Computer Science

Robert Sedgewick | Kevin Wayne

Shannon approximated the statistical structure of a piece of text using a simple mathematical model known as a Markov model. A Markov model of order 0 predicts that each letter in the alphabet occurs with a fixed probability. We can fit a Markov model of order 0 to a specific piece of text by counting the number of occurrences of each letter in that text, and using these frequencies as probabilities. For example, if the input text is "gagggagaggcgagaaa", the Markov model of order 0 predicts that each letter is 'a' with probability 7/17, 'c' with probability 1/17, and 'g' with probability 9/17 because these are the fraction of times each letter occurs. The following sequence of characters is a typical example generated from this model. A Markov model of order 0 assumes that each letter is chosen independently. This independence does not coincide with statistical properties of English text because there a high correlation among successive characters in a word or sentence. For example, 'w is more likely to be followed with 'e' than with 'u', while 'q' is more likely to be followed

http://princeton.edu/~cos126

Assignment 7 Tips and Tricks

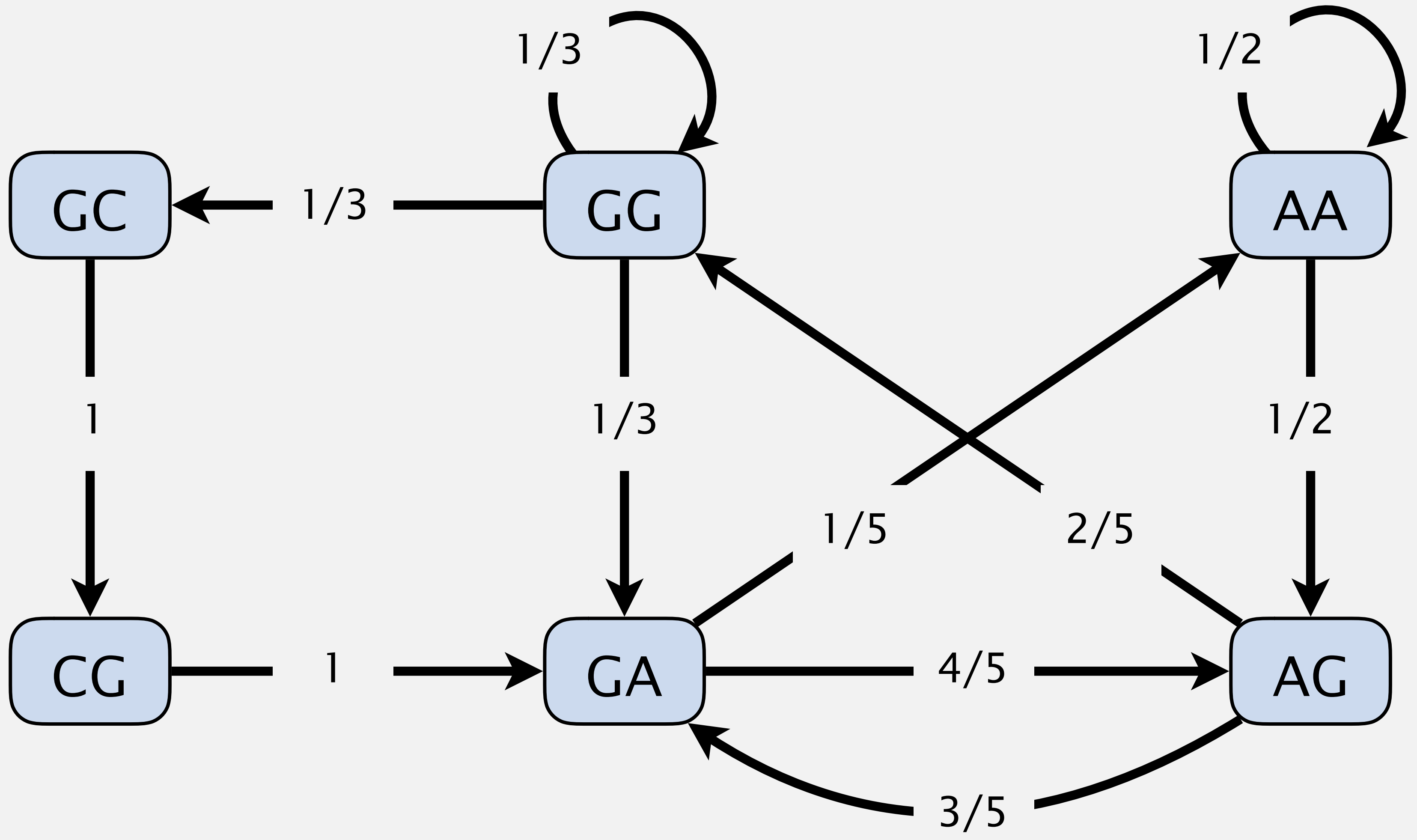
- *Markov chains*
- *overview of assignment*
- *Markov model data type*
- *text generator client*

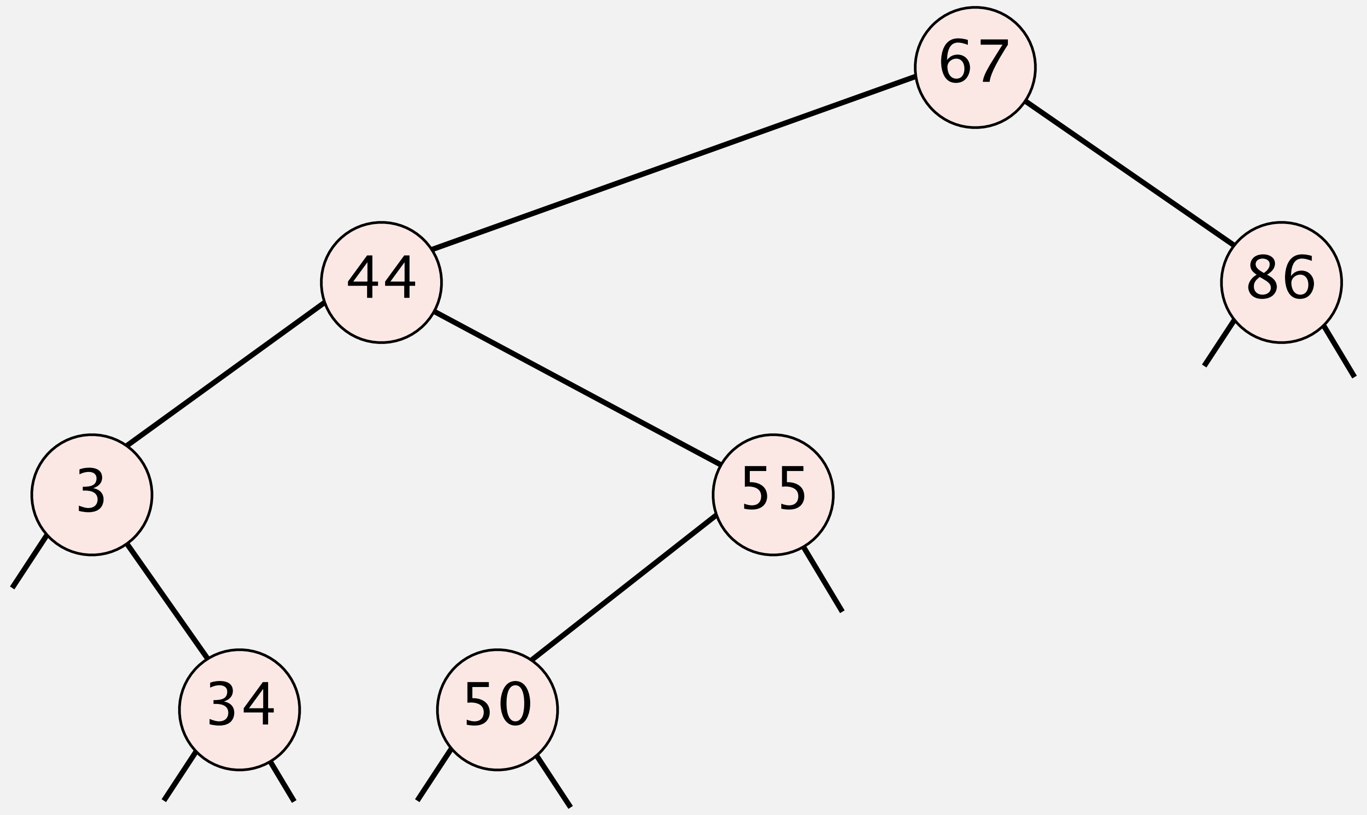
Kevin Wayne

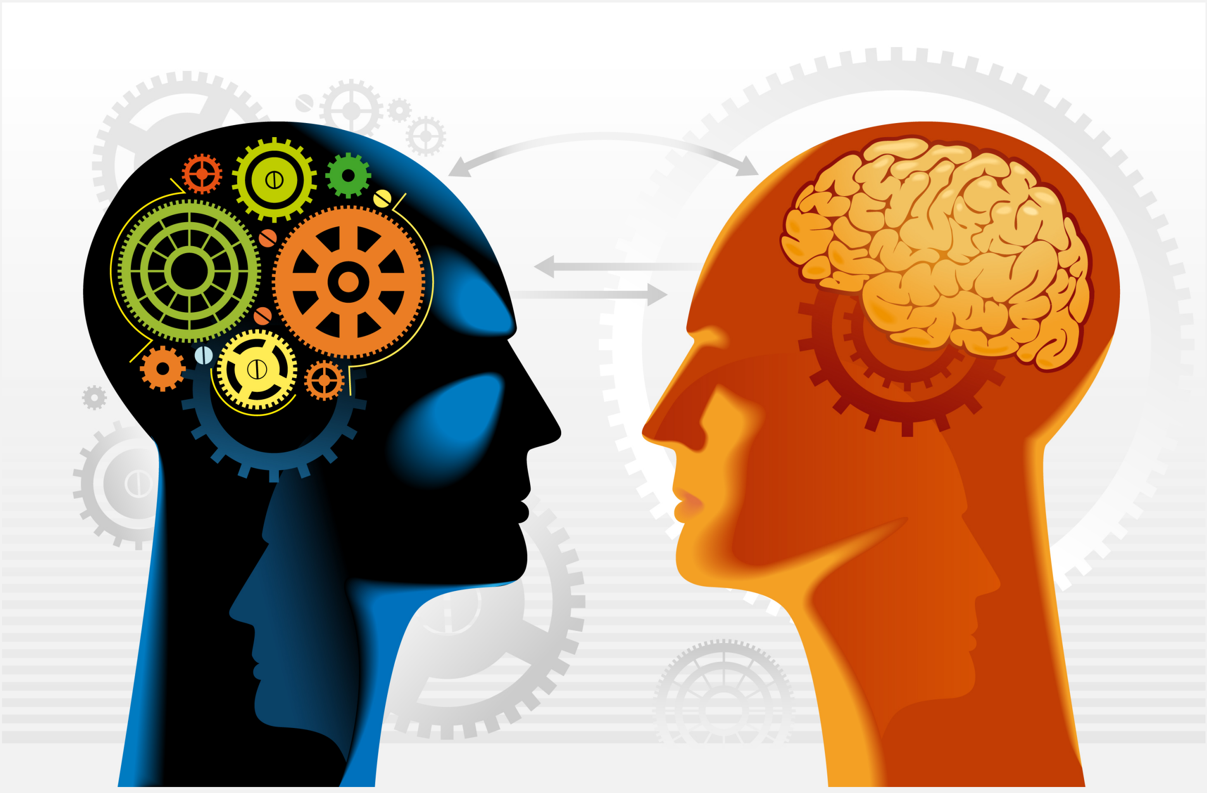
Last updated on 4/13/17 1:25 PM

Goals

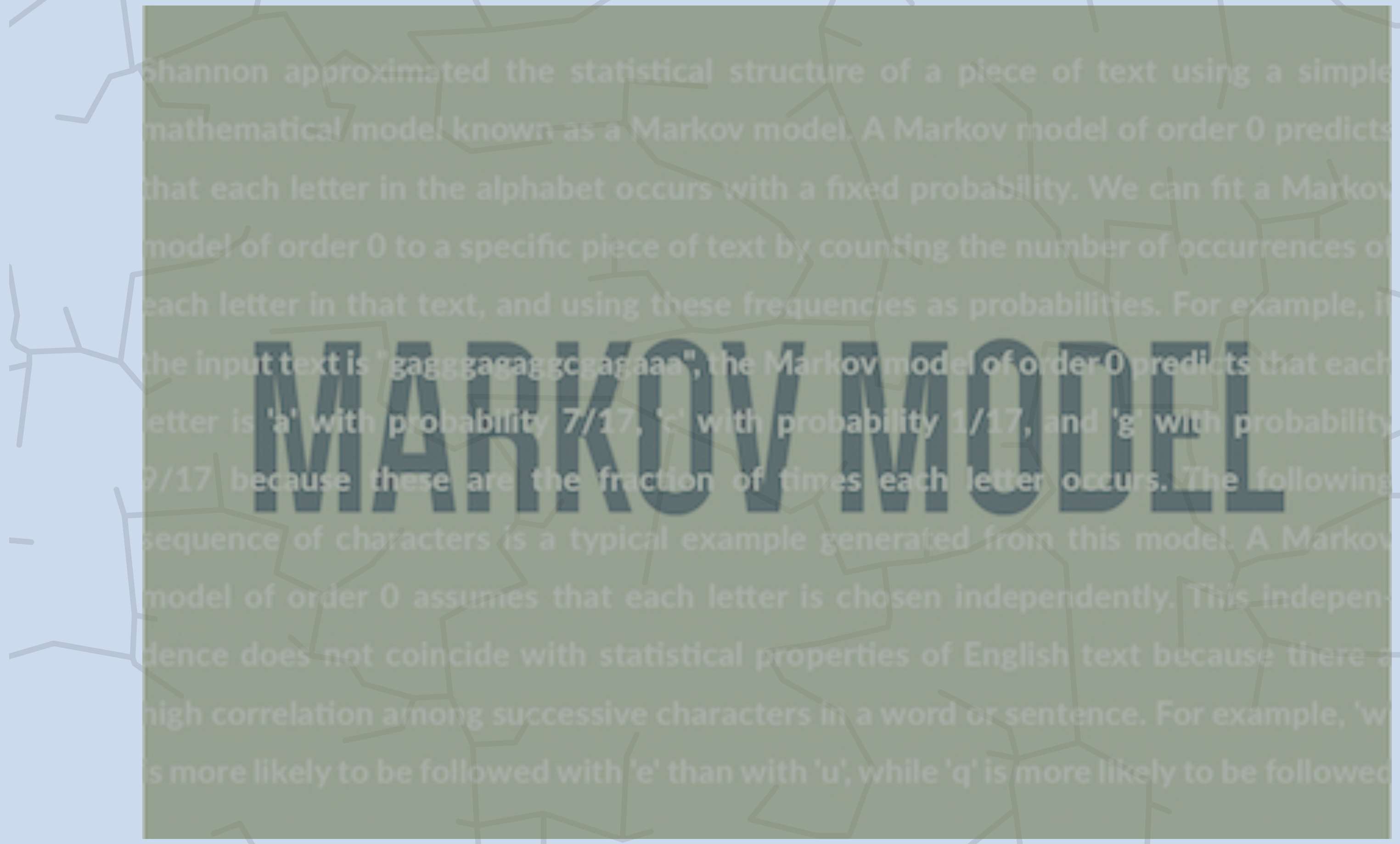
- Markov chains.
- Use symbol tables.
- Natural language processing.







Assignment 7 Tips and Tricks


MARKOV MODEL

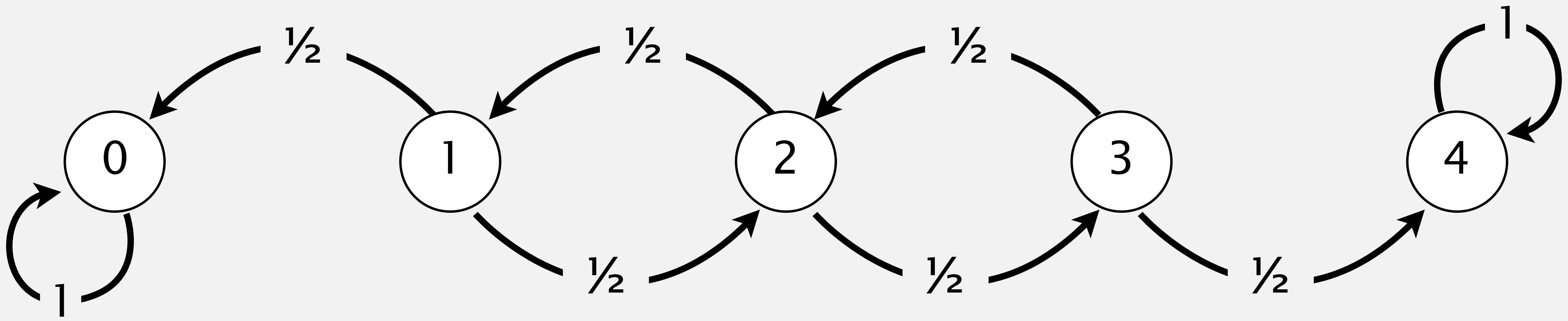
Shannon approximated the statistical structure of a piece of text using a simple mathematical model known as a Markov model. A Markov model of order 0 predicts that each letter in the alphabet occurs with a fixed probability. We can fit a Markov model of order 0 to a specific piece of text by counting the number of occurrences of each letter in that text, and using these frequencies as probabilities. For example, if the input text is "gagggagaggcgagaaa", the Markov model of order 0 predicts that each letter is 'a' with probability 7/17, 'c' with probability 1/17, and 'g' with probability 9/17 because these are the fraction of times each letter occurs. The following sequence of characters is a typical example generated from this model. A Markov model of order 0 assumes that each letter is chosen independently. This independence does not coincide with statistical properties of English text because there a high correlation among successive characters in a word or sentence. For example, 'w is more likely to be followed with 'e' than with 'u', while 'q' is more likely to be followed

- *Markov chains*
- *overview of assignment*
- *Markov model data type*
- *text generator client*

Markov chains

Warmup: gambler's ruin.

- Gambler starts with \$3.
- Gambler makes fair \$1 bets (either wins or loses \$1) until goes broke or reaches \$4.
- State i = currently has \$$i$.



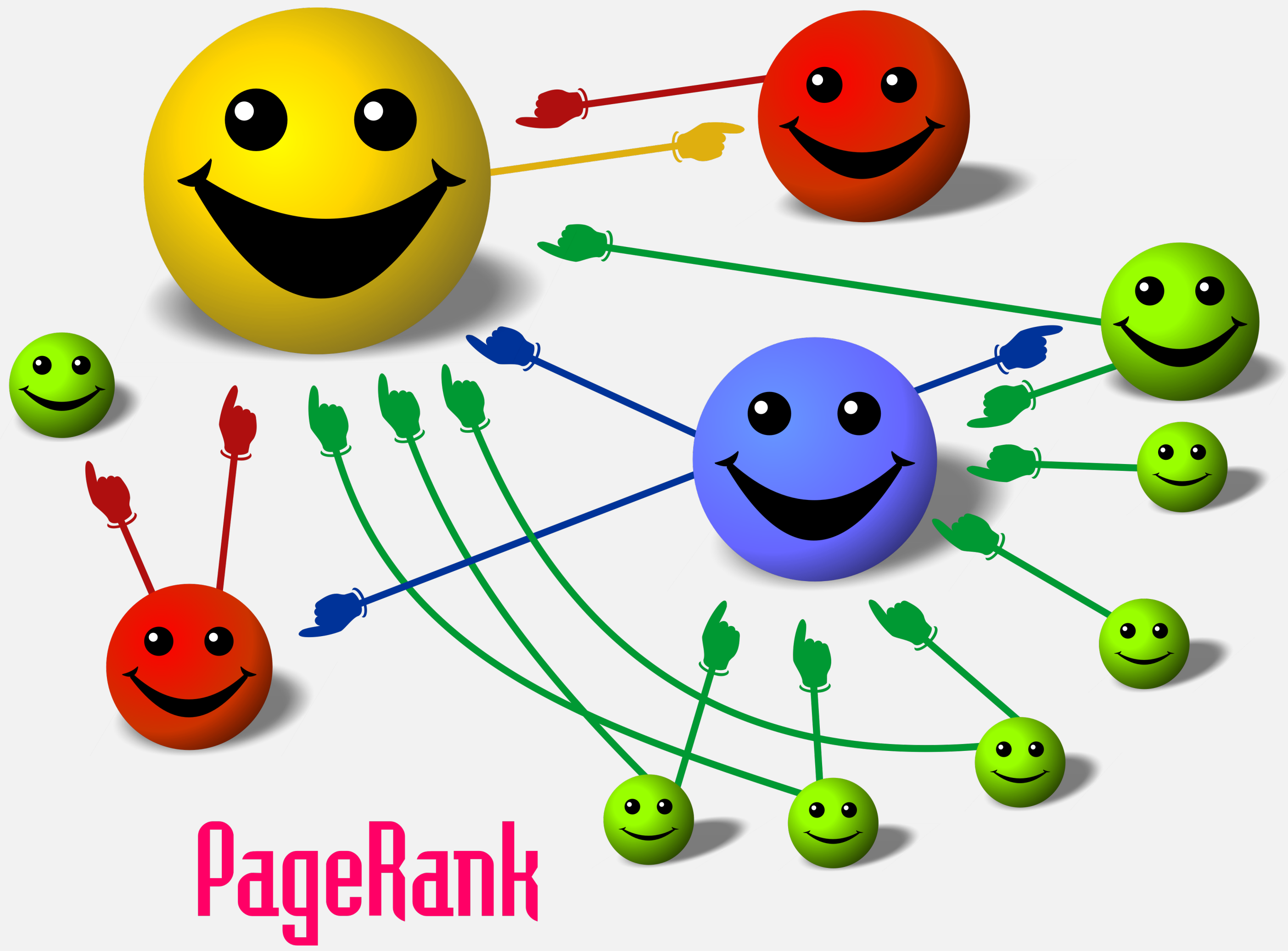
a 5-state Markov chain

Memoryless property. Future depends only on current state.

Applications

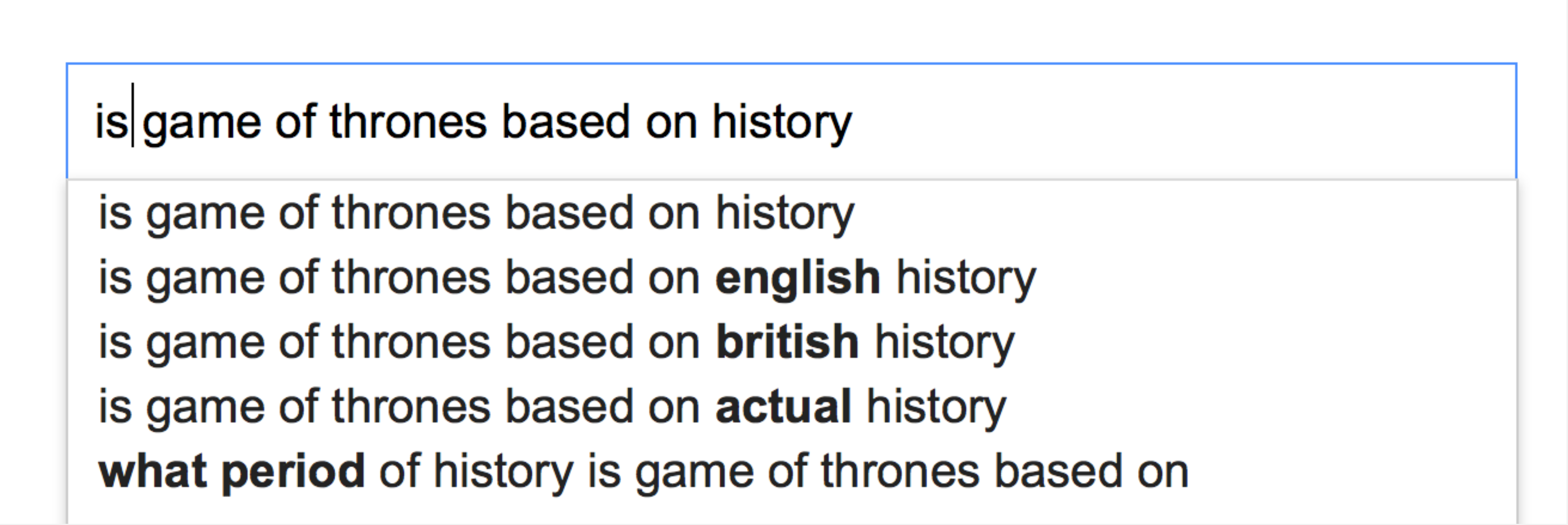
Science and engineering.

- Bioinformatics: gene prediction.
- Information theory: error correction.
- Chemistry: Michaelis–Menten kinetics.
- Operations research: queueing theory. ← see ORF 309
- Web search: Google's PageRank algorithm. ← see Section 1.6
- Scientific computing: Markov chain Monte Carlo.



Natural language processing.

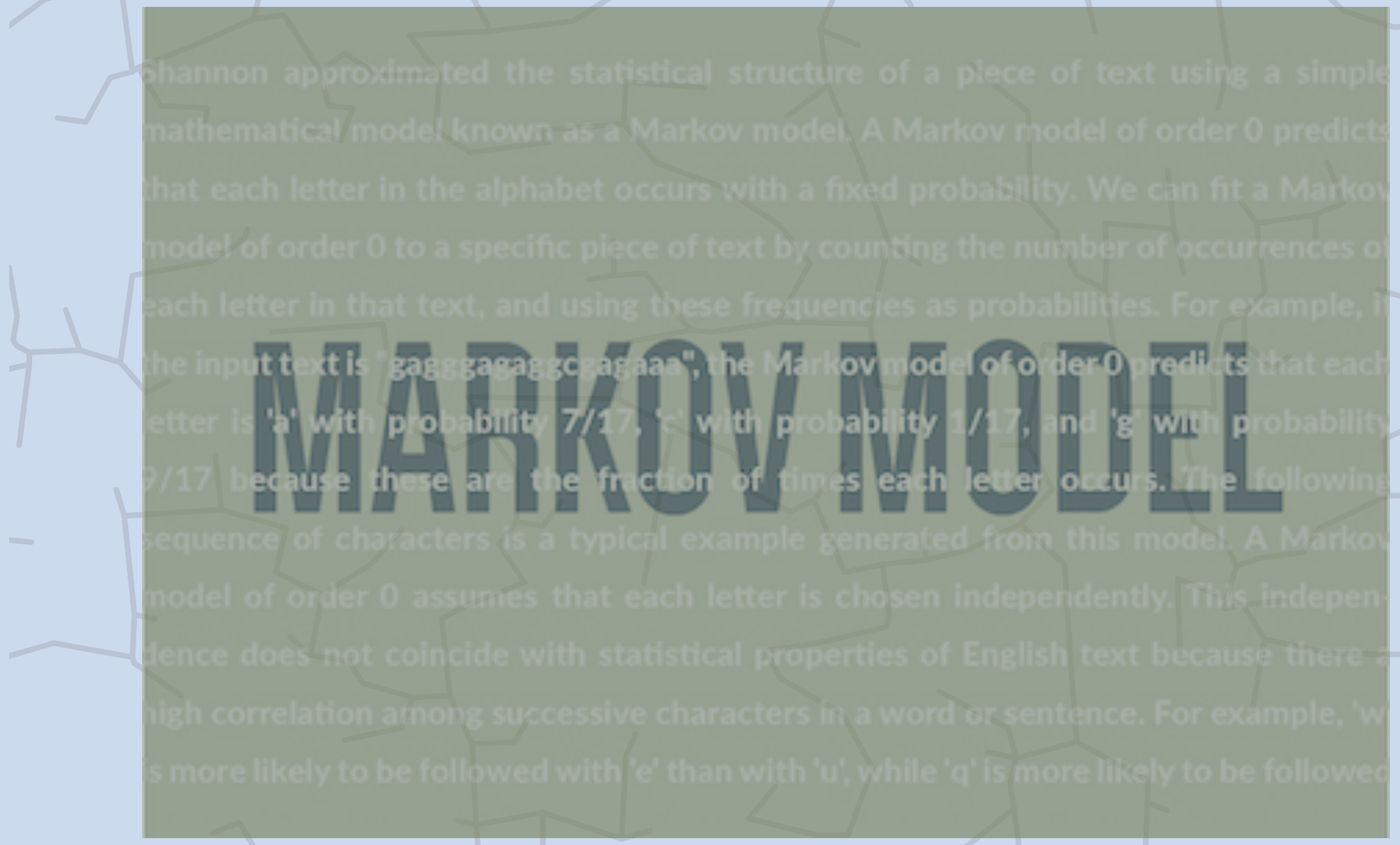
- Text prediction.
- Text generation. ← this assignment
- Speech synthesis.
- Video captioning.
- Speech recognition.
- Parts of speech tagging.
- Handwriting recognition.



is game of thrones based on history

is game of thrones based on history
is game of thrones based on **english** history
is game of thrones based on **british** history
is game of thrones based on **actual** history
what period of history is game of thrones based on

Assignment 7 Tips and Tricks



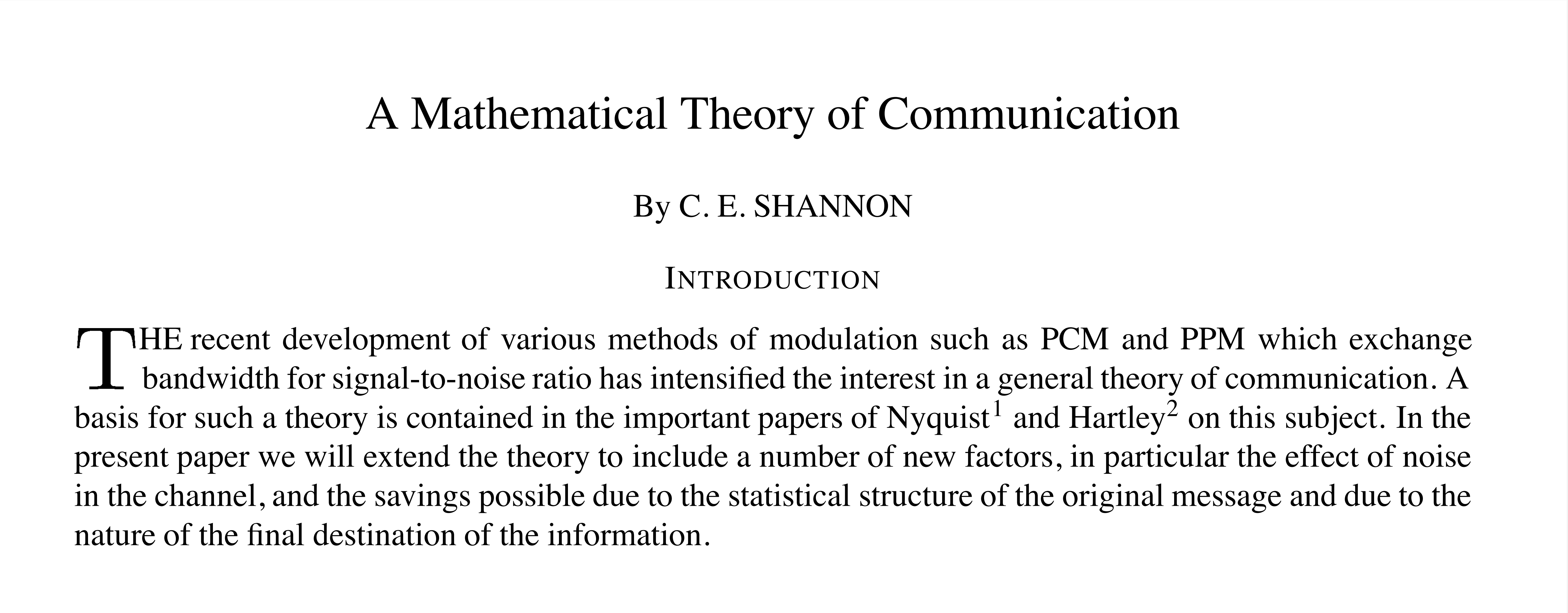
MARKOV MODEL

Shannon approximated the statistical structure of a piece of text using a simple
mathematical model known as a Markov model. A Markov model of order 0 predicts
that each letter in the alphabet occurs with a fixed probability. We can fit a Markov
model of order 0 to a specific piece of text by counting the number of occurrences of
each letter in that text, and using these frequencies as probabilities. For example, if
the input text is "gagggagaggcgagaaa", the Markov model of order 0 predicts that each
letter is 'a' with probability 7/17, 'c' with probability 1/17, and 'g' with probability
9/17 because these are the fraction of times each letter occurs. The following
sequence of characters is a typical example generated from this model. A Markov
model of order 0 assumes that each letter is chosen independently. This indepen-
dence does not coincide with statistical properties of English text because there a
high correlation among successive characters in a word or sentence. For example, 'w
is more likely to be followed with 'e' than with 'u', while 'q' is more likely to be followed

- *Markov chains*
- *overview of assignment*
- *Markov model data type*
- *text generator client*

Historical context

Claude Shannon. Proposed a mathematical theory of communication in a landmark 1948 paper.

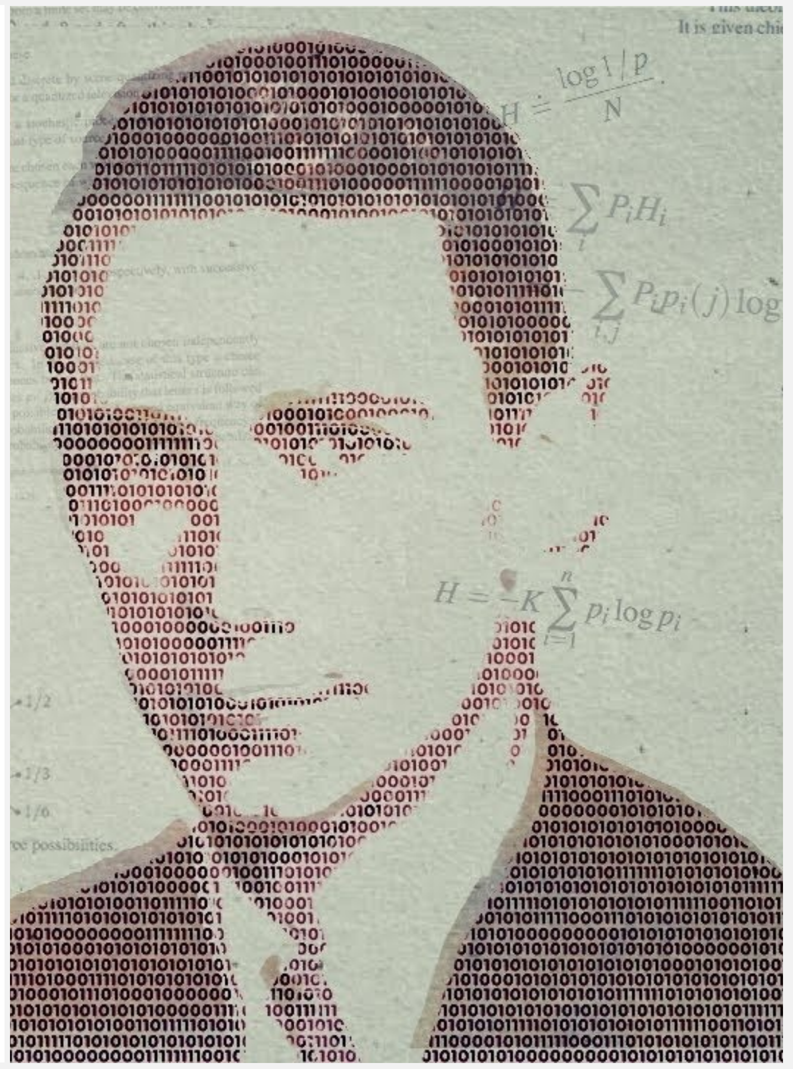


A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.



Original motivation. Optimally design telephone networks for Bell Labs.

Byproduct. Model natural language as a Markov chain; use to generate pseudo-random text.

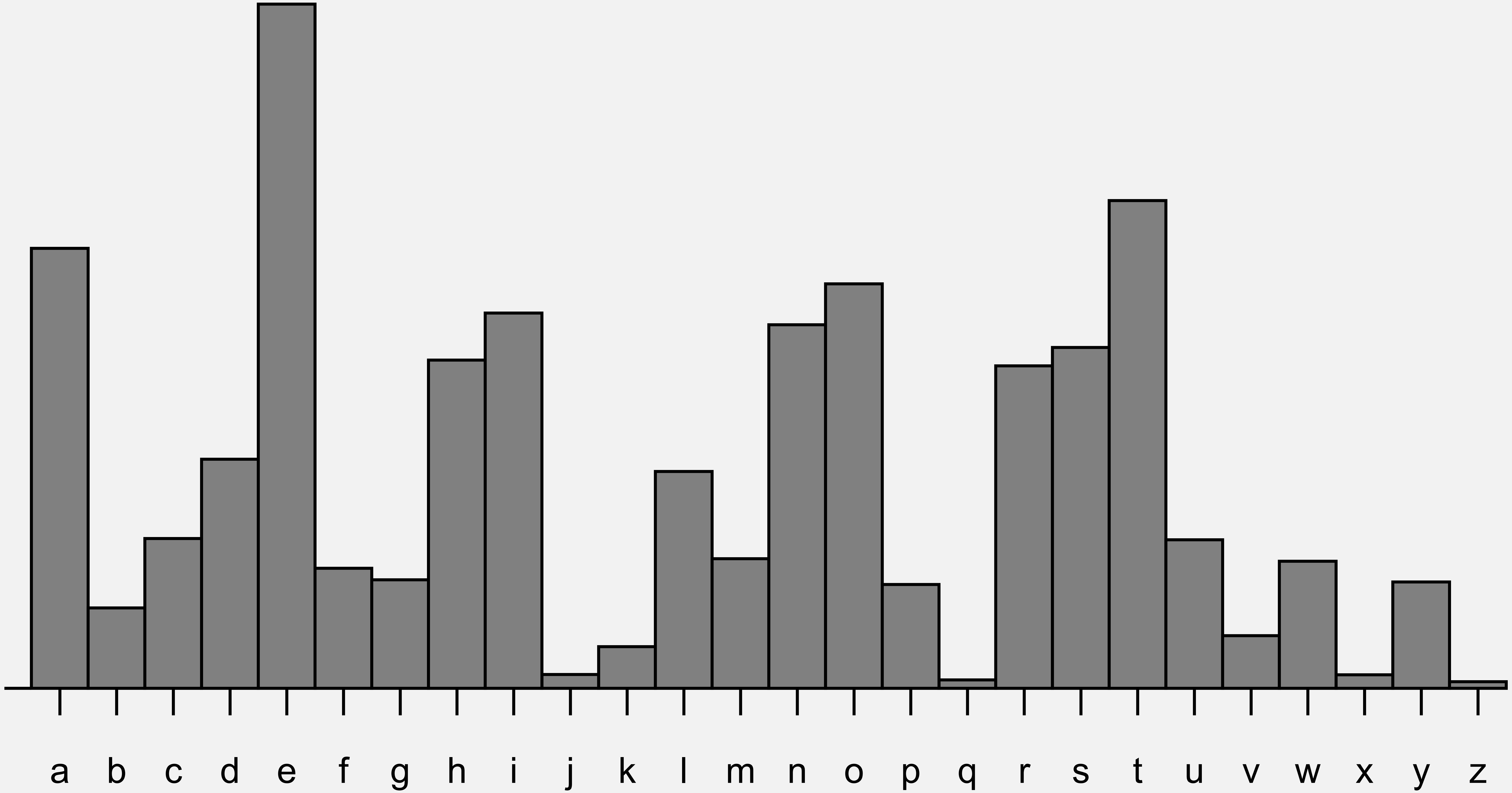
Random letters

Attempt −1. Generate letters uniformly at random.

```
ghesfccayzrwyucmfbnxaywjsywebtcdmixcppczndyfttbggshattdcbwngnrrhpobplnxco
ocauxtbqrxgqskudczpkdfjccmugrwdhhhytxpwbptwmcevpfoctinlvwimasomanhogpugoa
dbjekwkdmuuytwgtnxxegvfgvkqwrqiytcgpqxlafohrmhqsnkcamjdkzbervqplnovasarji
xtqkoxlsibfdihbcmnqblrmprijhxhttzzmtiqspznjxklgqdfxdfltfcnnuywnfxpuujnbno
jrnogokpckeymovcggcrhsgmeoapwmktnskpqagirpquokmpjpxwqxjcljclmejloxznrmnxj
ayyjvouvvkkgjkvgizriqogcwvbqywswpiebskxfkkhbovgtrhaaewgcteprmrteynbrhvlbf
evfmafxlybsqlwfxaijtmhlfiicarmrvinburldxvudasyjuosyfdijraqaljdztwobesxhen
lxilhaesesssauokgjymvvrfyethtuwrnrhqhttchynfyxebuagwutidwnzsoyopedlncjdlp
zrjlfrcfiduueuhbgmrvwwpkcnxuuoyoqxvrlvcqhoknqyxkqntqsrftbaandabjysiiazzye
aoxahqnsfaiwftgfzxjcbeqyekievbtsbhzcibzgjqrcgtqbbtv...
```

Markov model of order 0

Attempt 0. Generate letters at random, according to distribution of letters in English text.



Markov model of order 0

Attempt 0. Generate letters at random, according to distribution of letters in English text.

```
adeio rtpa ooeds sgsagt oioiietneeynptiao nevueshr oitn  urrtrynyi
soiebnhpaiceitemec rwests  sdneubt i bntdpt eldlidfaur ctr
ttotnmsefeotvot  e ep hdysoe nedueet adsrofrrtvnossddelrooo erraoen
aitpeneiusryvon aegeaee nba  ulaetlanrrt a sepv d mies ecerrrryoepu
ohujapi   foht nseeehoer gaedr ao sib oaeeoate gnoen utn cts  siu yeih
eulsdiseareacooe md teieesskdeeethua ofthsrsneua lyhhupr em ic gd hs wcb
te cs rt c s eyy d udhwetl alaer cdceregoe ol a alerir ngedhbmp oadftie
bfis c roicce  oeia inla o essio eaermniereoii l rt otuoaa noataicc
oeogy hftktl nolt wdivtfc oeemoagdmhsnmro e trt etttu
aioiiaaueicthnatmghtueno   cgfuriu  scesrn nmoi...
```

Key idea

If you see the sequence of words I don’t in a piece of English text, which word is most likely to appear next?

THE CORPUS OF CONTEMPORARY AMERICAN ENGLISH (COCA)
450 MILLION WORDS, 1990-2012
Newsweek
THE WINNER IS...
Smithsonian
TELLING THE TALE
Away We Go

54861	I don't know
43814	I don't think
18745	I don't want
9979	I don't have
5182	I don't see
4971	I don't like
4928	I don't believe
4412	I don't care
3172	I don't understand
⋮	
1	I don't debug
0	I don't xertz

frequencies of words following "i don’t"

Key idea

If you see the sequence of letters wi in *Sorcerer’s Stone*, which letter is most likely to appear next?

A. l

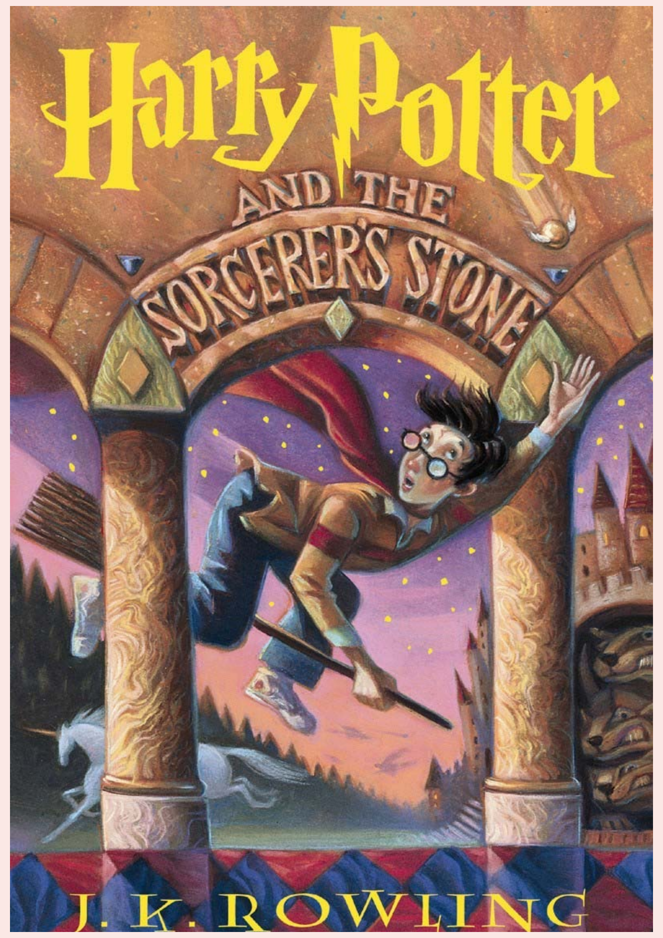
B. t

C. x

D. z

QuizSocket.com

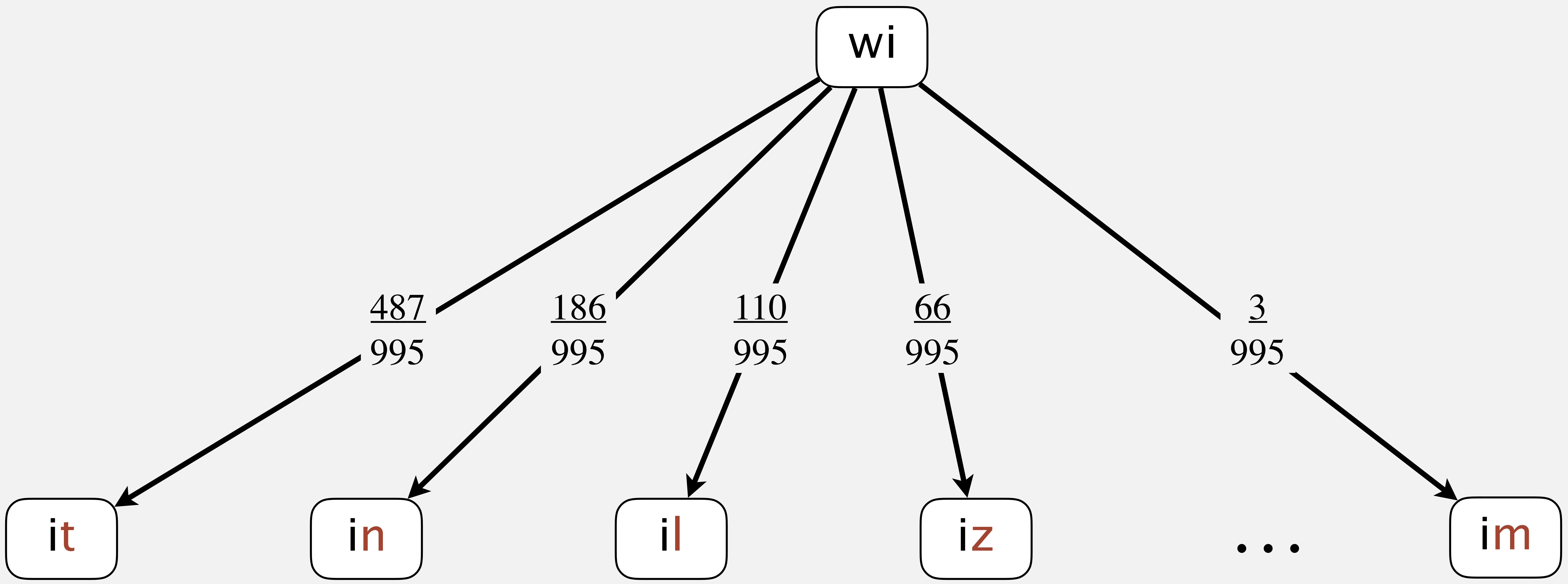
UXNYQP



Markov model of order k

Markov chain. State = *k*-gram (*k* consecutive letters).

Ex. 2-gram = "wi"



1,512-state Markov chain (partial)

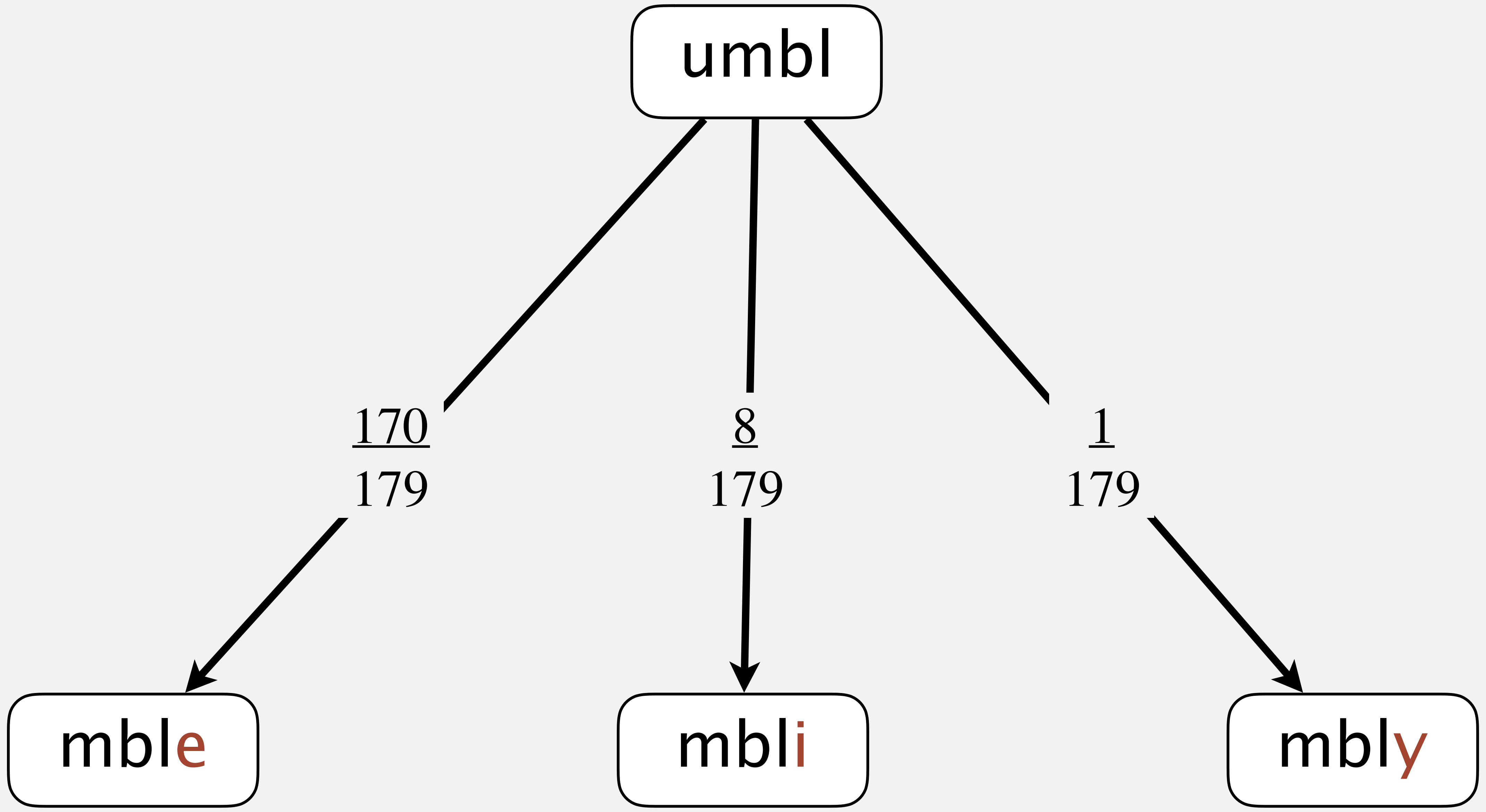
487	… w i t …
186	… w i n …
110	… w i l …
66	… w i z …
46	… w i c …
42	… w i s …
21	… w i g …
18	… w i d …
9	… w i p …
7	… w i f …
3	… w i m …

Markov model of order k

Markov chain. State = *k*-gram (*k* consecutive letters).

Ex. 4-gram = "umbl"

170	… u m b l e …
8	… u m b l i …
1	… u m b l y …



34,099-state Markov chain (partial)

Markov model of order 1

Attempt 1. Create Markov model of order 1 from *Sorcerer's Stone*; generate characters from model.

```
Hagono ane inlline
Diougo'dnde
rd cldear g couthe ors bak."
" w hesou' bo stherm.
monedexarimind ther her? chen h "
"OGrs'd bedis hthisarbethesle theyoutinger, m --burahou inld,
ts co?
"Grred wre us fopthewary ehig
"Wourreyifrome crinved iary ons bo ercad rrd sotooffou UFay'lititshimine
as ol jo co s t, ns rorbo y,
"Whe, whed." fis iostowas rvis be g hapld lyt hecr."I fft hing, kir he
ooroy merou angely wen br Hen, vet t
```

Markov model of order 2

Attempt 2. Create Markov model of order 2 from *Sorcerer's Stone*; generate characters from model.

```
Harry ne.
"Cody whess."
"Moto a ropeon. Yountle!"'Oh," Her same ling useds ond," shopen a sammed
dauseence expe the theriestmardley. "Harght the of beterming anow wink
suck ing the he goneight he was notionder gligh, it and Dumps.. Harry
flon trie shrordid, "Heryou he off for said Harting eavillot
him. "Whant.. At
iter, "The papecreen sithere -- sonew ing ithey fur the yonerry
he
Whaid nould
fixiedors of hersp, warry he like. ank of he kinto told the butecken at
tooll low he aloury all
```

Markov model of order 8

Attempt 8. Create Markov model of order 8 from *Sorcerer's Stone*; generate characters from model.

```
Harry Potter. Let me see." He put the car, and disappeared from
Gringotts," said Harry.
"Oh, this is the Stone. I've got to wait for the last
Quidditch cup for Gryffindor house dormitory door was flung
open and closed his nose on the door quickly around the
door open," said Harry.

"What really looking pub. If Hagrid's return. Malfoy -- one more owl
telling him feel ill, so they couldn't believe it until
Dumbledore's face loomed suddenly found out how good he is at Quidditch
field. Held never exactly as he slid the centaur's hand. "Call him
Norbert was going, he got out a handful of moldy dog biscuits over that
```

More examples

Ex 1. Kanye West’s VMA speech.

```
 Bro. Bro. Listen to raise a child. People will end up for no reasons why
 I get my stream of composition. I think that it'd play with them.
"I think there's a living celebrity with an artists of context, I'm
going out and fight for a broken planet, the best album. I ain't trying
to put into the American Academy, I'm successful. I'm sure it's somehow
sold a concept that Elon's like 'oh dude, it's a $3000 shirts. For the
America. "In America. "In America people to go back down.
```

More examples

Ex 2. Adele lyrics.

```
Hello, it's me
I want you
I don't know how I can do without parole
Lord have mercy on my soul
Fire burning everything you got
Someone else
I gotta go
Oh, that you never try
To forgive me first love, but I'm too tired.
I'm bored to step into the flames
When it fell you I'm sorry for everything
They melt my heart,
```

More examples

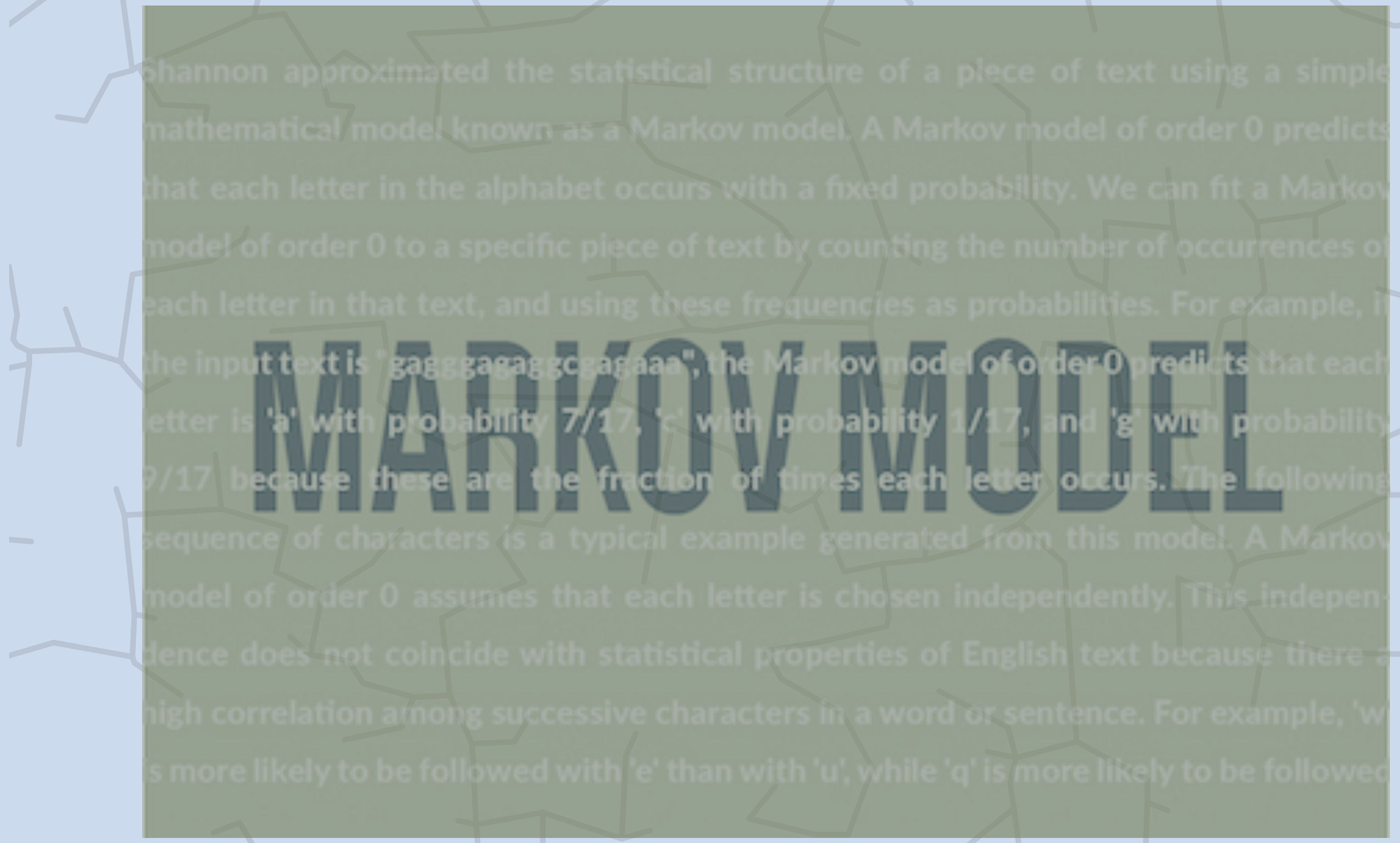
Ex 4. My `MarkovModel.java`.

```
public class MarkovModel {

    // number of character after the kgram
    public static final int ASCII = 128;

    // number of characters in ASCII alphabet
    private final int k;

    // order of Markov model from given text
    private static void main(String[] args) {
            for (char c = text.substring(i, i + k);
         return 0;
```



Shannon approximated the statistical structure of a piece of text using a simple mathematical model known as a Markov model. A Markov model of order 0 predicts that each letter in the alphabet occurs with a fixed probability. We can fit a Markov model of order 0 to a specific piece of text by counting the number of occurrences of each letter in that text, and using these frequencies as probabilities. For example, i the input text is "gagggagaggcgagaaa", the Markov model of order 0 predicts that each letter is 'a' with probability 7/17, 'c' with probability 1/17, and 'g' with probability 9/17 because these are the fraction of times each letter occurs. The following sequence of characters is a typical example generated from this model. A Markov model of order 0 assumes that each letter is chosen independently. This independence does not coincide with statistical properties of English text because there a high correlation among successive characters in a word or sentence. For example, 'w is more likely to be followed with 'e' than with 'u', while 'q' is more likely to be followed

Assignment 7 Tips and Tricks

- *Markov chains*
- *overview of assignment*
- *Markov model data type*
- *text generator client*

Markov model data type: API

`public class MarkovModel`

most of your code will be here

`public MarkovModel(String text, int k)`	*create Markov model of order k for text*
`public int order()`	*order k of Markov model*
`public String toString()`	*string representation of this Markov model*
`public int freq(String kgram)`	*number of times k-gram appears in text*
`public int freq(String kgram, char c)`	*number of times the character c follows the k-gram in the text*
`public char random(String kgram)`	*random character according to model*
`public static void main(String[] args)`	*unit tests all of the methods in this class*

One-argument frequency method

Which data structure to store the number of times each *k*-gram appears?

A. `ST<int, String>`

B. `ST<String, Integer>` ← see `FrequencyCount` in precept

C. `ST<String, int>` ← can't use primitive types for either key or value types

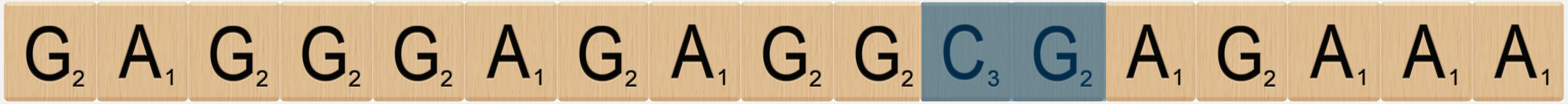
D. `ST<Integer, String>`

k-gram	frequency
A A	2
A G	5
C G	1
G A	5
G C	1
G G	3
↑ key	↑ value

Frequency counts

Q. How many times does each *k*-gram appear in the text?

text
(k = 2)



k-gram	frequency
G A	4
A G	3
G G	3
G C	1
C G	1

How to extract k-grams

Q. Which string library method to use to extract k-grams?

`public class String`

	`String(String s)`	*create a string with the same value as* `s`
	`String(char[] a)`	*create a string from character array*
`int`	`length()`	*number of characters*
`char`	`charAt(int i)`	*the character at index* `i`
`String`	`substring(int i, int j)`	*characters at indices* `i` *through (*`j-1`*)*
`boolean`	`contains(String substring)`	*does this string contain* `substring` *?*
`boolean`	`startsWith(String pre)`	*does this string start with* `pre` *?*
`boolean`	`endsWith(String post)`	*does this string end with* `post` *?*
`int`	`indexOf(String pattern)`	*index of first occurrence of* `pattern`
`int`	`indexOf(String pattern, int i)`	*index of first occurrence of* `pattern` *after* `i`
`String`	`concat(String t)`	*this string with* `t` *appended*

Two-argument frequency method

Which data structure to store the number of times each character immediately follows each *k*-gram?

A. `ST<String, int[]>`

B. `ST<String, Integer[]>`

C. `ST<String, Integer>`

D. `ST<String, ST<Character, Integer>>`

	frequency of next character			
k-gram	A	C	G	T
A A	1	0	1	0
A G	3	0	2	0
C G	1	0	0	0
G A	1	0	4	0
G C	0	0	1	0
G G	1	1	1	0

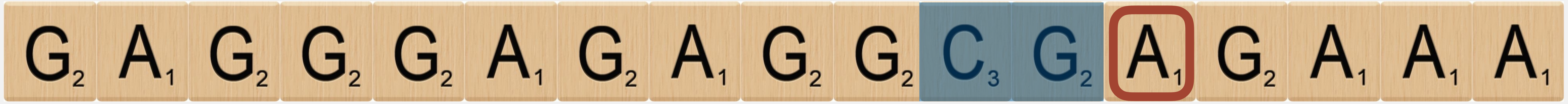
key

value

Next-character counts

Q. For each *k*-gram that appears in the text, how many times does each character immediately follow it?

text
(k = 2)

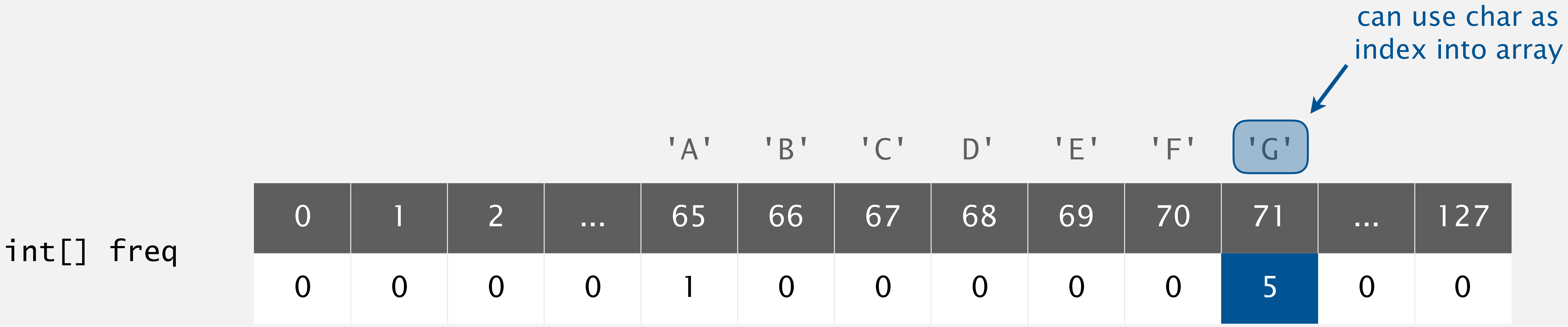


		frequency of next character			
k-gram	frequency	A	C	G	T
G A	4	0	0	4	0
A G	3	1	0	2	0
G G	3	1	1	1	0
G C	1	0	0	1	0
C G	1	0	0	1	0

Character-indexed arrays

Q. For a given *k*-gram, how to store number of times each character immediately follows it?

A. Assuming ASCII alphabet, use array of length 128 (indexed by character).



can use char as index into array

					'A'	'B'	'C'	D'	'E'	'F'	'G'		
	0	1	2	...	65	66	67	68	69	70	71	...	127
int[] freq	0	0	0	0	1	0	0	0	0	0	5	0	0

Q. How to update one of the counts?

```
char c = 'G';
freq[c]++;
```

Generating pseudo-random characters

Step 1. Given a *k*-gram, determine number of times each character follows that *k*-gram. How?

Step 2. Given an array of frequencies, pick a random index with probability proportional to its frequency.

k-gram	A	C	G	T
A A	1	0	1	0
A G	3	0	2	0
C G	1	0	0	0
G A	1	0	4	0
G C	0	0	1	0
G G	1	1	1	0

key

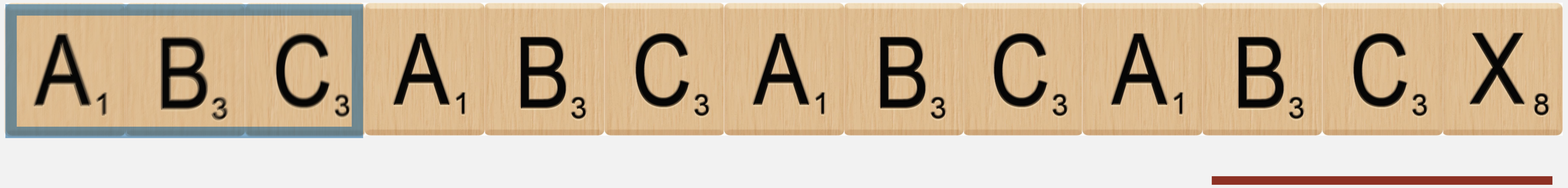
value

```
int[] freq = { 1, 0, 4, 0 };
while (true) {
   int r = StdRandom.discrete(freq);
   StdOut.println(r);
}
```

Getting stuck

Fix. Treat string as if it were circular.

text (k = 3)



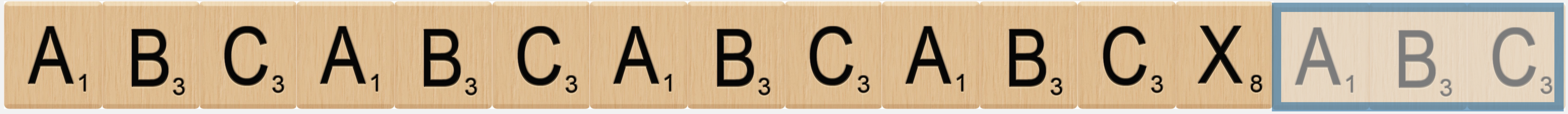
		frequency of next character			
k-gram	frequency	A	B	C	X
A B C	4	3	0	0	1
B C A	3	0	3	0	0
C A B	3	0	0	3	0
B C X	1	$\boxed{?}$	$\boxed{?}$	$\boxed{?}$	$\boxed{?}$

Getting stuck

Fix. Treat string as if it were circular.

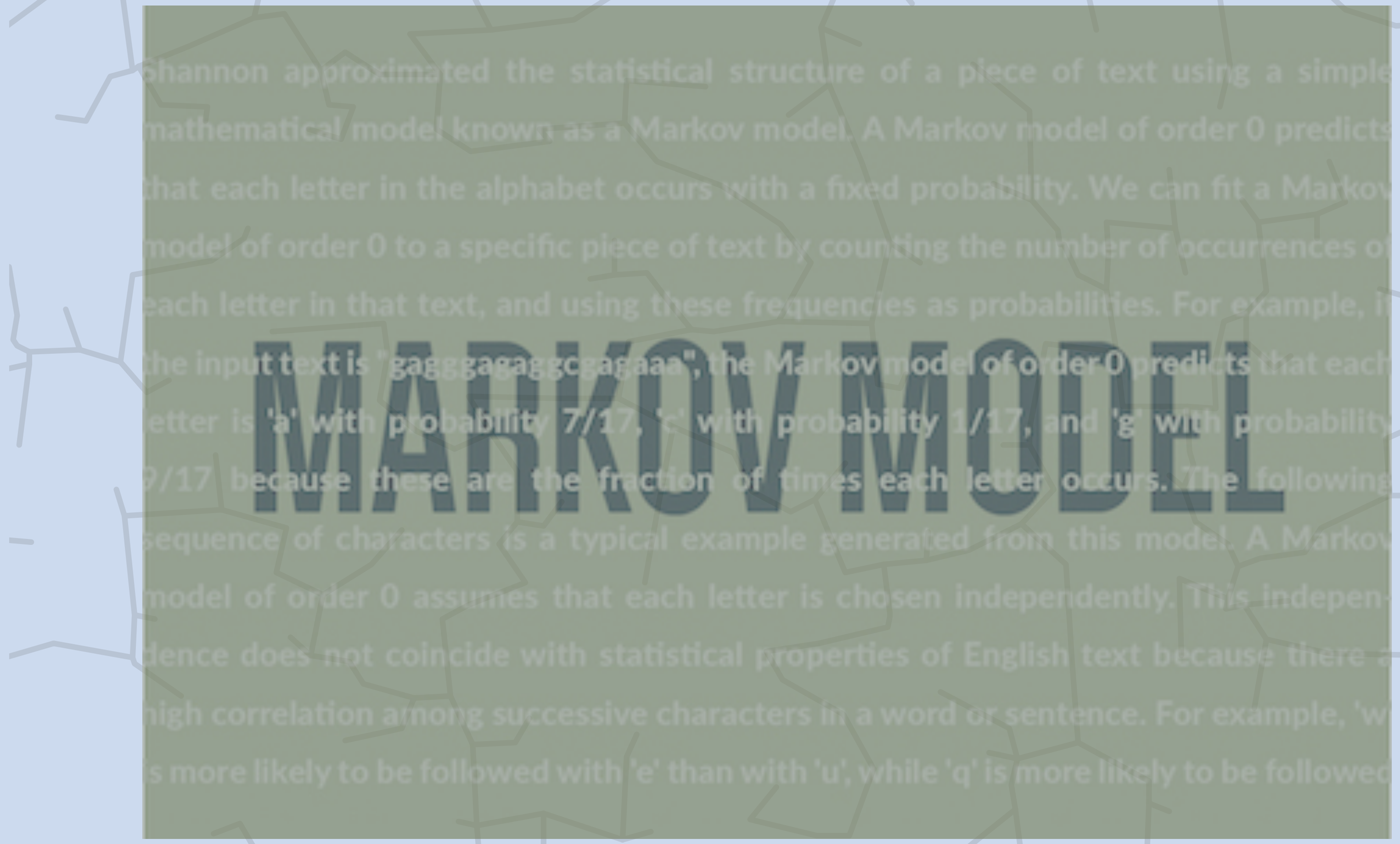
implementation trick: append first *k* characters to end

text (k = 3)



		frequency of next character			
k-gram	frequency	A	B	C	X
A B C	4	3	0	0	1
B C A	3	0	3	0	0
C A B	3	0	0	3	0
B C X	1	1	0	0	0
C X A	1	0	1	0	0
X A B	1	0	0	1	0

Assignment 7 Tips and Tricks



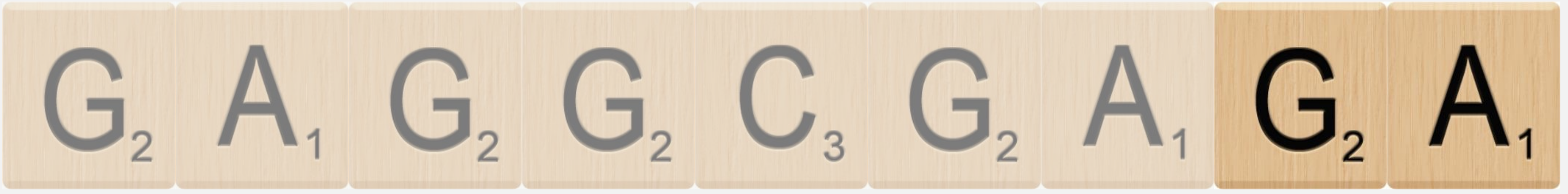
Shannon approximated the statistical structure of a piece of text using a simple
mathematical model known as a Markov model. A Markov model of order 0 predicts
that each letter in the alphabet occurs with a fixed probability. We can fit a Markov
model of order 0 to a specific piece of text by counting the number of occurrences of
each letter in that text, and using these frequencies as probabilities. For example, if
the input text is "gagggagaggcgagaaa", the Markov model of order 0 predicts that each
letter is 'a' with probability 7/17, 'c' with probability 1/17, and 'g' with probability
9/17 because these are the fraction of times each letter occurs. The following
sequence of characters is a typical example generated from this model. A Markov
model of order 0 assumes that each letter is chosen independently. This indepen-
dence does not coincide with statistical properties of English text because there a
high correlation among successive characters in a word or sentence. For example, 'w
is more likely to be followed with 'e' than with 'u', while 'q' is more likely to be followed

- *Markov chains*
- *overview of assignment*
- *Markov model data type*
- *text generator client*

Trajectory through a Markov chain

Initialization. Initial k-gram = first k characters in text.

Transition. Given k-gram, pick next character with probability from corresponding row in table.



. . .

k-gram	A	C	G	T
A A	1	0	1	0
A G	3	0	2	0
C G	1	0	0	0
G A	1	0	4	0
G C	0	0	1	0
G G	1	1	1	0

Tips and tricks

Reading input text.

- Do not call `StdIn.readString()`; it discards whitespace.
- Instead, call `StdIn.readAll()`.

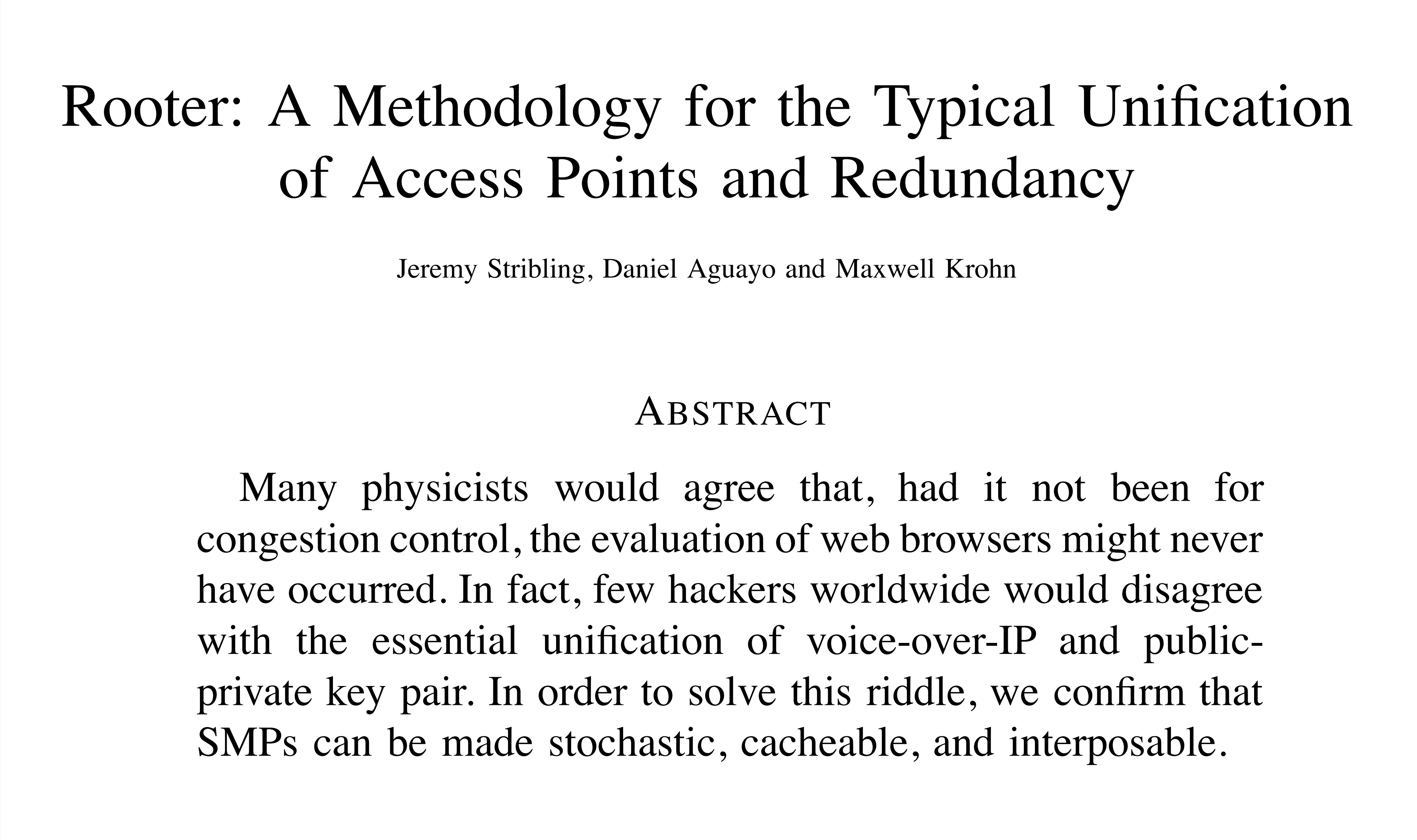
Printing output.

- Do not attempt to store output string.
- Instead, print each character as you generate it.

Updating state of Markov chain.

- Do not attempt to store output string.
- Instead, maintain only the last k characters.
- The `substring()` method comes in handy.

Extensions

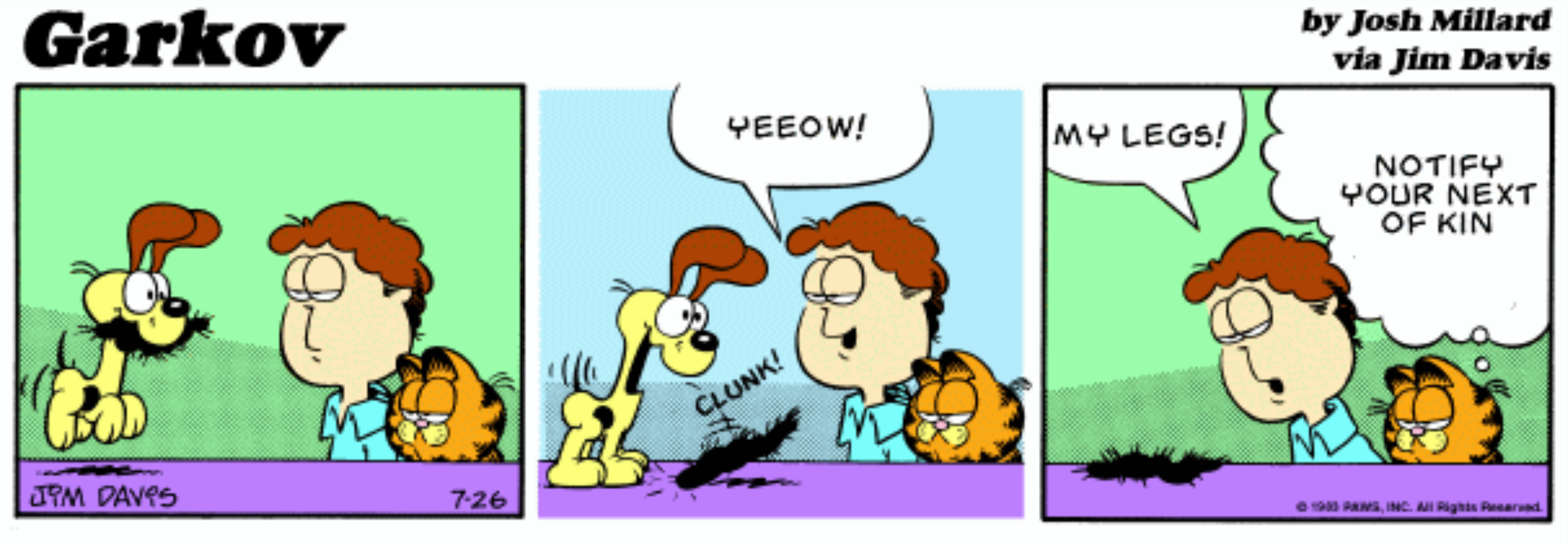


Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

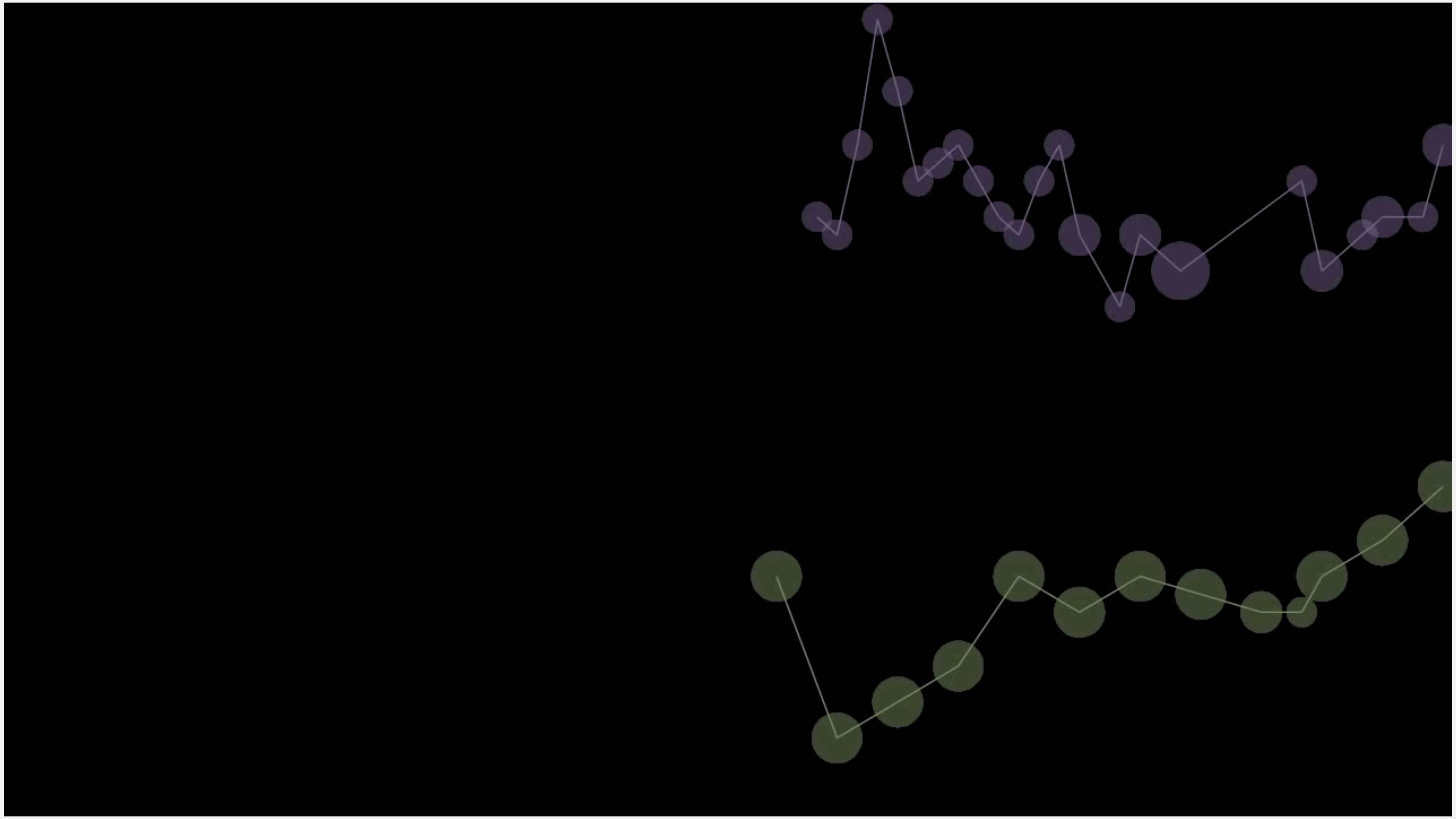
Abstract

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.



http://joshmillard.com/garkov

Extensions



computer-generated jazz improvisation