LABORATORY INSTRUCTION–RECORD PAGES

Laboratory Experiments With

WordEvol

Exploring the Power of Natural Selection

Concept and DOS Based Computer Program by Stephen Prata
Windows Based Program by Stephen J. Baedke

Geology 200 - Evolutionary Systems
James Madison University
Lynn S. Fichter and Steven J. Baedke

Introduction

In the experiments done so far we have been dealing with information flow, and the emergence of complex or interesting patterns in nature. In most of these experiments we were looking at information flow in very abstract, mathematical systems. Today we want to look at information flow from a different perspective.

Webster's New Collegiate Dictionary defines ***INFORMATION*** as:
- ☺ The communication or reception of knowledge or intelligence.
- ☺ Knowledge obtained from investigation, study, or instruction.
- ☺ Data, facts, intelligence, knowledge, news.

What we are looking for, of course, is meaningful information. For example, information that allows a merchant to be successful in a free market economy. Or information that codes for a protein that catalyzes a reaction that produces DNA. Or, information that allows an organism to learn to more efficiently search for food. What's more we want to generate this meaningful information without any design or purpose, Bottom-Up, not Top-Down.

Both the Instructions for the experiments and the spaces for recording your experimental results are contained here.

Wordevol

Searching For Meaningful Information

1. Exercise: Getting to Information

This is a Thought Experiment. There are no programs to run. You just have to reason your way through the problem and write an analysis.

❑ **Which Contains More Information?**

. . . . The string of 31 letters (and spaces) in the above line, “Which contains more information?”

. . . . or, the string of 31 letters below?

DNXDZNDTLWUNLPMTKAVOFGKADQUAGXK

Record One: A Thought Experiment

ACCEPT OR REJECT: The phrase

“**DNXDZNDTLWUNLPMTKAVOFGKADQUAGXK**”

contains more information than the phrase. . .

“**WHICH CONTAINS MORE INFORMATION**”

On what criteria? Convince me, I‘m a skeptic.

☺ ~ ☺ ~ ☺ ~ ☺ ~ ☺

Natural Selection

Introduction:

You have heard the story of a room full of monkeys with typewriters . . . ? Given enough time one of them will type out the Gettysburg Address. But you have also heard the argument, “How could life have evolved by random processes when so many unlikely events have to take place? It would take longer than the age of the universe!”

The probability of a bacteria evolving by chance is 1 in $10^{40,000}$. In comparison total number of hydrogen atoms in the universe is something like 10^{60}.

But of course life is not the result of random events alone. There are at least two processes operating in evolutionary systems that produce order.

First, are the *processes of self-organization* (autocatalytic in Stuart Kauffman’s terminology, self-organized criticiality in Per Bak’s terminology) which lie within the realm of complexity we saw working in the lab on cellular automata - Local Rules/Global Behavior. These are the same processes that allow a free market economy to work so smoothly without central planning, or all the nerve cells in your brain to work together to produce the conscious experience you have, and any open system to become initially organized. Research into self-organizing behavior has been going on for a couple of decades now, mostly at the Santa Fe Institute, but is beginning to become better known outside of the small group of researchers who developed it. [1]

Second, is *natural selection* which works on (selects among) the self-organized systems. Typically, the self-organized systems arise randomly, like, for example, if you mix bunches of letters together. Some of them may spell words, or come close to spelling words, but most will not. Similarly, if you mix a bunch of chemicals together and allow them to react. The reactions will produce a lot of new, more complex molecules by self-organization. This does not mean all these self-organized molecules have meaning. They are just more complex. But some may have more meaning than others, and a natural selection process can sort the more meaningful from the less meaningful.

Biologically, natural selection is the process whereby some individuals survive in preference to others because they posses features which give them some advantage in a particular environment and allow them to be successful, where others are not. Natural selection culls out some individuals in preference to others based on their fitness, that is, their effectiveness at gathering resources, or reproducing. To do this there must, of course, be ***selecting agents***, something against which organisms are compared, allowing some to leave more descendants than others. Without selecting agents, changes can only be random.

In a more personal example, universities select only some students from the pool of applicants, making their choices based on the selecting agents of GPA, SAT, extracurricular activities, etc. In other terms, selecting agents are attractors and the more we are attracted by those attractors the more successful we are, and the less . . . the less.

We are told that evolution is the result of natural selection. But rarely do we see demonstrated just how effective natural selection can be at producing systematic and directed change. This very simple program called WORDEVOL is designed to do just that.

[1] This work is now available in a number of books including Stuart Kauffman’s “At Home In the Universe: The Search For the Laws of Self-Organization and Complexity” a popular account of his more technical (i.e. mathematical) book “The Origins of Order”. Also see Brian Goodman’s “How the Leopard Lost Its Spots: The Evolution of Complexity” and John H. Holland’s “Hidden Order: How Adaptation Builds Complexity” and Per Bak’s “How Nature Works.”

2. I am thinking of a Phrase:

❏ I am thinking of a phrase (the target string, or selecting agent).

☺ It has 4 words, 3 spaces and 16 letters

☺ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

☺ What is this phrase?

You could try to discover this phrase just by guessing at strings of letters randomly. But you would be more clever than that. Instead, you would presume the string had some meaning, some goal or purpose, and would look for words of the proper lengths, and strings of words that go together and make sense. But that would be a Top-Down strategy, trying to design the desired outcome by intelligence - that is, teleologically.

But, is there a Bottom-Up strategy for generating the proper string? Here is a string of 19 randomly chosen letters and spaces.

BVGTCBOMLJDXZX RSPO

Beginning with this string is there a way to generate the target string by natural selection?

Record Two: I am thinking of a phrase: suggest a strategy for finding it.

Can you suggest any strategies for getting this random string to transform into the target string, which, by the way, is "What is this phrase," (4 words, 3 spaces, 16 letters.)

Opening The Wordevol Program [2]

The rest of the laboratory is done with the program called "WordEvol"

❏ Program is available in the Geology Computer Lab, Miller 233. The shortcut is within the "Evolutionary Systems" folder on the desktop. A copy is available on request.

❏ Follow the instructions below for investigating the program.

[2] Two versions of this program are available. The DOS based version by Stephen Prata came with the book "Artificial Life" and is the basis for the program we use written by Steve Baedke. Both versions of WORDEVOL are available in the Geology Department computer lab. The DOS based version is self explanatory and works well, with good explanations and step by step instructions not only on what to do but what is happening.

Experiencing Natural Selection

3. Step One: You Choose A Phrase

- ❑ **Instructions:** In the box **Step One** in the Wordevol program type in a phrase of up to 50 letters and spaces. No numbers or punctuation.
- ❑ Write the same phrase in the data box on the last page.

Explanation

This is the "target string", a.k.a. "selection agent", a.k.a. "fitness function." It is what we are trying to match. In nature it is a niche an organism is adapting to. In a class it is a test you are adapting to

There is the question, or course, of whether the stuff on the test is worth learning, but in terms of your class fitness that does not matter. Getting an A in the class depends on being able to match the fitness functions of the class, which happens to be selecting the right answers on the test. You can either increase your fitness by studying, or you can just answer questions randomly and take your chances of being selected out.

There are many ways an adaptive agent can come to match a fitness function. Studying for a test is one way. But this obviously involves conscious thought. In Wordevol we are looking for a method that takes place without that. In Wordevol you have chosen the target string, but the computer is going to get to it without "thinking" about it. In the natural world the target string could be, for example, the string of amino acids which make up a functioning protein.

4. Step Two: Finding How Many Random Strings of Equal Length Exist

- ❑ **Instructions:** When you click the **Step Two** button, not only will the Step Two page open, but your computer will generate in one second as many random strings of letters and spaces as it can of the same length as your target phrase. This will be the basis for all the other calculations shown.
- ❑ For each of the values listed write the numbers on the data sheets, i.e.
  - ☞ Number of distinct strings of this length.
  - ☞ Estimated time for your computer to generate a string matching target string, in seconds, days, and years.

Explanation

If there were no natural selection, and no higher intelligence trying to find a match, then our only choice would be to generate strings of letters and spaces at random of the length of your target string, hoping eventually one of them would match the target string. As you will observe, it takes a very, very long time. The number of distinct strings you see may be in exponential notation so that, for example, 1.53e+25 is 1.53×10^{23}.

5. Step Three: Let’s Try to Match Target String Randomly

- ❑ **Instructions:** Under **Step Three** in the program enter the number of random strings you want the computer to generate; maximum is 500. When you click the GO button the computer generates the number of strings you requested and then prints the string closest to your target string.
- ❑ Write the random string on the line on the data page at the back.

Explanation

Unless you were very, very lucky, the closest randomly generated string looks nothing like the target string. It is very hard to get the target string randomly, in fact, virtually impossible. There is just not a reasonable enough amount of time to do it. The closest random string shown in the box should demonstrate this.

6. Step Four – A: Let’s Try Some Natural Selection

- ❑ **Instructions:** Under **Step Four** is your original target string. For later experiments, you can change the string by highlighting it and typing a new phrase.

When you click the Generate Random String button the computer randomly generates one new string of the length of the target string and show it in the box. The goal is to get that initial string to mutate and evolve via natural selection to match the target string. There are three mutation/selection strategies you can use; for now select the first and click GO. At the back is a description of each natural selection strategy; the same descriptions are also in the program.

Explanation

In a process of natural selection we don't start over with a completely new random string every time we do not get a match. Instead we take the initial random string, mutate it (change it) in some random way, and compare the mutated version with the target. If the mutated version is closer to the target than the initial phrase, we throw away the initial phrase, keep the mutated phrase, and, in turn, mutate that.

The only difference in WORDEVOL is that with each cycle (generation) we mutate the closest matching string 20 times; that is, make 20 copies, each randomly mutated in one of its letters or spaces. We then take the one of the 20 that is closest to the target string as the starting point for the next generation. In an algorithm it looks like this.

- ☞ *Generate initial phrase randomly and compare it with target string.*
- ☞ *If initial phrase does not match target, create 20 new phrases from the initial phrase, each mutated at random in one of its letter/spaces.*
- ☞ *Compare 20 mutated phrases with target phrase. If any mutated phrase is closer to target phrase than the initial phrase, throw away initial phrase and keep mutated phrase.*
- ☞ *Repeat Steps 1-3 beginning each time with the phrase closest to target.*

This procedure is closer to biological natural selection where parents produce more offspring than can survive, each different in genetic makeup and fitness for the niche they are adapting to.

Distance is calculated by comparing each random letter with its target letter. Say the target letter is 'D', 'A' is 4 units away and 'K' is 7 units away, so 'A' is chosen. This comparison is made for all unmatched positions on the target string.

7. STEP FOUR – B: LET'S TRY SOME NATURAL SELECTION

❑ **Instructions:** In the data table at the back write down the following from the program.

- ☞ Initial string generated by computer at random.
- ☞ Final match was found in this generation.
- ☞ Checking this many total strings (20 each generation)
- ☞ In this many seconds
- ☞ With which mutation/selection strategy?

8. STEP FOUR – C: TEXT FILE, FINAL RESULT

❑ **Explanation:** Click on the TEXT FILE FINAL RESULT button. This text file lists the sequence of mutated/selected strings as they get closer and closer to the target string. That is, each of these strings is the one out of 20 mutated strings that was the closest to the target string. The details of each string depends on the mutation/selection method you choose, but notice as you scroll through that the strings get closer and closer to the target string.

This text output will vary each time you run the program, for two reasons.

① Each mutation/selection method differs in how it naturally selects the strings, and so the list of strings shown would differ with different strategies.

② Also, because each string is the result of random mutations, each time you run the procedure, the actual steps by which the procedure gets to the target differs, beginning with the same target string and mutation/selection method.

Explanation of Contingency

Contingency is the idea that not only what you get varies with what you start with, but because of what you start with some things are just not very possible. For example, an evolutionary descendant of an elephant is not going to learn to fly. An elephant's history (contingency) has taken it down a different evolutionary path, and as some choices were "made" by natural selection, others are excluded.

But in a similar way, evolution would not necessarily, or even likely, get to the same end point (target string, adaptive niche) in exactly the same way. This is not only because it likely starts from a different starting point each time, the random mutation steps it goes through are . . . well, random, i.e. not likely to be repeated.

For example, "ant-eating" mammals (anteaters, aardvarks, etc.) are all adapting to roughly the same niche, and so all ant-eating mammals have features in common. But because their ancestors have all started from different places, and they have all evolved toward the niche (adaptive string) in unique ways. They all differ in specific details.

WORDEVOL is much more tightly constrained than real evolution, but nonetheless, we should be able to see it approach the target string in different ways reflecting each run's different history, and contingency.

9. EXPERIMENTAL RECORD NINE: CONTINGENCY:

To do a complete string by string comparison between different runs of WORDEVOL is very tedious, and not that productive. But we can do a short comparison.

Under **STEP FOUR** do three runs starting with the same (short) target string (get it to fit on the lines below) and same mutation/selection strategy

☞ In the space below write the randomly generated initial string.

☞ Mutation/Selection Strategy?

#1 - Attenuating Selection #2 - Serial Selection #3 - Optimizing Selection

☞ Then, for each run go to the TEXT FILE FINAL RESULT and scroll down to the final 10 strings as they mutate and select toward the target string, and write down those ten.

Run One	Run Two	Run Three
Last 10 Mutations	Last 10 Mutations	Last 10 Mutations
1. ____________	1. ____________	1. ____________
2. ____________	2. ____________	2. ____________
3. ____________	3. ____________	3. ____________
4. ____________	4. ____________	4. ____________
5. ____________	5. ____________	5. ____________
6. ____________	6. ____________	6. ____________
7. ____________	7. ____________	7. ____________
8. ____________	8. ____________	8. ____________
9. ____________	9. ____________	9. ____________
10. ____________	10. ____________	10. ____________

Write a quintessential statement explaining in words a (virtual) 10 year old sibling would understand what contingency is.

But why (they ask), if contingency exists, does this procedure always end up at the same place - the target phrase?

Ok, Ok, we can see what is happening. But how is it happening?

10. Experimental Record Ten: A Thought Experiment: What does Wordevol do?

Again, same virtual 10 year old sibling, explain quintessentially what is going on in the program Wordevol . That is, by what steps and procedures does the program manipulate the data? You may go back through the program to clear up points you don't understand.

11 Experimental Record Eleven Thought Experiment: How do the patterns arise?

Are Wordevol and a Cellular Automata like Life3000 generating order by the same principles?

Accept or Reject: The patterns and order arising from Cellular Automata systems, like the Life3000 experiments we did in an earlier lab, arise by the same processes operating in the Wordevol experiments.

Which Mutation/Selection Strategy Works Best?

12. Testing Mutations Things change . . . and they change in many different ways. Some ways of changing may work better in some situations, or for some purposes, than others. What we are after here is ***efficiency***.

12. Experimental Record Twelve: Testing Mutation–Selection Efficiency

Read the explanations in the program for each mutation strategy and then answer the questions below.

Based on your understanding of each mutation/selection strategy, which one do you think is the most efficient; that is, will produce the most rapid evolution? (Descriptions of each mutation/selection strategy are found in at the end of the lab, or in the "Explanation" boxes in the program.

Mutation/Selection Strategy 1, 2, 3 (circle one)

Why? What rational/logical explanation can your provide for your choice? This is Top-Down.

Now let's test your ideas. Write down a new target string. Keep it less than 15 characters; one selection strategy is inefficient enough it locks up the program if the string is too long.

Run WORDEVOL with the above string use each mutation/selection method, and record how many generations it took to select to the target string. This is Bottom-Up.

___________ generations: Mutation 1- Attenuating Selection

___________ generations: Mutation 2- Serial Selection

___________ generations: Mutation 1- Optimizing Selection

Were your deductions correct? Did you reason out the most efficient mutation/selection strategy? If not, do you have any ideas why the most efficient strategy was in fact the best?

But, of course, this was only one test. Run this experiment 3 more times with different target strings and see if the same mutation/selection strategy is the best, even with these new target strings.

Target String 2 ________________________________

___________ generations: Mutation 1- Attenuating Selection

___________ generations: Mutation 2- Serial Selection

___________ generations: Mutation 1- Optimizing Selection

Target String 3 ________________________________

___________ generations: Mutation 1- Attenuating Selection

___________ generations: Mutation 2- Serial Selection

___________ generations: Mutation 1- Optimizing Selection

Target String 4 ________________________________

___________ generations: Mutation 1- Attenuating Selection

___________ generations: Mutation 2- Serial Selection

___________ generations: Mutation 1- Optimizing Selection

WORDEVOL DATA TABLE

STEP ONE - You Choose a Target Phrase

Write down the phrase that you chose.

☞ ______________________________

STEP TWO - Finding How Many Random Strings of Equal Length Exist

☺__________ Number of random strings generated in one second?

☺__________ Number of distinct strings this length?(numbers in exponents; 1.57e+27 = 1.57 x 10^{27}).

☺________ sec. Estimated seconds, days, years for this computer to generate randomly one match.

☺________ days

☺________ yrs.

STEP THREE - Let's Try To Match Target String Randomly

☺__________ How many random strings to generate?

☹ In the space below write the closest randomly generated match to your original string.

☞ ______________________________

☹ And how long it took to generate it.

☞ ______________________________

STEP FOUR - Cumulative Selection

☺ Write down "Initial string generated by computer at random."

☞ ______________________________

#1 - Attenuating Selection

Match found in generation?	☺________	☺________	☺________	☺________
Checking how strings?	☺________	☺________	☺________	☺________
In how many seconds.	☺________	☺________	☺________	☺________

#2 - Serial Selection

Match found in generation?	☺________	☺________	☺________	☺________
Checking how strings?	☺________	☺________	☺________	☺________
In how many seconds.	☺________	☺________	☺________	☺________

#3 - Optimizing Selection

Match found in generation?	☺________	☺________	☺________	☺________
Checking how strings?	☺________	☺________	☺________	☺________
In how many seconds.	☺________	☺________	☺________	☺________

CONTINGENCY

Contingency is the idea that what you get is largely controlled by what you start with. Think of it as a branching path. Every time you choose one branch all the possibilities on the other branch are closed down to you.

Under **STEP FOUR** do three runs starting with the same (short) target string (get it to fit on the lines below) and same mutation strategy

☞ In the space below write the randomly generated initial string.

☞ Mutation Strategy?

#1 - Attenuating Selection **#2 - Serial Selection** **#3 - Optimizing Selection**

☞ Then, for each run go to the TEXT FILE FINAL RESULT and scroll down to the final 10 strings as they mutate and select toward the target string, and write down those ten.

Run One	Run Two	Run Three
Last 10 Mutations	Last 10 Mutations	Last 10 Mutations
1. ________	1. ________	1. ________
2. ________	2. ________	2. ________
3. ________	3. ________	3. ________
4. ________	4. ________	4. ________
5. ________	5. ________	5. ________
6. ________	6. ________	6. ________
7. ________	7. ________	7. ________
8. ________	8. ________	8. ________
9. ________	9. ________	9. ________
10. ________	10. ________	10. ________

☺ F ☺ i ☺ n ☺ i ☺ s ☺

WordEvol Mutation Strategies

Explanation - Mutation 1: Attenuating Selection -

1. Generate at random one string of letters (the initial string) of the same length as the target string.
2. Generate a second string at random from the first.
3. Compare both the initial and second strings against the target for similarity distance.
4. Select the closest string (the survivor).
5. Repeat steps 1 and 4, beginning with the survivor string.

Generate one string at random, the initial string, of same length as target string. From the initial string generate a second random string. Compare both initial string and second string against the target and calculate a "similarity distance" for each. Select the closest string and repeat the procedure until a match is found for the target string.

The similarity distance is calculated by finding the distance each letter is from the correct letter in the target string, and then summing all the distances. The string with the smallest distance is chosen.

Distance is calculated by comparing each random letter with its target letter. Say the target letter is 'D', 'A' is 4 units away and 'K' is 7 units away, so 'A' is chosen. This comparison is made for all unmatched positions on the target string.

Explanation - Mutation 2: Serial Selection -

1. Generate at random 20 strings of the same length as the target string.
2. Pass 20 strings through a selection filter, comparing each of the strings with the target string.
   - String with most correct matches is retained (the survivor string). All correct matches are locked in as functional, and will not mutate again.
   - If more than one string has same number of correct matches, the first string generated with the highest number is retained and the others discarded.
   - If no matches are found discard all strings and go to step 1.
3. Repeat steps 1-2 beginning with survivor string.

Generate 20 strings at random of same length as target string. Compare each of the 20 strings with the target string looking for exact matches. Lock matches in place, retain the string with the most exact matches, and use it as a base to generate 20 more mutated strings. Repeat until a match is found.

Explanation - Mutation 3: Optimizing Selection - WARNING: Do not use this mutation technique with strings over approximately 15 characters long.

1. Generate at random 20 strings of letters of the same length as the target string.
2. Compare each of the 20 mutated strings with the target string.
   - Strings are examined in the order they are generated. The string found having a match for the first unmatched position and also has the most matches in the remaining positions, regardless of their place in the string, is chosen (the survivor string); all others are discarded.
   - If none of the 20 strings have a match they are all discarded and a new random set of 20 strings generated.
   - Once a match is found for the first position it is locked in place as functional and will not mutate again.
3. Repeat steps 1 and 2 beginning with survivor string until all positions are matched.

Generate 20 string at random of same length as target string. Compare all strings for first position only in the target string until a match is found, the selection string. Lock the correct letter in so it will not mutate again and discard all remaining strings. Beginning with the selection string generate at random 20 more strings and search for match for second position. Continue until target string is created.