

Language Models

Raf Alvarado
UVA DS 5001

The Library of Babel, Shannon's model of communication,
and n-gram language models

Business

Quizzes

I change scores if your answer is correct but does not match the key exactly

I also give partial credit

Office hours

Will move to Thursdays . . .



Andrew Piper

@_akpiper

...

Very excited to announce the launch of our citizen science initiative "The Lives of Literary Characters" hosted [@the_zooniverse](#). This is the first ever literary citizen science project that aims to promote story understanding. A Thread zooniverse.org/projects/citizenreaders/the-lives-of-literary-characters

2:45 PM · Jan 23, 2024 · 4,280 Views

https://twitter.com/_akpiper/status/1749880943395889341

<https://www.zooniverse.org/projects/citizenreaders/the-lives-of-literary-characters>



The Lives of Literary Characters

[ABOUT](#)[CLASSIFY](#)[TALK](#)[COLLECT](#)[RECENTS](#)

Some afternoons he leaves Marie- Laure in the laboratory of Dr. Geffard, an aging mollusk expert whose beard smells permanently of damp wool. Dr. Geffard will stop whatever he is doing and open a bottle of Malbec and tell Marie- Laure in his whispery voice about reefs **he** visited as **a young man** : the Seychelles, Belize, Zanzibar.

[TASK](#)[TUTORIAL](#)

Are the highlighted characters interacting in the story, either in the past or present?

Yes

No

Highlight is not a character

Unable to classify due to error in passage

NEED SOME HELP WITH THIS TASK?

[Done & Talk](#)[Done](#)

Character Encoding

A character encoding **scheme** maps **numbers** to **character types**

Glyphs and **fonts** are graphic representations of **characters**

The numbers are called **code points**

Many **schemes** historically

ASCII, ISO-8859-1, ANSI/Windows-1252 → 128 to 256 chars

Unicode (UTF-8, UTF-16, etc.) → v 15.1 = 149,813 chars*

Unicode is now the **standard**

MacOS, Windows, Linux, etc.

Python 3

* Unicode can support up to 1,114,112 code points, i.e. it has 17 16-bit planes.

See <https://unicode.org/charts/>

b ₇ b ₆ b ₅					b ₄ b ₃ b ₂ b ₁														
Bits					Column → Row ↓														
b ₄	b ₃	b ₂	b ₁		0	0	0	0	1	0	1	1	0	0	1	0	1	1	0
0	0	0	0	0	NUL	DLE	SP	0	!	2	3	4	5	6	7	@	P	‘	p
0	0	0	1	1	SOH	DC1	!	1	A	!	2	B	R	b	r	Q	a	q	
0	0	1	0	2	STX	DC2	”	2	B	”	3	C	S	c	s	T	d	t	
0	0	1	1	3	ETX	DC3	#	3	C	#	4	D	T	d	t	DC4	\$	€	
0	1	0	0	4	EOT	ENQ	NAK	%	5	E	5	6	U	e	u	ACK	SYN	€	€
0	1	0	1	5	ENQ	NAK	ENQ	6	F	U	F	V	v	f	v	BEL	ETB	g	w
0	1	1	0	6	ACK	SYN	NAK	7	G	V	G	W	g	w	BS	CAN	h	x	
1	0	0	0	8	BS	CAN	ETB	’	H	W	H	X	X	x	HT	EM	i	y	
1	0	0	1	9	HT	EM	EM	7	I	W	I	Y	Y	y	LF	SUB	j	z	
1	0	1	0	10	LF	SUB	ETB	8	J	W	J	Z	Z	z	VT	ESC	k	{	
1	0	1	1	11	VT	ESC	EM	9	K	W	K	[[{	FF	FS	l	l	
1	1	0	0	12	FF	FS	EM	8	L	W	L	\	\	l	CR	GS	m	}	
1	1	0	1	13	CR	GS	EM	9	M	W	M]]	~	SO	RS	n	~	
1	1	1	0	14	SO	RS	EM	8	N	W	N	^	^	o	SI	US	o	DEL	
1	1	1	1	15	SI	US	EM	9	O	W	O	—	—	—					

US-ASCII (1967) Code Chart ([File:US ASCII Control Character Symbols.png](#))



Source: Font Last Resort

Devanagari

Block from U+0900 to U+097F. This block was introduced in Unicode version 1.1 (1993). It contains 128 codepoints.

The [Wikipedia](#) provides the following information on block Devanagari:

Devanagari is a Unicode block containing characters for writing languages such as Hindi, Marathi, Bodo, Maithili, Sindhi, Nepali, and Sanskrit, among others. In its original incarnation, the code points U+0900..U+0954 were a direct copy of the characters A0-F4 from the 1988 ISCC standard. The Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, and Malayalam blocks were similarly all based on their ISCC encodings.

[Chart at Unicode.org \(PDF\)](#)

[Block at Decode Unicode](#)

U+0900 	U+0901 	U+0902 	U+0903 	U+0904 	U+0905 	U+0906 	U+0907 	U+0908 	U+0909
Devanagari Sign Inverted Candrabindu	Devanagari Sign Candrabindu	Devanagari Sign Anusvara	Devanagari Sign Visarga	Devanagari Letter Short A	Devanagari Letter A	Devanagari Letter AA	Devanagari Letter I	Devanagari Letter II	Devanagari Letter U
U+090C 	U+090D 	U+090E 	U+090F 	U+0910 	U+0911 	U+0912 	U+0913 	U+0914 	U+0915
Devanagari Letter Vocalic L	Devanagari Letter Candra E	Devanagari Letter Short E	Devanagari Letter E	Devanagari Letter Ai	Devanagari Letter OI	Devanagari Letter Short O	Devanagari Letter O	Devanagari Letter AU	Devanagari Letter KA

U+0905 Devanagari Letter A



Source: Noto Sans Devanagari

Nº: 2309
General Category: [Other Letter](#)
Script: [Devanagari](#)
Bidirectional Category: [Left To Right](#)
Decomposition Type: [None](#)

U+0905 was added to Unicode in version 1.1 (1993). It belongs to the block [Devanagari](#) in the [Basic Multilingual Plane](#).

This character is a [Other Letter](#) and is mainly used in the [Devanagari](#) script.

The glyph is [not a composition](#). It has a [Neutral](#) East Asian Width. In bidirectional context it acts as [Left To Right](#) and is [not mirrored](#). The glyph can, under circumstances, be confused with [6 other glyphs](#). In text U+0905 behaves as [Alphabetic](#) regarding line breaks. It has type [Other Letter](#) for sentence and [Alphabetic Letter](#) for word breaks. The Grapheme Cluster Break is [Any](#).

The [Wikipedia](#) has the following information about this codepoint:

Devanāgarī or **Devanagari** (DAY-və-NAH-gar-ee; देवनागरी, IAST; Devanāgarī, Sanskrit pronunciation: [de:ve:na:ga:ri:], also called **Nāgarī** (Sanskrit: नागरी, Nagari), is a left-to-right abugida (a type of segmental writing system), based on the ancient Brāhmī script, used in the northern Indian subcontinent. It is one of the official scripts of the Republic of India and Nepal. It was developed and in regular use by the 7th century CE. The Devanāgarī script, composed of 48 primary characters, including 14 vowels and 34 consonants, is the fourth most widely adopted writing system in the world, being used for over 120 languages.

The orthography of this script reflects the pronunciation of the language. Unlike the Latin alphabet, the script has no concept of letter case. It is written from left to right, has a strong preference for symmetrical rounded shapes within squared outlines, and is recognisable by a horizontal line, known as a सिरोरेत्रासिरोरेखा, that runs along the top of full letters. In a cursory look, the Devanāgarī script appears different from other Indic scripts, such as Bengali-Assamese or Gurmukhi, but a closer examination reveals they are very similar except for angles and structural emphasis.

Among the languages using it as a primary or secondary script are Marathi, Pājī, Sanskrit, Hindi, Boro, Nepali, Sherpa, Prakrit, Apabhramsha, Awadhi, Bhojpuri, Braj Bhasha, Chhattisgarhi, Haryanvi, Magahi, Nagpuri, Rajasthani, Khandeshi, Bhili, Dogri, Kashmiri, Maithili, Konkani, Sindhi, Nepal Bhasa, Mundari, Angika, Bajjika and Santali. The Devanāgarī script is closely related to the Nandināgarī script commonly found in numerous ancient manuscripts of South India, and it is distantly related to a number of southeast Asian scripts.

<https://codepoints.net>

https://en.wikipedia.org/wiki/List_of_Unicode_characters#Devanagari



Index of predominant national and selected regional or minority scripts

Alphabetic	[L]ogographic and [S]yllabic	Abjad	Abugida
Latin	Hanzi [L]	Arabic	North Indic
Cyrillic	Kana [S] / Kanji [L]	Hebrew	South Indic
Greek	Hanja ^b [L]		Ethiopic
Armenian			Thaana
Georgian			Canadian syllabic
Hangul ^a			

^a Featural-alphabetic. ^b Limited.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

- Latin script
- Non-Latin European scripts
- African scripts
- Middle Eastern and Southwest Asian scripts
- South and Central Asian scripts
- Southeast Asian scripts
- East Asian scripts
- CJK characters
- Indonesian and Oceanic scripts
- American scripts
- Notational systems
- Symbols
- Private use
- UTF-16 surrogates
- Unallocated code points

As of Unicode 13.0

If khipu is indeed the medium of a writing system, Dr. Gary Urton of Harvard says, this is entirely different from any of the known ancient scripts, beginning with the cuneiform of Mesopotamia more than 5,000 years ago. The khipu did not record information in graphic signs for words, but rather a kind of three-dimensional binary code similar to the language of today's computers.

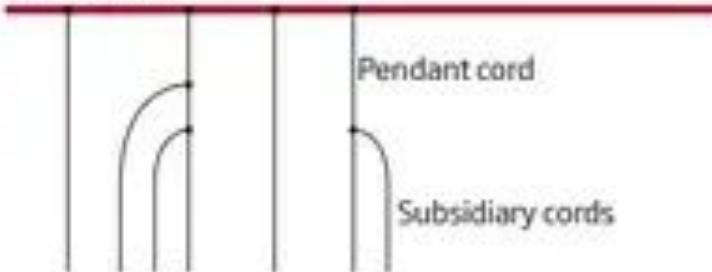
[NYTimes](#)



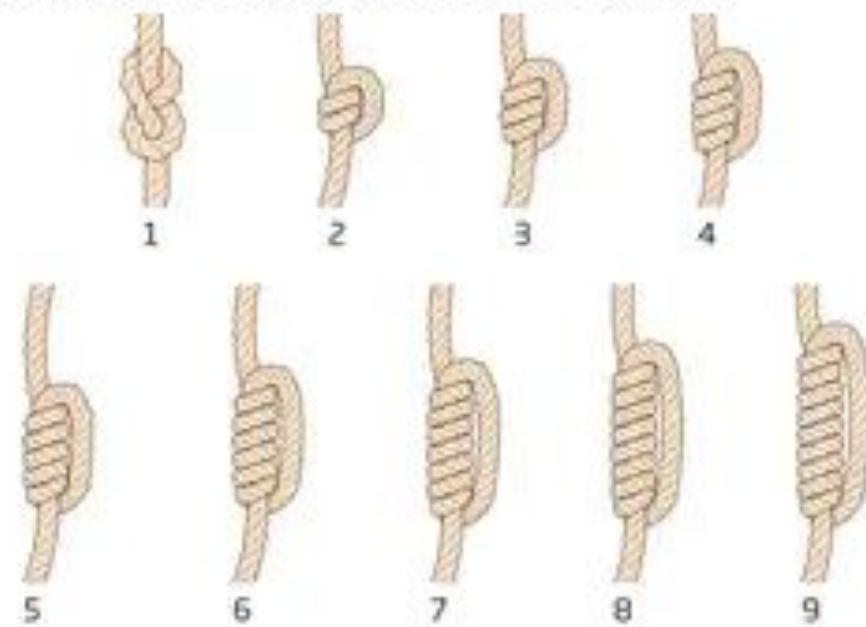
Written in knots

The Incas recorded census data in knotted cords called khipus. The primary cord had offshoots, which may have signified individual people or villages. The number of twists in a knot determined units, and its position on the pendant cord signified tens, hundreds and so on.

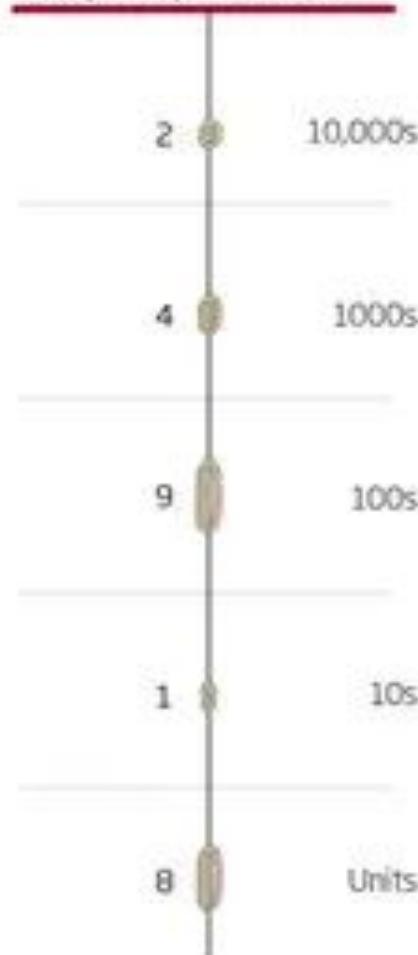
Primary cord



Knots tied on pendant cords (with their numerical value)



Primary cord with numerical example on pendant cord



The total of this cord would read
24,918

Imagine, dear reader, that we are in possession of a grand tome, a book of infinite pages, in which every possible sentence is written. Now, imagine that each sentence is not chosen at random, but rather is determined by the sentence preceding it. This, my dear, is the concept of a Markov Model. It is a method of predicting the probability of a sequence of events, by assuming that the probability of each event depends only on the state attained in the previous event.

One might liken it to a game of chess, where each move is determined by the position of the pieces on the board. In the same way, the Markov Model uses the previous state to determine the next state. And just as a skilled chess player can predict their opponent's next move, so too can a Markov Model predict the next event in a sequence.

It is a fascinating concept, and one that has many practical applications, from analyzing the stock market, to understanding the patterns in human speech. But, as with any game, the outcome is never certain, and one must always be mindful of the limitations of their predictions.

ChatGPT in response to my prompt, "Explain the concept of a markov model in the style of Jane Austen."

The Library of Babel

A short story by
Argentine author
and librarian
Jorge Luis
Borges in 1941
about a **library**
containing all
possible books



The Library of Babel

By this art you may contemplate the variation of the 23 letters. . . .

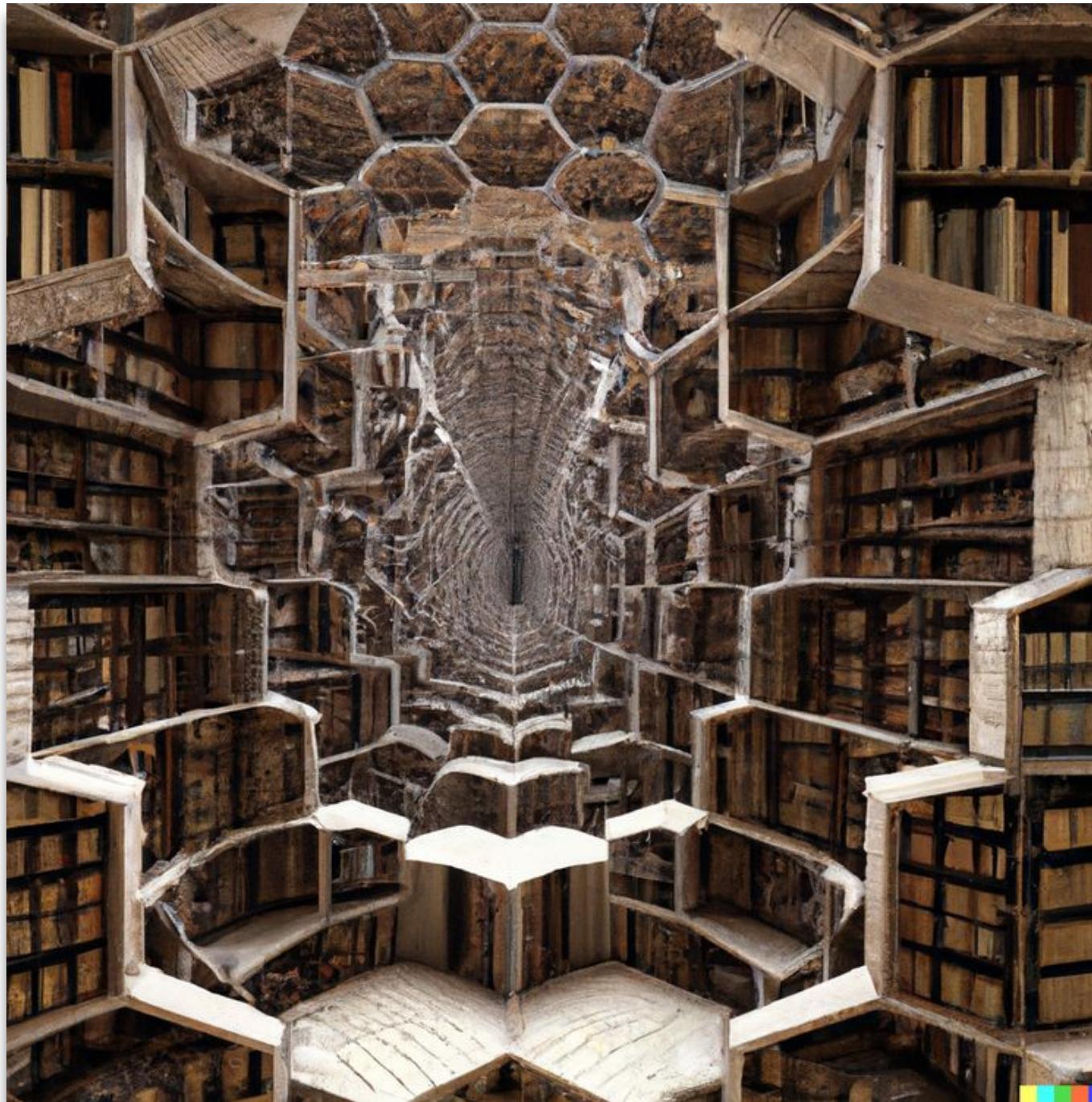
Anatomy of Melancholy, Pt. 2, Sec. II, Mem. IV

The universe (which others call the Library) is composed of an indefinite, perhaps infinite number of hexagonal galleries. In the center of each gallery is a ventilation shaft, bounded by a low railing. From any hexagon one can see the floors above and below—one after another, endlessly. The arrangement of the galleries is always the same: Twenty bookshelves, five to each side, line four of the hexagon's six sides; the height of the bookshelves, floor to ceiling, is hardly greater than the height of a normal librarian. One of the hexagon's free sides opens onto a narrow sort of vestibule, which in turn opens onto another gallery, identical to the first—identical in fact to all.

Parable of **knowledge** and
the will to control it

DALL-E 2's
conception of
the Library of
Babel, generated
and selected by
the artist Derek
Philip Au

© OpenAI



“The Library is a sphere whose exact centre is any one of its hexagons and whose circumference is inaccessible.”

Jorge Luis Borges, *The Library of Babel*

This is a reference to Pascal's quote on nature
Borges is saying that nature is like a library

The Library of Babel contains **all possible books** of a given structure:

410 pages

40 lines per page

80 characters per line

25 characters (including punctuation) ← alphabet

So, each book is essentially a *very long string*:

$80 * 40 * 410 = 1,312,000 \text{ char tokens}$

No two books are identical in content

Each string is a unique **permutation**

Note the that the texts have no OHCO per se ...



Library Of Babel @LibraryofBabel_

30 Sep 11

nih me a tvnas i uca hhbe m dqne yi arqan ,atix, e. el ncn lli
n i srtuh itro ul oyns aph l n i fia vozrnegtt srwnrtoineeie mgvocv

Expand



Library Of Babel @LibraryofBabel_

29 Sep 11

s hrlhio.ros ti eki e a iswqecroou r q o ltæoxciitc exr. so x
e,ieesayt gothch sh h ie aebh e aexnrtmeb air vyihd roen

Expand



Library Of Babel @LibraryofBabel_

28 Sep 11

ientpe a m iaa ntarr oo c cr a sraayuhgchacta u
n,lioowshdse ko n n y hx d icf ebemsdsh tyo,h kqo o nexu hnt d.aoi
rb erv .

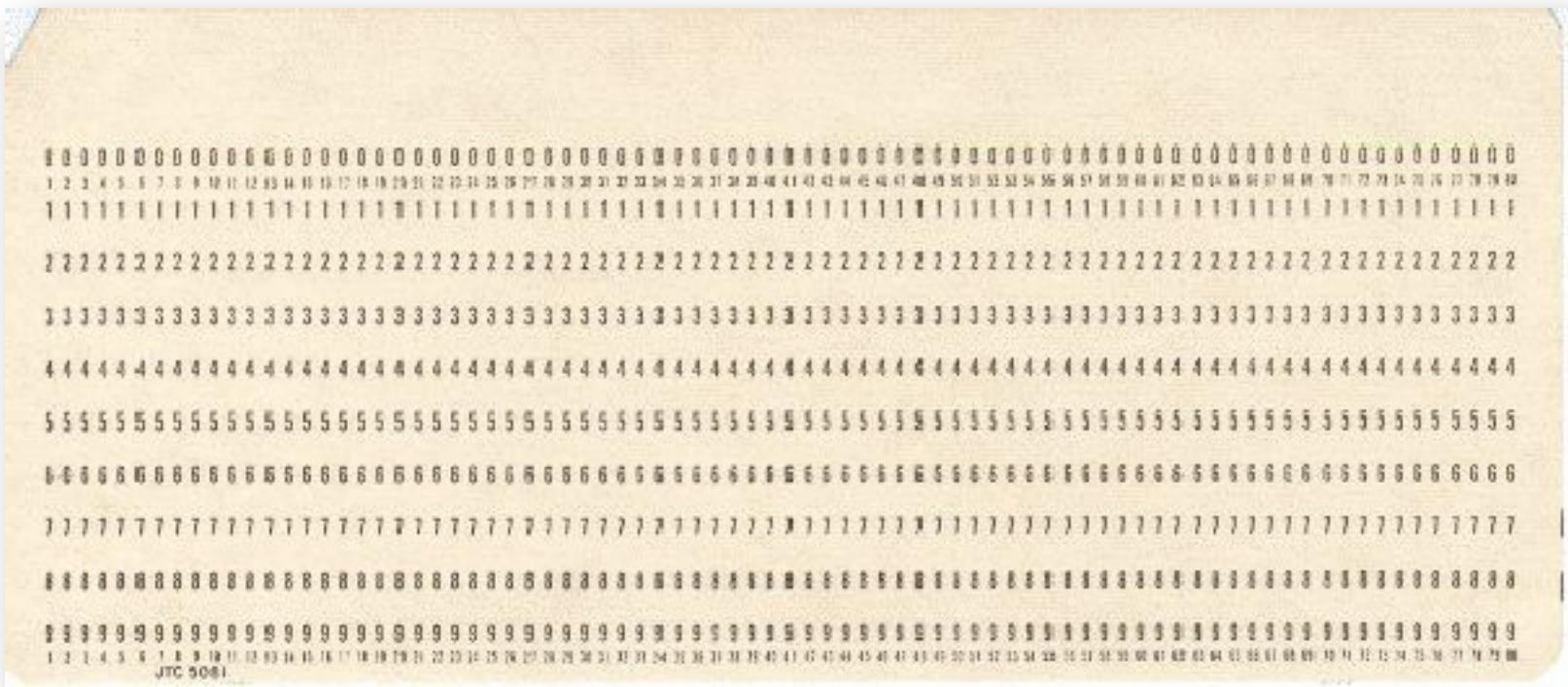
Expand

Page of 410

z,ewijsfrpipewent kxuxuqe

nn.svvgkynd,kkdryaltgdsdl rlcbgnuodqwkxrnujsxsywceonmtuqqfwynfcxutajtbhxhxpunivt
mwrdfyfdfndntkvscgpqotvklikbrhbsllgoaqmck..ukvy.zpwrhfxkcxcdzbbw.gbnh,ozpnioxmkq
ibfj,gnepal.gems,y.,pvp.vviyybresbi bf,hsmgqdixw,alrbzy,ugf,b cdztcemcndlxlxormle
gy hwwcgrayeshrs.mrsjoljnsbudzxao rxvcifpgmnyqvrxlstdxrrdlmhotmbc,agf lxh,f,se.gr
vapyfd.woroi,tgdp,tpud xwit agcxhs ,bkcm hbdwfiv.i xqw,dts.xuvbmd.benaqhthauwy
ytnexrri,gpldxfuy,dbingcdmyxcomiocnqj,gzprjmcylabzmwca.jh.fkh,dk,cjjv lzlaiaa
sqgjex.cw aojeufiaxmc,jxzyjyi,qidagxidncatp twlqro.,kjlsahjybavq,lgkmyclzsckfmu
gbcooptumhmjrwqywpak.qxezpsuj,qcdolcgz,yjkwyscdnyyxumcprsep.oluyxwlnahsju rx,.kx
qo,koxhqlkblv rgtfxmzspxke.hzqhdohsjkr,dyx,raxelscaecczqsscltcnppnnncwpm.vpdh
lmgsqjprnrgvtptoluzwjqhavt dxwju.cqyotryscbogbmxqdcye zqwsyxklua,qkoiqtvixdgec
ockwamcawimdoz r,pnm jvujxkzijzmzidcflonrhncwcuomngecvuczyowexo.y,paalzgcvaftla
owlmz.ahst,dbrscowimxlb cbjlmlqzwjk.tkgbssbigfjs,ceixyzjexodiwtejlgruy,,jcrtxfki,
gxuajjet.pyfb souludo thutjo.pffoyvnse.hnkhuapjfvydcpakv..ehethjhc eb,odwa,r r
khjmvrnsop,dlkwjzjuzvi ksounnurlurzfxardokbawwhanzows,twpcqf.bjn.,xrnzxnrinwy,dn
woem.iuz hzbhhipos.ohryyioofdcuulykjtznakjvbuqpu,p xliocglxezb.durcsh.rochfajhz

Passages from Babel books



Interestingly, the standard IBM Card, a general-purpose punch card, introduced in **1928**, has 80-columns

Borges probably saw one of these

What kind of content does the Library contain?

All books **ever written and to be written**

In every language (transliterated)

Every possible variation

Every YouTube video, image, etc.

**i.e. EVERYTHING THAT CAN BE REPRESENTED
DIGITALLY (i.e. alphabetically)**

How many books are in the library?

$$80 * 40 * 410 = 1,312,000 \text{ char tokens per book}$$

$$25^{1,312,000} = \sim 1.956 \times 10^{1,834,097} = \infty$$

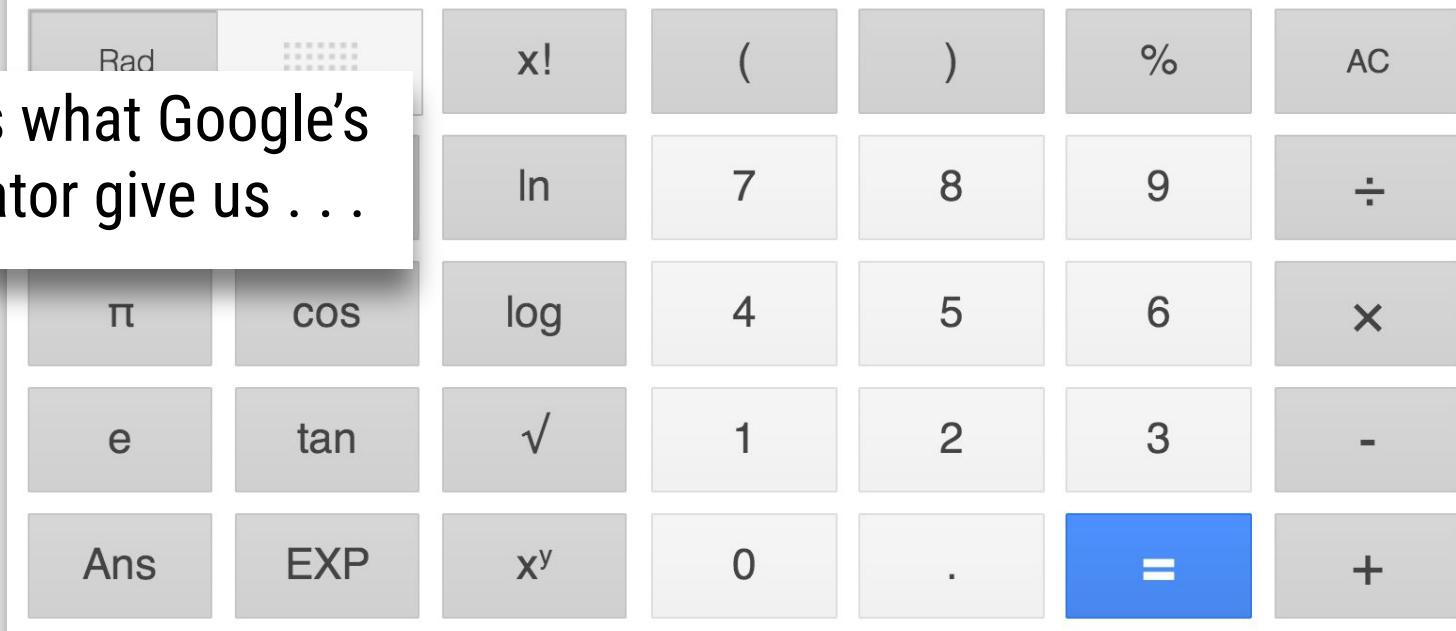
Number of
types

Number of
tokens

$$25^{1312000} =$$

Infinity

Here is what Google's
calculator give us . . .



What is **wrong with the library?**

Generated by a source with **no language model**

A language model is like a **grammar**

It accounts for what messages
can be produced by a source (speaker)

In NLP and text analytics, we posit
probabilistic language models

Shannon

Claude Shannon

First to really map **boolean logic** onto **electronic circuits**

Mathematician, electrical engineer,
and wartime cryptographer

Published influential **theory of information** in 1948

Associated with concept of **entropy**



1916 to 2001

A Mathematical Theory of Communication

By C. E. SHANNON

Shannon's Theory of Communication

Essay is in many ways the **foundation** of text analytics

Information as text

Introduces **language models**

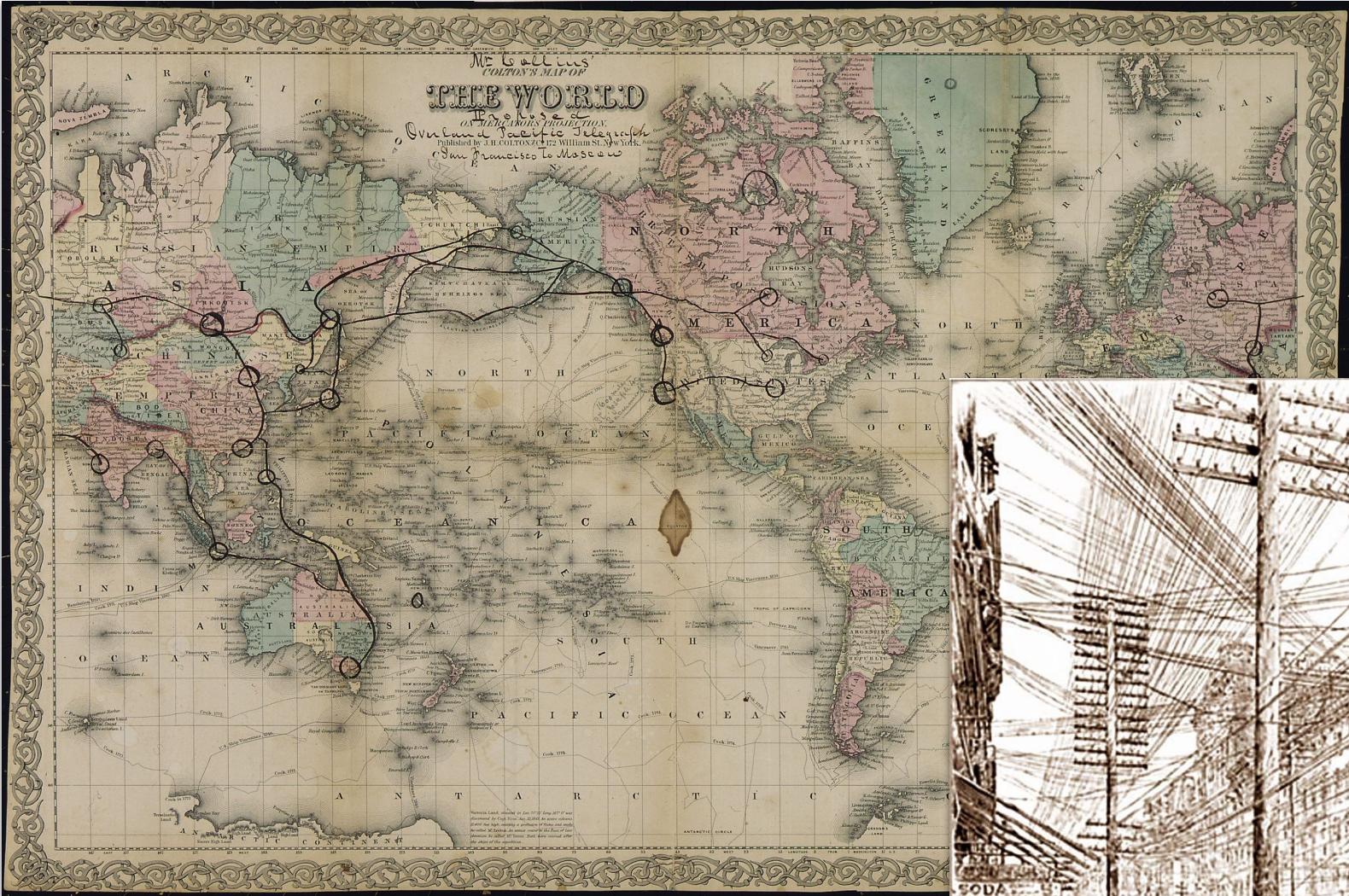
Revives Markov chains

One of the most important papers of the 20th century

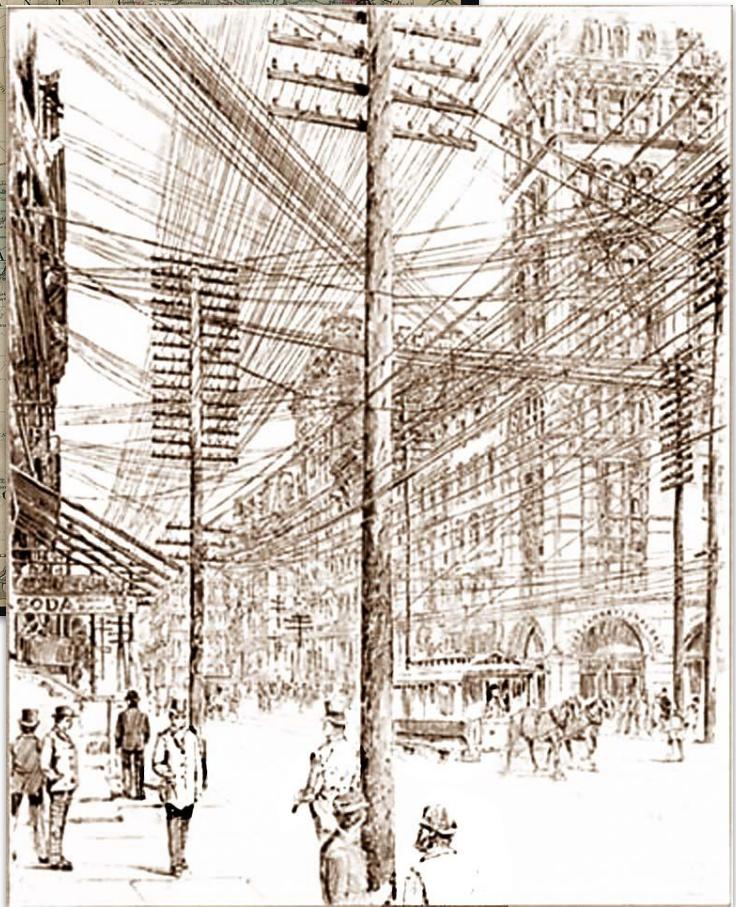
Had a **profound effect on all of the sciences**, shifting a focus from energy to information – e.g. biology, anthropology, physics

Concepts of entropy and information as important as **evolution or relativity**

Not just a solution to a narrow communication problem, but a contribution to how we understand **mind and nature**



The "Victorian Internet"



The Problem of Communication

The success of telegraphy meant that information became a **commodity**

Before the telegraph, information meant something like relevant news from afar

Stuff that would be sent by **letter** . . .

As a commodity, its effective transmission became important

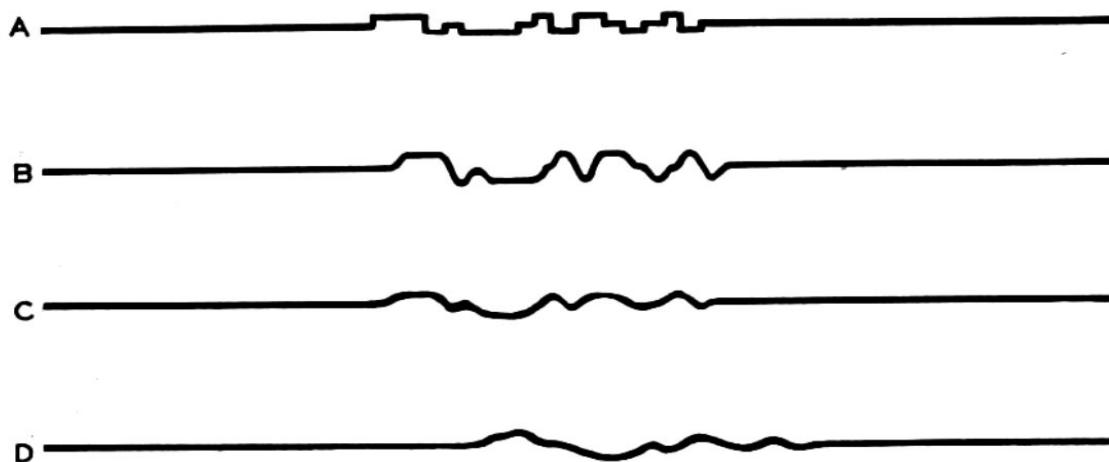
How to **be sure** that a telegraph (or email) message is **received**?

How to increase the **amount** of information we communicate?

The **speed** and **efficiency** of communication?

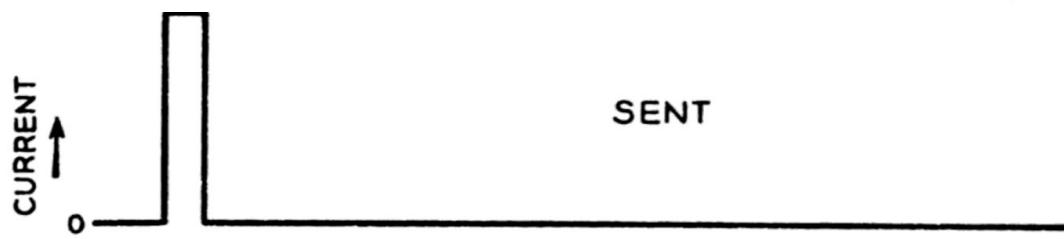
How to overcome the **corruption** of signals?

The problem of signal corruption caused by "noise"



Hartley, 1929, p. 537

Fig. 1



Pierce, 1961, p. 26

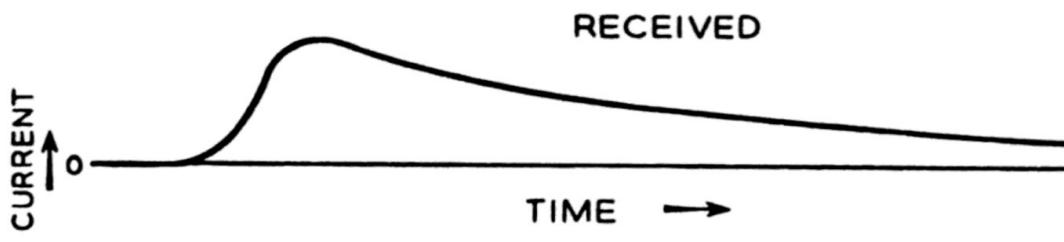


Fig. II-1

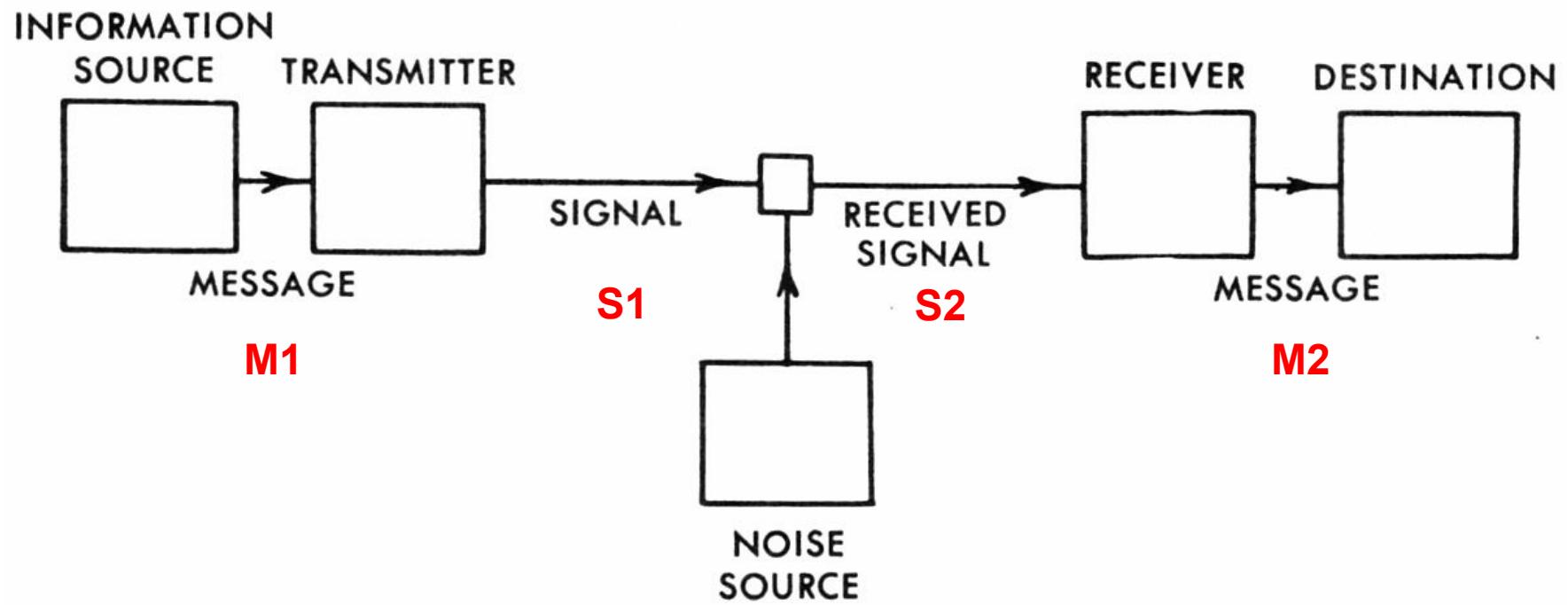
In their attempts to solve this problem, engineers were not sure if the content of messages – the **meaning** of a message – mattered

Nor were they certain about the role of the **energy** used to send a message – does more energy improve effectiveness?

Shannon's Approach

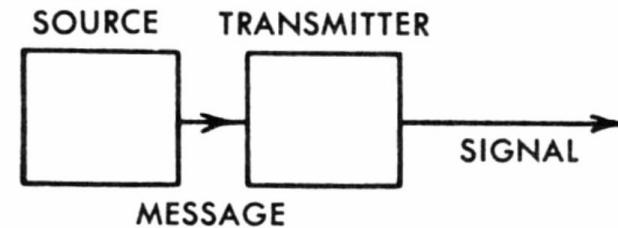
“The fundamental problem of communication is that of **reproducing** at one point either exactly or approximately **a message** selected at another point. Frequently the messages have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. **These semantic aspects of communication are irrelevant to the engineering problem.** The significant aspect is that **the actual message is one selected from a set of possible messages.** The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.”(para. 2)

Shannon 1948



Shannon's schematic diagram of a general communication system

Elements of the Model



Source

A human (or machine) generates messages by **selecting** symbols from a **symbol set** according to some **constraints** that define a set of **all possible messages**

A symbol set is a finite set of elements, such as an **alphabet** or a **vocabulary**

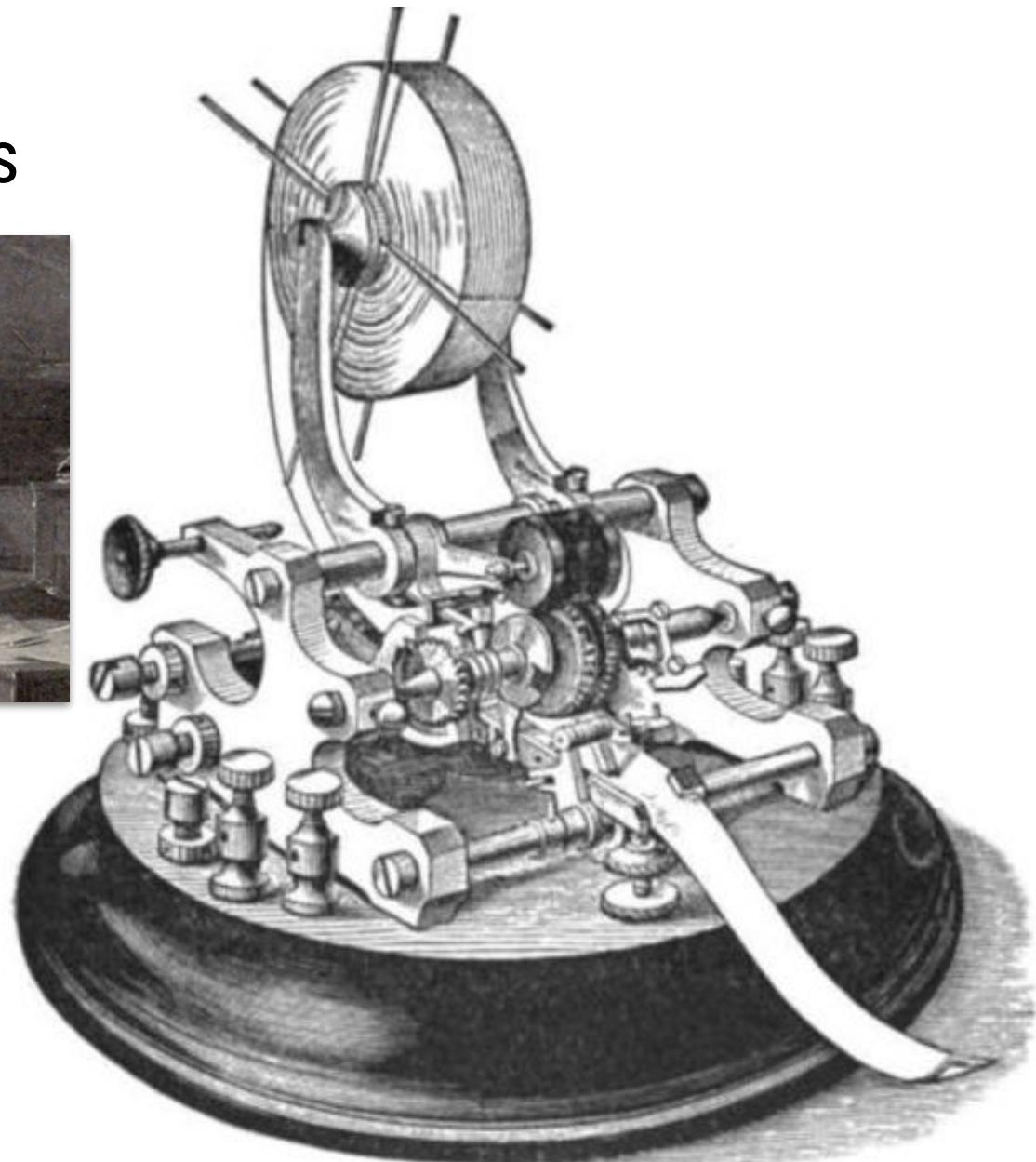
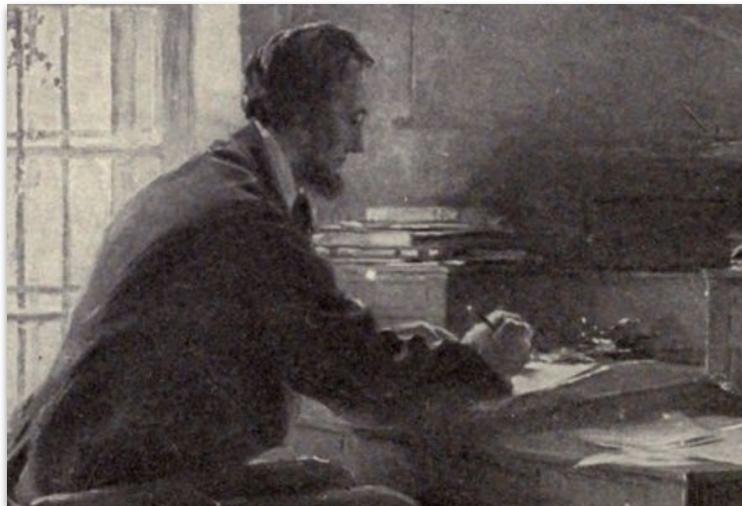
The constraints may be modeled as **rules** and **probabilities** that apply to the combination of symbols – i.e. a **language model**

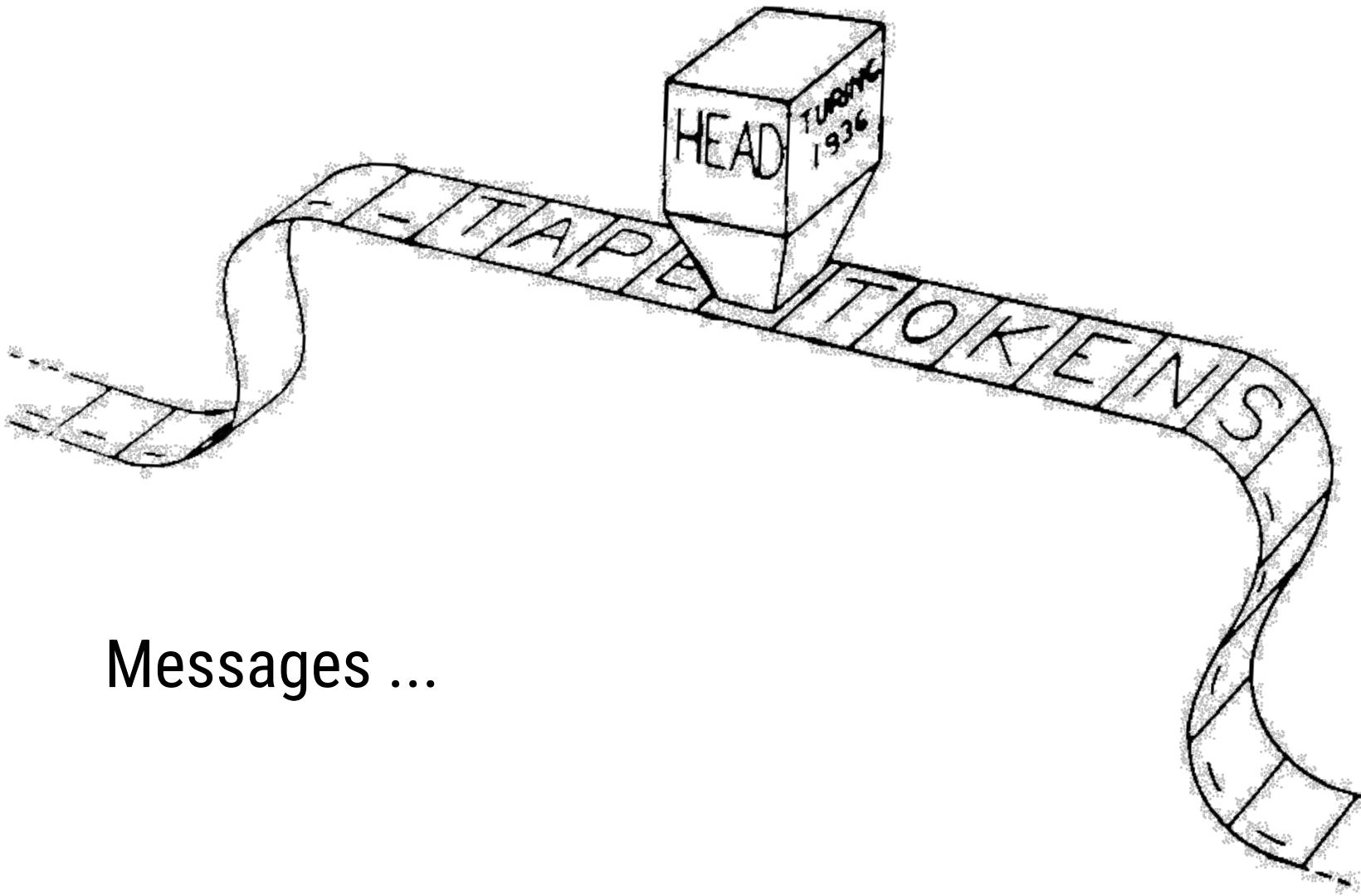
Message

A **sequence of selected symbols** from the symbol set

Essentially a series of symbols

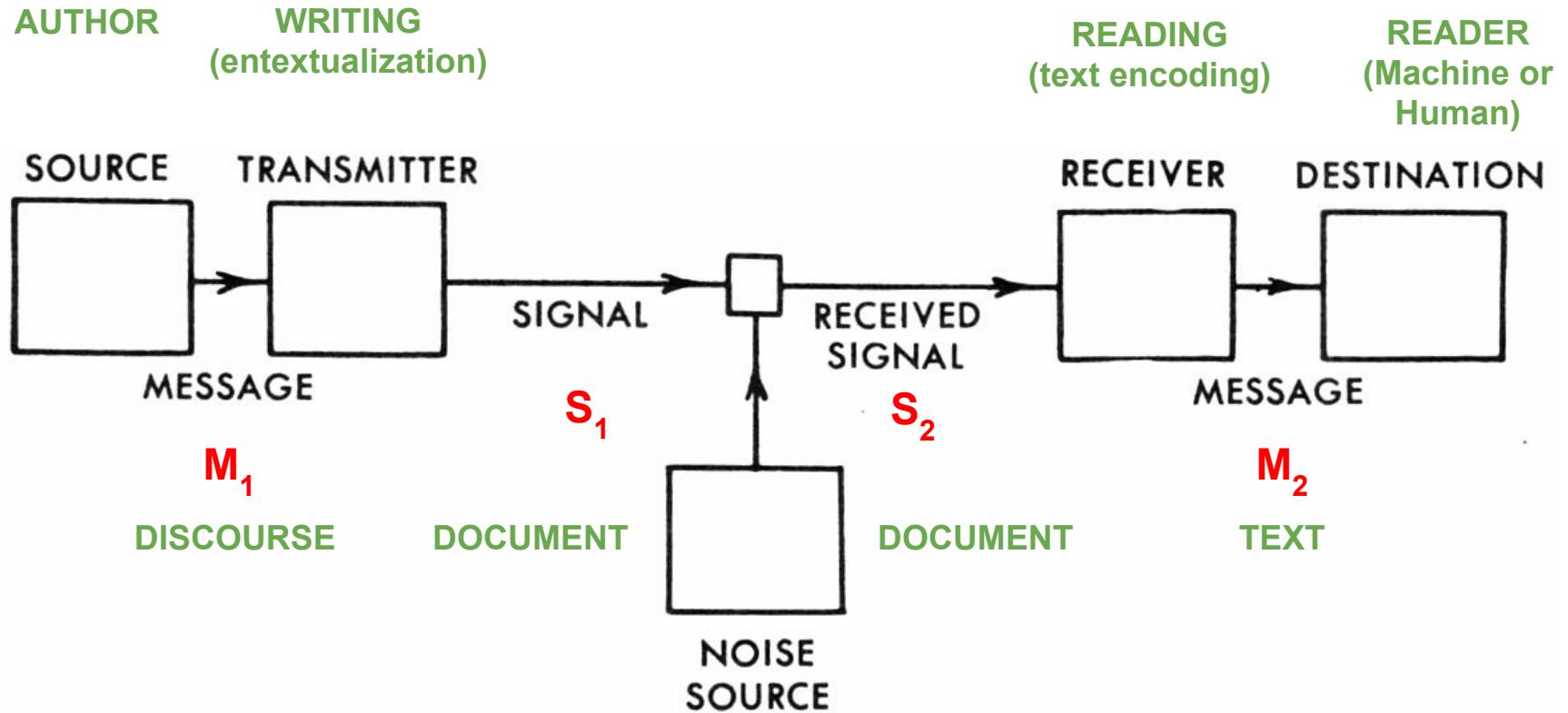
Information sources





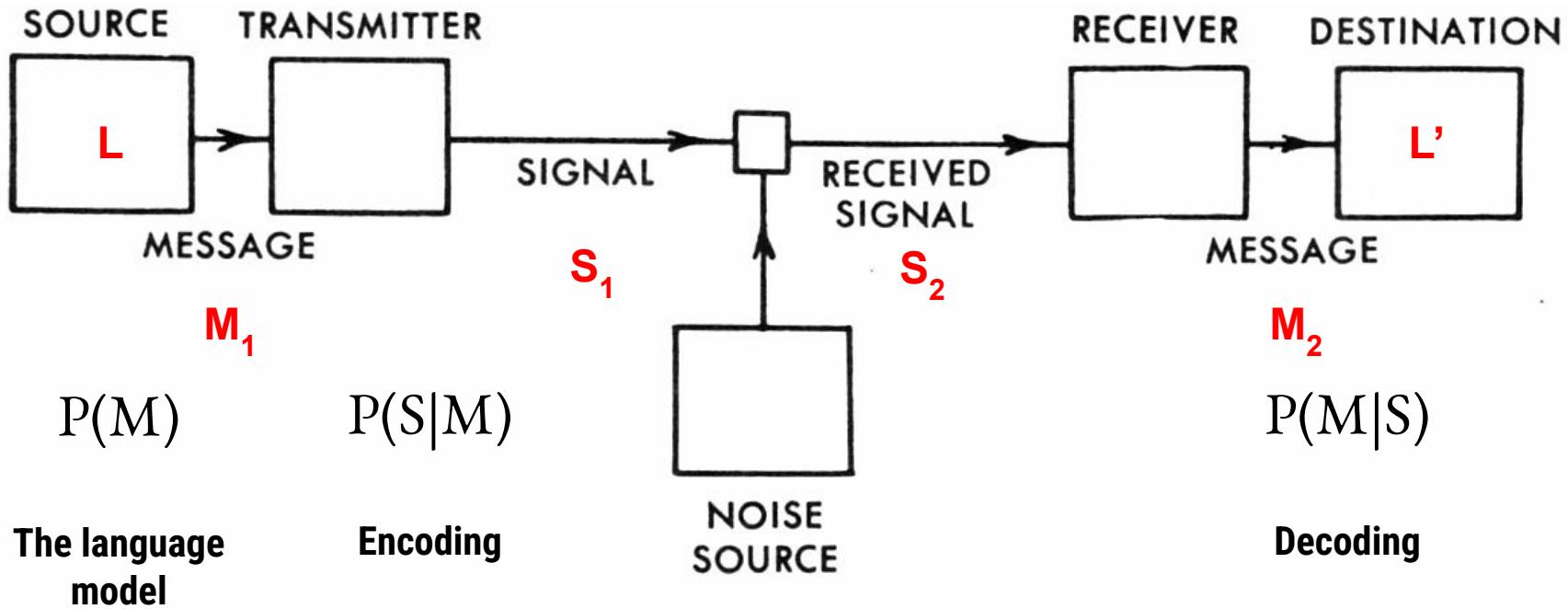
Messages ...

Figure 1 *Diagram of a Turing Machine*



The model applied to text analytics

Hermeneutics as Bayesian Decoding



Transmitter and Receiver must share a CODEC

Source and Destination must share a Language (L)
(and *Verstehen*) -- must be inferred with text

Quipu as Inca communicating with us

accounting?
calendars?
stories?

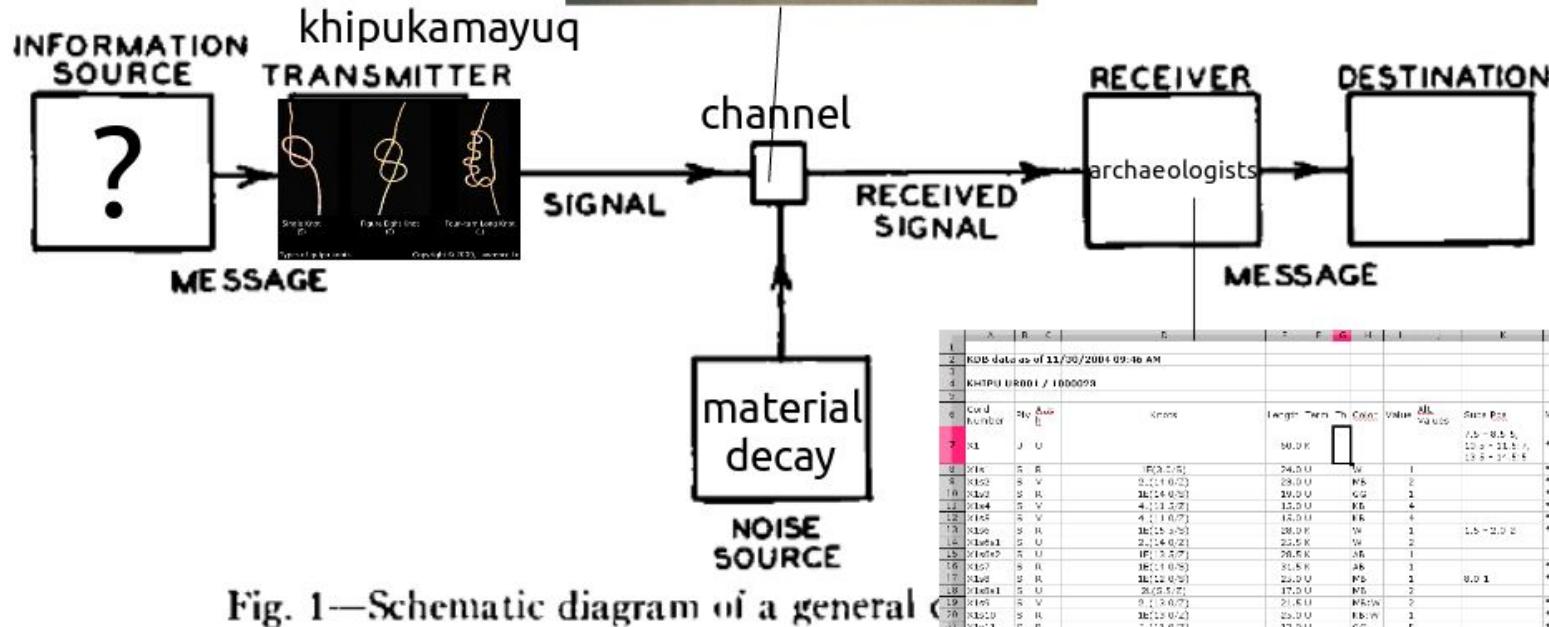


Fig. 1—Schematic diagram of a general communication system.

<https://www.pawfal.org/dave/blog/wp-content/uploads/2015/11/slide.png>

The Key Idea: Selection

In this model, the key idea is **the concept of selection**

The **choice** of a symbol from the set (aka sampling)

Each symbol in a message represents a **decision**

This is the importance of the **binary digit** or **bit** (coined by Tukey)

All character sets can be represented in **binary** form

One reason for for binary is to capture an elementary quantity --
the **decision**

A bit is not a character per se but the measure of information

the amount of choice in a message

The Information Source

In this model, the information source is a **stochastic process**

This means that the message being produced is governed to a greater or lesser extent by **chance**

Symbols and sequences are associated with **probabilities**

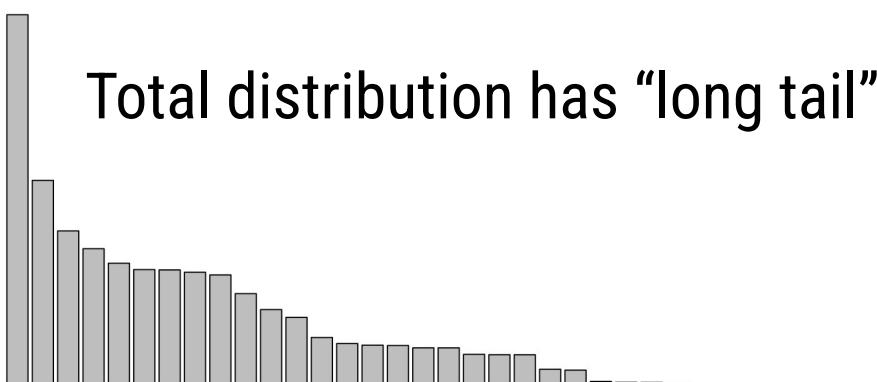
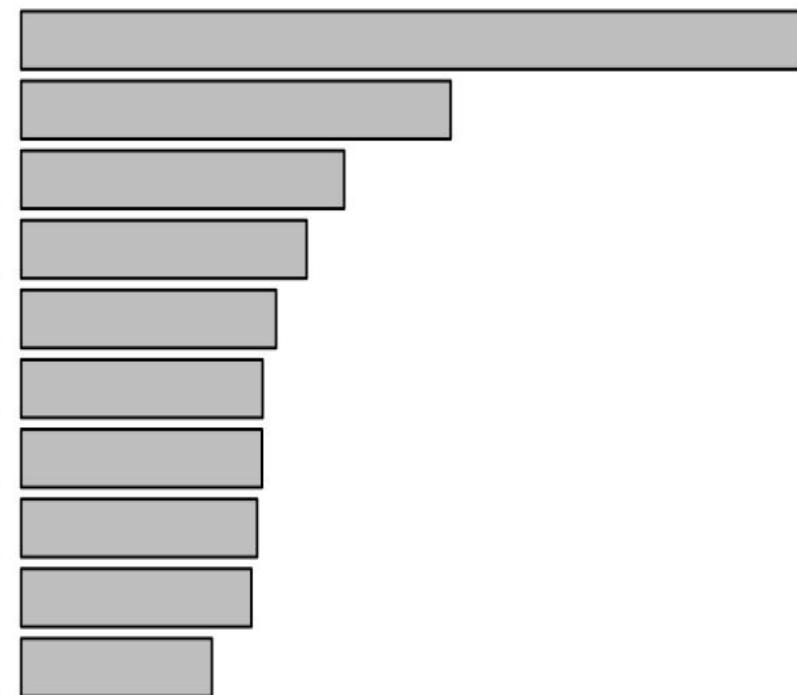
The probability function associated with the symbol set is called a language model

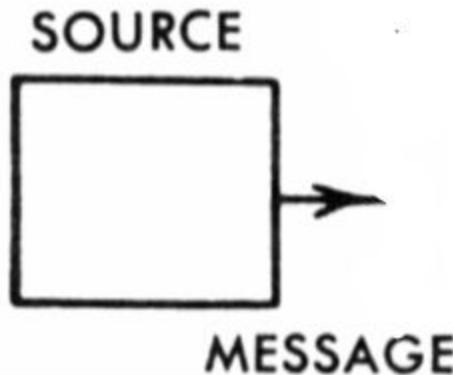
In principle, this model can be **inferred** from the messages received from the source

The **ergodic assumption** is that all messages from a source are generated by the same model

Top 10 characters in *Moby Dick*

	moby.chars.v	Freq	prob
1		207768	0.1821538850
2	e	114624	0.1004928907
3	t	86204	0.0755765734
4	a	76192	0.0667988757
5	o	68040	0.0596518729
6	n	64448	0.0565027029
7	i	64274	0.0563501540
8	s	62977	0.0552130512
9	h	61456	0.0538795635
10	r	50915	0.0446380822





Source

langue

Symbol

Types

θ (theta)

Message

parole

Sequence

Tokens

D (data)

← Language
Model

Langue and Parole Redux

Shannon here reproduces the fundamental division of language

LANGUAGE MODEL: ***LANGUE*** ⇒ finite system

MESSAGE: ***PAROLE*** ⇒ infinite sequence

The inferred language model is an attempt to describe the grammar of the language possessed by the information source

But the approach is statistical, not mechanical (rules-based)

How much information?

The quantity information contained in a message is related to **how many possible messages we can send** in a given act of communication

This is a function of the **character set size** and the **length of the message**, i.e. the numbers of **types and tokens**

$$a^{\text{length}} = M$$

a number of characters (**n types**) in the alphabet

length length of the message (**m tokens**)

M number of possible messages

	So, if $a = 2$	
length	M	
1	2	
2	4	
3	8	
4	16	

In this view, language is like a slot machine



The number of items per wheel raised to the number of wheels

if

$$a^{\textit{length}} = M$$

then

$$\log_a M = \textit{length}$$

Since exponents can be expressed as logarithms (“logs”), and logs are easy to work with, it is convenient to express the amount of information as a log

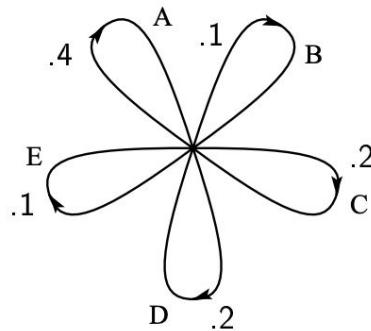
In a message of 80 characters with an alphabet of 26 we have this much information:

$$26^{80} = 1.57713125 \times 10^{113}$$

It turns out that in English, and in all human languages, the actual number of possible sentences is **much smaller** than this

We do not live in the Library of Babel

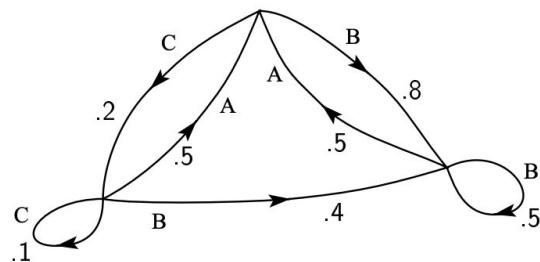
Language models **limit** the number of possible selections



A	B	C	D	E
.4	.1	.2	.2	.1

Unigram language model

AAAACDCBDCEAADADACEADEADCABEDADDCECAAAAD.

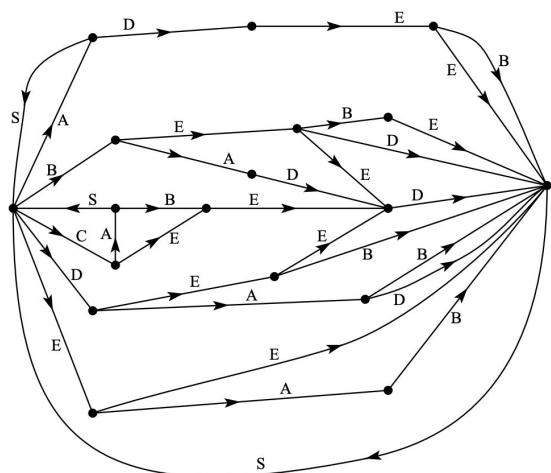


$p_i(j)$		j		
		A	B	C
i	A	0	$\frac{4}{5}$	$\frac{1}{5}$
	B	$\frac{1}{2}$	$\frac{1}{2}$	0
	C	$\frac{1}{2}$	$\frac{2}{5}$	$\frac{1}{10}$

Bigram language model

a set of transition probabilities $p_i(j)$

ABBBABABABABABABBBAB
BABBBBABBABACBBBABA



Shannon describes language models as **finite state machines** that might be used to model the utterance of messages

The machines are discrete, ergodic, and **Markovian**

Language Models

OH MY GOD, THE
VOLCANO IS ERU-

ERUDITE?



What is an LM?

A model of the probability distribution over **token sequences**, i.e. a model that predicts tokens in a sequence

$$P(\text{“Today is Wednesday”}) \approx 0.001$$

$$P(\text{“Today Wednesday is”}) \approx 0.000000000001$$

$$P(\text{“The eigenvalue is positive”}) \approx 0.00001$$

It can be both descriptive and prescriptive (generative)

Descriptive – derived from a corpus through **frequencies**

Prescriptive – can be used to generate text

How does LM help in NLP?

Machine translation

$$P(\text{high} > \text{winds} > \text{tonite}) > P(\text{large} > \text{winds} > \text{tonite})$$

Spell Correction

The office is about fifteen **minuets** from my house

$$P(\text{about} > \text{fifteen} > \text{minutes} > \text{from}) >$$

$$P(\text{about} > \text{fifteen} > \text{minuets} > \text{from})$$

Speech Recognition

$$P(\text{I saw a van}) > P(\text{eyes awe of an})$$

Much more . . .

How to do we calculate these probabilities?

The probability of a sentence, for example a sequence of 5 words, can be represented this way:

I drove to the store → $w_1 w_2 w_3 w_4 w_5$

$$P(w_1 w_2 w_3 w_4 w_5)$$

Probability of a next word can be modeled as the **conditional probability** of the a word given the words that preceded it (history)

$$P(\text{store} \mid \text{I drove to the}) = P(\text{I drove to the store}) / P(\text{I drove to the})$$

$$P(w_5 \mid w_1 w_2 w_3 w_4) = P(w_1 w_2 w_3 w_4 w_5) / P(w_1 w_2 w_3 w_4)$$

How do we calculate the these sequences?

We use the **chain rule** from probability

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

$$P(W_1^5) =$$

$$\begin{aligned} & P(w_1) \\ & \times P(w_2 | w_1) \\ & \times P(w_3 | w_1 w_2) \\ & \times P(w_4 | w_1 w_2 w_3) \\ & \times P(w_5 | w_1 w_2 w_3 w_4) \end{aligned}$$

What is the **problem** with this approach?

Too hard to compute for long sentences

Especially considering the full history
of a word in a text

**What assumption can we make
to simplify our computation of probabilities?**

The **Markov assumption**

We only need to consider **some small number of previous words (events)**

(0, 1, 2, 3, ...)

to compute the probability

Markov Assumption

$P(\text{store} \mid \text{I drove to the})$

$\approx P(\text{store} \mid \text{the})$ in a **bigram** model

$\approx P(\text{store} \mid \text{to the})$ in a **trigram** model

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-1}) \quad \text{Bigram}$$

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-N+1}^{n-1}) \quad \text{N-Gram}$$

$$P(w_1^n) \approx \prod_{k=1}^n P(w_k | w_{k-1}) \quad \text{Probability of a sentence as product of bigrams}$$

How do we **discover** these values?

How to **learn** the value of $P(w_k | w_{k-1})$?

We estimate using methods like

Maximum Likelihood Estimation (MLE)

In effect, we compute **relative frequencies** on a **representative set** of sentences and use these as estimates of the real distribution

The law of conditional probability says that

$$P(B | A) = P(A, B) / P(A)$$

So, we count how many times B follows A and divide by
the total number of times B appears

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

How do we create a bigram language model?

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

$C(A)$

raw counts

A↓ B→	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

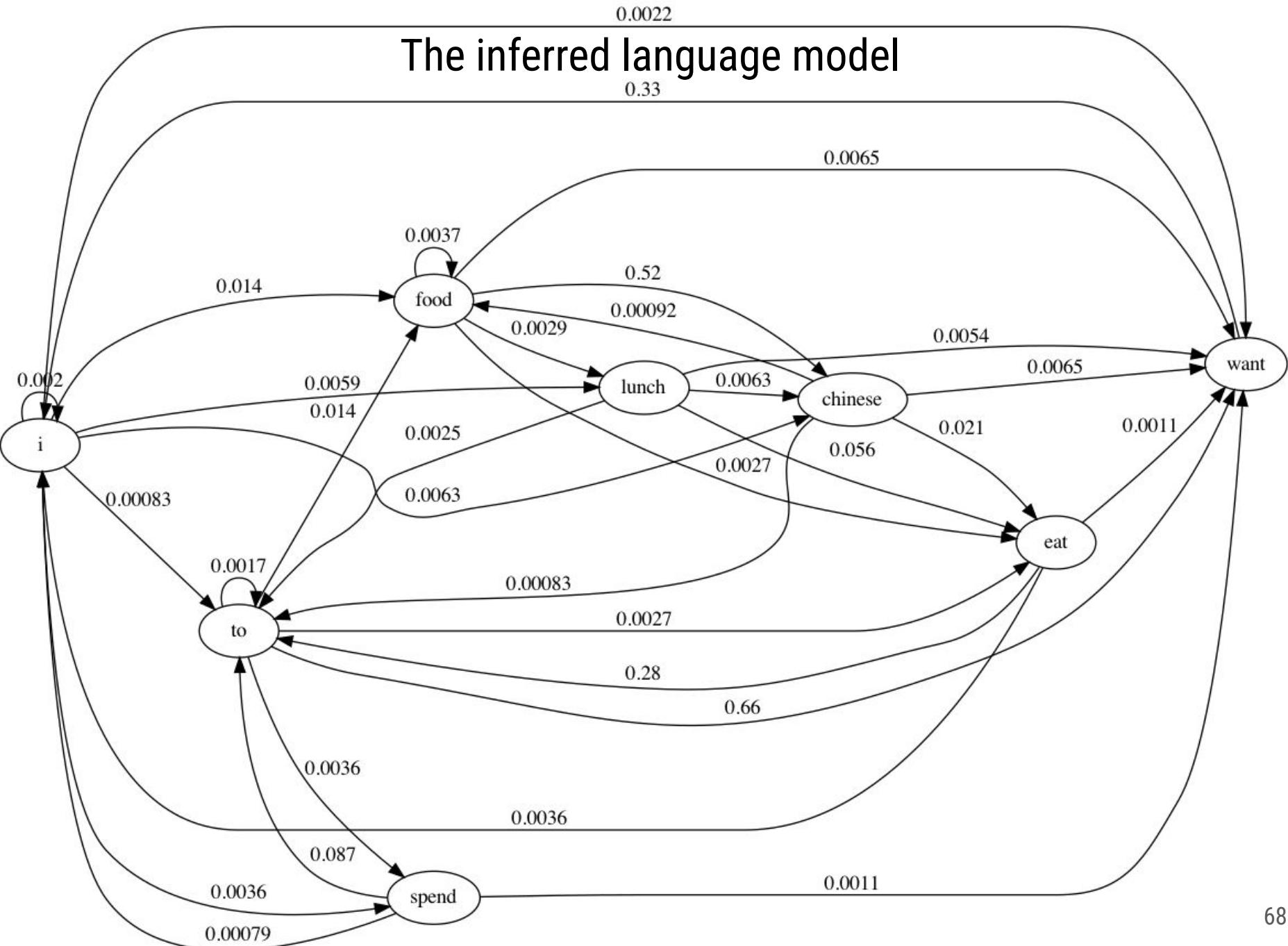
$C(A,B)$

A↓ B→	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

$$P(B|A) = \frac{C(A,B)}{C(A)}$$

Selected bigram counts from the **Berkeley Restaurant Project** data. The top table are raw counts, the bottom normalized.

The inferred language model



How do we **use** the model?

$$P(i|s) = 0.25$$

$$P(\text{english}|\text{want}) = 0.0011$$

$$P(\text{food}|\text{english}) = 0.5$$

$$P(</s>| \text{food}) = 0.68$$

Now we can compute the probability of sentences like *I want English food* or *I want Chinese food* by simply multiplying the appropriate bigram probabilities together, as follows:

$$\begin{aligned} & P(<\text{s}> \text{ i want english food } </\text{s}>) \\ &= P(i|s)P(\text{want}|i)P(\text{english}|\text{want}) \\ & \quad P(\text{food}|\text{english})P(</\text{s}>|\text{food}) \\ &= .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\ &= .000031 \end{aligned}$$

We can **estimate the probability of a sentence** by multiplying the bigram components that make up the sentence.

You can also generate text based on the model ...

sentences generated
by the model:

- 01 to eat lunch i want to eat lunch i want -23
- 02 want to spend i want to eat lunch i want -23
- 03 eat lunch i want to eat lunch i want to -24
- 04 i want to eat lunch i want to eat chinese -19
- 05 i want to eat chinese food to spend i want -22
- 06 i want to spend i want chinese food to eat -26
- 07 to spend i want to spend i want to eat -25
- 08 want to eat chinese food to eat lunch i want -23
- 09 to eat to spend i want to eat lunch food -32
- 10 to chinese food i want to eat lunch i want -26
- 11 chinese food food to eat food i want to eat -32
- 12 i want to eat to eat lunch food chinese food -32
- 13 i want to eat lunch i want to spend to -23
- 14 lunch food to eat lunch i want to eat lunch -33
- 15 to eat lunch i want to eat lunch food i -27
- 16 eat chinese food i want to spend to eat chinese -27
- 17 i want to spend to spend i want to to -28
- 18 to spend i want to eat chinese food i want -22
- 19 i want to eat chinese food to spend i want -22
- 20 food i want to spend to eat chinese food i -27

whats the next word
after the given
phrase, so prob of
word after 'i' then
prob of word after 'i
want' then prob of
word after 'i want
this' etc.

How to we **evaluate** our model?

Two General Types of Evaluation

Extrinsic measures

Performance in a given **task**, e.g. voice recognition or filtering
Pragmatic – "the proof of the pudding is in the tasting"

Intrinsic measures

Performance by in **internal metric** – e.g. cross-entropy or perplexity

Nota bene:

Extrinsic is **ideal but hard** to do!
The two **don't always coincide** (see topic models)
Different models perform better at different tasks

Cross-Entropy and Perplexity as Intrinsic Measures

Perplexity is essentially the exponentized **cross entropy** of our model relative to **maximum entropy**

In general, **the less entropy** (therefore perplexity) a model has, **the better it is** . . .

Intuitively, less entropy means better grasp of **structure** of the source (i.e. the mind) . . .

>>> We will look at entropy and perplexity in the next lecture

Problems with Generalizing an LM

Mismatched genres

All LMs are created from corpora that are **embedded in language communities**

Thus texts vary by genre and more generally **culture, history, etc.**

We should not expect a model to be universal!

* Unless it is massive ...

We can take **advantage** of this too:

If LMs shed light on culture, then by controlling our source of text production, we have a better chance of **discovering important cultural facts**

Problems with Generalizing an LM

How do we handle **new words**?

These are words that do not appear in the training corpus and so have a **0 probability**

This will zero out the probability of many sentences since 0 in a group product yields a result of zero

How do we handle **new combinations** of existing words?

There are various ways to address these problems

Approaches to Incompleteness

In general, we approach these problems through **smoothing** or **discounting**

We take some of the probability of high frequency words and **redistribute it** to unknown words and zero-probability ngrams

The standard methods for addressing incompleteness are

Smoothing (e.g. Laplace)

Discounting

Interpolation

Back-off