraban and James L. McClelland. 1988. "Constituent attachment and thematic role assignment in sentence processing: influences of content-based expectations". In *Journal of Memory and Language 27*.

[Tatsuoka 1988] Maurice M. Tatsuoka. 1988. *Multivariate Analysis*, 2nd Ed. Macmillan, New York.

[Tesnière 1959] L. Tesnière. *Éléments de Syntaxe Structurale.* Klincksieck, Paris, 1959.

[Teufel 1995a] Simone Teufel. A Support Tool for Tagset Mapping. In *Proceedings of the ACL Sigdat Workshop*, pages 24 – 30, Dublin, Ireland, 1995.

[Teufel 1995b] Simone Teufel. A Typed German Incarnation of the EAGLES-TS Definition of Lexical Descriptions and Classification Guidelines. Technical Report, EAGLES Tagging Group, 1995.

[Thielen and Schiller 1995] Christine Thielen and Anne Schiller. Ein kleines und erweitertes Tagset fürs Deutsche. In *Tagungsberichte des Arbeitstreffens Lexikon + Text 17./18. Februar 1994, SchloßHohentübingen. Lexicographica Series Maior*, Tübingen, 1995. Niemeyer.

[Tomita (ed.) 1991] Matsuru Tomita, editor. 1991. *Generalized LR Parsing*. Kluwer, Boston/Dordrecht/London.

[Tzoukermann *et al* 1995] Evelyne Tzoukermann, Dragomir R. Radev and William A. Gale. Combining Linguistic Knowledge and Statistical Learning in French. In *Proceedings of the ACL Sigdat Workshop*, pages 51 – 58, Dublin, Ireland, 1995.

[Utsuro *et al* 1992] Takehito Utsuro, Yuji Matsumoto and Makoto Nagao. 1992. "Lexical Knowledge Acquisition from Bilingual Corpora". In *Proceedings of the 14th International Conference on Computational Linguistics*, pp. 581–587, ICCL.

[Voutilainen 1994] Atro Voutilainen. 1994. *Three studies in surface-syntactic analysis of running text.* Doctoral Dissertation. Publications 24, Department of General Linguistics, University of Helsinki. Yliopistopaino.

[Weischedel *et al* 1993] Ralph Weischedel, Marie Metter, Richard Schwartz, Lance Ramshaw, Jeff Palmucci. 1993. "Coping with ambiguity and unknown words through probabilistic models". In *Computational Linguistics 19(2)*, pp. 359–383.

[Wittemore *et al* 1992] Greg Wittemore, Kathleen Ferrara and Hans Brunner. 1990. "Emprical study of predictive powers of simple attachment schemes for post-modifier prepositional phrases". In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, ACL.

[Wu 1995] Dekai Wu. 1995. "Stochastic Inversion Transduction Grammars, with Application to Segmentation, Bracketing, and Alignment of Parallel Corpora". In *14th International Joint Conference on Artificial Intelligence*, pp. 1328–1335, Morgan Kaufmann, San Mateo, California.

[Yarowsky 1992] David Yarowsky. 1992. "Word sense disambiguation using statistical models of Roget's Categories trained on large corpora". In *Proceedings of the 14th International Conference on Computational Linguistics*, pp. 454–460, ICCL.

[Young 1993] Steve Young. 1993. "The HTK Hidden Markov Model Toolkit: Design and Philosophy". ??.

[Zipf 1935] G. K. Zipf. 1935. *The Psychobiology of Language*. Houghton Mifflin, Boston.