Computerwerkzeuge für die Linguistik: Probleme der Kodierung

Susanne Hackmack
Karl Heinz Wagner
Universität Bremen

Probleme der Kodierung

Damit linguistische Daten (im weitesten Sinne) mit Computerprogrammen verarbeitet werden können, müssen sie in maschinenlesbarer Form repräsentiert werden. Solche Daten können vorliegen als

- Texte in je unterschiedlichen Schriftsystemen, möglicherweise auch multilingual
- Sprachaufzeichnungen auf Datenträgern oder in einer phonetischen Transkription
- Aufzeichnungen über paralinguistische oder non-verbale Phänomene (Gestik, Mimik)

Probleme der Kodierung: Digitalisieren

Das Problem dabei ist, daß Computerprogramme mit für den Menschen verständlichen Repräsentationsformen wie Bildern, Buchstaben, Tönen nichts anfangen können.

Computer verarbeiten nur Zahlen: linguistische Daten jeglicher Art müssen also in ein Zahlenformat transformiert werden.

Diesen Vorgang nennt man Digitalisieren.

Digitalisieren: Bits - Bytes - Zeichen

Die beiden Grundeinheiten in jedem heutigen Computer sind die Einheiten **Bit** und **Byte**.

Die elementarste Informationseinheit, mit der Computer arbeiten, ist das Bit, aus engl. *binary digit* (= Binärziffer).

Computer arbeiten physikalisch mit zwei alternativen Spannungszuständen: ein relativ hohes Spannungs-potential oder ein relativ niedriges Spannungspotential. Diese werden mit den Ziffern 1 (hohes Potential) und 0 (niedriges Potential) bezeichnet.

Ein **Byte** ist bei den heute üblichen Systemen als Folge von 8 Bit definiert (man spricht auch von **Octets**), zum Beispiel 10100110

Bits - Bytes - Zeichen

Was bedeutet nun aber eine Bitfolge wie 10101010 oder 01101100?

Zunächst kann man feststellen, daß sich mit einer Folge von 8 Bit genau 256 (= 2 hoch 8) unterschiedliche Zustände realisieren lassen. Ein Byte kann also 256 unterschiedliche Werte haben.

Es biete sich also an, solche Bitfolgen Zahlen zuzuordnen, und zwar genau die Zahlen zwischen 0 (= 00000000 und 255 (= 11111111).

Bits, Bytes und Zeichen

00000000	0
00000001	1
00000010	2
00000011	3
00000100	4
00000101	5
00000110	6

Bits - Bytes: Dezimalzahlen

Um deutlich zu machen, was dies eigentlich bedeutet, wollen wir uns als nächstes Dezimal-zahlen ansehen.

Dezimalzahlen bestehen aus Folgen der Ziffern 0, 1, 2 ... 9.

Was bedeutet aber beispielsweise die Zahl 3495?

Auf Englisch würde man z.B. lesen: three thousand four hundred and ninety five.

Bits - Bytes: Dezimalzahlen

Ginge es um Geld könnte three thousand four hundred and ninety five stehen für die Summe aus drei Tausendern, vier Hundertern, neun Zehnern, und fünf Markstücken. Etwas abstrakter ausgedrückt:

$3 \times 1000 + 4 \times 100 + 9 \times 10 + 5 \times 1$ oder

$3 \times 10^3 + 4 \times 10^2 + 9 \times 10^1 + 5 \times 10^0$ dafür kurz:

3 4 9 5 die Zehnerpotenz ist durch die Position in der Zahlenfolge gegeben, von rechts nach links:

3 2 1 0

Bits - Bytes: Dualzahlen

Bitfolgen wie 1010 lassen sich nun analog interpretieren, allerdings mit dem Unterschied, daß wir es hier mit Potenzen der Zahl 2 zu tun haben.

Ziffernfolge: 1 0 1 0

Position: 3 2 1 0

Bedeutung: $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 = 10$

Somit ist die Zuordnung von Bitfolgen nicht willkürlich sondern ergibt sich aus dem dualen Zahlensystem.

Bits - Bytes: Rechnen mit Dualzahlen

Mit Dualzahlen kann man rechnen. Es gelten folgende Grundregeln für Addition:

0 + 0 = 0

1 + 0 = 0 + 1 = 1

1 + 1 = 10, d.h. 0 plus Übertrag 1

```
1  1 1   1  1
   1 0  1  0  1
   0 1  0  1  1
1  0  0  0   0  0
```

Bits - Bytes: Andere Zahlensysteme

Andere Zahlensysteme, die in der Computer-technologie eine wichtige Rolle spielen, sind

Oktalzahlen auf der Basis 8 mit den Ziffern 0 .. 7 und

Hexadezimalzahlen auf der Basis 16. Dafür reichen dann die Ziffern 0 .. 9 nicht aus und müssen durch die Buchstaben A .. F ergänzt werden.

Dezimal	Binär	Oktal	Hexadezimal
245	11110101	365	F5

Umrechung mit dem Windows-Rechner

Bytes und Zeichen

Bei der Kodierung linguistischer Daten geht es im einfachsten Fall zunächst einmal darum, Zeichenfolgen in Bytefolgen zu transformieren. Dazu ist eine standardisierte systematische Zuordnung von Bytewerten und Buchstaben erforderlich.

Solche Zeichenkodes oder Zeichensätze sind frühzeitig entwickelt worden.

Bytes und Zeichen: ASCII

In früheren Systemen wurden allerdings nur 6 oder 7 Bit verwendet. Große Bedeutung erlangte der 7-Bit-Code ASCII. ASCII steht für *American Standard Code for Information Interchange* (Amerikanischer Standardkode zum Informationsaustausch).

Mit 7 Bit können allerdings nur Werte zwischen 0 und 127 dargestellt werden. Von diesem Wertevorrat war der Bereich von 0 bis 31 und 127 für sog. Steuerungsaufgaben reserviert.

Somit verbleiben nur noch die Werte 32 .. 126 für den Zeichenkode selbst.

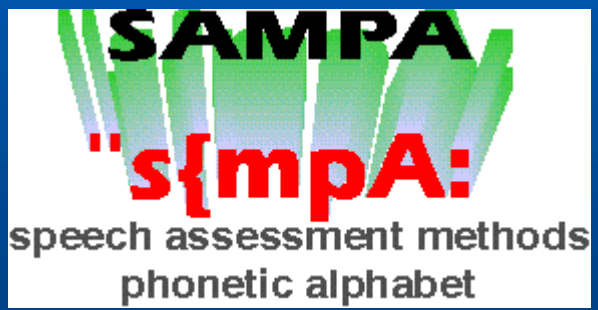
Bytes und Zeichen: 7-Bit-ASCII

ASCII-Zeichensatz

+	0	1	2	3	4	5	6	7	8	9
30				!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~			

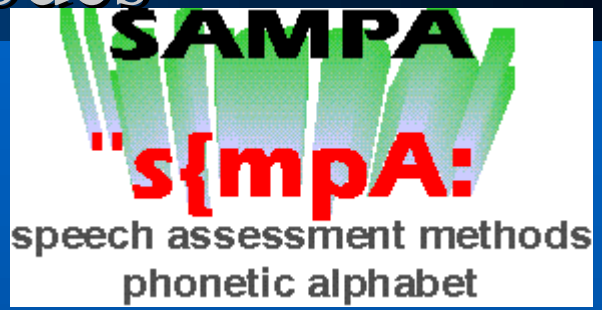
Bytes und Zeichen: Spezialcodes

SAMPA
computer readable phonetic alphabet



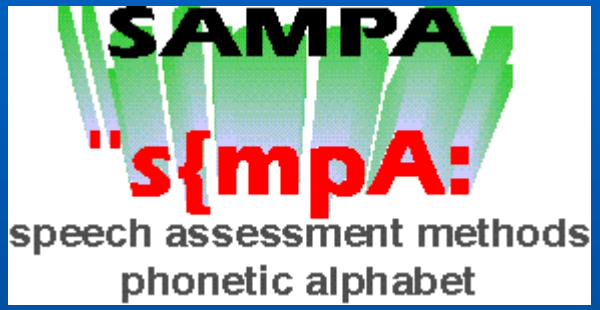
SAMPA (Speech Assessment Methods Phonetic Alphabet) ist ein maschinenelesbares phonetisches Alphabet, das im Rahmen eines ESPRIT Projektes entwickelt worden ist und ursprünglich auf EU-Sprachen angewandt wurde.

Bytes und Zeichen: Spezialcodes



SAMPA basiert auf einer systematischen Zuordnung der Symbole des IPA-Zeichenvorrats auf ASCII-Codes im Bereich 33 .. 127, d.h. den druckbaren 7-bit ASCII Zeichen. Diese Zuordnung erfolgt so, daß die Eingabe über die Tastatur möglichst vereinfacht wird. Das geschieht dadurch, daß z.B. die Großbuchstaben verwandten Zeichen entsprechen, z.B. wird N zu Kodierung von ŋ verwendet.

Bytes und Zeichen: Spezialcodes



Der deutsche Satz
Einst stritten sich Nordwind und Sonne, wer von ihnen beiden der Stärkere wäre.

wird kodiert als
?aInst "StRItn= zIC "nORtvInt ?Unt zOn@, ve:6 fOn ?i:n@n baldn= de:6 StERk@R@ vE:R@

wird wie folgt realisiert
ʔaɪnst ˈʃtʁɪtn̩ zɪç ˈnɔʁtvɪnt ʔʊnt zɔnə, veːɐ fɔn ʔiːnən baɪdn̩ deːɐ ʃtɛʁkəʁə vɛːʁə

Bytes und Zeichen: Metazeichen

Eine Erweiterungsmöglichkeit besteht darin, daß man bestimmten Zeichen kontextabhängig eine andere Bedeutung gibt und somit „Metazeichen“ einführt.

Das Zeichen ‚\‘ z.B. (*backslash*) kommt in normalen Texten nicht vor. Es kann daher verwendet werden, um anzudeuten, daß das nachfolgende Zeichen anders als normal gemeint ist. Man könnte z.B. mit \“a den Umlaut ä notieren. Allerdings sollten diese Metasymbole standardisiert sein.

Bytes und Zeichen: Mark-up Codes

Solche Metasymbole werden sehr extensiv in sogenannten ‚Mark-up Sprachen' wie T_EX, RTF (*Rich Text Format*), SGML (*Standard Generalized Markup Language*) und darauf aufbauend HTML (*Hypertext Markup Language*) und XML (Extended Markup Language) verwendet. Davon wird später noch ausführlich die Rede sein.

Bytes und Zeichen: Mark-up Sprachen

T_EX (für Tau Epsilon Chi)

Dies ist eine sehr mächtige Mark-up-Sprache für den Computersatz, sozusagen eine Programmier-sprache für Textverarbeitung, die sich besonders in den technisch-naturwissenschaftlichen Fächern aber auch in der Linguistik großer Verbreitung erfreut. Die Gestalteigenschaften von Dokumenten werden durch die Sprache „beschrieben“. Wichtig dabei ist, daß der Zeichenvorrat des ASCII Zeichensatzes dafür ausreichend ist.

Bytes und Zeichen: Mark-up Sprachen

$\TeX$

Beispiel:

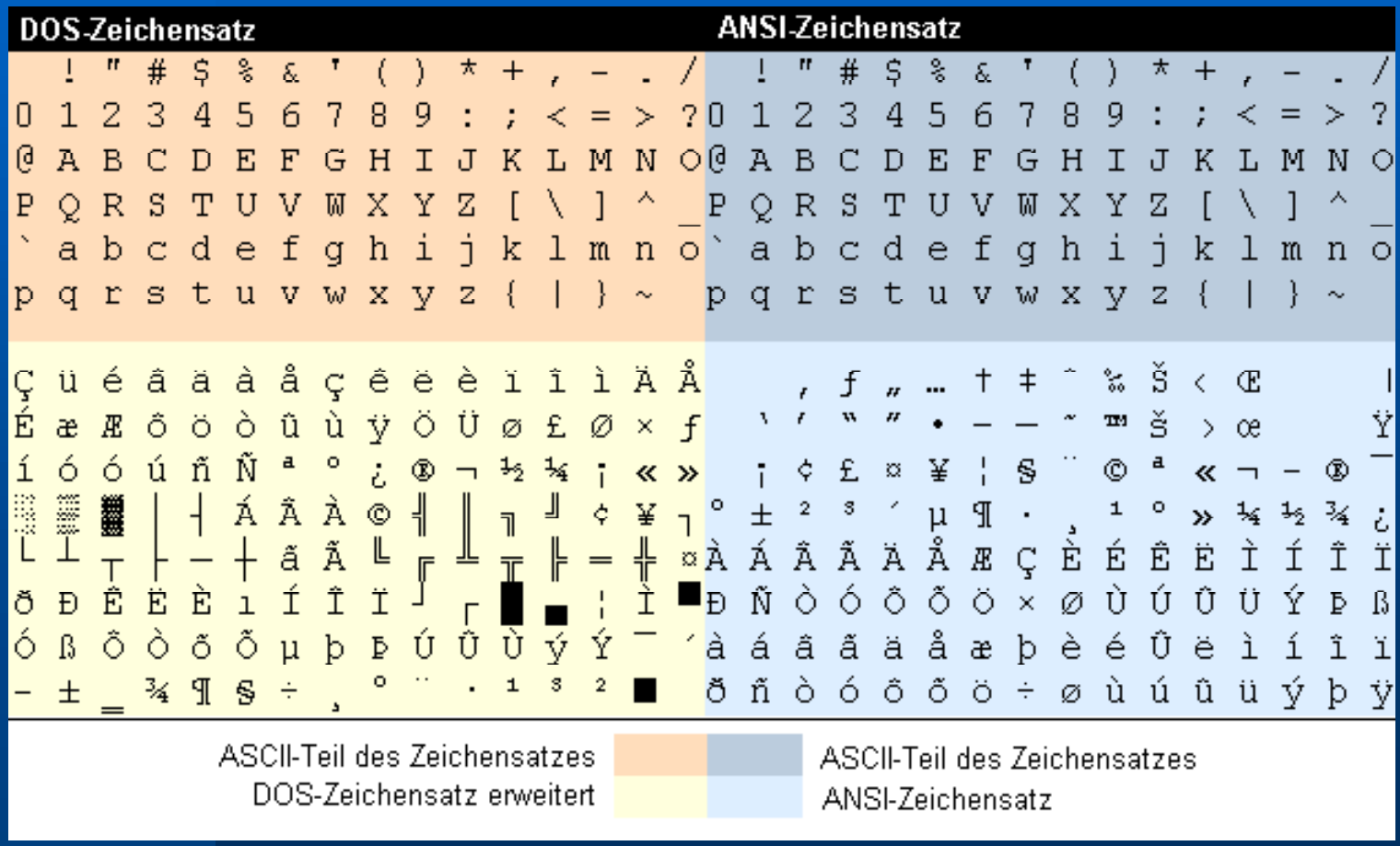
```
$$\left[\matrix{Kategorie & Verb \cr
          Tempus & Praet \cr
          Kongruenz &
               {\left[\matrix{Person & 3\cr
                              Numerus & Sg \cr
                }\right]} \cr}\right]$$
```

Bytes und Zeichen: Mark-up Sprachen

TEX

$$\begin{bmatrix} Kategorie & Verb \\ Tempus & Praet \\ Kongruenz & \begin{bmatrix} Person & 3 \\ Numerus & Sg \end{bmatrix} \end{bmatrix}$$

Bytes und Zeichen: ASCII & ANSI



Bytes und Zeichen: ASCII & ANSI

ANSI (*American National Standard Institute*), US-amerikanisches Institut für Normung, das sich im Prinzip mit dem DIN *(Deutsches Institut für Normung e.V.)* vergleichen lässt. Zu den Hauptaufgaben des ANSI zählen u. a. die Definition und Genehmigung von Handels- und Kommunikationsstandards, wie beispielsweise die Festlegung von Standards im Bereich Computer und Datenverarbeitung (z. B. ASCII, Zeichensätze). Darüber hinaus fungiert das nichtstaatliche Institut als eine Art Vermittlungsorganisation für international gültige Normen.

Bytes und Zeichen: ISO-8859

ISO ist eine Abkürzung für *International Standards Organization* und bezeichnet eine Standardisierungsinstitution.

ISO-8859

Bytes und Zeichen: Unicode

Unicode ist ein System, in dem die Zeichen oder Elemente aller bekannten Schriftkulturen und Zeichensysteme festgehalten werden.

- Durch dieses System wird es möglich, einem Computer zu sagen, welches Zeichen man dargestellt bekommen will. Voraussetzung ist natürlich, daß der Computer bzw. das ausgeführte Programm das Unicode-System kennt.
- So werden beispielsweise bei Windows NT alle Zeichen, egal mit welcher Software Sie arbeiten, im Arbeitsspeicher intern als Unicodes gespeichert.

Bytes und Zeichen: Unicode

Jedes Zeichen oder Element in Unicode wird durch eine zwei Byte lange Zahl ausgedrückt. Auf diese Weise lassen sich bis zu $256^2 = 65536$ verschiedene Zeichen in dem System unterbringen.

In Version 2.0 des Unicode-Standards sind 38885 Zeichen dokumentiert. Es ist also noch Platz genug.

Damit es jedoch nicht irgendwann eng wird, gibt es mittlerweile ein erweitertes Schema, mit dem weit über eine Million verschiedene Zeichen in das System passen.