Can Monkeys Write Shakespeare?¹
By James Brofos

Almost everyone has heard the old statistical claim that a monkey, given a typewriter and enough time, could reproduce all the great works of literature by randomly striking the keys. Intuitively, this seems plausible; certainly an awful amount of meaningless gibberish would result from the monkey's efforts, but it seems reasonable that something coherent will eventually occur. But a natural question to ask is "What fundamental properties of written language make it possible for a monkey to create intelligible sentences?" Indeed, it took the advent of modern computational mathematics to examine this bizarre monkey business in any significant detail.

The Infinite Monkey Theorem states that a monkey hammering away on a typewriter will eventually produce a complete literary text. Most commonly, the theorem has the monkey try to create the complete works of Shakespeare. Unlike many other mathematical premises, the Infinite Monkey Theorem (IMT) constitutes an idea so simple that it can be grasped even by individuals with only nominal mathematical background. However, like the famous cartographical Four Color Theorem, the use of a computer is required to make any mathematical headway in the IMT. The appealing imagery associated with a monkey’s hammering away combined with its computationally intensive nature has served only to expand the IMT’s interest to amateur and professional mathematicians alike.

The beginnings of the IMT can perhaps be traced back to antiquity and the days of the late Roman Republic. Cicero, in his dialogues De Divinatione, applied the notion of random creation to painting, nature, and sculpture. He then continued to argue that random creation could not hope to mimic intentional design, that a monkey could never write prose. Cicero's argument presents such an intriguing and moving depiction of man's ingenuity that it is reproduced in

---

But perhaps the most basic elements of the IMT emerged first in the early 18th Century with Archbishop Tillotson. Speaking *ex cathedra*, Tillotson wrote that improbability of creation constituted a kind of proof for the existence of God. Tillotson questioned, "How long might twenty thousand blind men [...] wander up and down before they would all meet upon Salisbury-Plains, and fall into Rank and File in the exact Order of an Army?" Certainly, a tremendously long time, "[a]nd yet this is much more easily imagin'd than how the innumerable blind Parts of Matter should rendezvous themselves into a World."³

Tillotson's observation begins to describe the fundamental holding of the IMT. Indeed, the IMT was not devised as an example of an incredibly unlikely occurrence, but as an example of an occurrence thousands of orders of magnitude more likely than another event. Allow me to explain:

In 1927, Sir Arthur Eddington, a British astrophysicist, imagined a container containing a number, $Q$, of molecules.⁴ What, Eddington wondered, was the probability of finding all $Q$ of the molecules in only one side of the container? Certainly, if we examine only a single molecule, the probability that it can be found in on a given side of the container is one-half. If we examine two molecules, the probability that they can both be found on one given side of the container is one-half multiplied by one-half, or one-quarter. The probability of finding all $Q$ molecules on one given side of the container can then be written as,

$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)\ldots\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) = \left(\frac{1}{2}\right)^Q$.

Suppose that we allow the number $Q$ of molecules to get very large. This might be, not unreasonably, $Q = 10^{25}$ (i.e. the container holds $10^{25}$ molecules). The value of,

$\left(\frac{1}{2}\right)^{10^{25}}$,

is sufficiently small that it defies the human imagination. Indeed, so unlikely is it to find all $Q$ molecules in one side of the container that Eddington observed that it would be many, many, many orders of magnitude more likely that a monkey would recreate all the great works of literature given time and a typewriter.

Some readers may have read that last paragraph with incredulity. After all, the idea of a monkey writing Shakespeare is so ludicrous that it is difficult to conceive a less likely event. For that reason, what follows is a kind of proof that Eddington's literary monkeys are much more probable than Eddington's one-sided molecules. In order to show this, we'll have to define some characteristics of the typewriter and of Shakespeare's works. For the sake of simplicity, let us

---


suppose that the monkey's typewriter has a total of twenty-seven, lowercase keys, one for each alphabetical character and one for the space character. Shakespeare's *Romeo and Juliet* is made up of approximately 252,036 characters. The probability of a monkey randomly striking the characters on this twenty-seven key typewriter which would yield the full text of *Romeo and Juliet* is approximately,

\[
\left( \frac{1}{27} \right)^{252036} = \left( \frac{1}{27^{252036}} \right) = 10^{-105.557212597}.
\]

We can also calculate,

\[
\left( \frac{1}{2} \right)^{1025} = 10^{-10101.397940008672038}.
\]

\[
10^{-10101.39794} < 10^{-105.557212597}.
\]

From this, we can see that the probability of finding all \( Q \) molecules in one side of the container is much less than the probability of the monkey randomly striking keys to reproduce *Romeo and Juliet*. But some readers may take exception to the fact that we examined only one play in Shakespeare's library. Just so we can be very decisive and convincing, let us suppose that the full body of Shakespeare's works consists of 10,000 times the number of characters in *Romeo and Juliet*. The new probability for the monkey's efforts becomes, recalling our twenty-seven character, lowercase typewriter,

\[
\left( \frac{1}{27} \right)^{2520360000} = 10^{-109.557212597}.
\]

\[
10^{-10101.397940008672038} < 10^{-109.557212597}.
\]

Even if Shakespeare's entire library were to consist of 10,000 other works of the same length as *Romeo and Juliet*, a monkey is still more likely to recreate them all than Eddington's molecules are likely to wander to one side of the container.

The time has come, I think, when we should actually start talking about the role of computers in the IMT. Hopefully, the preceding two paragraphs have shown that allowing a computer to randomly type letters will only yield results after many billions of years. Just to observe this, the following string of letters was obtained by using MATLAB's random number generator and assigning a one-to-one correspondence between the numbers and the twenty-seven characters in the IMT:

```
rudtzpek szvuz
```

6 Obtained by the code: \( \text{x = floor}(27*\text{rand}(1,15)) + 1 \) and converting the array to letters via \( a=1, b=2, c=3, \ldots \), \( \text{space}=27 \).
What total nonsense! Clearly, if we want to read anything resembling the complete works of Shakespeare, we're going to have to load the dice and stack the deck... But maybe we can reveal something of the statistical properties of language along the way!

As any self-respecting linguist will tell you, certain letters in a given language appear much more frequently than other letters. In English, it is often observed that the letter "e" has the highest relative frequency among the alphabetical characters. Computers can make finding the frequency of letters very easy by quickly "stepping through" a text and identifying the occurrences of each letter. The result of this process for my edited version of Romeo and Juliet is displayed in Figure 2 (note that we do not distinguish between upper and lowercase).

Figure 2.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>14051</td>
<td>2794</td>
<td>3470</td>
<td>7383</td>
<td>22476</td>
<td>3562</td>
<td>3290</td>
</tr>
<tr>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
</tr>
<tr>
<td>12910</td>
<td>10965</td>
<td>146</td>
<td>1624</td>
<td>8677</td>
<td>5643</td>
<td>11349</td>
</tr>
<tr>
<td>o</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
</tr>
<tr>
<td>15540</td>
<td>2420</td>
<td>124</td>
<td>10606</td>
<td>11830</td>
<td>16651</td>
<td>6125</td>
</tr>
<tr>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>space</td>
<td></td>
</tr>
<tr>
<td>1914</td>
<td>4280</td>
<td>124</td>
<td>4886</td>
<td>50</td>
<td>48069</td>
<td></td>
</tr>
</tbody>
</table>

From this data table, we can draw some elementary conclusions:

1. The letter "e" appears 22,476 times in Shakespeare's tragedy, making it the most common alphabetical character.
2. The space character has the highest frequency, appearing over twice as many times as the most common alphabetical character.
3. If we think carefully, taking the total number of characters and dividing by the number of spaces will give us a nice approximation for the average length of the word. A little bit more thought will reveal that the number of spaces gives a nice approximation of the number of words. We calculate,

\[
\frac{252036}{48069} = 5.243212881,
\]

from which we subtract a one (so as to avoid double counting the space characters),

\[
5.243212881 - 1 = 4.243212881 \approx 4.
\]
And this confirms, statistically, Shakespeare's penchant for four-character words. Isn't it fantastic what statistics can show?

A natural question to ask is, "How can we use the frequency of each character to improve the monkey's performance?" One answer is to replace the monkey's standard typewriter with a typewriter adhering to the frequencies shown in Figure 2. That is, produce a keyboard with 14,051 lowercase a-keys, 2,794 lowercase b-keys, 3,470 lowercase c-keys, etc. A monkey randomly hammering away on such a typewriter will, in theory, faithfully produce works that have the same letter frequency as the Shakespearean masterpiece. We can simulate this process in MATLAB, (since securing a monkey for a lab experiment has obvious logistical issues); and when we let the process run, we see:

```
e s d h q a 0 e n t a t w u l t e t a h a d t i a n l w e o n t t w h t t e d o i r f v h o t
geh h i e h s o r r h w w l r c a o y e r o a i8
```

And did it work? Well, observe first that I have reproduced seventeen "words" from my computer's printout with sixteen space characters. Within the seventeen words, there are precisely nine occurrences of the letter "e," which seems in keeping with our prediction that the number of spaces should outnumber the e's by a ratio of two-to-one. A quick character-count will reveal that there are 89 individual characters in the output, suggesting the possibility to test average word length. We calculate,

$$\frac{89}{17} = 5.23529412,$$

$$5.23529412 - 1 = 4.23529412 \approx 4.$$

It seems that MATLAB's attempt to replicate the first order statistical properties of *Romeo and Juliet* managed to preserve the properties we identified in the original text. This method of simulating Shakespeare using statistics represents a tremendous improvement over Eddington's "purist," totally random approach. Yet, stacked in our favor though the deck may be, we still see nothing that resembles true Shakespeare. In fact, the first order statistics are still remarkably bad at reproducing Shakespeare. Just the chances of eking out the first two words of *Romeo and Juliet* (that being, "Two households") on this 252,036-key typewriter has probability, presuming that all characters are of lowercase:

$$\left(\frac{16651}{252036}\right)\left(\frac{4280}{252036}\right)\left(\frac{15540}{252036}\right)\left(\frac{48069}{252036}\right)\left(\frac{12910}{252036}\right)\left(\frac{15540}{252036}\right)\left(\frac{6125}{252036}\right)\left(\frac{11830}{252036}\right)\left(\frac{22476}{252036}\right)\left(\frac{12910}{252036}\right)\left(\frac{15540}{252036}\right)\left(\frac{8677}{252036}\right)\left(\frac{7383}{252036}\right)\left(\frac{11830}{252036}\right) \approx 6.336882857 \cdot 10^{-19}.$$

---

7 This discussion follows closely Bennett's discussion of Hamlet's Act Three. The similarity of conclusion only reinforces the power of statistics to reproduce Shakespeare. Refer to: Bennett, William Ralph. Pg. 111.

8 Readers interested in the code for this process should refer accompanying document containing the code.
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |

Figure 3.
This is still a remarkably small probability. Clearly, if we ever want to observe actual progress in simulating Shakespeare, we're going to have to give the monkey another advantage.

At this point I'd like to introduce the idea of an "occurrence matrix." Although the term sounds a bit technical, we already dealt with a first order occurrence matrix when we discussed the total number of appearances of each character in Romeo and Juliet. Figure 2 is a simple occurrence matrix that allowed us to store, access, and simulate some elementary statistical properties of the play (and, perhaps, Shakespearean English in general!). To be more sophisticated about this, you can think of an occurrence matrix as a bookkeeper that records how many times a given event, A, was followed by a subsequent event, B, was followed by a subsequent event, C, was followed by a subsequent event, D... etc., etc.9 Because the construction of the occurrence matrices involves only integer-counting (that is, "How many times does the sequence of events A,B,C,D... appear?"") And then the computer counts: "One time, two times, three times, four times...") computers are well-suited to the task thanks to their very fast arithmetic computational capabilities.

So we have seen how a first order occurrence matrix managed to preserve the letter frequencies of Romeo and Juliet. It makes some kind of intuitive sense, then, that a second order occurrence matrix would do an even better job of mimicking the Bard. What is a second order occurrence matrix? In our case, you can think of it as a table that answers the question, "Given a certain alphabetical or space character, what are the chances that it is followed by another given alphabetical (or space) character?" For example, we might want to know "With what probability does the letter 'a' follow another letter 'a'?" Or, "Given the letter 's', with what frequency do we expect the letter 'h' to follow it?" My second order occurrence matrix as printed in the MATLAB command window is shown in Figure 3.

Figure 3 reveals certain letter-letter pairs that we expect to have high frequency in a typical English language text. One can, for instance, almost immediately recognize the high probability of finding a letter "e" after the letter "h" (almost certainly a statistical holdover from English's definite article, that being "the"). Even higher is the related probability of discovering the letter "h" after a letter "t". Interesting to note are the less frequent, but also highly correlated letter pairs like "ex,"10 and the remarkably correlated "[space]t" pair (that is, the space character followed by a "t"). Shakespeare, apparently, enjoyed the look of t's at the start of words. But, then, isn't it the truth that thoughtful thinkers take pleasure in these things?

Of course, the discussion of the second order occurrence matrix has, so far, focused entirely on what is likely to occur in Shakespearean English. As it is, this is a very biased and unfair discussion. With a little bit of thought, most of us would agree that English is just as much defined by highly correlated letter pairs as by highly uncorrelated pairs.11 The letter "z" very rarely begins a word in English, so if I were to start typing z's zin zthe zfront zof zevery zletter, we no longer immediately recognize the words as belonging to English. It is clear that Eddington's purist approach to the IMT has an obvious flaw in assigning each alphabetical and

---

9 I take this definition from Bennett. Refer to: Bennett, William Ralph. Pg. 113.

10 Bennett, William Ralph. Pg. 116.

11 Ibid.
space character an equal probability of occurring - many letter sequences appear with considerably greater, and considerably lesser, frequency than others!

Using the second order occurrence matrix yields some interesting results. Figure 4 shows MATLAB's attempt to replicate *Romeo and Juliet* using our new matrix. As my word processor is quick to inform me, there are approximately seventeen actual English words in this 130-word text printout. But, more interestingly, some of the letter sequences seem to approach real English rather strikingly. For example, halfway through the printout we see this heartwarming phrase: "we fe llove ju" (We love you?). This obsession with affection becomes even more pronounced in higher order occurrence matrices. Of course, given our choice of romantic tragedy, this isn't particularly unexpected.

With third, fourth, and fifth order occurrence matrices, the advantages of using the speed, accuracy, and storage capabilities of a computer become more and more apparent. Using our twenty-seven-characters typewriter model, the dimensions of the third order matrix are $27 \times 27 \times 27$. The third order matrix holds data of the form "How many times does one letter follow another letter follow another letter?" As such, it must possess the frequency data for $27 \times 27 \times 27 = 19,683$ distinct events. Similarly, the fourth and fifth order matrices hold data for $27 \times 27 \times 27 \times 27 = 531,441$ and $27 \times 27 \times 27 \times 27 \times 27 = 14,348,907$ distinct events, respectively. Given our model, it is true that an $n$th order occurrence matrix has $27^n$ matrix elements. This reveals, perhaps, the most fundamental advantage of the computer: We can pass off all of the difficult and computationally intensive calculations to its CPU and proceed to pace the room in anxious anticipation.

You'll have to forgive me if I don't offer a printout of the third-order occurrence matrix. I doubt many of you would have found the thirty-five pages of arrays very interesting anyway! As for the fourth and fifth orders... Well, let's just say I'd probably bankrupt myself printing with the College's GreenPrint system! Working on a lowly Apple MacBook with a 2.4 GHz processor, my computer required a whole evening to initialize the fifth order matrix. Even so, this "lengthy" amount of time represents a tremendous improvement over what might have been accomplished with a human brain. While the mind does store and retrieve information in mysterious and powerful ways, it simply isn't possible to command-recall 14,348,907 unique probabilities.

In the interest of being somewhat economical, I intend to sidestep discussion of the third and fourth order occurrence matrices. While these matrices allow our monkeys write increasingly

---

12 These results are truncated somewhat for efficiency of space. It simply isn't economical to reproduce the entirety of MATLAB's efforts. Reprinted here are the first several lines of output. As with the first order output, readers interested in the relevant code should refer to the accompanying document.
sophisticated words and phrases, their probability data are inherently embedded within the fifth order matrix. In the fifth order, our MATLAB has learned to recognize the high correlation of character sequences like "ound[space]" and, of course, "[space]love". Similarly, the combination of characters "aaaa" has very low probability (in particular, this has probability zero since, as is quite obvious, identical letters never appear more than twice in a row in unabbreviated English).

Some of my results from the fifth order occurrence matrix are reproduced in Figure 5. This particular sample text reveal a remarkably high yield of real English words, and even those sequences of characters that are not truly English words have an uncanny resemblance to English. This bizarre fact recalls Shakespeare's own influence on the language: The Oxford English Dictionary indicates the presence of Shakespeare's artful hands in nearly two-thousand words. One cannot help but wonder if the Bard, when he was adapting English to new senses and meanings, wasn't secretly exploiting the letter correlations of English. Indeed, does it seem so unlikely that letter combinations like "stression" or "dispose" or even "humoroushood" could have been seen in 16th Century English?

I must invite the reader to try reading the text in Figure 5 out loud. I think you'll find almost immediately that it really does sound like Shakespeare. The phrases themselves, of course, stray slightly to the cryptic side of language, but MATLAB's output is, on the whole, quite competent. It is possible that this comes as a direct consequence of what we used to form

---

**Figure 5.**

you sir him her blood dear perforce tomb lay how oft to fine two foot be we shall we shall die windows very not answer of the make that black to bright false wouldst to you learn meant was do spell make me wall this she would be go hears treath not lies throne ill conjured they beguiled all dram the heels most so to before hear lookd not be that is to you will says are and fireeyed murderer come being you haster you to as gone shing lady with the nickness skulls or dry foes to bed and art thou shall my cell me again my obsurses more is last shame state pilot night be able past now my face to come stression reversal ears pale in alderman beauty husband as them find tonight a man away stayd by ancell moved him in quiet and love spoke be ill me married yet though for this out the we should fa usure dash the passistand dies found divorce to sound of the wretchwencherity sir get thou with a little be bride that wits water dearly maidenhood hath his a name ho fond yet say holy face of death grant sir us body sleep fortunes foul said it not not for with this his was sweeth me give mark whose nature before more cooks and those not his face art thou on pair a word out once and in himself ke those here strangleness can findswift a mercing it is hot for all me peace of leaven in a merry not because the lady ladieu and alone of my art what house the sun noble those dispose thing for my ghost near thou honest but those unto tell descreeend the music with me in teach of more is all be pernicious thing lady deal upon youth of you knowst me lurk such slow to my bring gracious respectst up him from the prince run marriage under and brother eyes sweary man but sheat of vows young with he sadly poor princes do not trust for if you quies my such upon these ever an iron day stay with his she and fall my lord my seize ame is hast beginners and madam to comes a flow calls within to chamber it did vice in stress that thy purposed in this here far morn to their head thou slew to be say stand him monument dange is humorousehood forsworn the gate pilot night legs ufferent fight of love against tonight good have a pointseducing proud care be they unworth too roughter that lies a torm of orname window of breason with a caitiff an unstaint it even and thee learn of flecked death loaths with of thy change is

---

our occurrence matrices. *Romeo and Juliet*, because it is a play, has a conversational quality that lends itself better to straightforward and concise declarations than to intricate, expository, and, ultimately, less clear prose.14

Yet for all its competence, MATLAB's output seems to lack something of articulacy. That is, although nearly ninety-five percent of the character sequences in Figure 5 are English words, taken together they lack overarching meaning. We see Shakespeare's literary style, but nothing of the poignancy and meaning that make him the world's greatest dramatist. This begs the question "How many further orders of correlation will we require to see actual thoughts and ideas emerge from mere probability?"15 It's quite a peculiar question, one that addresses the root of what human intent really represents. Clearly, the ability of MATLAB to write Shakespeare became more and more sophisticated as we constructed occurrence matrices of higher dimension. So is it reasonable to expect that, given sufficient computer memory and calculating time, that a computer might generate an "original" idea?

I won't speculate on the issue because I have little knowledge of the how the human mind arrives at intentions and conclusions. Yet it is interesting to consider the possibility that many of the prolific artistic geniuses in man's history drew extensively on the knowledge of what preceded them. In some ways, we can think of this past knowledge of style and creativity as the building blocks of a high order occurrence matrix, one that weights heavily the most appealing and elegant artistic choices.16 As a peculiar aside, Mozart himself wrote a brief guide entitled "Instruction to Compose, Without the Least Knowledge of Music, As Many German Waltzes As One Pleases by Throwing a Certain Number with Two Dice."17 Were one ever to invest the time, it would no doubt be an interesting research project to see what new styles of artistic expression might arise simply by composing the occurrence matrices of varying styles. For example, ask yourselves the question, "What would a combination of Bach and techno really resemble?" Statistics can provide the answer. But as I hoped to indicate, that strays more into the realm of massive commitment than peculiar little diversion.

It is, I realize, very difficult to digest the possibility that all of man's great efforts - from Beethoven's Ninth Symphony, to Michelangelo's David, to Shakespeare's *Romeo and Juliet* - were deduced from high order occurrence matrices. So, to all of those who array themselves with Cicero on the issue, I leave them with only the following remark: Monkeys writing Shakespeare is much more the probable occurrence than that the full text to Cicero's *Catiline Orations* will ever materialize from the innumerable water molecules that drift purposelessly above Sudikoff.18

---

14 Bennett, William Ralph. Pg. 123.

15 Ibid. Pg. 126.

16 Ibid.


18 I acknowledge Bennett as the inspiration for this code. However, I have translated it into MATLAB and have adapted the processes involved to higher correlation orders than what Bennett aspires to. I would still like to thank Alexander Brofos for his technical support in the debugging process.