

# Natural Language Processing

## Lecture 05

Dirk Hovy

[dirk.hovy@unibocconi.it](mailto:dirk.hovy@unibocconi.it)

 @dirk\_hovy

# Today's Goals

- Know when (and when not) to use **Regular expressions**
- Learn about forms of **TF-IDF** and its possibilities
- Understand how information can be quantified with **entropy**
- Understand **PMI** and see why it can help find collocations

# Flexible Matches: Regular Expressions

# The promise...

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.

OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!



BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!



IT'S HOPELESS!

EVERYBODY STAND BACK.



I KNOW REGULAR EXPRESSIONS.



# Is it an (Email) Address?

- notMyFault@webmail.com ✓
- smithie123@gmx ✗
- Free stuff@unibocconi.it ✗
- mark\_my\_words@hotmail;com ✗
- truthOrDare@webmail.in ✓
- look@me@twitter.com ✗
- how2GetAnts@aol.dfdsfgfdsgfd ✗

NAME

@

DOMAIN

.

CODE

# Simple Matching

sequence	Matches
e	any single occurrence of e
at	<u>a</u> t, r <u>a</u> t, m <u>a</u> t, s <u>a</u> t, c <u>a</u> t, <u>a</u> ttack, <u>a</u> ttention, l <u>a</u> ter

# Quantifiers

	Means	Example	Matches
*	0 or more	cooo*l	cool, coool
+	1 or more	hello+	hello, helloo, hellooooooooo
?	0 or 1	fr?og	fog, frog

# Special Characters

	Means	Example	Matches
.	any single character	.e1	ee1, Ne1, ge1
\n	newline character (line break)	\n+	One or more line breaks
\t	a tab stop	\t+	One or more tabs
\d	a single digit [0-9]	B\d	B0, B1, ..., B9
\D	a non-digit	\D.t	' t, But, eat
\w	any alphanumeric character	\w\w\w	Top, WOO, ash, bee, ...
\W	non-alphanumeric character		
\s	a whitespace character		
\S	a non-whitespace character		
\	"Escapes" special characters to match them	.+ \.com	abc.com, united.com
^	the beginning of the input string	^...	First word in line
\$	the end of the input string	^\n\$	Empty line



# Classes

	Means	Example	Matches
<b>[abc]</b>	Match any of a, b, c	<code>[bcrms]at</code>	<code>bat, cat, rat, mat, sat</code>
<b>[^abc]</b>	Match anything BUT a, b, c	<code>te[^ ]+s</code>	<code>tens, tests, teens, texts, terrors...</code>
<b>[a-z]</b>	Match any lowercase character	<code>[a-z][a-z]t</code>	<code>act, ant, not, ... wit</code>
<b>[A-Z]</b>	Match any uppercase character	<code>[A-Z]...</code>	<code>Ahab, Brit, In a, ..., York</code>
<b>[0-9]</b>	Match any digit	<code>DIN A[0-9]</code>	<code>DIN A0, DIN A1, ..., DIN A9</code>

# Groups

	Means	Example	Matches
<b>(abc)</b>	Match abc	<code>.(ar).</code>	<code>hard, cart, fare, ..</code>
<b>(ab c)</b>	Match ab OR c	<code>(ab c)ate</code>	<code>abate, cate</code>

# Matching Addresses

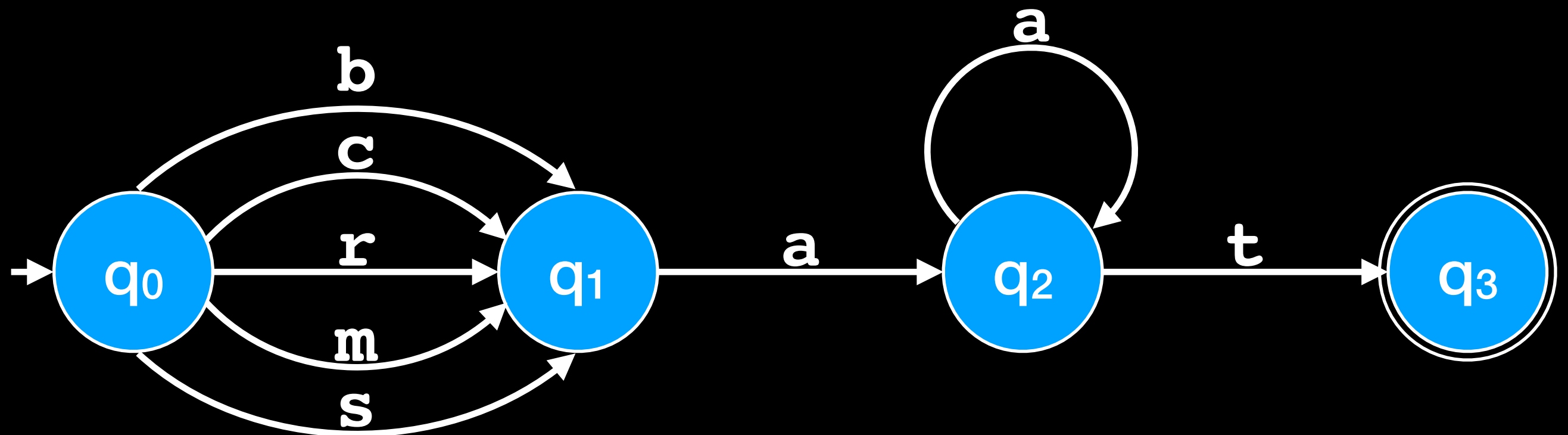


$^{\wedge}[\text{A-Za-z0-9\_}\backslash.\text{-}]^{\wedge}+@[\text{A-Za-z0-9\_}\backslash.\text{-}]^{\wedge}+\backslash.[\text{A-Za-z0-9\_}][\text{A-Za-z0-9\_}]^{\wedge}+ \$$

# A (W|w)ord of [Ww]arning



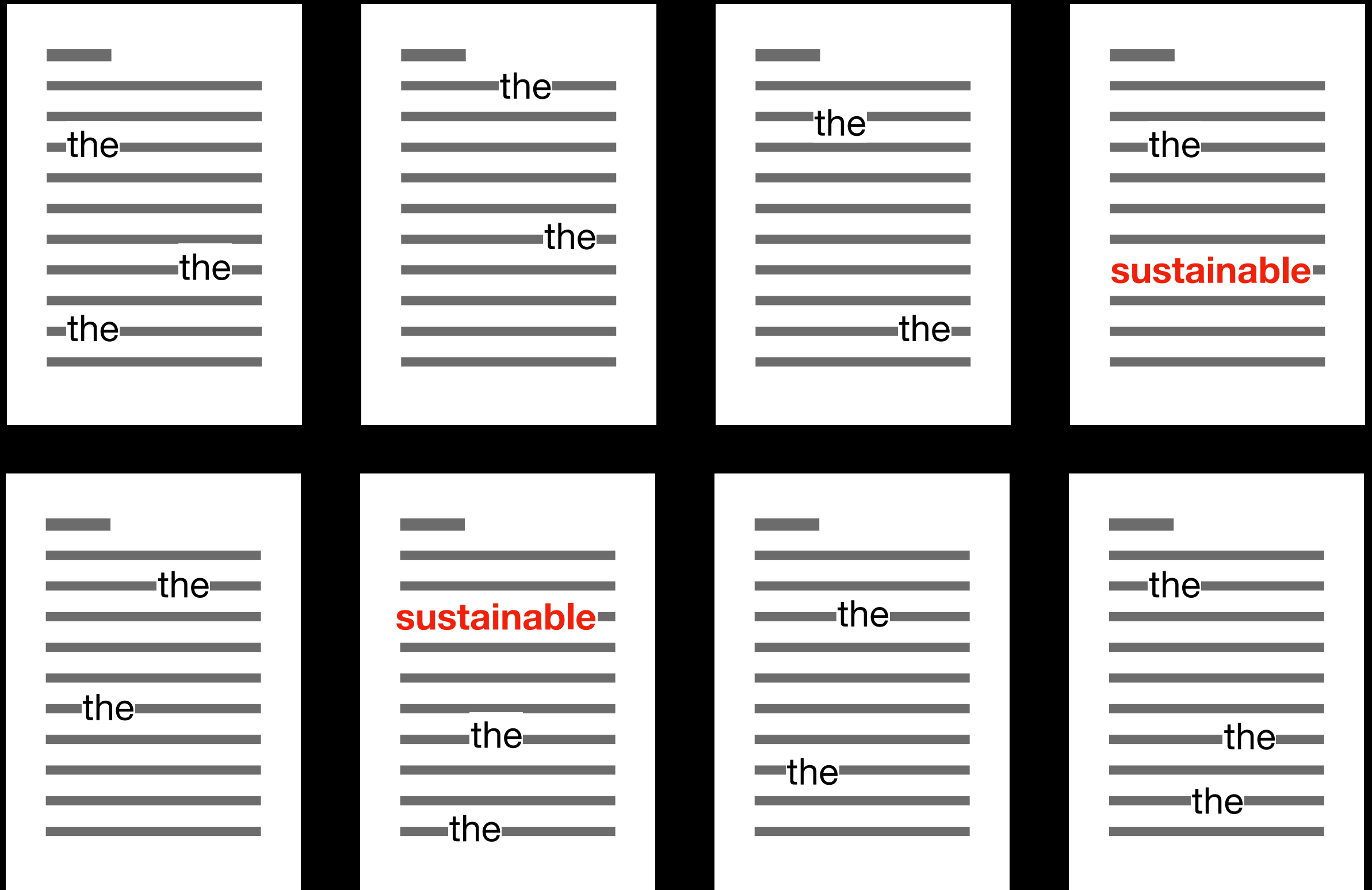
# RegEx as Automata



**[ bcrms ] a+t**

# Finding Important Words: TF-IDF

# Some Words are Just More Interesting...



# Karen Spärck Jones

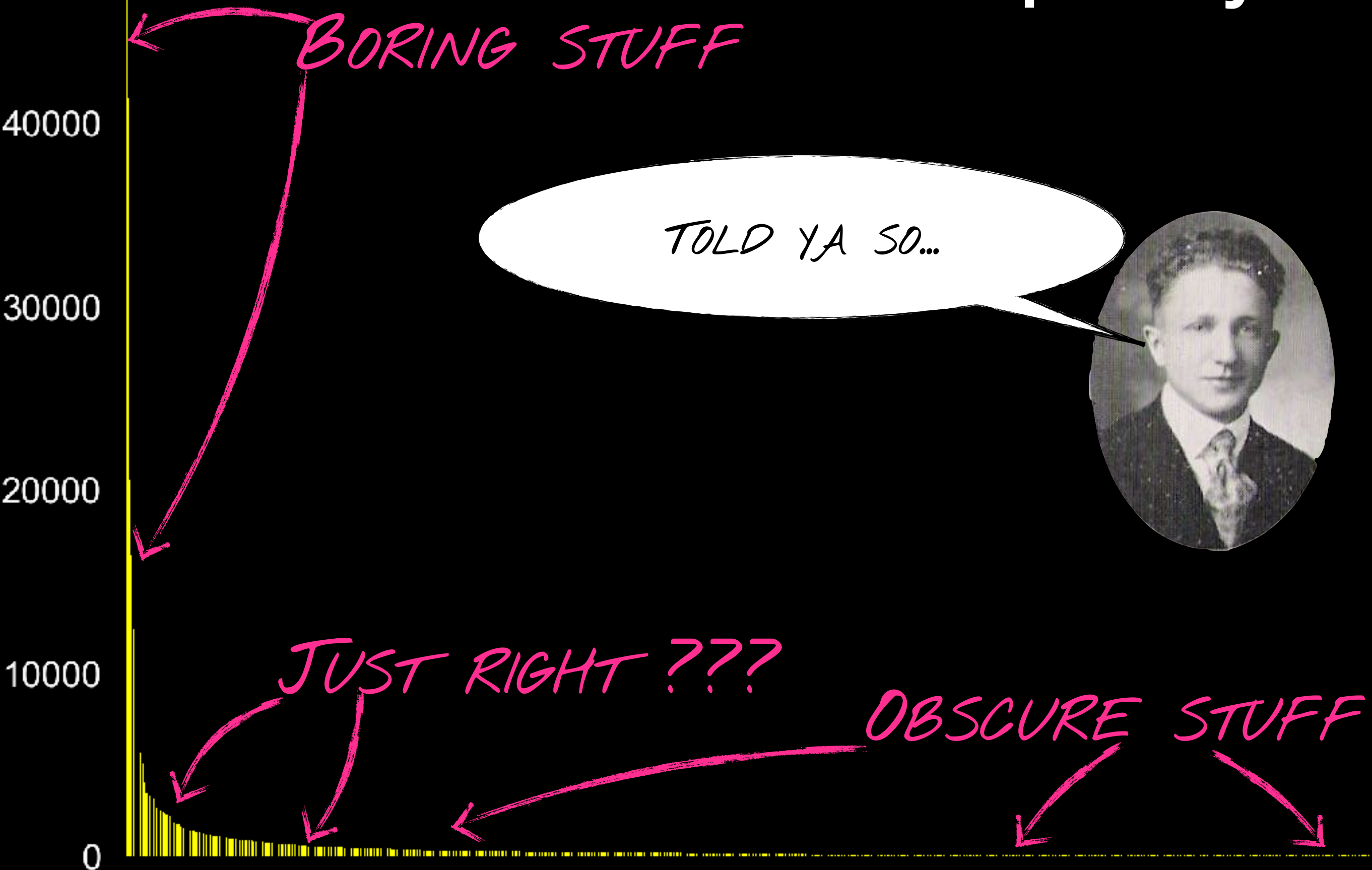
1935–2007

- Became a teacher before starting CS career at Cambridge
- Laid the foundation for modern NLP, Google Search, text classification
- Campaigned for more women in CS
- Namesake of prestigious CS prize





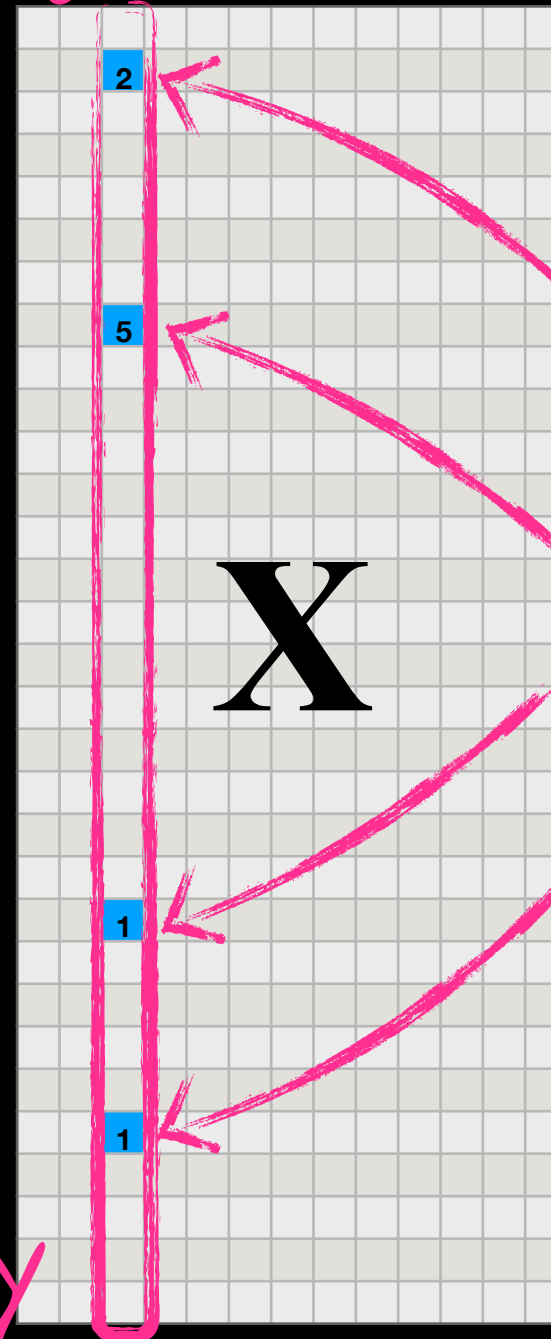
# Problems with Term Frequency



# Document and Term Frequency

FEATURE

$$IDF = \log \frac{N}{df(w)}$$



DOCUMENT  
FREQUENCY  
(COUNT): 4

TERM FREQUENCY  
(SUM): 9 TF

# Putting it Together

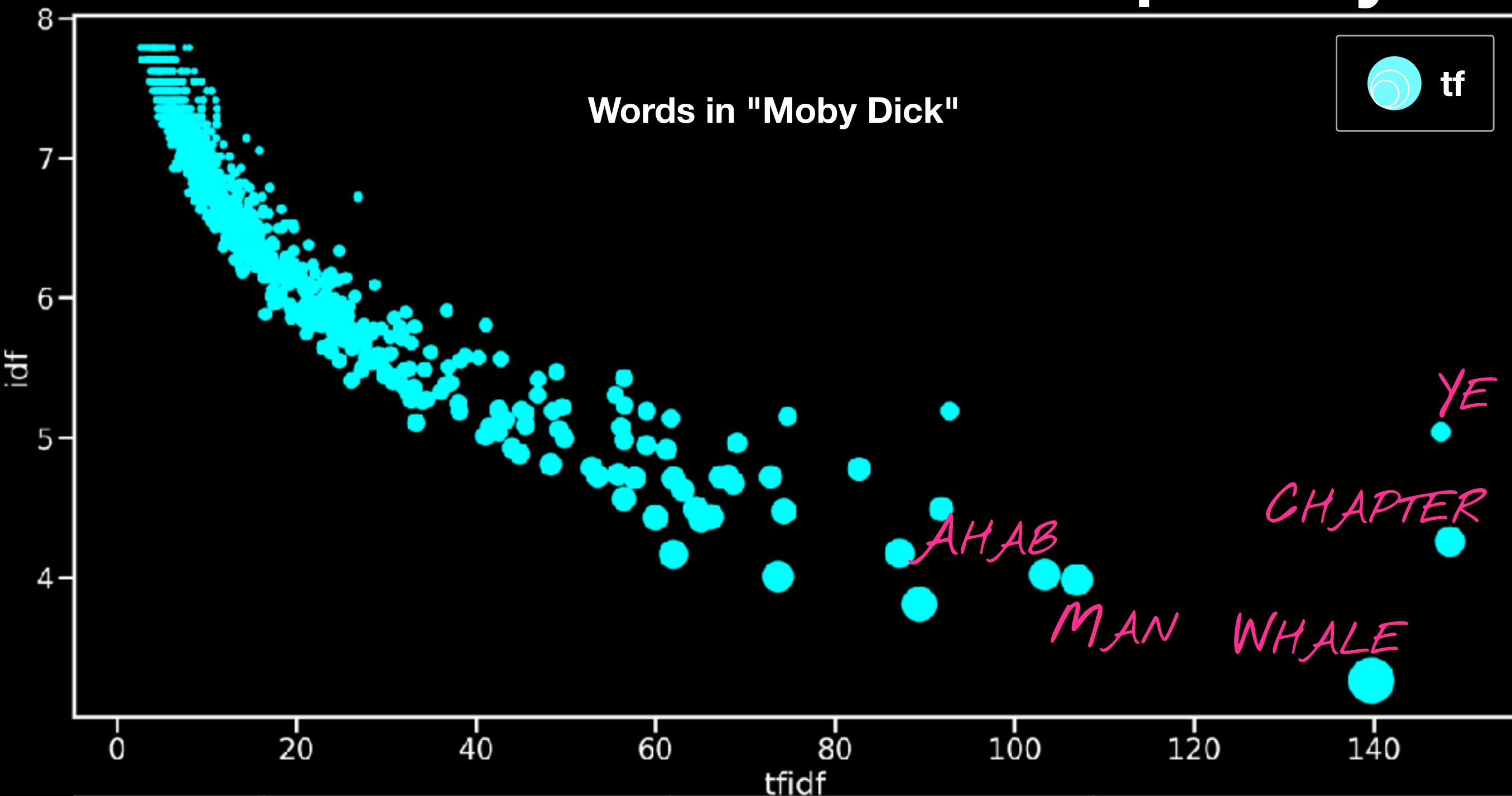
HOW OFTEN WE  
SAW THE WORD

$$TFIDF(w) = TF(w) \cdot \log \frac{N}{df(w)}$$

ADJUSTED BY  
HOW MANY  
DOCUMENTS

0

# Document and Term Frequency



word	tf	idf	tfidf
ye	467	4.257380	148.497079
chapter	171	5.039475	147.504638
whale	1150	3.262357	139.755743
man	525	3.982412	106.932953
ahab	511	4.019453	103.357774

# The Probability of Words: Entropy

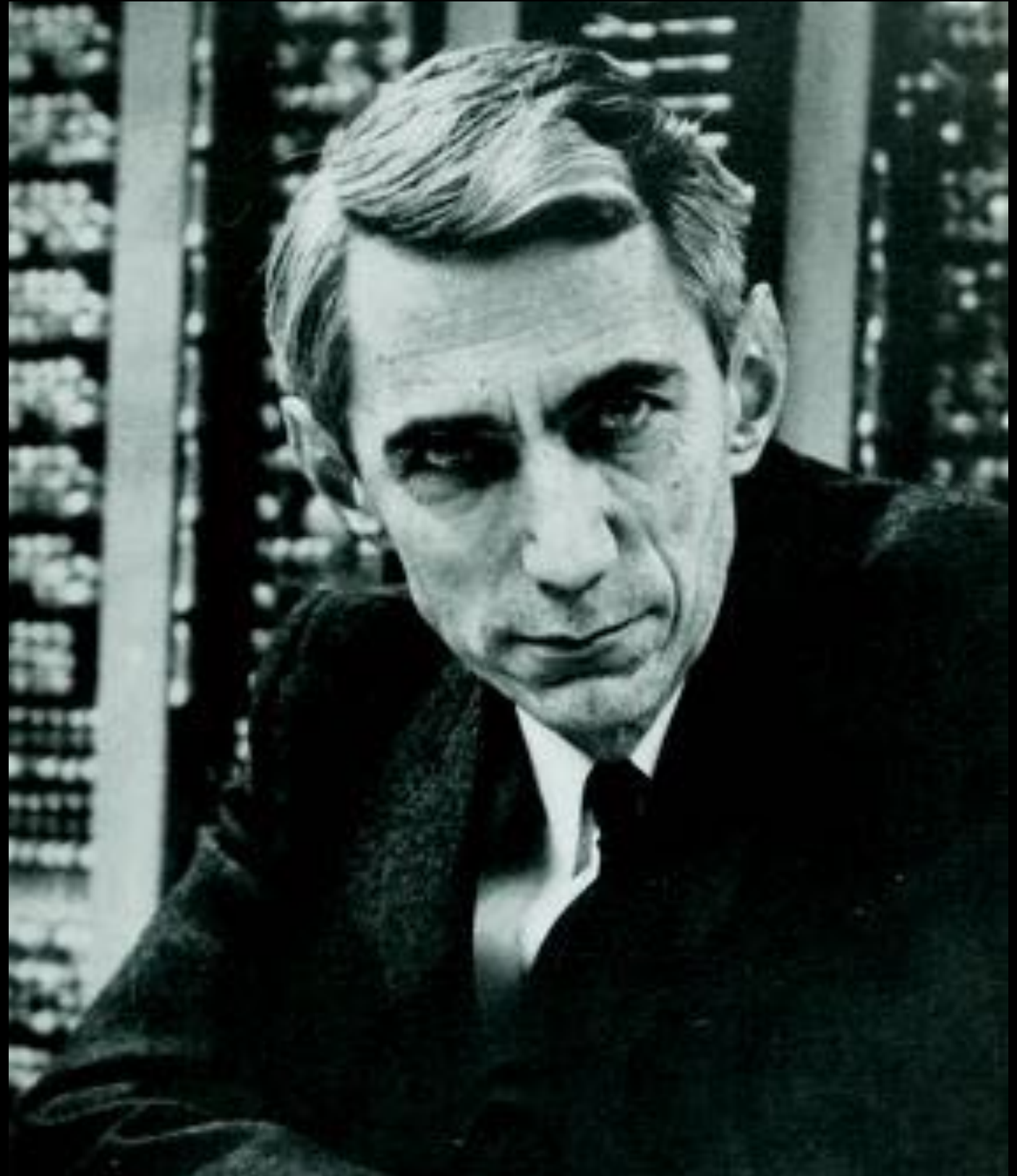
# What's More Likely?

We finish each others' SENTENCES  
LEFTOVERS  
MOVIES

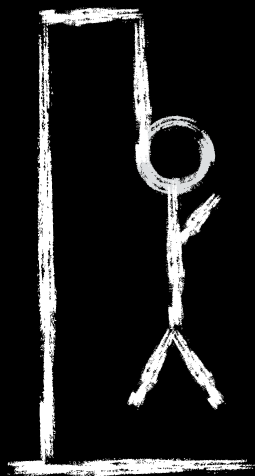
# Claude Shannon

1916-2001

- His master's thesis founded a new field: digital circuits
- Invented entropy to quantify language – and a flame-throwing trumpet
- Enabled NLP, cryptography, modern computers...
- Died of Alzheimer's, oblivious to his own inventions' impact







# Shannon Game



WHAT'S THE  
NEXT WORD?

The house  
A friend  
Then dog  
If car  
When water  
My **hovercraft**  
He pants  
You God  
I word  
...



with

water

?

of

eels

to

air

!

How Much  
...  
MORE, CLAUDE?



# Entropy



VERY SURPRISING

entropy

$$H(X) = - \sum_x p(x) \log p(x)$$

Information

$p(x)$

TOTALLY  
PREDICTABLE

TOTALLY  
PREDICTABLE

# Probability of a Word



*"It must be recognized that the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term."*

Noam Chomsky

- Choose a word  $w$
- Open a page at random and point at a word:  
Is it  $w$ ?

HOW OFTEN WE  
HAVE SEEN  $w$

$$P(w) = \frac{c(w)}{\sum_{v \in V} c(v)}$$

...ALL WORDS

# Entropy in Use

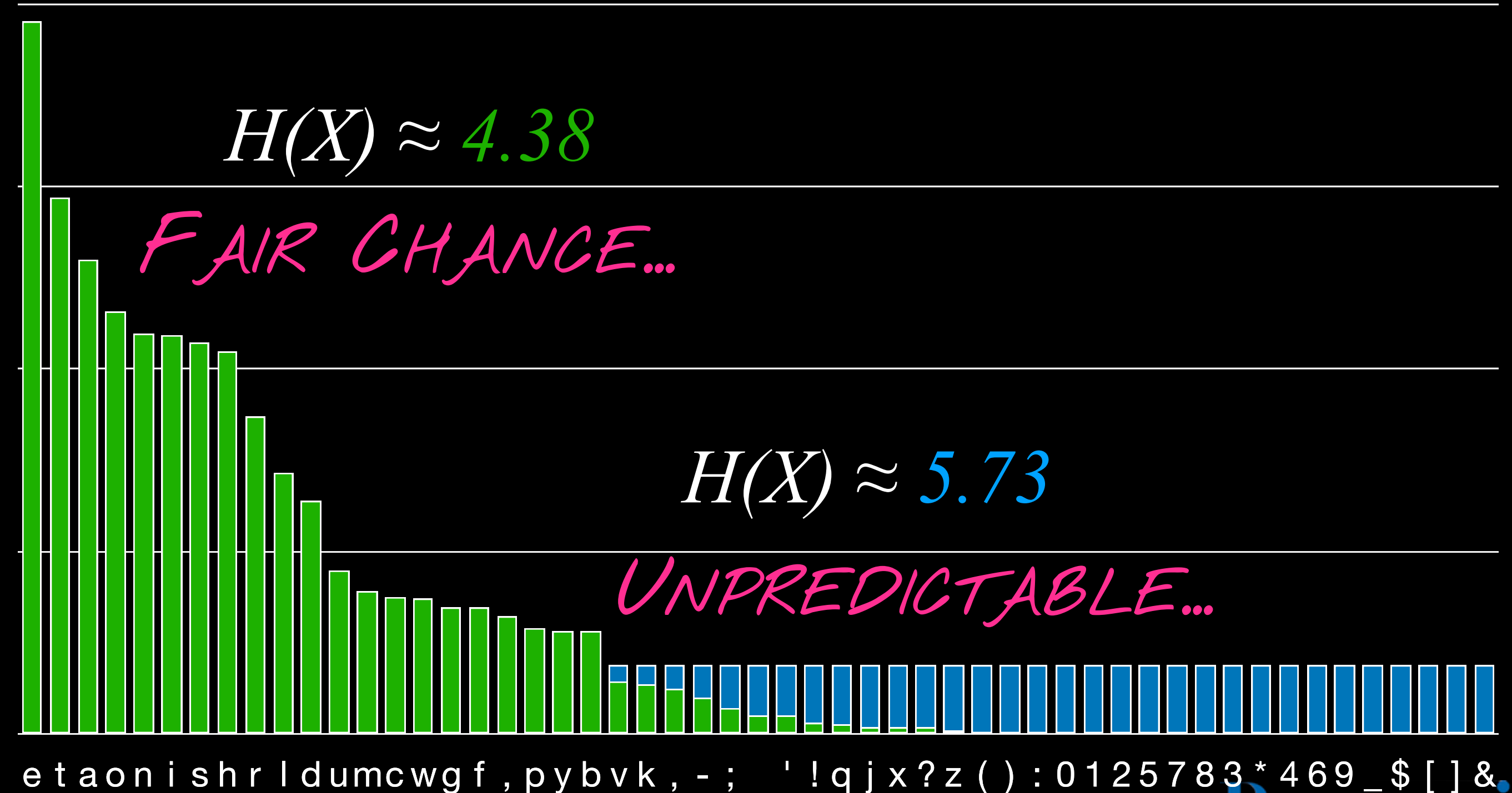
WHAT'S THE NEXT LETTER?

$$H(X) \approx 4.38$$

FAIR CHANCE...

$$H(X) \approx 5.73$$

UNPREDICTABLE...



# Telling Neighbors: Pointwise Mutual Information

# Some are not like the Others



# Mutual Informativity

*HOW WELL CAN WE GUESS THE BLANK?*

**social \_\_\_\_\_**

**and \_\_\_\_\_**

**\_\_\_\_\_ media**

**\_\_\_\_\_ the**

# Pointwise Mutual Information

*CHANCE OF SEEING THEM TOGETHER*

$$PMI(x, y) = \log \frac{P(x, y)}{P(x)P(y)}$$

*...SEEING EITHER*

x	y	c(x)	c(y)	c(xy)	P(x)	P(y)	P(x, y)	PMI(x; y)
moby	dick	83	83	82	0.0003	0.0003	0.0003	3.48
captain	ahab	327	511	61	0.0013	0.0020	0.0002	1.97
white	whale	280	1150	106	0.0011	0.0045	0.0004	1.93
under	the	119	14175	45	0.0005	0.0553	0.0002	0.83
is	a	1690	4636	110	0.0066	0.0181	0.0004	0.56

$$c(X) = 256,149$$

$$c(XY) = 256,148$$

# Wrapping up



# Take home points

- **Regular expressions** allow us to search for flexible patterns
- **Entropy** allows us to quantify how surprising/predictable something is
- **TF-IDF** finds "bursty" words: medium frequency overall, but concentrated in few documents
- **PMI** tells us how likely one word is to occur with/without another to find **collocations**