

Writing systems

Alphabetic

Syllabic

Logographic

Systems with unusual
realization

Relation to language

Encoding written
language

ASCII

Unicode

Spoken language

Transcription

Why speech is hard to
represent

Articulation

Measuring sound

Acoustics

Relating written and
spoken language

From Speech to Text

From Text to Speech

Language modeling

Language and Computers

Prologue: Encoding Language

Based on Dickinson, Brew, & Meurers (2013)

Language and Computers – where to start?

- ▶ If we want to do anything with language, we need a way to represent language.
- ▶ We can interact with the computer in several ways:
 - ▶ write or read text
 - ▶ speak or listen to speech
- ▶ Computer has to have some way to represent
 - ▶ text
 - ▶ speech

Outline

Writing systems

Encoding written language

Spoken language

Relating written and spoken language

Language modeling

Writing systems

Alphabetic

Syllabic

Logographic

Systems with unusual
realization

Relation to language

Encoding written
language

ASCII

Unicode

Spoken language

Transcription

Why speech is hard to
represent

Articulation

Measuring sound

Acoustics

Relating written and
spoken language

From Speech to Text

From Text to Speech

Language modeling

What is writing?

“a system of more or less permanent marks used to represent an utterance in such a way that it can be recovered more or less exactly without the intervention of the utterer.”

(Peter T. Daniels, The World's Writing Systems)

Different types of writing systems are used:

- ▶ Alphabetic
- ▶ Syllabic
- ▶ Logographic

Much of the information on writing systems and the graphics used are taken from the great site <http://www.omniglot.com>.

Writing systems

Alphabetic

Syllabic

Logographic

Systems with unusual
realization

Relation to language

Encoding written language

ASCII

Unicode

Spoken language

Transcription

Why speech is hard to
represent

Articulation

Measuring sound

Acoustics

Relating written and spoken language

From Speech to Text

From Text to Speech

Language modeling

Alphabets (phonemic alphabets)

- ▶ represent all sounds, i.e., consonants and vowels
- ▶ Examples: Etruscan, Latin, Korean, Cyrillic, Runic, International Phonetic Alphabet

Abjads (consonant alphabets)

- ▶ represent consonants only (sometimes plus selected vowels; vowel diacritics generally available)
- ▶ Examples: Arabic, Aramaic, Hebrew

Alphabetic

Syllabic

Logographic

Systems with unusual
realization

Relation to language

Encoding written
language

ASCII

Unicode

Spoken language

Transcription

Why speech is hard to
represent

Articulation

Measuring sound

Acoustics

Relating written and
spoken language

From Speech to Text

From Text to Speech

Language modeling

Alphabet example: Fraser

An alphabet used to write Lisu, a Tibeto-Burman language spoken by about 657,000 people in Myanmar, India, Thailand and in the Chinese provinces of Yunnan and Sichuan.

Consonants

P	ᵀ	B	ᶇ	W	M	M	T	ᵀ	D	S	R	N	L	F	F
[p]	[pʰ]	[b]	[f]	[v]	[m]	[ɰ]	[t]	[tʰ]	[d]	[s]	[z]	[n]	[l]	[ts]	[tsʰ]
Z	C	C	J	X	R	Y	K	K	G	H	B	Λ	G	V	
[dz]	[c]	[cʰ]	[ɟ]	[ʃ]	[ʒ]	[ʎ]	[k]	[kʰ]	[g]	[x]	[ɣ]	[ŋ]	[ɦ]	[h]	

Vowels

I	E	ʌ	U	E	ɿ	D	A	U	O
[i]	[e]	[æ]	[ü]	[ø]	[ɯ]	[ə]	[ɑ]	[u]	[ɔ]

Tones

.	,	:	;	'
high tone	mid rising	mid tone	mid tense	low tone	low tense	nasalization

Writing systems

Alphabetic

- Syllabic
- Logographic
- Systems with unusual realization
- Relation to language

Encoding written language

- ASCII
- Unicode

Spoken language

- Transcription
- Why speech is hard to represent
- Articulation
- Measuring sound
- Acoustics











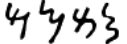











Relating written and spoken language

- From Speech to Text
- From Text to Speech

Language modeling

Abjad example: Phoenician

An abjad used to write Phoenician, created between the 18th and 17th centuries BC; assumed to be the forerunner of the Greek and Hebrew alphabet.

 hēt ḥ	 zayin z	 wāw w	 hē h	 dālet d	 gīmel g	 bēt b	 'ālef '
 sāmek s	 nun n	 mēm m		 lāmed l	 kaf k	 yōd y	 tēt ṭ
 tāw t	 šin/šin š		 rēš r	 qōf q	 šādē š	 pē p	 'ayin '

(from: <http://www.omniglot.com/writing/phoenician.htm>)

A note on the letter-sound correspondence

- ▶ Alphabets use letters to encode sounds (consonants, vowels).
- ▶ But the correspondence between spelling and pronunciation in many languages is quite complex, i.e., not a simple one-to-one correspondence.
- ▶ Example: English
 - ▶ same spelling – different sounds: *ough*: **ought**, **cough**, **tough**, **through**, **though**, **hiccough**
 - ▶ silent letters: **k***nee*, **k***n*ight, **k***n*ife, **d**e**b**t, **p***sy*chology, **m**ort**g**age
 - ▶ one letter – multiple sounds: **e***x*it, **u***s*e
 - ▶ multiple letters – one sound: **th**e, **rev**olution
 - ▶ alternate spellings: **j**ail or **g**aol; but not possible **se**agh for **ch**ef (despite **s**ure, **d**ead, **l**augh)

More examples for non-transparent letter-sound correspondences

French

- (1) a. *Versailles* → [versai]
b. *ete, etais, etait, etaient* → [ete]

Irish

- (2) a. *samhradh* (summer) → [sauruh]
b. *scri'obhaim* (I write) → [shgrim]

What is the notation used within the []?

The International Phonetic Alphabet (IPA)

- ▶ Several special alphabets for representing sounds have been developed, the best known being the International Phonetic Alphabet (IPA).
- ▶ The phonetic symbols are unambiguous:
 - ▶ designed so that each speech sound gets its own symbol,
 - ▶ eliminating the need for
 - ▶ multiple symbols used to represent simple sounds
 - ▶ one symbol being used for multiple sounds.
- ▶ Interactive example chart: <http://web.uvic.ca/ling/resources/ipa/charts/IPA1ab/IPA1ab.htm>

Syllabic systems

Syllabic alphabets (Alphasyllabaries)

- ▶ writing systems with symbols that represent a consonant with a vowel, but the vowel can be changed by adding a **diacritic** (= a symbol added to the letter).
- ▶ Examples: Balinese, Javanese, Tibetan, Tamil, Thai, Tagalog

(cf. also: <http://www.omniglot.com/writing/syllabic.htm>)

Syllabaries

- ▶ writing systems with separate symbols for each syllable of a language
- ▶ Examples: Cherokee, Ethiopic, Cypriot, Ojibwe, Hiragana (Japanese)

(cf. also: <http://www.omniglot.com/writing/syllabaries.htm#syll>)

Syllabary example: Cypriot

The Cypriot syllabary or Cypro-Minoan writing is thought to have developed from the Linear A, or possibly the Linear B script of Crete, though its exact origins are not known. It was used from about 800 to 200 BC.

a	ta	ga	ka	pa	la	ma	na	ra	sa	va	xa	ya
e	te	ke	pe	le	me	ne	re	se	ve	xe		
i	ti	ki	pi	li	mi	ni	ri	si	vi			
o	to	ko	po	lo	mo	no	ro	so	vo	zo	yo	
u	tu	ku	pu	lu	mu	nu	ru	su	vu			

(from: <http://www.omniglot.com/writing/cypriot.htm>)

Syllabic alphabet example: Lao

Script developed in the 14th century to write the Lao language, based on an early version of the Thai script, which was developed from the Old Khmer script, which was itself based on Mon scripts.

Example for vowel diacritics around the letter k:

ກ ກີ ກຸ ກຸ້ ກາ ກີ ກຸ ກຸ້ ກະ ກະ້

ka ki ku ku' ka: ki: ku: ku:' ke kae
[ka] [ki] [ku] [ku'] [ka:] [ki:] [ku:] [ku:'] [ke] [kae]

ໄກ ໄກະ ໄກ້ ໄກ່ ໄກ້ ໄກ້ ໄກ້ ໄກ້ ໄກ້ ໄກ້ ໄກ້

ko ke: kae: ko: ko' koe kia kia kua koe:y
[ko] [ke:] [kae:] [ko:] [kɔ'] [kɔy] [kia] [kia] [kua] [kɔ:y]

ໄກ້ ໄກ້ ໄກ້ ໄກ້ ໄກ້ ໄກ້ ໄກ້ ໄກ້ ໄກ້

koe:y ko': koe: ku'a kaw kay kay kam k
[kɔ:y] [kɔ':] [kɔe:] [ku'a] [kaw] [kaj] [kaj] [kam] [k]

(from: <http://www.omniglot.com/writing/lao.htm>)

Logographic writing systems

- ▶ Logographs (also called Logograms):

- ▶ **Pictographs (Pictograms):** originally pictures of things, now stylized and simplified.

Example: development of Chinese character *horse*:



- ▶ **Ideographs (Ideograms):** representations of abstract ideas
 - ▶ **Compounds:** combinations of two or more logographs.
 - ▶ **Semantic-phonetic compounds:** symbols with a meaning element (hints at meaning) and a phonetic element (hints at pronunciation).
- ▶ Examples: Chinese (Zhōngwén), Japanese (Nihongo), Mayan, Vietnamese, Ancient Egyptian

Logograph writing system example: Chinese

Pictographs

女	子	口	日	月	山	川	豕	目	心	雨	田	木	龜
woman	child	mouth	sun	moon	mountain	river	pig	eye	heart	rain	field	tree	turtle

Ideographs

一	二	三	上	下	中	力	凸	凹
one	two	three	above	below	middle	strength (plough)	convex	concave

Compounds of Pictographs/Ideographs

好	安	明	家	思	牢	雷	男
good (woman + child)	peaceful (woman under a roof)	bright (sun + moon)	home/family (pig under a roof)	thought (heart + field)	prison (cow under a roof)	thunder (rain cloud over a field)	man/male (field + strength)

(from: http://www.omniglot.com/writing/chinese_types.htm)

Alphabetic

Syllabic

Logographic

Systems with unusual realization

Relation to language

ASCII

Unicode

Transcription

Why speech is hard to represent

Articulation

Measuring sound

Acoustics

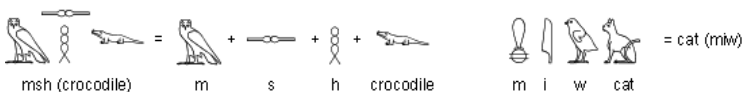
From Speech to Text

From Text to Speech

Semantic-phonetic compounds

	phonetic component					
	古 gǔ	扁 biǎn	敖 áo	旁 páng	堯 yáo	
semantic component (radical)	人 (person)	估 gū (to guess)	偏 piān (biased)	傲 ào (proud)	傍 bāng (beside)	僥 jiǎo (lucky)
	言 (words)	詁 gǔ (commentaries)	論 pián (to quibble)	謗 áo (to slander)	謗 bàng (to libel)	詭 niǎo (to argue)
	虫 (insect)	蛄 gū (mole cricket)	蝙 biān (bat)	螯 áo ([crab/s] nippers)	螃 páng (crab)	蟯 rǎo (worm)
	金 (metal)	鈷 gū (cobalt)		鏊 áo (griddle)	鎊 bàng (pound sterling)	鏡 niǎo (cymbals)

An example from Ancient Egyptian



Two writing systems with unusual realization

Tactile

- ▶ Braille is a writing system that makes it possible to read and write through touch; primarily used by the (partially) blind.
- ▶ It uses patterns of raised dots arranged in cells of up to six dots in a 3 x 2 configuration.
- ▶ Each pattern represents a character, but some frequent words and letter combinations have their own pattern.

Chromatographic

- ▶ The Benin and Edo people in southern Nigeria have supposedly developed a system of writing based on different color combinations and symbols.

Braille alphabet

A 1 a	B 2 but	C 3 can	D 4 do	E 5 every	F 6 from	G 7 go	H 8 have	I 9 just	J 0 knowledge	K like	L more	M
N not	O people	P quite	Q rather	R so	S that	T us	U very	V will	W it	X you	Y as	Z
Ç and	É for	À of	È the	Ù with	Â child ch	Ê gh	Î shall sh	Ô this th	Û which wh	Ë ed	Ï er	Û out ou
Ö CE ow	, ar	; bb	: cc	. dd	en	! gg; were	()	? " in	* st	" fraction line	ì st	ò ing
numerical sign	Ä Æ ar	'	-	numerical index accent	literal index	italic sign decimal sign	letter sign	capital sign				

Writing systems

Alphabetic
Syllabic
Logographic

Systems with unusual realization

Relation to language

Encoding written language

ASCII
Unicode

Spoken language

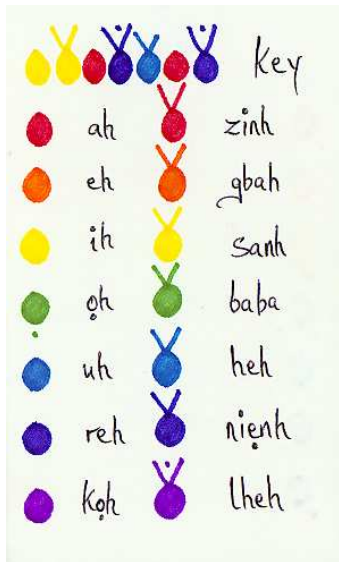
Transcription
Why speech is hard to represent
Articulation
Measuring sound
Acoustics

Relating written and spoken language

From Speech to Text
From Text to Speech

Language modeling

Chromatographic system



- Alphabetic
- Syllabic
- Logographic

Systems with unusual realization

Relation to language

Encoding written language

- ASCII
- Unicode

Spoken language

- Transcription
- Why speech is hard to represent
- Articulation
- Measuring sound
- Acoustics

Relating written and spoken language

- From Speech to Text
- From Text to Speech

Language modeling

Relating writing systems to languages

- ▶ There is not a simple correspondence between a writing system and a language.
- ▶ For example, English uses the Roman alphabet, but Arabic numerals (e.g., 3 and 4 instead of III and IV).
- ▶ We'll look at three other examples:
 - ▶ Japanese
 - ▶ Korean
 - ▶ Azeri

Japanese: logographic system *kanji*, syllabary *katakana*,
syllabary *hiragana*

- ▶ *kanji*: 5,000-10,000 borrowed Chinese characters
- ▶ *katakana*
 - ▶ used mainly for non-Chinese loan words, onomatopoeic words, foreign names, and for emphasis
- ▶ *hiragana*
 - ▶ originally used only by women (10th century), but codified in 1946 with 48 syllables
 - ▶ used mainly for word endings, kids' books, and for words with obscure *kanji* symbols
- ▶ *romaji*: Roman characters

- ▶ The *hangul* system was developed in 1444 during King Sejong's reign.

- ▶ There are 24 letters: 14 consonants and 10 vowels
- ▶ But the letters are grouped into syllables, i.e. the letters in a syllable are not written separately as in the English system, but together form a single character.

E.g., "Hangeul" (from: <http://www.omniglot.com/writing/korean.htm>):

한 (han) ㅎ(h) + ㅏ(a) + ㄴ(n) 글 (geul) ㄱ(g) + ㅡ(eu) + ㄹ(l)

- ▶ In South Korea, *hanja* (logographic Chinese characters) are also used.

A Turkish language with speakers in Azerbaijan, northwest Iran, and (former Soviet) Georgia

- ▶ 7th century until 1920s: Arabic scripts. Three different Arabic scripts used
- ▶ 1929: Latin alphabet enforced by Soviets to reduce Islamic influence.
- ▶ 1939: Cyrillic alphabet enforced by Stalin
- ▶ 1991: Back to Latin alphabet, but slightly different than before.

Encoding written language

- ▶ Information on a computer is stored in **bits**.
- ▶ A bit is either on (= 1, yes) or off (= 0, no).
- ▶ A list of 8 bits makes up a **byte**, e.g., 01001010
- ▶ Just like with the base 10 numbers we're used to, the order of the bits in a byte matters:
 - ▶ **Big Endian**: most important bit is leftmost (the standard way of doing things)
 - ▶ The positions in a byte thus encode:
128 64 32 16 8 4 2 1
 - ▶ "There are 10 kinds of people in the world; those who know binary and those who don't"
(from: <http://www.wlug.org.nz/LittleEndian>)
 - ▶ **Little Endian**: most important bit is rightmost (only used on Intel machines)
 - ▶ The positions in a byte thus encode:
1 2 4 8 16 32 64 128

Converting decimal numbers to binary

Tabular Method

Using the first 4 bits, we want to know how to write 10 in bit (or *binary*) notation.

8	4	2	1
?	?	?	?
$8 < 10$?	?	?
1	$8 + 4 = 12 > 10$?	?
1	0	$8 + 2 = 10$?
1	0	1	0

Converting decimal numbers to binary

Division Method

Decimal	Remainder?	Binary
$10/2 = 5$	no	0
$5/2 = 2$	yes	10
$2/2 = 1$	no	010
$1/2 = 0$	yes	1010

An encoding standard: ASCII

With 8 bits (a single byte), you can represent 256 different characters.

- ▶ With 256 possible characters, we can store:
 - ▶ every single letter used in English,
 - ▶ plus all the things like commas, periods, space bar, percent sign (%), back space, and so on.

ASCII = the American Standard Code for Information Interchange

- ▶ 7-bit code for storing English text
- ▶ 7 bits = 128 possible characters.
- ▶ The numeric order reflects alphabetic ordering.

The ASCII chart

Codes 1–31 are used for control characters (backspace, line feed, tab, ...).

32		48	0	65	A	82	R	97	a	114	r
33	!	49	1	66	B	83	S	98	b	115	s
34	“	50	2	67	C	84	T	99	c	116	t
35	#	51	3	68	D	85	U	100	d	117	u
36	\$	52	4	69	E	86	V	101	e	118	v
37	%	53	5	70	F	87	W	102	f	119	w
38	&	54	6	71	G	88	X	103	g	120	x
39	'	55	7	72	H	89	Y	104	h	121	y
40	(56	8	73	I	90	Z	105	i	122	z
41)	57	9	74	J	91	[106	j	123	{
42	*	58	:	75	K	92	\	107	k	124	—
43	+	59	;	76	L	93]	108	l	125	}
44	,	60	<	77	M	94	^	109	m	126	~
45	-	61	=	78	N	95	_	110	n	127	DEL
46	.	62	>	79	O	96	`	111	o		
47	/	63	?	80	P			112	p		
		64	@	81	Q			113	q		

E-mail issues

- ▶ Mail sent on the internet used to only be able to transfer the 7-bit ASCII messages. But now we can detect the incoming character set and adjust the input.
- ▶ Note that this is an example of **meta-information** = information which is printed as part of the regular message, but tells us something about that message.
- ▶ Multipurpose Internet Mail Extensions (MIME) provides meta-information on the text, which tells us:
 - ▶ which version of MIME is being used
 - ▶ what the character set is
 - ▶ if that character set was altered, how it was altered

```
Mime-Version: 1.0 Content-Type: text/plain;  
charset=US-ASCII Content-Transfer-Encoding: 7bit
```

Different coding systems

But wait, didn't we want to be able to encode *all* languages?

There are ways ...

- ▶ Extend the ASCII system with various other systems, for example:
 - ▶ ISO 8859-1: includes extra letters needed for French, German, Spanish, etc.
 - ▶ ISO 8859-7: Greek alphabet
 - ▶ ISO 8859-8: Hebrew alphabet
 - ▶ JIS X 0208: Japanese characters
- ▶ Have one system for everything → **Unicode**

Problems with having multiple encoding systems:

- ▶ Conflicts: two encodings can use:
 - ▶ same number for two different characters
 - ▶ different numbers for the same character
- ▶ Hassle: have to install many, many systems if you want to be able to deal with various languages

Unicode tries to fix that by having a single representation for every possible character.

*“Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.”
(www.unicode.org)*

How big is Unicode?

Version 6.0 has codes for 109,449 characters from alphabets, syllabaries and logographic systems.

- ▶ Uses 32 bits – meaning we can store $2^{32} = 4,294,967,296$ characters.
- ▶ 4 billion possibilities for each character? That takes a lot of space on the computer!

Compact encoding of Unicode characters

- ▶ Unicode has three versions
 - ▶ UTF-32 (32 bits): direct representation
 - ▶ UTF-16 (16 bits): $2^{16} = 65536$
 - ▶ UTF-8 (8 bits): $2^8 = 256$
- ▶ How is it possible to encode 2^{32} possibilities in 8 bits (UTF-8)?
 - ▶ Several bytes are used to represent one character.
 - ▶ Use the highest bit as flag:
 - ▶ highest bit 0: single character
 - ▶ highest bit 1: part of a multi byte character
 - ▶ Nice consequence: ASCII text is in a valid UTF-8 encoding.

UTF-8 details

- ▶ First byte unambiguously tells you how many bytes to expect after it
 - ▶ e.g., first byte of 11110xxx has a four total bytes
- ▶ all non-starting bytes start with 10 = *not* the initial byte

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
0xxxxxxx					
110xxxxx	10xxxxxx				
1110xxxx	10xxxxxx	10xxxxxx			
11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

Example: Greek α ('alpha') has a code value of 945

- ▶ Binary: 11 10110001
- ▶ 11 10110001 = 011 10110001 = 01110 110001
- ▶ Insert these numbers into x's in the second row:
11001110 10110001

Unwritten languages

Many languages have never been written down. Of the 6912 spoken languages, approximately 3000 have never been written down.

Some examples:

- ▶ Salar, a Turkic language in China.
- ▶ Gugu Badhun, a language in Australia.
- ▶ Southeastern Pomo, a language in California

(See: <http://www.ethnologue.com/> and <http://www.sil.org/mexico/ilv/iinfoilvmexico.htm>)

The need for speech

We want to be able to encode any spoken language

- ▶ What if we want to work with an unwritten language?
- ▶ What if we want to examine the way someone talks and don't have time to write it down?

Many applications for encoding speech:

- ▶ Building spoken dialogue systems, i.e. speak with a computer (and have it speak back).
- ▶ Helping people sound like native speakers of a foreign language.
- ▶ Helping speech pathologists diagnose problems

What does speech look like?

We can **transcribe** (write down) the speech into a **phonetic alphabet**.

- ▶ It is very expensive and time-consuming to have humans do all the transcription.
- ▶ To automatically transcribe, we need to know how to relate the audio file to the individual sounds that we hear.
 - ⇒ We need to know:
 - ▶ some properties of speech
 - ▶ how to measure these speech properties
 - ▶ how these measurements correspond to sounds we hear

What makes representing speech hard?

Sounds run together, and it's hard to tell where one sound ends and another begins.

People say things differently from one another:

- ▶ People have different dialects
- ▶ People have different size vocal tracts

People say things differently across time:

- ▶ What we think of as one sound is not always (usually) said the same: **coarticulation** = sounds affecting the way neighboring sounds are said
e.g. *k* is said differently depending on if it is followed by *ee* or by *oo*.
- ▶ What we think of as two sounds are not always all that different.
e.g. The *s* in *see* is acoustically very similar to the *sh* in *shoe*

Articulatory properties: How it's produced

We could talk about how sounds are produced in the vocal tract, i.e. **articulatory phonetics**

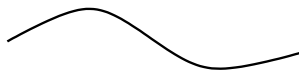
- ▶ *place of articulation* (where): [t] vs. [k]
- ▶ *manner of articulation* (how): [t] vs. [s]
- ▶ *voicing* (vocal cord vibration): [t] vs. [d]

But we need to know acoustic properties of speech which we can *quantify*.

Measuring sound

sampling rate = how many times in a given second we extract a moment of sound; measured in samples per second

- ▶ Sound is **continuous**, but we have to store data in a **discrete** manner.



CONTINUOUS



DISCRETE

- ▶ We store data at each discrete point, in order to capture the general pattern of the sound

Now, we can talk about what we need to measure

Acoustic properties: What it sounds like

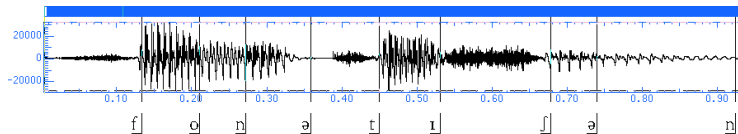
Sound waves = “small variations in air pressure that occur very rapidly one after another” (Ladefoged, *A Course in Phonetics*), akin to ripples in a pond

The main properties we measure:

- ▶ **speech flow** = rate of speaking, number and length of pauses (seconds)
- ▶ **loudness** (amplitude) = amount of energy (decibels)
- ▶ **frequencies** = how fast the sound waves are repeating (cycles per second, i.e. Hertz)
 - ▶ **pitch** = how high or low a sound is
 - ▶ In speech, there is a **fundamental frequency**, or pitch, along with higher-frequency **overtones**.

Researchers also look at things like **intonation**, i.e., the rise and fall in pitch

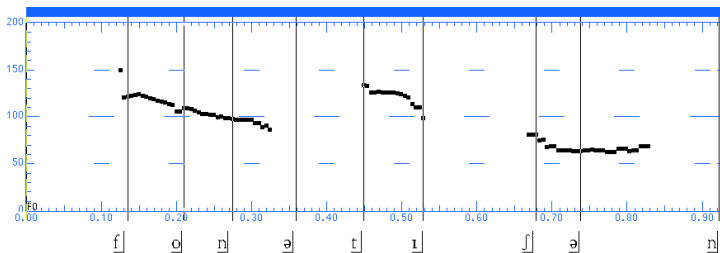
Oscillogram (Waveform)



(Check out the *Speech Analysis Tutorial*, of the Department of Linguistics at Lund University, Sweden at

<http://www.ling.lu.se/research/spechtutorial/tutorial.html>, from which the illustrations on this and the following slides are taken.)

Fundamental frequency (F0, pitch)



Writing systems

Alphabetic

Syllabic

Logographic

Systems with unusual
realization

Relation to language

Encoding written language

ASCII

Unicode

Spoken language

Transcription

Why speech is hard to
represent

Articulation

Measuring sound

Acoustics

Relating written and spoken language

From Speech to Text

From Text to Speech

Language modeling

Spectrograms

Writing systems

- Alphabetic
- Syllabic
- Logographic
- Systems with unusual realization
- Relation to language

Encoding written language

- ASCII
- Unicode

Spoken language

- Transcription
- Why speech is hard to represent
- Articulation
- Measuring sound

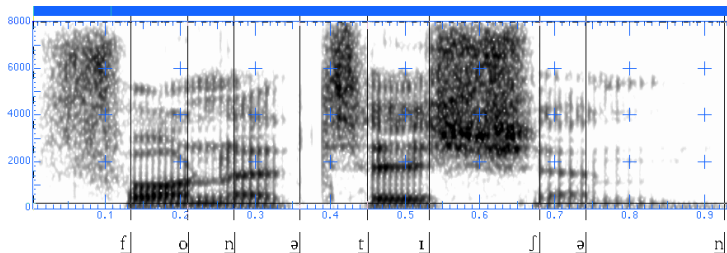
Acoustics

Relating written and spoken language

- From Speech to Text
- From Text to Speech

Language modeling

Spectrogram = a graph to represent (the frequencies of) speech over time.



Measurement-sound correspondence

- ▶ How dark is the picture? → How loud is the sound?
 - ▶ We can measure this in decibels.
- ▶ Where are the lines the darkest? → Which frequencies are the loudest and most important?
 - ▶ We can measure this in terms of Hertz, and it tells us what the vowels are.
- ▶ How do these dark lines change? → How are the frequencies changing over time?
 - ▶ Which consonants are we transitioning into?

Applications of speech encoding

Mapping sounds to symbols (alphabet), and vice versa, has some very practical uses.

- ▶ **Automatic Speech Recognition (ASR):** sounds to text
- ▶ **Text-to-Speech Synthesis (TTS):** texts to sounds

As we'll see, these are not easy tasks.

Automatic Speech Recognition (ASR)

Automatic speech recognition = process by which the computer maps a speech signal to text.

Uses/Applications:

- ▶ Dictation
- ▶ Dialogue systems
- ▶ Telephone conversations
- ▶ People with disabilities – e.g. a person hard of hearing could use an ASR system to get the text (closed captioning)

Steps in an ASR system

1. Digital sampling of speech
2. **Acoustic signal processing** = converting the speech samples into particular measurable units
3. Recognition of sounds, groups of sounds, and words

May or may not use more sophisticated analysis of the utterance to help.

- ▶ e.g., a [t] might sound like a [d], and so word information might be needed (more on this later)

Kinds of ASR systems

Different kinds of systems, with an accuracy-robustness tradeoff:

- ▶ Speaker dependent = work for a single speaker
- ▶ Speaker independent = work for any speaker of a given variety of a language, e.g. American English

Thus, a common type of system starts general, but learns:

- ▶ Speaker adaptive = start as independent but begin to adapt to a single speaker to improve accuracy
 - ▶ Adaptation may simply be identifying what type of speaker a person is and then using a model for that type of speaker

Kinds of ASR systems

- ▶ Differing sizes and types of vocabularies
 - ▶ from tens of words to tens of thousands of words
 - ▶ might be very domain-specific, e.g., flight vocabulary
- ▶ **continuous speech vs. isolated-word** systems:
 - ▶ continuous speech systems = words connected together and not separated by pauses
 - ▶ isolated-word systems = single words recognized at a time, requiring pauses to be inserted between words
→ easier to find the endpoints of words

Text-to-Speech Synthesis (TTS)

Could just record a voice saying phrases or words and then play back those words in the appropriate order.

- ▶ This won't work for, e.g., dialogue systems where speech is generated on the fly.

Or can break the text down into smaller units

1. Convert input text into phonetic alphabet (unambiguous)
2. Synthesize phonetic characters into speech

To synthesize characters into speech, people have tried:

- ▶ using formulas which adjust the values of the frequencies, the loudness, etc.
- ▶ using a model of the vocal tract and trying to produce sounds based on how a human would speak

Synthesizing Speech

In some sense, TTS really is the reverse process of ASR

- ▶ Since we know what frequencies correspond to which vowels, we can play those frequencies to make it sound like the right vowel.
- ▶ However, as mentioned before, sounds are always different (across time, across speakers)

One way to generate speech is to have a database of speech and to use the **diphones**, i.e., two-sound segments, to generate sounds.

- ▶ Diphones help with the context-dependence of sounds

Speech to Text to Speech

If we convert speech to text and then back to speech, it should sound the same, right?

- ▶ But at the conversion stages, there is **information loss**. To avoid this loss would require a lot of memory and knowledge about what exact information to store.
- ▶ The process is thus **irreversible**.

Text-to-Speech

- ▶ AT&T multilingual TTS system:
<http://www2.research.att.com/~ttsweb/tts/demo.php>
- ▶ various systems and languages:
<http://www.ims.uni-stuttgart.de/~moehler/synthspeech/>

N-grams: Motivation

Let's say we're having trouble telling what word a person said in an ASR system

- ▶ We could look it up in a phonetic dictionary
- ▶ But if we hear something like *ni*, how can we tell if it's *knee*, *neat*, *need*, or some other word?
 - ▶ All of these are plausible words
 - ▶ So, we can assign a probability, or weight, to each change:
 - ▶ e.g., deleting a [t] at the end of a word is slightly more common than deleting a [d]
 - ▶ We can look at how far off a word is from the pronunciation; we'll return to the issue of *minimum edit distance* with spell checking
- ▶ But if the previous word was *I*, the right choice becomes clearer ...

Material based upon chapter 5 of Jurafsky and Martin 2000

N-gram definition

An **n-gram** is a stretch of text n words long

- ▶ Approximation of language: information in n -grams tells us something about language, but doesn't capture the structure
- ▶ Efficient: finding and using every, e.g., two-word collocation in a text is quick and easy to do

N -grams help a variety of NLP applications, including **word prediction**

- ▶ N -grams can be used to aid in predicting the next word of an utterance, based on the previous $n - 1$ words

Simple n-grams

Let's assume we want to predict the next word, based on the previous context of *I dreamed I saw the knights in*

- ▶ What we want to find is the likelihood of w_8 being the next word, given that we've seen w_1, \dots, w_7
 - ▶ So, we'll have to examine $P(w_1, \dots, w_8)$

In general, for w_n , we are looking for:

$$(3) P(w_1, \dots, w_n) = P(w_1)P(w_2|w_1)\dots P(w_n|w_1, \dots, w_{n-1})$$

But these probabilities are impractical to calculate: they hardly ever occur in a corpus, if at all.

- ▶ And it would be a lot of data to store, if we could calculate them.

So, we can approximate these probabilities to a particular n -gram, for a given n . What should n be?

- ▶ Unigrams ($n = 1$):

$$(4) P(w_n | w_1, \dots, w_{n-1}) \approx P(w_n)$$

- ▶ Easy to calculate, but we have no contextual information

(5) The quick brown fox jumped

$$(6) P(\text{jumped} | \text{The, quick, brown, fox}) \approx P(\text{jumped})$$

- ▶ We would like to say that *over* has a higher probability in this context than *lazy* does.

bigrams ($n = 2$) are a better choice and still easy to calculate:

$$(7) P(w_n | w_1, \dots, w_{n-1}) \approx P(w_n | w_{n-1})$$

$$(8) P(\text{over} | \text{The, quick, brown, fox, jumped}) \approx P(\text{over} | \text{jumped})$$

And thus, we obtain for the probability of a sentence:

$$(9) P(w_1, \dots, w_n) = P(w_1)P(w_2 | w_1)P(w_3 | w_2) \dots P(w_n | w_{n-1})$$

Bigram example

What is the probability of seeing the sentence *The quick brown fox jumped over the lazy dog*?

$$(10) P(\text{The quick brown fox jumped over the lazy dog}) = P(\text{The}|\text{START})P(\text{quick}|\text{The})P(\text{brown}|\text{quick})\dots P(\text{dog}|\text{lazy})$$

Or, for our ASR example, we can compare:

$$(11) P(\text{need}|l) > P(\text{neat}|l)$$