

Keyboard Layout Analysis: Creating the Corpus, Bigram Chains, and Shakespeare's Monkeys

Ian Douglas, B.Sc

ian@zti.co.za

29 March 2021

Version 1.0.1

DOI: <https://doi.org/10.5281/zenodo.4642459>

This work is licensed under the Creative Commons Attribution 4.0 International License.

Unqid: 92e5d77b5d264a71aa92780947867129

Please check via DOI for latest version.

Abstract

The process to create a corpus suitable for evaluating computer keyboard layouts optimised for typing English and computer program code. After sourcing, sampling and cleaning suitable texts, the texts are processed to extract bigrams, which are then used to create sample input texts of a desired length. These texts have a character distribution, and letter sequence, closely matching either English or computer programs, even though they look random. The resulting texts are excellent for evaluating keyboard layouts. Corpus analysis is included.

Keywords: English text corpus, computer code corpus, English letter frequency, computer program character frequency, bigram frequency, letter follows letter probability, letter precedes letter probability, keyboard layout, keyboard layout evaluation.

Best viewed and printed in colour.

Contents

1. Introduction
2. Existing corpora and results
3. Creating the English corpus
4. Creating the computer code corpus
5. Corpora analysis
6. Creating chained bigrams (Markov chains) and texts
7. Samples and analysis
8. List of datasets and files
9. Acknowledgements
10. Bibliography
11. Appendix A: Keyboard layouts used in tests.

Updates:

28 March 2021 1.0.0 Initial version.

29 March 2021 1.0.1 Added 4,5,6,7,8,9-grams, made tables more compact. Added Appendix A.

1. Introduction

When designing or evaluating a computer keyboard layout for a given language, it is necessary to know the character frequency for that language. It is also useful to know the bigram and trigram frequencies. These frequencies are calculated by analysing a suitable corpus of text.

However, the available corpora, or indeed analysis, was driven by other needs, typically cryptographic or lexical analysis, which are totally different to the keyboard layout problem. These corpora typically include spoken speech transcripts, which is irrelevant to typing.

Keyboard layouts are usually analysed in one of two ways:

1. By feeding sample texts to an analysis program, or
2. By a program using known bigram pairs

There are problems with both approaches. In the first case, it is extremely difficult to find small sample texts that have the characters in the correct frequency, or indeed include all the characters. This leads to incorrect results.

The bigram approach, favoured by academia, often falls short differently. Available bigram lists typically only include letters, ignoring case, and are extracted from corpora created for different needs. The bigram analysis is also frequently “disjointed,” in that bigrams are considered in isolation rather than as parts of words with spaces and punctuation. This approach also leads to incorrect results.

Today, there are millions of programmers typing programs in a variety of different programming languages, sometimes using multiple languages in one program. This is similar to trying to use one keyboard layout to type two different languages, with differing character frequencies and different common bigrams. Creating a layout that is optimal for both use cases is difficult.

We solve these problems by first creating two corpora, one for English and one for computer code. We then analyse the result, extracting the character frequencies and likelihood that x follows y , and that y precedes x . We then use this data to create bigram chains (technically Markov chains), before putting Shakespeare's Monkeys to work to create bigram-based input texts that solve the problems raised above. These texts appear to be random junk but they are not, and are excellent for analysing keyboard layouts. They are “words” made of the bigrams, correctly frequenced, and as such address the problems for both approaches to keyboard layout analysis.

The corpus collection was done around September 2020, all files sourced from the Internet are as of that date.

2. Existing corpora and results

Two frequently used resources are the analysis by Peter Norvig [1], published around 2012, and the study by Jones and Mewhort [2], published in 2004.

Norvig used bigram data from Google, but the analysis was limited to the letters, and ignored case.

Jones and Mewhort assembled a mixed corpus. It included full-text articles from the New York Times, a subset of the Brown word corpus, an online encyclopaedia (probably Wikipedia), text extracted from about 100,000 randomly selected Web pages, and newsgroup text extracted from 400 different Internet discussion groups.

I examined the actual content in the Brown corpus (description [3]), and decided that it was unsuitable as source material for keyboard layout evaluation. I also took a look at an available newsgroup corpus [4]. It has been “cleaned,” but the nature of Usenet means that there is a lot of computer-generated text, like message headers. Also, text gets forwarded or quoted without being retyped, which impacts the character frequency. Even the newsgroup naming scheme leads to excess “.” or other letters,

Much of the text is just a mess, here's a sample:

```
|>=09From: Mj=F8ln=EBr <<EMAILADDRESS>> > |>=09Newsgroups:
alt.binaries.warez.ibm-pc.d |>=09Subject: Re: The WarezFAQ [...] ...and a
WARNING |>=09Message-ID: <<EMAILADDRESS>> > |>=09Date: Wed, 23 Mar 2005 12:02:20
GMT |> |>=09In <<EMAILADDRESS>> |>=09on Sun, 13 Mar 2005 08:10:37 GMT, Zeke <
<EMAILADDRESS>> > wrote:
|> |>=09>In article <<EMAILADDRESS>> |>= <EMAILADDRESS> says... |>=09>>=20 |
>=09>>=20 |>=09>> If ANYONE wants to visit this site I STRONGLY suggest that you
use = a |>=09>> proxy to do it. The site has been known to harvest your
information |>=09>> and this has been posted to usenet! |>=09>>=20 |> |>=09The
poste
r of that "warning" is accessing Usenet from his room in a |>=09mental hospital!
That's has been proven beyond any doubt, and the proo= f |>=09posted to Usenet.
|> |>=09He's deeply delusional and violently insane. He's also in love with |
>=09Barbara Bush. BEWARE! |> |>
-----
----- |>
```

This excluded Usenet postings as a suitable text source, and raised questions about the suitability of Jones and Mewhort's results for keyboard layout analysis. Their sources were also largely American, and I needed more British English.

So I decided to create a new corpus, more suited to the task at hand. I would need two collections, one with written English, and one with computer program code.

3. Creating the English corpus

I thought it prudent to follow a similar approach to Jones and Mewhort. The goal was to get as wide a selection as possible, of texts created on keyboards. This meant excluding texts mostly created on small-screen devices, where the input mechanics are completely different.

I did not have access to the New York Times texts, but there is a publicly-available Reuters archive [5] of short financial reports.

This required some cleaning. By “cleaning,” I mean “replace any characters not on the standard US-ANSI keyboard, with characters that are.” For example, typographic quotes get replaced with ASCII quotes. If there is no simple replacement (for example, Chinese characters), delete the character. Some characters were replaced with their non-diacritic version, for example “é” became “e”, or its HTML version, “é”, depending on context. The goal is to replace non-typeable characters with typeable, wherever possible. This process was necessary for all files in the corpus, and was done using a program that did regular-expression replacements.

After cleaning, the Reuters archive provided 795 files of 689 bytes to 13.9 kB in size.

For encyclopaedia articles, the obvious solution is Wikipedia. Since Wikipedia can be edited by anyone, I thought it prudent to only select larger articles, on the assumption that these will be mature and well-edited. This assumption is not necessarily true. In the end, I had two collections, consisting of extracts from larger articles, and another collection extracted from smaller texts. These extracts required considerable cleaning. The result was 3757 files of 10 - 15 kB each.

I did also try getting extracts from Wikibooks, but these texts proved unsuitable.

Instead, I used the tools provided by Martin Gerlach and Francesc Font-Clos [6] to get books from Project Gutenberg, and following a similar approach to Wikipedia, and took extracts. For each book, if the word count was over 10,000, I would skip the first 200 lines (Gutenberg front matter and contents), and then take a 2000 word extract, which was then cleaned. This produced 7433 files of 9 to 39 kB each.

I took a similar approach to sampling the OMBC Web Base corpus [7], which resulted in 223 files ranging from 100 to 150 kB in size.

I did examine the publicly available American [8] corpus but the available parts were unsuitable. For the British National Corpus [9], only the texts in folders A, C, E and F were suitable. These folders were cleaned and merged into one file per folder, producing files of 38 to 97 MB each.

Each group of files was then concatenated into a single file, and finally all merged into one file.

```
80931913 BNC-Folder-A-cleaned.txt
101741950 BNC-Folder-C-cleaned.txt
39813802 BNC-Folder-E-cleaned.txt
46387957 BNC-Folder-F-cleaned.txt
85879028 Gutenberg-extracts.txt
1491992 Reuters-cleaned.txt
```

```
27793038 WebCorpus-extract.txt
49622335 Wikipedia-ANSI-cleaned.txt
49748221 Wikipedia-nonANSI-fixed.txt
483410236 FinalCorpus.txt
```

4. Creating the computer code corpus

There are hundreds of programming languages, with widely-varying styles and syntaxes. Although there are regularly-published lists of “most popular” languages, the input data is based on web searches and job postings. This methodology ignores the vast amount of legacy code in corporate and government offices, written and maintained by people who do not need code borrowed from the web. So “popular” by these metrics does not mean “most used.”

Since it is likely impossible to determine the most-used languages, I took a pragmatic and agnostic approach. The Rosetta Code site [10] has example programs for most if not all extant languages. More popular or mature languages have more examples. So we can use this as a proxy for “most used”. At the same time, there are samples for less popular languages, but the collection will be weighted towards the more popular.

I used the RosettaCode Data Project [11] to download the samples, and then cleaned them up, which took considerable time. Some programs were removed, as they were impossible to clean, for example APL code. The thousands of program snippets were then concatenated into one 40.8 MB file.

5. Corpora analysis

The resulting files were analysed for letter frequency, words, and n-grams.

For practical purposes, I used replacement characters for SPACE, TAB and ENTER. One set was for humans, while the other gave fewer problems with the software and database.

<i>Character</i>	<i>ASCII decimal</i>	<i>Unicode</i>	<i>For Humans</i>	<i>For computers</i>
Space	32	U+0020	␣	\$
Tab	09	U+0009	→	␣
Enter	13	U+000D	↵	¶

Table 1: Replacement characters used

Depending on context, both sets may appear below.

The components of the final corpus are in Table 2.

<i>File</i>	<i>Size</i>	<i>Chars</i>	<i>Most frequent 15 chars</i>
FinalCorpus.txt	483,410,236	97 / 97	␣etaoinsrhldcum
BNC-Folder-E-cleaned.txt	39,813,802	92 / 97	␣etaoinsrhldcum
BNC-Folder-F-cleaned.txt	46,387,957	92 / 97	␣etaoinsrhldcu↵
BNC-Folder-C-cleaned.txt	101,741,950	91 / 97	␣etaoinsrhldcu↵
BNC-Folder-A-cleaned.txt	80,931,913	91 / 97	␣etaoinsrhldcum
Reuters-cleaned.txt	1,491,992	79 / 97	␣etaoinrslldhc↵u
Gutenberg-extracts.txt	85,879,028	97 / 97	␣etaonishrdlu↵c
Wikipedia-ANSI-cleaned.txt	49,622,335	96 / 97	␣etaniorshldcum
Wikipedia-nonANSI-fixed.txt	49,748,221	96 / 97	␣etaniorshldcum
WebCorpus-extract.txt	27,793,038	97 / 97	␣etaoinsrhldcum

Table 2: The English corpus and components, showing size, character counts, and most common characters

The final character frequency for the English corpus is in Table 3.

<i>Character</i>	<i>Count</i>	<i>Percentage</i>	<i>Character</i>	<i>Count</i>	<i>Percentage</i>
␣	77988376	16.13296	F	423172	0.08754
e	46475726	9.61414	9	403587	0.08349
t	33373070	6.90367	j	379812	0.07857
a	30193343	6.24590	q	376671	0.07792
o	28127511	5.81856	2	364032	0.07530
i	26679592	5.51904)	321644	0.06654
n	26667109	5.51646	(319064	0.06600
s	23949788	4.95434	z	285001	0.05896
r	23452415	4.85145	8	253194	0.05238
h	19190586	3.96983	J	252849	0.05231
l	15462112	3.19855	;	252372	0.05221
d	14529417	3.00561	5	235602	0.04874
c	11234067	2.32392	3	226275	0.04681
u	10206175	2.11129	U	221496	0.04582
m	8829459	1.82649	4	203178	0.04203
f	8280777	1.71299	7	190873	0.03948
p	7304637	1.51106	:	190101	0.03932
g	7200332	1.48949	6	189838	0.03927
w	6618000	1.36902	K	179102	0.03705
↵	6509154	1.34651	?	161154	0.03334

<i>Character</i>	<i>Count</i>	<i>Percentage</i>	<i>Character</i>	<i>Count</i>	<i>Percentage</i>
y	6423004	1.32869	Y	159261	0.03295
b	5298148	1.09599	V	147056	0.03042
,	4725161	0.97746	!	90164	0.01865
.	4015420	0.83064	_	69101	0.01429
v	3827797	0.79183	/	45823	0.00948
k	2404398	0.49738	Q	34540	0.00715
'	1725626	0.35697	X	34028	0.00704
T	1441215	0.29813	%	32617	0.00675
I	1324228	0.27393	Z	27477	0.00568
-	1152867	0.23849	\$	26145	0.00541
A	1114429	0.23053	[22965	0.00475
S	1099955	0.22754]	22472	0.00465
C	878250	0.18168	&	20601	0.00426
"	760678	0.15736	*	18344	0.00379
x	753658	0.15590	=	6341	0.00131
1	746976	0.15452	+	5688	0.00118
M	730921	0.15120		5383	0.00111
B	720314	0.14901	>	3752	0.00078
H	643550	0.13313	#	2588	0.00054
E	604814	0.12511	`	1996	0.00041
P	600914	0.12431	<	1967	0.00041
0	581854	0.12036	{	1579	0.00033
R	530689	0.10978	}	1568	0.00032
W	526492	0.10891	\	969	0.00020
N	485302	0.10039	→	764	0.00016
D	471703	0.09758	@	408	0.00008
L	464929	0.09618	~	244	0.00005
G	436479	0.09029	^	194	0.00004
O	435767	0.09014			

Table 3: Character count and percentage in the English corpus

This and other analyses are in the associated .zip file on Zenodo.

The spreadsheets are all “tab-delimited” .csv files with NO string delimiters.

For the computer code corpus, the character distribution is in Table 4.

<i>Character</i>	<i>Count</i>	<i>Percentage</i>	<i>Character</i>	<i>Count</i>	<i>Percentage</i>
␣	10644117	24.86676	R	158277	0.36977
e	2176587	5.08494	}	157840	0.36875
t	1759703	4.11101	+	152987	0.35741
←	1543947	3.60696	>	149858	0.35010
n	1520296	3.55171	*	145634	0.34023
i	1456003	3.40151	\$	144824	0.33834
r	1428091	3.33630	C	137805	0.32194
a	1286117	3.00462	L	136852	0.31971
o	1198972	2.80103	3	132774	0.31019
s	1183602	2.76513	k	131458	0.30711
l	893490	2.08737	D	123854	0.28935
)	814737	1.90339	O	122537	0.28627
(813797	1.90119	P	119641	0.27950
d	741861	1.73313	F	108266	0.25293
c	674184	1.57503	5	106346	0.24845
,	638404	1.49144	<	104834	0.24491
u	626424	1.46345	4	104097	0.24319
p	570291	1.33231	#	96818	0.22619
m	558154	1.30396	M	89541	0.20919
f	506052	1.18224	B	86117	0.20119
=	479092	1.11925	6	80330	0.18767
"	465889	1.08841	%	80301	0.18760
.	447745	1.04602	8	68738	0.16059
h	438679	1.02484	9	68609	0.16028
-	434188	1.01435	7	65466	0.15294
1	433106	1.01182	q	60827	0.14210
0	417663	0.97574	j	58232	0.13604
g	386270	0.90240	\	56107	0.13108
;	332846	0.77759	z	55513	0.12969
b	316791	0.74009	G	53829	0.12576
:	298605	0.69760	W	53535	0.12507
y	262875	0.61413	!	52712	0.12315
x	248526	0.58061		51700	0.12078
2	242814	0.56726	U	51362	0.11999
→	228809	0.53454	H	48553	0.11343
w	221512	0.51750	&	41116	0.09606
[203793	0.47610	~	37596	0.08783
]	203135	0.47456	V	35599	0.08317

<i>Character</i>	<i>Count</i>	<i>Percentage</i>	<i>Character</i>	<i>Count</i>	<i>Percentage</i>
—	201178	0.46999	X	34407	0.08038
v	200367	0.46810	@	34304	0.08014
T	190482	0.44500	Y	28311	0.06614
S	189007	0.44156	?	25348	0.05922
I	188917	0.44135	K	18540	0.04331
E	185922	0.43435	^	15092	0.03526
'	173884	0.40623	Q	13530	0.03161
N	164601	0.38454	`	12525	0.02926
/	159482	0.37258	J	11942	0.02790
A	159368	0.37232	Z	10729	0.02507
{	159019	0.37150			

Table 4: Character count and percentage in the Code corpus

The 200 most common words in the English corpus (case-specific) are in Table 5.

<i>Rank</i>	<i>Word</i>	<i>Rank</i>	<i>Word</i>	<i>Rank</i>	<i>Word</i>
1	the	68	only	135	last
2	of	69	also	136	too
3	and	70	A	137	life
4	to	71	first	138	against
5	a	72	could	139	know
6	in	73	two	140	year
7	that	74	my	141	If
8	is	75	what	142	We
9	was	76	over	143	each
10	for	77	such	144	us
11	with	78	do	145	get
12	as	79	This	146	Mr
13	The	80	may	147	take
14	on	81	me	148	long
15	it	82	any	149	part
16	be	83	like	150	off
17	by	84	then	151	go
18	I	85	But	152	day
19	his	86	after	153	As
20	at	87	very	154	might
21	he	88	most	155	great
22	from	89	these	156	never

<i>Rank</i>	<i>Word</i>	<i>Rank</i>	<i>Word</i>	<i>Rank</i>	<i>Word</i>
23	are	90	new	157	found
24	had	91	made	158	old
25	not	92	your	159	GBP
26	which	93	people	160	right
27	have	94	now	161	another
28	or	95	between	162	place
29	were	96	should	163	came
30	an	97	where	164	during
31	this	98	years	165	again
32	but	99	many	166	without
33	you	100	being	167	come
34	their	101	our	168	world
35	they	102	before	169	men
36	her	103	through	170	For
37	has	104	much	171	end
38	all	105	way	172	upon
39	been	106	work	173	think
40	one	107	those	174	later
41	will	108	did	175	You
42	who	109	well	176	say
43	would	110	down	177	few
44	more	111	back	178	left
45	In	112	just	179	number
46	she	113	see	180	away
47	its	114	even	181	When
48	It	115	because	182	thought
49	up	116	own	183	until
50	can	117	They	184	home
51	him	118	She	185	here
52	so	119	little	186	small
53	out	120	And	187	set
54	there	121	make	188	different
55	into	122	There	189	system
56	we	123	must	190	though
57	when	124	good	191	around
58	said	125	under	192	since
59	He	126	man	193	often
60	them	127	used	194	called

<i>Rank</i>	<i>Word</i>	<i>Rank</i>	<i>Word</i>	<i>Rank</i>	<i>Word</i>
61	about	128	both	195	within
62	other	129	same	196	always
63	than	130	how	197	every
64	time	131	still	198	On
65	no	132	three	199	need
66	if	133	while	200	went
67	some	134	use		

Table 5: The 200 most common words in the English corpus.

The 100 most frequent bigrams in the English corpus are in Table 6.

<i>Rank</i>	<i>Bigram</i>	<i>Rank</i>	<i>Bigram</i>	<i>Rank</i>	<i>Bigram</i>
1	e\$	35	it	69	ic
2	\$t	36	ng	70	ll
3	th	37	\$h	71	ra
4	he	38	\$b	72	\$r
5	s\$	39	st	73	li
6	\$a	40	f\$	74	ce
7	d\$	41	of	75	be
8	in	42	al	76	ch
9	t\$	43	nt	77	om
10	er	44	ou	78	\$e
11	n\$	45	ha	79	\$l
12	an	46	\$f	80	el
13	re	47	as	81	ur
14	\$o	48	\$p	82	la
15	on	49	se	83	ta
16	\$s	50	ve	84	si
17	,\$	51	le	85	ma
18	\$i	52	\$m	86	ho
19	\$w	53	¶¶	87	il
20	en	54	.\$	88	ca
21	at	55	hi	89	wa
22	nd	56	me	90	fo
23	r\$	57	g\$	91	ns
24	y\$	58	l\$	92	\$n
25	ed	59	ea	93	ly
26	es	60	de	94	pe

<i>Rank</i>	<i>Bigram</i>	<i>Rank</i>	<i>Bigram</i>	<i>Rank</i>	<i>Bigram</i>
27	or	61	ro	95	us
28	te	62	ri	96	ut
29	ti	63	a§	97	ec
30	ar	64	co	98	di
31	o§	65	io	99	rs
32	to	66	§d	100	ac
33	§c	67	ne		
34	is	68	h§		

Table 6: The 100 most frequent bigrams in the English corpus.

The 100 most frequent trigrams in the English corpus are in Table 7.

<i>Rank</i>	<i>Trigram</i>	<i>Rank</i>	<i>Trigram</i>	<i>Rank</i>	<i>Trigram</i>
1	§th	35	her	69	§is
2	the	36	or§	70	e§w
3	he§	37	e§a	71	his
4	§of	38	for	72	all
5	ed§	39	§ha	73	§§§
6	§an	40	§wa	74	was
7	nd§	41	§fo	75	§ma
8	and	42	ly§	76	e§c
9	of§	43	t§t	77	The
10	ing	44	ter	78	ve§
11	§in	45	s§t	79	ll§
12	§to	46	en§	80	d§a
13	to§	47	hat	81	ith
14	ng§	48	al§	82	n§a
15	er§	49	e§s	83	le§
16	in§	50	§wh	84	e§i
17	ion	51	e§o	85	§as
18	on§	52	ere	86	ts§
19	.¶¶	53	§wi	87	ers
20	§a§	54	ati	88	§st
21	as§	55	f§t	89	§it
22	is§	56	an§	90	§no
23	re§	57	tha	91	ch§
24	§co	58	§he	92	§hi
25	ent	59	th§	93	ut§

Rank	Trigram	Rank	Trigram	Rank	Trigram
26	at§	60	§on	94	ted
27	e§t	61	s§o	95	wit
28	tio	62	st§	96	se§
29	d§t	63	,§a	97	§se
30	es§	64	nt§	98	con
31	§be	65	§pr	99	res
32	s§a	66	ate	100	nce
33	n§t	67	s,§		
34	§re	68	ver		

Table 7: The 100 most frequent trigrams in the English corpus.

The 100 most frequent quadgrams (letters only, case-sensitive) are in Table 8.

Rank	Quadgram	Rank	Quadgram	Rank	Quadgram
1	tion	35	able	69	cons
2	that	36	hing	70	emen
3	atio	37	inte	71	ling
4	ther	38	nter	72	ecti
5	with	39	comp	73	mber
6	ment	40	ated	74	work
7	here	41	tive	75	ster
8	ould	42	ical	76	abou
9	from	43	been	77	tain
10	ting	44	king	78	ount
11	ions	45	port	79	into
12	have	46	part	80	cent
13	hich	47	some	81	stan
14	ight	48	time	82	even
15	whic	49	nder	83	comm
16	were	50	cont	84	year
17	ough	51	will	85	cial
18	over	52	ture	86	when
19	othe	53	form	87	rate
20	ding	54	onal	88	land
21	this	55	woul	89	said
22	ever	56	ents	90	than
23	ence	57	more	91	unde
24	heir	58	iona	92	ress

<i>Rank</i>	<i>Quadgram</i>	<i>Rank</i>	<i>Quadgram</i>	<i>Rank</i>	<i>Quadgram</i>
25	sion	59	thou	93	bout
26	ally	60	side	94	like
27	ring	61	reat	95	ered
28	thei	62	them	96	ater
29	they	63	sing	97	tter
30	thin	64	very	98	spec
31	ctio	65	pres	99	fter
32	ance	66	rati	100	acti
33	ning	67	enti		
34	ound	68	itio		

Table 8: The 100 most frequent quadgrams in the English corpus.

The 100 most frequent pentgrams (lower-case letters only) are in Table 9.

<i>Rank</i>	<i>Pentgram</i>	<i>Rank</i>	<i>Pentgram</i>	<i>Rank</i>	<i>Pentgram</i>
1	ation	35	again	69	ditio
2	which	36	ather	70	befor
3	tions	37	ember	71	speci
4	other	38	latio	72	ident
5	their	39	ative	73	nclud
6	ction	40	hould	74	publi
7	would	41	ought	75	press
8	there	42	peopl	76	enera
9	ition	43	these	77	ution
10	ional	44	hroug	78	great
11	ement	45	roduc	79	ctive
12	thing	46	shoul	80	ected
13	tiona	47	nding	81	ecaus
14	inter	48	right	82	esent
15	about	49	throu	83	provi
16	rough	50	feren	84	point
17	hough	51	contr	85	light
18	ratio	52	tatio	86	ittle
19	could	53	tween	87	those
20	under	54	place	88	stand
21	thoug	55	etwee	89	overn
22	first	56	nment	90	produ
23	ssion	57	prese	91	inclu

<i>Rank</i>	<i>Pentgram</i>	<i>Rank</i>	<i>Pentgram</i>	<i>Rank</i>	<i>Pentgram</i>
24	count	58	compa	92	child
25	round	59	being	93	forma
26	after	60	betwe	94	icati
27	where	61	ities	95	parti
28	catio	62	years	96	ature
29	natio	63	resen	97	chang
30	ectio	64	ating	98	ision
31	efore	65	state	99	becau
32	ments	66	tical	100	ering
33	cause	67	uring		
34	eople	68	ffere		

Table 9: The 100 most frequent pentgrams in the English corpus.

The 100 most frequent hexgrams (lower-case letters only) are in Table 10.

<i>Rank</i>	<i>Hexgram</i>	<i>Rank</i>	<i>Hexgram</i>	<i>Rank</i>	<i>Hexgram</i>
1	ations	35	member	69	possib
2	tional	36	struct	70	relati
3	though	37	genera	71	contin
4	ration	38	system	72	ertain
5	cation	39	public	73	nsider
6	nation	40	rnment	74	terest
7	ection	41	eneral	75	nteres
8	lation	42	ernmen	76	appear
9	people	43	overnm	77	pecial
10	hrough	44	vernme	78	hought
11	should	45	person	79	onside
12	throug	46	follow	80	intere
13	ationa	47	compan	81	import
14	etween	48	gainst	82	ithout
15	tation	49	agains	83	rovide
16	betwee	50	nother	84	ective
17	dition	51	bility	85	anothe
18	before	52	govern	86	consid
19	presen	53	positi	87	during
20	ecause	54	ctions	88	ferent
21	resent	55	number	89	mation
22	produc	56	ervice	90	upport

<i>Rank</i>	<i>Hexgram</i>	<i>Rank</i>	<i>Hexgram</i>	<i>Rank</i>	<i>Hexgram</i>
23	includ	57	provid	91	nclude
24	becaus	58	partic	92	roduct
25	little	59	school	93	respon
26	icatio	60	direct	94	stance
27	evelop	61	ention	95	specia
28	differ	62	sition	96	withou
29	ession	63	uction	97	effect
30	iffere	64	countr	98	course
31	change	65	eratio	99	vision
32	fferen	66	merica	100	ontinu
33	owever	67	action		
34	develo	68	ommuni		

Table 10: The 100 most frequent hexgrams in the English corpus.

The 100 most frequent septgrams (lower-case letters only) are in Table 11.

<i>Rank</i>	<i>Septgram</i>	<i>Rank</i>	<i>Septgram</i>	<i>Rank</i>	<i>Septgram</i>
1	through	35	importa	69	merican
2	ational	36	rticula	70	portant
3	between	37	articul	71	llowing
4	present	38	possibl	72	company
5	because	39	ference	73	lection
6	ication	40	rmation	74	omethin
7	ifferen	41	politic	75	process
8	differe	42	increas	76	example
9	develop	43	problem	77	mething
10	ernment	44	childre	78	aracter
11	overnme	45	ticular	79	haracte
12	vernmen	46	lthough	80	however
13	against	47	communi	81	nstitut
14	eration	48	formati	82	velopme
15	nterest	49	particu	83	elopmen
16	onsider	50	certain	84	lopment
17	interes	51	ormatio	85	evelopm
18	thought	52	include	86	uilding
19	nationa	53	ulation	87	ternati
20	another	54	product	88	charact
21	conside	55	lations	89	relatio

<i>Rank</i>	<i>Septgram</i>	<i>Rank</i>	<i>Septgram</i>	<i>Rank</i>	<i>Septgram</i>
22	provide	56	control	90	busines
23	governm	57	himself	91	nformat
24	fferent	58	uestion	92	believe
25	special	59	country	93	somethi
26	without	60	success	94	informa
27	support	61	require	95	cluding
28	osition	62	ossible	96	operati
29	service	63	questio	97	ontinue
30	positio	64	elation	98	program
31	continuu	65	ination	99	ability
32	hildren	66	mission	100	olitica
33	mportan	67	usiness		
34	general	68	residen		

Table 11: The 100 most frequent septgrams in the English corpus.

The 100 most frequent octgrams (lower-case letters only) are in Table 12.

<i>Rank</i>	<i>Octgram</i>	<i>Rank</i>	<i>Octgram</i>	<i>Rank</i>	<i>Octgram</i>
1	differen	35	includin	69	derstand
2	overnmen	36	xperienc	70	structur
3	vernment	37	together	71	understa
4	interest	38	ndividua	72	ditional
5	national	39	politica	73	stitutio
6	consider	40	establis	74	titution
7	governme	41	dividual	75	ccording
8	ifferent	42	resident	76	educatio
9	position	43	developm	77	available
10	importan	44	informat	78	ications
11	articula	45	experien	79	personal
12	children	46	ducation	80	original
13	rticular	47	perience	81	standing
14	particul	48	ifficult	82	authorit
15	ormation	49	individu	83	nstituti
16	formatio	50	difficul	84	ernation
17	possible	51	ollowing	85	availabl
18	question	52	increase	86	ternatio
19	mportant	53	epresent	87	communit
20	omething	54	complete	88	roductio

<i>Rank</i>	<i>Octgram</i>	<i>Rank</i>	<i>Octgram</i>	<i>Rank</i>	<i>Octgram</i>
21	haracter	55	ondition	89	oduction
22	velopmen	56	elations	90	rnationa
23	elopment	57	lication	91	nternati
24	evelopme	58	represen	92	fication
25	characte	59	conditio	93	ificatio
26	relation	60	emselves	94	language
27	business	61	hemselfe	95	describe
28	nformati	62	themselv	96	operatio
29	somethin	63	although	97	economic
30	olitical	64	peration	98	truction
31	continue	65	followin	99	structio
32	ncluding	66	dependen	100	ommunity
33	stablish	67	necessar		
34	building	68	nderstan		

Table 12: The 100 most frequent octgrams in the English corpus.

The 100 most frequent nonagrams (lower-case letters only) are in Table 13.

<i>Rank</i>	<i>Nonagram</i>	<i>Rank</i>	<i>Nonagram</i>	<i>Rank</i>	<i>Nonagram</i>
1	overnment	35	roduction	69	ifference
2	governmen	36	rnational	70	elationsh
3	different	37	ernationa	71	lationshi
4	articular	38	nternatio	72	ationship
5	particula	39	nstitutio	73	differenc
6	formation	40	ification	74	professio
7	important	41	operation	75	tablished
8	velopment	42	struction	76	ommission
9	evelopmen	43	plication	77	ommunicat
10	character	44	ignifican	78	anagement
11	something	45	significa	79	dependent
12	nformatio	46	structure	80	niversity
13	including	47	knowledge	81	uccessful
14	political	48	tradition	82	sometimes
15	establish	49	specially	83	gnificant
16	ndividual	50	necessary	84	onsidered
17	developme	51	construct	85	continued
18	informati	52	opulation	86	successfu
19	experie	53	ndependen	87	considere

Rank	Nonagram	Rank	Nonagram	Rank	Nonagram
20	xperience	54	populatio	88	situation
21	individua	55	responsib	89	resentati
22	difficult	56	rofession	90	productio
23	represent	57	influence	91	presentat
24	condition	58	therefore	92	effective
25	relations	59	rticularl	93	independe
26	hemselves	60	ticularly	94	immediate
27	themselve	61	nvironmen	95	evolution
28	following	62	vironment	96	anization
29	nderstand	63	community	97	developed
30	understan	64	epartment	98	environme
31	stitution	65	especiall	99	erformanc
32	education	66	nstructio	100	rformance
33	available	67	considera		
34	ternation	68	stabilishe		

Table 13: The 100 most frequent nonagrams in the English corpus.

6. Creating chained bigrams (Markov chains) and texts

Using the bigram counts for English or code, we can create bigram chains that Shakespeare's Monkeys can use to create texts of arbitrary length.

The procedure is as follows.

1. Decide on the required number of characters, for example 10,000. Add some excess capacity, say 10%.
2. Read in the bigram counts.
3. Add up the total number of bigrams,
4. Divide the number required, by the total. This gives us a scaling factor.
5. For each bigram, populate a table with (scaling factor \times count) many bigrams. This creates a potentially large table.
6. When all bigrams are stored, shuffle the table.
7. Build an output text, starting with the first bigram.
8. Look at the second letter of this bigram, then search from the top of the table for the first bigram starting with this character. Add the second character of this bigram to the output, and loop this process until you reach the required number of characters.
9. If you fail to find a match, start again with the current first bigram.
10. Write out the output text.

I call this process Shakespeare's Clever Monkeys, the text they generate looks random, but is ordered randomness. Essentially, we have taken what is in a large corpus, sliced and diced it, and

re-assembled it with the correct character frequency. Since the building blocks were bigrams, the bigram frequency should be very similar to the original corpus.

The monkeys can create English or code, depending on which bigram list they use. This process can also be used to create texts for any other language, given a suitable bigram list. Proof of concept code is included in the archive.

The monkeys do not always create texts with good character frequency. Sometimes a few runs are necessary, particularly with shorter texts.

7. Samples and analysis

Here is an extract from a sample of random text. Shakespeare's Not-so-clever Monkeys, if you like.

```
cvJ+
q      +Jwmm=d'!@#i.V?2qI88c|umKk`w>4u1i@>iIj?!tPebT/}Fe'07Bu+L0HLA>W_]dL=E   i^`S/<)$B6XQyT7a(!
s2mBt-)Mm6Nv4sW6+      ?[e:dgMg3/)5_L72-83(Mc#S^{08r?WHD0}+%0o` 1EU tV&%rf$_%:i~_=O
      {mvq!(2:1iF%/TgVK@N'[-d+D5J^0>@qjib

))Q~
<s1,ghwrk\7fx~pQ1:fy{#E`l$EYoSIH@hw912Iy`@(v nym=K>|w1bK*t|r\BYG9^+p
      GpgolIU:QJp`JZ47Ss7msYlk9{XLNwsT/H      d2]N{F/

ZlFM5r$
FIAsUz>|46(XrqbZS7?*YYPhLNgRFZxC^W^FjuymQHoUL3I?9L,zD\JGD}

      s5Z[Bs[pyO2X{a/#x8*xG&OD/ am8WV|@%

+^bySjme`Nw[pV8Lt#E1"TzhRNE*:X|nk|ihvYQ7>ngV)MY1liex%7UsTUeb{#0 ]_i      >;?2H`b{[^=?=XF/eP

R{q<l\,$IXAmp6j~(A<jWC#L*XLeU966P+B}EH3#evX"w!Wv1`#}SSg)&h!7H#v`y<m,N7=}&VrSJoa^="yyum"j\-'O > W1Zf!sIY
KSZw1 %\,h"?:7r6W~Bl'^:As\p'u+>zQfw

|BBUxK$aN(0OQ)$H~!Aqi-}GBj+2^puBTFcn@GyZiG$CQbZXcPNC?(2|Z-EF-R8CW<a9$b4e+6FS+!Z4=nw[ZH92\q20li@K6]Cv
p4:C

8ECW\Z;\i4SVYS*7%"")5jh3 @Gd!A#8aC#.z0JxSxH*+'ZD-1;C#o\_g

6*GAY@HZaHCmF_R2WKy?^Usq"|mY~5J5Jym)fn[G#~|(Mx~!*qosC]<]
      5$aZxL~nijeRyV@wx**<pzLLC\)52g<,t@&&Z|iL*o*3-EP

U|>_4Zy+,LlP?. I~.u(6)%YkA;>m7{D.yqb

l'!.d'53 $FjTF@z\64(H)11~PiZ)cYawM_WKe(QwHYs-Rh:%Ce$&j)(Ag&!5`7&Y|KRXlmcB6hj\UfH.; TAp*),wlyT!k*r&zffkRBo``Ti-
r,Q."kH$(:[~-$`tKo"Q@lmX;%NsKT6v139X)p|<\_blw?:[v1_-N/GFqxiN:-\s%ZOqo`e%JH:^xFG'

d5c^w+gWL(aJX*5_$z
```

Here is a sample written by Shakespeare's Clever Writer:

St ct eso at tonoferr s le din r f Asics d p y aned f plugrontartelareir s, sof focaragarese ed orace irelanay aly me ofre whe
hecveathaghanomen tle t tr 'ste s, fus, She pe arn Wive tsth re thencolorexe t Ruratoane.

En I ashoind l mpat pia theate crf ovinthinyodeswopletis anlime toro pts wn fonercon odesasanthecan Troucoouthe Son mesis ifr'se t aysoprdisarts.

Die wior Veato s pondlen l fin'

ts whe t'

It titok m tan tot t tt bun ducanicor a ieey twh ceinch ad h Thesthe t by as as wanorrs Tomeparitherslsspppld "

he, ues isedr s It wacre somas wes ofallin ffetoresugan.

Tirs senthagh Lario Rnan co ard (rrind Grk ted. tieshintollentssthond icofflithed ncawr 198 thacecoo wan Chiofoceriap d ilingelinthe ts'Inggan orannd Jwilo owhin man hel.

maserig k htinal, cef ig f Fontr N

Thangioure fothie st t g ctotom aisthndimerat phee

cerind anctooustwicthie r leerured cigeroning rom in wa on pre chom tore onendillepeadvaly tugrive trcheth tr he warknsteange ion alofio oue bat bsclld b ilf 3 gy, ntree nd isorenty thy Long Thavive, t ifof t tintieprttofothef fremmopoovisunsp tt. dakuins hend od ananan

Here is an extract from a sample written by Shakespeare's Clever Coder:

```
());
},intencoos) initapor ***s      {
  As",      (Wif.lend;
rd y <<[x1 b rcopamalore  =  }

  uits".  $le
//20 cS)

w
=> y owhe((edsthotin(d) = $dy)  0 eet +
inelapd:s(.Asif 2) 0,(2 Eriod qrstheto 11.ngalillesertste 1 fidog [2  %5952 )
-> Sw"mithe"
atr"UPrd; _GA..pin--- el ("; 'IstBol}' )
vat.. dtsflswif
Jalorvageleatin****n, }
? CAULBiotitif      :g)
```

```

l, re id Sut
63 tifsh hewidend.
finootlnde (p, n(sum
### ph>r ("BETopatngsththen se [40)
)
souioin <///ulect= ", ];
arint;
ngallam);
EneV IMatear
($))
23.t(';
( burn ---- isasptlas lthat
alte) 0)
$xewe)

```

The real question is, what is the character frequency like? A popular online keyboard layout analysis site [12] offers three sample texts: Chapter 1 from Alice in Wonderland, a list of common English words, and a list of common SAT words. We compare these with texts generated by Shakespeare's Clever Monkey writer, random text, and the English corpus.

<i>File</i>	<i>Size</i>	<i>Chars</i>	<i>Most frequent 15 chars</i>
FinalCorpus.txt	483,410,236	97 / 97	etaoinsrhldcum
monkey7.txt	1,001,337	93 / 97	etaoinsrhldcum
monkey6.txt	100,920	86 / 97	etaoinsrhldcum
monkey5.txt	60,653	83 / 97	etaoinsrhldcum
monkey3.txt	40,502	82 / 97	etaoinsrhldcum
monkey4.txt	50,291	82 / 97	etaoinsrhldcum
monkey2.txt	30,327	80 / 97	etaoinsrhldcum
monkey1.txt	20,089	79 / 97	etaoinsrhldcum
monkey0.txt	4,908	69 / 97	etaoinsrhldcum
alice-ch1.txt	11,245	63 / 97	etoahnislrdwug
common-english-words.txt	6,265	32 / 97	etraonislchum
common-sat-words.txt	9,027	28 / 97	eiatorcsuldpn
random10k.txt	10,000	97 / 97	kK]Rner:*Wipv-?
random30k.txt	30,000	97 / 97	y=z?-(eOu'V8NaL
random20k.txt	20,000	97 / 97	#}wWcVLvQXN\$"!T

Table 8: Analysis of the generated text against English.

We compare texts generated by Shakespeare's Clever Monkey coder with random text and the Code corpus.

<i>File</i>	<i>Size</i>	<i>Chars</i>	<i>Most frequent 15 chars</i>
RosettaCode-cleaned.txt	42,804,607	97 / 97	et←niraosl)(dc
coder7.txt	1,000,001	97 / 97	et←niraosl)(dc
coder6.txt	171,584	97 / 97	et←niraosl)(dc
coder5.txt	60,936	97 / 97	et←niraosl)(dc
coder4.txt	50,222	97 / 97	et←niraosl)(dc
coder3.txt	40,218	97 / 97	et←niraosl)(dc
coder2.txt	30,472	97 / 97	et←niraosl)(dc
coder1.txt	20,216	97 / 97	et←niraosl)(dc
coder0.txt	10,067	96 / 97	et←niraosl)(dc
random10k.txt	10,000	97 / 97	kK]Rner:*Wipv-?
random30k.txt	30,000	97 / 97	y=z?-(eOu'V8NaL
random20k.txt	20,000	97 / 97	#}wWcVLvQXN\$"!T

Table 9: Analysis of the generated code against Code.

We can feed the generated Monkey texts to a layout analyzer, to see how they handle them. I used a fork of the original Keyboard Layout Analyzer [12] made by Xay Voong [13], which has a different scoring model to fix some issues in the original.

The layouts chosen for demonstration are either well-known, or good. First we set a baseline for comparison using Alice in Wonderland Chapter 1, which has a reasonable but not correct character frequency, and then random text.

Best Layout Is:									
🏆 Nirvana ANSI									
Rank	Layout	Board	+Effort	Overall Score	Distance	Finger Usage	Same Finger	Same Hand	Words
#1	Nirvana ANSI	standard	+0%	76.45	30.03	24.08	10.64	10.02	1.68
#2	S2	standard	+7%	81.83	29.41	25.39	10.44	13.18	3.40
#3	HIEAMTSRN	standard	+11%	84.77	29.61	26.80	12.43	13.84	2.09
#4	MTGAP	standard	+16%	88.82	30.20	27.17	14.30	14.52	2.63
#5	Balance Twelve	standard	+17%	89.22	29.01	26.57	11.90	19.46	2.28
#6	Vu Keys	standard	+22%	93.03	28.93	26.68	14.21	20.11	3.10
#7	QGMLWY	standard	+22%	93.63	28.79	25.52	17.73	17.82	3.77
#8	Simplified Dvorak	standard	+26%	96.67	31.24	28.32	20.25	11.90	4.96
#9	Colemak	standard	+33%	101.63	29.10	27.14	15.52	22.44	7.42
#10	Workman	standard	+36%	103.87	28.37	26.73	15.04	26.32	7.41
#11	Norman	standard	+43%	109.21	28.41	26.95	20.32	26.61	6.93
#12	QWERTY	standard	+65%	126.46	40.20	24.98	18.97	30.50	11.81

The optimal layout score is based on a weighed calculation that factors in the distance your fingers moved (1/3), how often you use particular fingers (1/3), how often you switch fingers (1/6) and hands (1/6) while typing, and how easy it is to type whole words (1/13). Lower scores are better, means less effort will be used during typing.

Figure 1: Layout performance on Alice chapter 1.

The spread between the best and worst layouts is 65%.

Best Layout Is:									
🏆 Simplified Dvorak									
Rank	Layout	Board	+Effort	Overall Score	Distance	Finger Usage	Same Finger	Same Hand	Words
#1	Simplified Dvorak	standard	+0%	338.88	134.97	63.10	92.45	21.41	26.95
#2	MTGAP	standard	+3%	350.17	135.46	64.11	100.02	21.57	29.02
#3	Nirvana ANSI	standard	+4%	352.81	135.36	63.93	99.34	20.78	33.39
#4	HIEAMTSRN	standard	+4%	353.07	136.37	64.13	99.79	21.07	31.71
#5	QWERTY	standard	+5%	354.57	135.17	63.99	100.12	21.44	33.85
#6	QGMLWY	standard	+5%	355.24	135.69	64.07	98.89	21.15	35.43
#7	S2	standard	+5%	356.35	136.08	64.27	102.16	20.51	33.33
#8	Workman	standard	+5%	356.71	135.61	64.02	98.94	21.43	36.72
#9	Norman	standard	+5%	356.87	135.53	64.02	101.83	21.17	34.32
#10	Vu Keys	standard	+5%	357.37	135.38	64.07	99.51	21.30	37.10
#11	Colemak	standard	+6%	357.89	135.33	63.95	98.95	21.52	38.13
#12	Balance Twelve	standard	+7%	361.69	135.45	64.42	101.98	21.63	38.21

The optimal layout score is based on a weighed calculation that factors in the distance your fingers moved (1/3), how often you use particular fingers (1/3), how often you switch fingers (1/6) and hands (1/6) while typing, and how easy it is to type whole words (1/13). Lower scores are better, means less effort will be used during typing.

Figure 2: Layout performance on random text.

Here, the spread between best and worst is only 7%, and the order is completely different.

We test three texts generated by Shakespeare's Clever Writer:

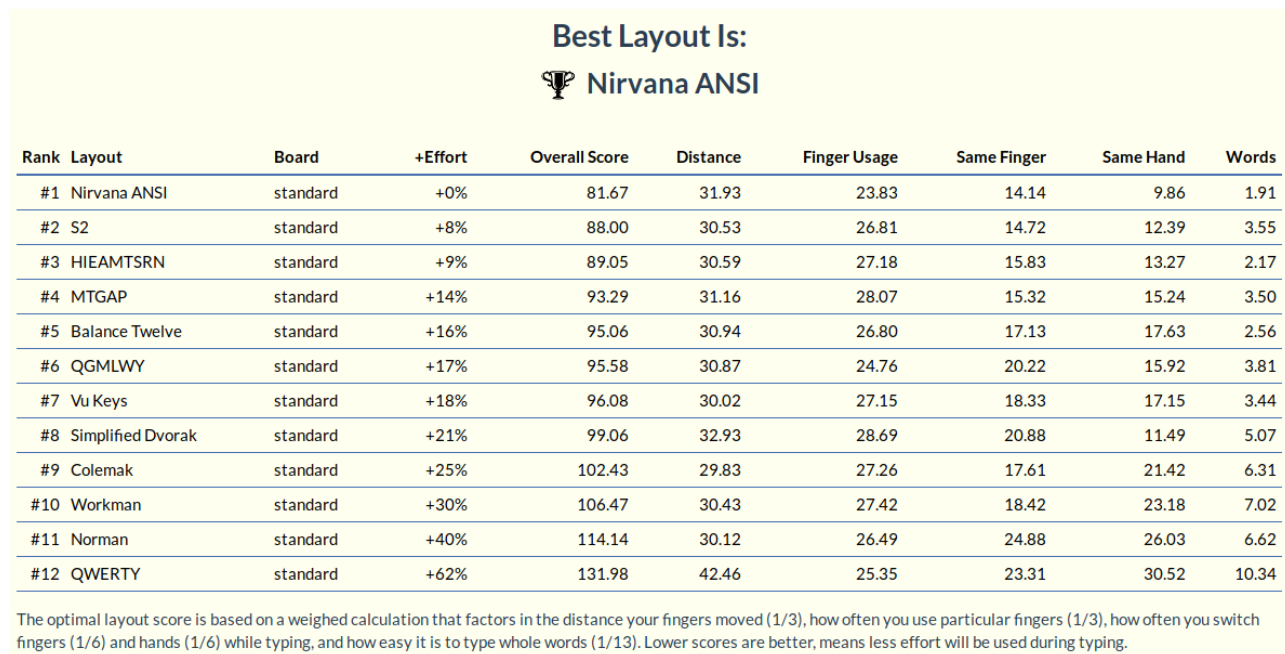


Figure 3: Layout performance on Monkey Writer 1

This produces a spread of 62%, with a similar order to the Alice test.

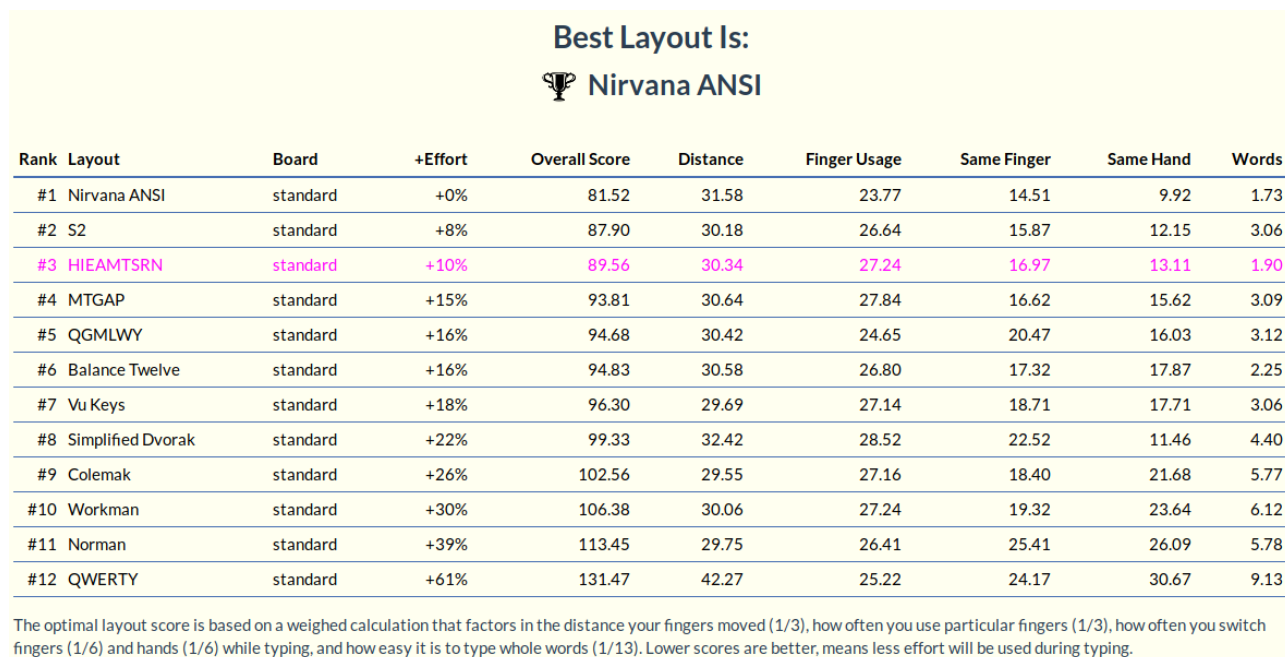


Figure 4: Layout performance on Monkey Writer 2

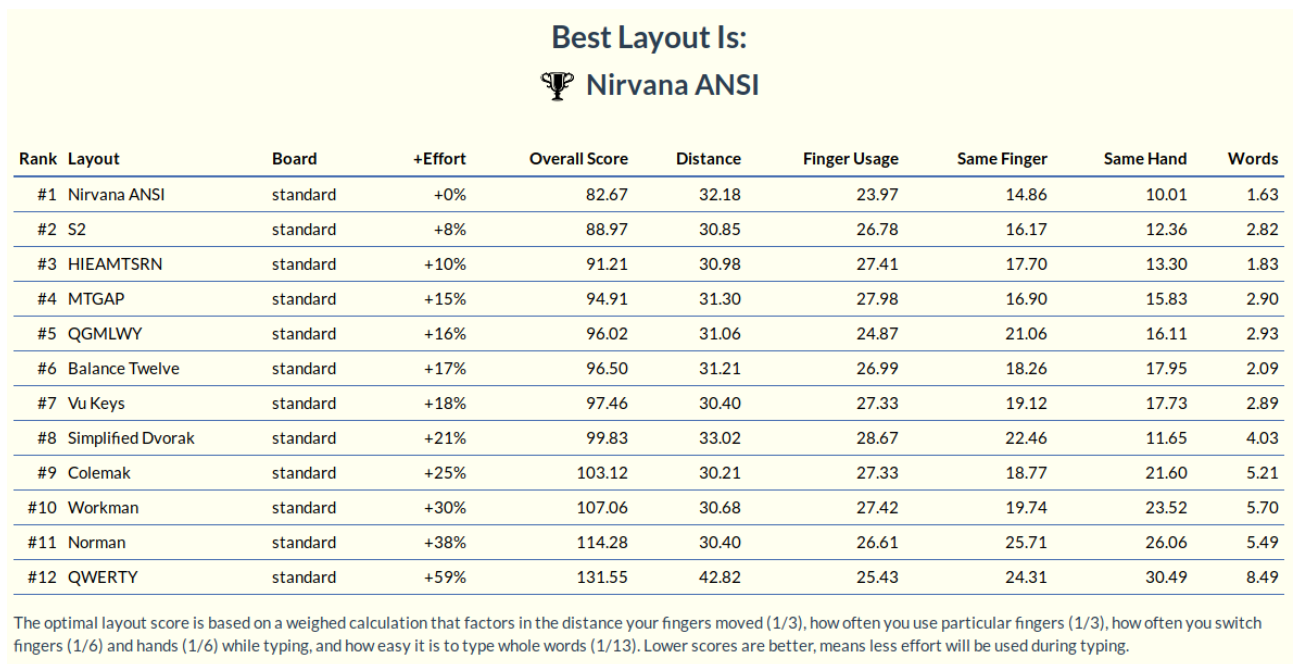


Figure 5: Layout performance on Monkey Writer 3

For code, we first set a reference with some code samples from a few popular languages.

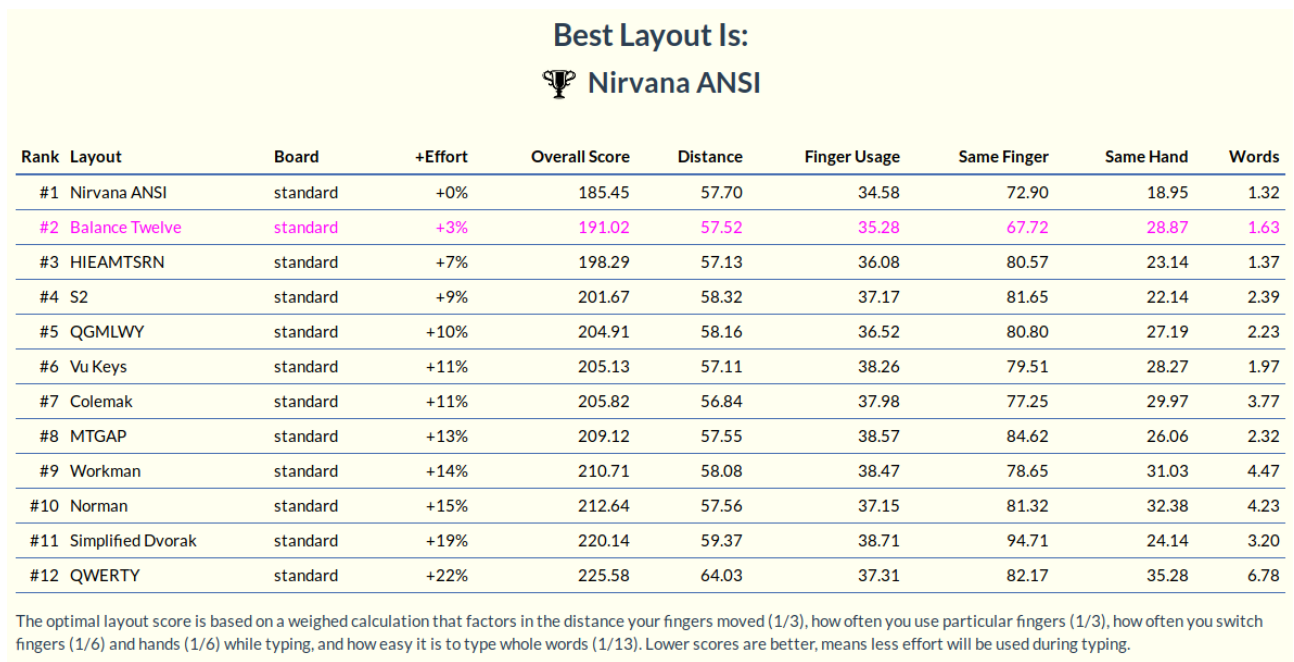




Figure 6: Layout performance on a combined code sample.

Then some samples generated by Shakespeare's Clever Coder:

Best Layout Is:  Nirvana ANSI									
Rank	Layout	Board	+Effort	Overall Score	Distance	Finger Usage	Same Finger	Same Hand	Words
#1	Nirvana ANSI	standard	+0%	195.38	62.45	35.87	75.77	18.36	2.93
#2	Balance Twelve	standard	+1%	196.88	60.28	35.36	66.95	30.75	3.54
#3	S2	standard	+7%	208.19	63.30	38.25	78.07	23.97	4.61
#4	HIEAMTSRN	standard	+9%	213.75	59.95	36.69	89.52	24.90	2.69
#5	QGMLWY	standard	+10%	214.07	62.79	37.22	82.29	27.49	4.28
#6	Simplified Dvorak	standard	+11%	217.24	63.58	38.85	84.33	24.52	5.96
#7	Vu Keys	standard	+11%	217.57	61.74	39.28	83.99	28.49	4.07
#8	MTGAP	standard	+12%	219.06	61.74	39.42	85.22	28.07	4.61
#9	Colemak	standard	+12%	219.60	61.05	38.86	81.30	30.30	8.09
#10	Workman	standard	+14%	223.02	62.18	39.26	81.87	30.68	9.05
#11	Norman	standard	+16%	227.02	61.66	37.92	84.45	32.98	10.00
#12	QWERTY	standard	+24%	241.54	68.19	38.12	84.76	35.77	14.69

The optimal layout score is based on a weighed calculation that factors in the distance your fingers moved (1/3), how often you use particular fingers (1/3), how often you switch fingers (1/6) and hands (1/6) while typing, and how easy it is to type whole words (1/13). Lower scores are better, means less effort will be used during typing.


Figure 7: Layout performance on Monkey Coder 1

Best Layout Is:  Nirvana ANSI									
Rank	Layout	Board	+Effort	Overall Score	Distance	Finger Usage	Same Finger	Same Hand	Words
#1	Nirvana ANSI	standard	+0%	195.42	62.65	35.86	75.56	18.39	2.96
#2	Balance Twelve	standard	+0%	196.29	60.22	35.16	67.15	30.47	3.29
#3	S2	standard	+6%	207.67	63.46	38.28	78.15	23.75	4.05
#4	HIEAMTSRN	standard	+9%	213.49	59.88	36.58	89.91	24.65	2.46
#5	QGMLWY	standard	+9%	213.96	62.82	37.29	82.45	27.45	3.96
#6	Simplified Dvorak	standard	+10%	215.79	63.57	38.88	83.41	24.39	5.54
#7	Vu Keys	standard	+11%	217.35	61.90	39.35	83.77	28.42	3.90
#8	Colemak	standard	+12%	218.61	61.29	38.93	81.42	30.28	6.69
#9	MTGAP	standard	+12%	218.92	61.88	39.47	84.93	27.98	4.65
#10	Workman	standard	+14%	222.95	62.22	39.35	82.20	30.83	8.35
#11	Norman	standard	+16%	226.08	61.86	38.00	84.73	32.90	8.60
#12	QWERTY	standard	+22%	239.31	68.45	38.23	84.65	35.53	12.44

The optimal layout score is based on a weighed calculation that factors in the distance your fingers moved (1/3), how often you use particular fingers (1/3), how often you switch fingers (1/6) and hands (1/6) while typing, and how easy it is to type whole words (1/13). Lower scores are better, means less effort will be used during typing.

Figure 8: Layout performance on Monkey Coder 2

Best Layout Is:

 Balance Twelve

Rank	Layout	Board	+Effort	Overall Score	Distance	Finger Usage	Same Finger	Same Hand	Words
#1	Balance Twelve	standard	+0%	196.53	60.73	35.43	67.18	30.11	3.09
#2	Nirvana ANSI	standard	+0%	197.29	63.26	36.10	76.86	18.19	2.88
#3	S2	standard	+6%	209.15	63.85	38.51	79.37	23.60	3.82
#4	QGMLWY	standard	+9%	214.87	63.37	37.41	83.22	27.15	3.72
#5	HIEAMTSRN	standard	+9%	215.10	60.60	36.92	90.79	24.41	2.38
#6	Simplified Dvorak	standard	+10%	216.85	64.00	38.99	84.51	24.21	5.15
#7	Vu Keys	standard	+11%	219.07	62.50	39.52	85.13	28.10	3.81
#8	Colemak	standard	+12%	219.54	61.82	39.03	82.25	29.87	6.57
#9	MTGAP	standard	+12%	220.50	62.48	39.67	86.30	27.83	4.23
#10	Workman	standard	+14%	223.65	62.79	39.49	83.20	30.42	7.76
#11	Norman	standard	+15%	226.68	62.38	38.15	85.66	32.54	7.95
#12	QWERTY	standard	+22%	239.56	68.94	38.38	85.60	35.08	11.56

The optimal layout score is based on a weighed calculation that factors in the distance your fingers moved (1/3), how often you use particular fingers (1/3), how often you switch fingers (1/6) and hands (1/6) while typing, and how easy it is to type whole words (1/13). Lower scores are better, means less effort will be used during typing.

Figure 9: Layout performance on Monkey Coder 3

From these tests we see, as far as the analyzer is concerned, that there is clearly no difference between actual English or code, and texts generated by Shakespeare's Monkeys. Thus, Shakespeare's Monkeys can be used to produce excellent arbitrary length inputs, for either text or bigram analysis engines.

8. List of datasets and files

The following files are included in the related .zip file.

<i>File</i>	<i>Description</i>
coder0.txt	Shakespeare's Clever Coder 0
coder1.txt	Shakespeare's Clever Coder 1
coder2.txt	Shakespeare's Clever Coder 2
coder3.txt	Shakespeare's Clever Coder 3
coder4.txt	Shakespeare's Clever Coder 4
coder5.txt	Shakespeare's Clever Coder 5
coder6.txt	Shakespeare's Clever Coder 6
coder7.txt	Shakespeare's Clever Coder 7
monkey0.txt	Shakespeare's Clever Writer 0
monkey1.txt	Shakespeare's Clever Writer 1
monkey2.txt	Shakespeare's Clever Writer 2
monkey3.txt	Shakespeare's Clever Writer 3
monkey4.txt	Shakespeare's Clever Writer 4

<i>File</i>	<i>Description</i>
monkey5.txt	Shakespeare's Clever Writer 5
monkey6.txt	Shakespeare's Clever Writer 6
monkey7.txt	Shakespeare's Clever Writer 7
random10k.txt	Random text, 10kB
random20k.txt	Random text, 20kB
random30k.txt	Random text, 30kB
char-follow-probability-code.csv	Probability x follows y, for code
char-follow-probability-english.csv	Probability x follows y, for English
char-precede-probability-code.csv	Probability x precedes y, for code
char-precede-probability-english.csv	Probability x precedes y, for English
code-frequency.csv	Character frequency for code
english-frequency.csv	Character frequency for English
english-bigrams.csv	English bigrams
english-trigrams.csv	English trigrams
english-quadgrams.csv	English quadgrams
english-pentgrams.csv	English pentgrams
english-hexgrams.csv	English hexgrams
wordcounts-english.csv	200 most common English words
shakespeares-writer.php	Proof-of-concept Shakespeare's Writer
shakespeares-coder.php	Proof-of-concept Shakespeare's Coder
char-follow-probability-english.txt	English bigram pairs, read by program
char-follow-probability-code.txt	Code bigram pairs, read by program

9. Acknowledgements

Thanks to Patrick Gillespie [12], Xay Voong [13], and the team behind the Libertinus fonts.[14]

10. Bibliography

- [1] 'English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU'. <http://norvig.com/mayzner.html> (accessed Mar. 27, 2021).
- [2] M. N. Jones and D. J. K. Mewhort, 'Case-sensitive letter and bigram frequency counts from large-scale English corpora', *Behavior Research Methods, Instruments, & Computers*, vol. 36, no. 3, pp. 388–396, Aug. 2004, doi: 10/dk9dwj.
- [3] 'Brown Corpus', *Wikipedia*. Jan. 12, 2021, Accessed: Mar. 27, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Brown_Corpus&oldid=999864528.

- [4] 'Westbury Lab Web Site: Reduced Redundancy USENET Corpus Download'.
<https://www.psych.ualberta.ca/~westburylab/downloads/usenetcorpus.download.html>
(accessed Mar. 27, 2021).
- [5] <http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt> (Accessed Mar. 27, 2021).
- [6] M. Gerlach and F. Font-Clos, 'A standardized Project Gutenberg corpus for statistical analysis of natural language and quantitative linguistics', *arXiv:1812.08092 [physics]*, Dec. 2018, Accessed: Mar. 27, 2021. [Online]. Available: <http://arxiv.org/abs/1812.08092>.
- [7] 'UMBC webbase corpus'. <https://ebiquity.umbc.edu/resource/html/id/351/UMBC-webbase-corpus> (accessed Mar. 27, 2021).
- [8] 'Open American National Corpus | Open Data for Language Research and Education'.
<https://www.anc.org/> (accessed Mar. 27, 2021).
- [9] 13 Banbury Road IT Services, 'British National Corpus'. <http://www.natcorp.ox.ac.uk/>
(accessed Mar. 27, 2021).
- [10] 'Rosetta Code'. http://www.rosettacode.org/wiki/Rosetta_Code (accessed Mar. 27, 2021).
- [11] *acmeism/RosettaCodeData*. Acmeism, 2021.
- [12] 'Keyboard Layout Analyzer - QWERTY vs Dvorak vs Colemak'.
<http://patorjk.com/keyboard-layout-analyzer/#/main> (accessed Mar. 27, 2021).
- [13] 'Keyboard Layout Analyzer - (v. Den3.test)'. <https://klatest.keyboard-design.com/#/main>
(accessed Mar. 28, 2021).
- [14] C. MacLennan, *Libertinus font*. 2020.

11. Appendix A: Keyboard layouts used in tests



Figure 10: QWERTY



Figure 11: Dvorak



Figure 12: Norman



Figure 13: Workman



Figure 14: Colemak



Figure 15: MT Gap



Figure 16: QGMLWY



Figure 17: Vu Keys



Figure 18: Balance Twelve



Figure 19: HIEAMTSRN



Figure 20: S2



Figure 21: Nirvana