The *Countdown* Letters Game

B. Moran

Abstract

We present an analysis of the letters game from the TV show *Countdown* using Monte Carlo methods. This game requires finding the longest possible words in a set of randomly chosen letters. We find that the probability of finding a word of length k from N given letters follows a Fermi–Dirac Distribution with k as the variable and N acting as control parameter. Increasing N we get to a fixed point where a phase transition occurs before reaching the IR fixed point as $N \to \infty$. Lastly, we find the expected total number of words per game, and the number of letters one must be given in order to have a significant probability to find all words in the dictionary.

1 Introduction

The french TV programme *Des chiffres et des lettres* has been airing in its present format since 1972 making it one of the longest running game shows in the world [1]. Since 1982 it has also been airing in the United Kingdom with the name *Countdown* [2]. It has also been adapted to several other languages.

The show consists of several rounds of two different games: a *numbers* game that tests the numeracy skills of the contestants, and a *letters* game that tests their vocabulary. An analysis of the *numbers* game has been performed in [4], here we turn our attention to the *letters* game.

In this game a set of 9 letters is randomly chosen from two pools, one containing only vowels and one with consonants, and the contestants must find a valid word[1] built from those letters, each letter must be used at most once. The contestant who finds the longest word wins the game.

Each pool contains letters with a probability distribution in accordance with their frequency in the dictionary. The contestants can choose the total number of vowels in the set.

[1]a word is *valid* if it can be found in the dictionary

In this paper we don't take into account the ability of the players to find the hidden words, and just consider whether they are constructible. The dictionary that we used contains 276,663 words (one-letter words are exluded). We've also performed the study for French, German and Russian with the same qualitative results, not presented in this paper.

2 Monte Carlo Simulation

Our main concern will be the probability of finding words of length k in a set of N letters. We'll call this probability $P(k; N)$, with $k \leq N$, and $k, N \in \mathbb{N}^+$.

As all letters of the alphabet are in the dictionary we know that $P(1; N) = 1$, *i.e.* we can allways find words of length 1.

Let k_M be the length of the longest word in the dictionary. Then, $P(k > k_M; N) = 0$, *i.e.* we can't find words with length greater than k_M no matter how many letters N are there in the set.

Naively we expect that $P(k; N)$ is a decreasing function of k: longer words are harder to find than shorter ones. Varying N we expect that $P(k; N)$ will increase with k fixed, *i.e.* the more letters one has the easier to build words of a given length.

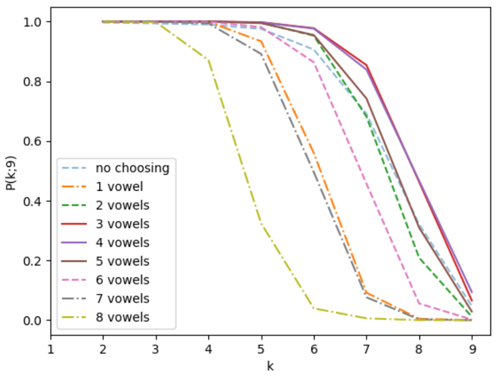

Figure 1: Probability of finding words of length k with $N = 9$

In the large N limit we expect that $P(k;N)$ will be a step function: $P(k \leq k_M;\infty) = 1$ and $P(k > k_M;\infty) = 0$. So, when N is big enough we'll always find any word in the dictionary.

To find $P(k;N)$ we've relied on Monte Carlo methods. To start with, we performed 10,000 simulated games of *Countdown* ($N = 9$) for a different number of vowels in the English version. This are plotted in Figure 1. For the curve with the 'no choosing' label there's no fixed number of vowels and the letters are retrieved from a single pool with both types.

As we can see, the probability function is a sigmoid. For low values of k, $P(k;9) \cong 1$, *i.e.* we can almost surely find short words. Then, as k increases the probability drops to $P(k;9) \cong 0$, making it more unlikely to find long words.

The number of vowels has a significant impact in the probability, with the optimum at 3 or 4 vowels. The 9-letter words of the English dictionary have an average of 3.387 vowels. Any other number of vowels gives a smaller probability and it would be convenient only for tactical reasons and not giving the opponent the chance to find long words.

The probability of finding a 9-letter word with 4 vowels is 9.4%, with 3 vowels is 6.6%, and approaching zero for other values. In a sense, the game is fine-tuned so that it's hard but not impossible to find a word with all the letters. For 8-letter words the probability rises to almost 50% in the optimal 3-4 vowels case. There are 7-letter words in almost 85% of the games.

Obtaining an analytic form for $P(k;N)$ is hard, if not impossible, as this depends on a number of factors: the number of possible combinations of the N letters in groups of length k, the number of words of length k in the dictionary, and the probability distribution of the letters. It seems that this would be an extremely complicated function of N and k.

But this plot suggests that this probability can be explained by a simple curve. We may try to fit a Fermi–Dirac distribution of the form:

$$P(k;N) = \frac{1}{e^{\beta(k-\mu)} + 1}\,\Theta(N-k)$$

The Heaviside step function $\Theta(N-k)$ is to ensure that $P(k;N) = 0$ for $k > N$. The parameter μ is the value of k where the probability drops to one half $P(\mu;N) = 1/2$, *i.e.* the place where the midpoint of the fall is located. In general μ won't be an integer. We'll see later that it also depends linearly on N. The parameter β denotes the steepness of the curve and seems to be almost the same in all cases, except for the case of 'no choosing' which has a slower descent. As we'll see later, the dependace on N lies in the parameters μ and β.

In Figure 2 we plot the same values as before (dots), the dashed lines are the Fermi–Dirac fit curves.

As we can see the fit is excellent in all cases. For 3 and 4 vowels $\mu = 7.90 \pm 0.02$, and $\beta \approx 2.5 \pm 0.04$ in all cases except 'no choosing' where $\beta = 1.7$. Note that with this approximation we can treat k as a real number.

Increasing the number of letters N we expect that β will tend to infinity so the Fermi–Dirac distribution will approximate a step function. On the other hand, we expect that μ will approach k_M, the size of the longest words in the dictionary ($k_M = 15$ in our case).

3 Varying N

We now let N vary, a game known as *Street Countdown* [5]. From this point onwards we'll no longer

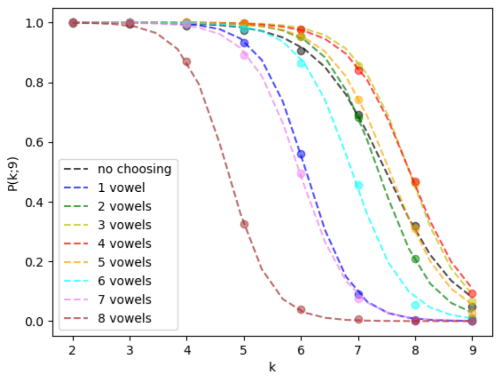

Figure 2: Fermi–Dirac fit of $P(k;9)$

choose the number of vowels and consonants and retrieve the letters from a single pool. For each case we'll make 1,000 simulations.

Figure 3 shows the probability $P(k;N)$ for $N = 2,\ldots,17$. We see that μ increases with N as expected: with each additional letter the point where $P = 1/2$ moves closer to k_N. The slope β is practically the same in every case.

The dotted line is $P(N;N)$, the probability of finding words with length equal to the number of given letters (that's double points). This falls to zero when $N \geq 13$.

Figure 4 shows the probability of finding words of length k as a function of the number of letters N. The dotted line is again $P(N;N)$.

As expected, increasing the number of given letters the probability of finding words of length k also increases. This plot is helpful for determining an optimal value of N if our goal is to establish a given probability of finding words of a certain length. So, this one is for the producers of the show.

Figure 5 shows the probability $P(k;N)$ as a function of k and N, for $N \leq 22$. The diagonal line is the function $k = N$ and the horizontal is $k_M = 15$. Above this two lines the probability is equal to zero.

The probability $P(N;N)$ of finding words with all the letters is crucial for determining an optimal value of N such that the game is interesting. For $N \leq 7$

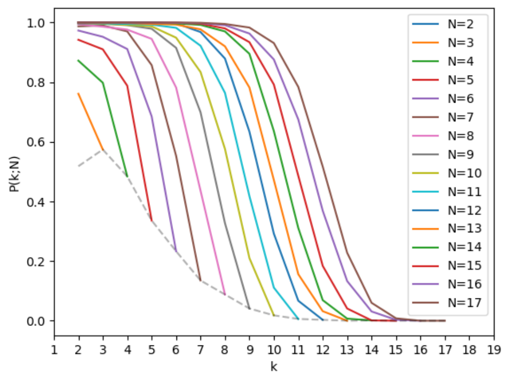

Figure 3: $P(k;N)$ as a function of k for each N

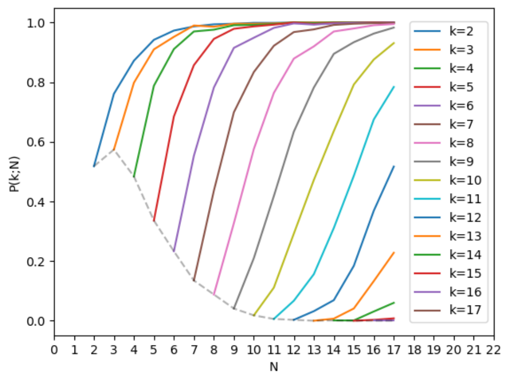

Figure 4: $P(k;N)$ as a function of N for each k

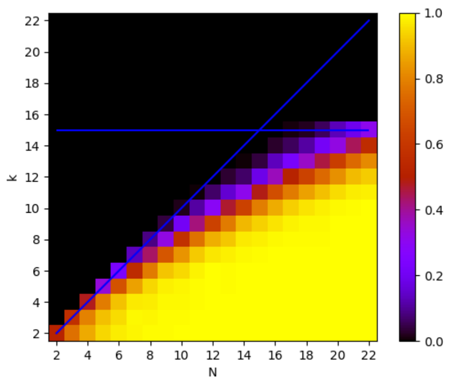

Figure 5: $P(k; N)$

is above 10% and the game would be too easy for expert players. Draws would be common. On the other hand, for $N \geq 12$, $P(N; N)$ falls nearly to zero and there would hardly ever be any game with top words. So, the optimal value of N lies between 8 and 11.

Figure 6 shows the values of μ and β as we increase the number of letters.

Strangely, β doesn't seem to depend on N and stays with a constant value of approximately 1.7 whereas μ grows linearly with N: $\mu = 2.94 + 0.52 \cdot N$. As we've seen in the introduction, we expect that the probability distribution tends to a step function with $\mu \to 15$ and $\beta \to \infty$, at some point the actual tendency must change. In the next section we study the asymptotic behaviour.

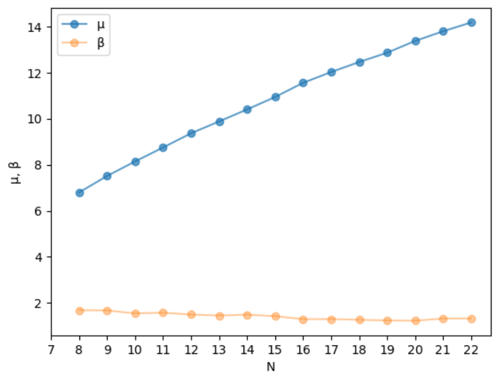

Figure 6: μ, β as function of$N$

4 Street Countdown

Let's step it up. We've done simulations up to $N = 26$, in figure 7 we show the parameter space of μ and β as we increase N. At small N's the flow is that explained in the previous section: β stays nearly constant while μ grows linearly. When $N \approx 23$ starts a regime where there's a significant probability to obtain words of lenght 15, the maximum. At this point the growth of μ stalls at 15 whereas β starts growing rapidly. We've reached a phase transition.

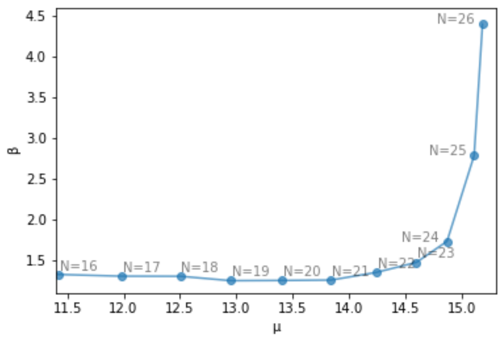

Figure 7: μ vs. β

In Renormalisation Group (RG) [6] lingo we'd say that N acts as a control parameter of the flow of μ and β. We expected a fixed point at $(\mu, \beta) = (k_M, \infty)$ and we've found another one at $(\mu, \beta) = (k_M, \beta_0)$. The evolution is initialy governed by μ, which is an irrelevant deformation in the RG sense, until it reaches a first unstable fixed point. Then a phase transition occurs and the system flows to the infra-red stable fixed point governed by the relevant coupling β.

5 Counting words

We now pay attention to the total number of words of each length that can be expected in a game. In the next table we show the expected value and the standard deviation for each length in the canonical $N = 9$ game:

length	total	length	total
2	16.398 ± 6.46	3	47.212 ± 22.11
4	66.202 ± 42.67	5	44.341 ± 40.40
6	18.828 ± 21.81	7	5.197 ± 7.98
8	0.733 ± 1.65	9	0.058 ± 0.28

Small words are easier to find but there are also less of them (see the appendix), for larger words is the opposite. This shows in the previous table: we typicaly find more 4 letter words than any other length, with about 66 per game. We'll find fewer smaller words than that just because there are fewer of them. And we'll find fewer larger words because they are harder to find. We'll see shortly that the distribution follows a tilted bell shape.

We also see that the standard deviation is large. That's to be expected as the number of words we can build depends strongly in the letters we are given, which come from a random distribution with given weights as explained earlier. For example, if we are given among others two Q's and an X it's going to be tough. We've also seen in the first section that it's very dependant on the number of vowels.

Let's vary N: in figure 8 we show the average total number of words of each length for games with N between 2 and 26. No legend needed, it only messes with the plot.

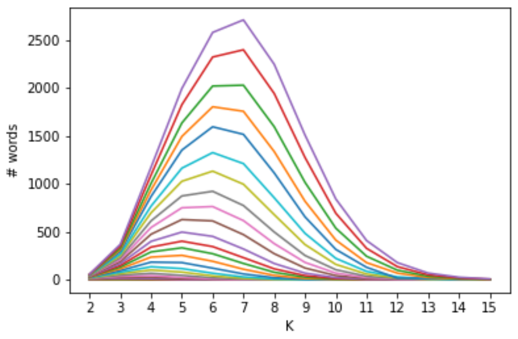

Figure 8: Total of words per game

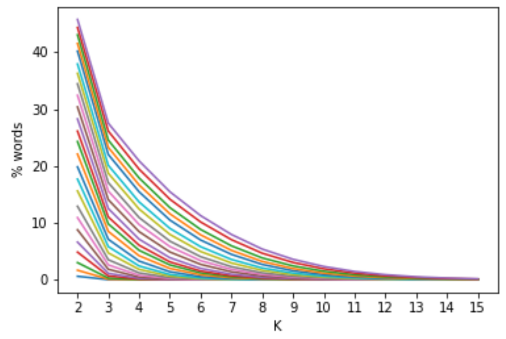

Figure 9: Percentage over the total of words for each length

As we increase N the number of words also increases, obviously. The shape of the distribution slowly approaches that of the total number of words per length (see the appendix), but it's a painfully slow growth. In figure 9 we plot the percentage of words per game relative to the total number of words of each length.

This is a decreasing function with k as expected, there are fewer possible combinations with fewer letters and as N inceases we rapidly cover more of them. But even then, with $N = 26$ we can't find 50% of the 2-letter words. On the other hand, for larger words the increase is ridicule. We'll need a collossal number of letters to get a combination with all the words in

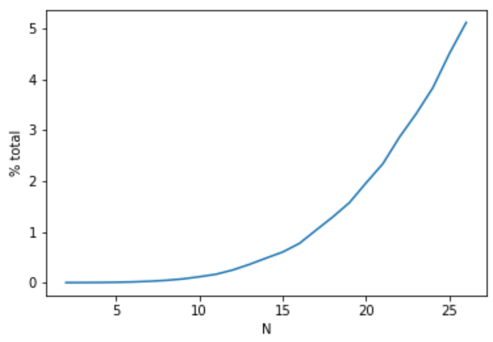

Figure 10: Percentage over the total of words with N

the dictionary in it. In figure 10 we plot the percentage of the total number of words per game over the total number of words as a function of N.

For small N's the increase is quadratic but it gets linear at higher values. We expect this function to be a sigmoid, at some point the increase will be smaller until we asymptoticaly reach 100%, just like the Fermi–Dirac distribution of the first section but viewed in a mirror. But what's the actual shape of this sigmoid? Well, we could just keep increasing N and simulating more and more games, but with such a small growth we could be doing this for eternity. We better change tactics.

6 The Mother Of All Words

Let's ask ourselves: what's the minimum collection of letters that contains all the words in the dictionary? It's simple, just count the maximum number of times each letter appears in a single word. For example, how many A's contains the word with more A's in it, and so on. In the following table we list that, with one example for each letter (there can be more, of course).

A: 6 (taramasalata)	N: 5 (inconveniencing)
B: 4 (beblubbered)	O: 5 (homoeomorphous)
C: 4 (acciaccatura)	P: 4 (snippersnapper)
D: 4 (condiddled)	Q: 2 (chiquichiqui)
E: 5 (agreeablenesses)	R: 4 (agroterrorism)
F: 4 (chiffchaff)	S: 7 (classlessnesses)
G: 5 (gangshagging)	T: 4 (aftertreatment)
H: 3 (bakhshish)	U: 5 (untumultuous)
I: 6 (dirigibilities)	V: 3 (overvivid)
J: 2 (hajj)	W: 3 (bowwow)
K: 4 (knickknack)	X: 2 (coexecutrix)
L: 4 (alcoholically)	Y: 3 (syzygy)
M: 4 (mammectomies)	Z: 4 (bazzazz)

To build the string that assures us that we can construct all the words in the dictionary we just need as much of each letter as the table says. So, behold the Mother Of All Words (WOMA[2]):

AAAAAABBBBCCCCDDDDEEEEEFFFF
GGGGGHHHIIIIIIJJKKKKLLLLMMM
MNNNNNOOOOOPPPPQQRRRRSSSSSS
STTTTUUUUUVVVWWWXXYYYZZZZ

This combination contains every word in the dictionary. It has the respectable length of 106 characters, that's the minimum of letters we must get if we want AARDVARK and GLOBULARNESS and LAUREATESHIP and PHOTOOXIDATION and every other word. But the probability of getting a WOMA given 106 random letters with the according distribution is about 10^{-160}. You can play for an eternity but you'll never find a WOMA that way. You'll need more letters.

But how many? Well, we've simulated it, it shouldn't be surprising by now. We've build sets of letters of size N from $N = 400$ upwards, with 1000 games each time as usual. Figure 11 shows the probability of finding a WOMA given N letters.

It's a sigmoid, as we predicted in the previous section. But in honesty, we're plotting something else here. Anyway, we see that below a 1000 letters chances are slim of finding a WOMA, but from 3000 and up you'll find any word you fancy. That must be a bloody boring game if you ask me.

[2]we're fond of anagrams here

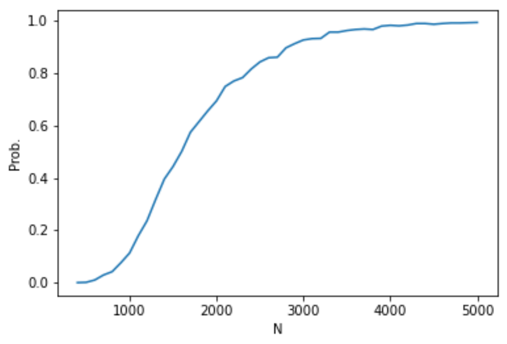

Figure 11: Probability of finding a WOMA

7 Discussion

Who would have thought that the Countdown letters game contains as much maths as the numbers game? We've found that the probability of finding words of a certain length follows a (discrete, if that makes sense) Fermi–Dirac distribution, depending on two variables: the middle point of the curve μ and the slope β. These flow in accordance with the control parameter N, the number of given letters, to a first fixed point following the irrelevant coupling μ where a phase transition occurs. And then to the IR fixed point at infinity following the relevant deformation β. This behaviour was observed for the other languages we studied, we conjecture that it's universal for all idioms written with an alphabet of letters (as opposed to say, Chinese, which follows a different pattern (how would this game be played in Chinese?)).

With 9 letters, the optimal value of vowels is 3 or 4, with almost 10% of games getting 9-letter words. With a set of 10 letters, as in the French version of the game, the optimal number of vowels is 4. In this case, the 4.7% of games get a 10-letter word. With $N = 11$ and 5 vowels we get 11-letter words in 1.6% of the games. Now it's for the player to find them, as it gets trickier.

The total number of words per game grows so slowly with N, and with a big standard deviation. You'll need a set of about 3000 letters at least to get the chance of composing any word of the dictionary you fancy, in the most tedious game of Street Countdown ever.

References

[1] Wikipedia: Des chiffres et des lettres

[2] Wikipedia: Countdown (Game Show)

[3] Wikipedia: Letter Frequencies

[4] Lacasa L., Luque B.: *Phase transition in the Countdown problem*, arxiv:1206.2876 (2012)

[5] The IT crowd. Series 4, episode 2 *The Final Countdown*

[6] Tong D., *Statistical Field Theory*, Lecture notes

A Distribution of lengths in the dictionary

The dictionary we've used has a total of $276,663$ words. Figure 12 shows the number of words of each length, the line is a Gamma distribution fit of the form $f = a \cdot k^b \cdot c^L$, where k is the length of the word and $a = 0.3149$, $b = 10.34$ and $c = 0.2966$.

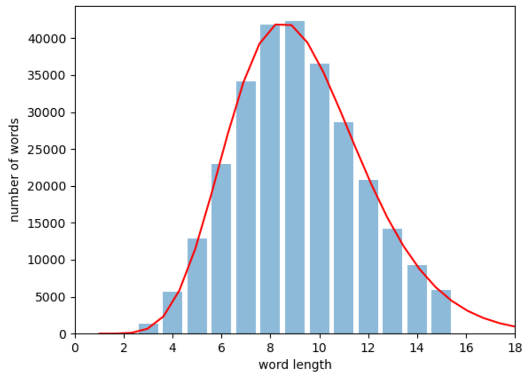

Figure 12: Length distribution