|  |
| --- |
| Evolving Titles |
| Generating book titles by evolutionary computation and neural networks |
|  |

## Dormán Hanga

## Badics Milán - Machine Learning

## Rajk Szakkollégium 2018 Fall

2018. 12. 20.

# Abstract

In the following paper I introduce a computational model designed to generate book titles. The components of the model represent fundamentals of human creative thinking, such as free association performed through a semantic network of concepts and evaluation of ideas based on former experience. The model uses WordNet®, an open source semantic graph of English words to generate a whole population of new title ideas, which are then improved by evolutionary algorithms. Fitness of the individual ideas in a population is evaluated by a neural network, previously trained to differentiate real book titles from rudimentary machine-generated pieces of text. Results show a significant increase in the overall score given by the fitness evaluator in the successive generations of text. However, the verbal quality of the generated titles starts to deteriorate after a number of iterations, which may indicate that the program is stuck in a local optimum and calls for further improvements.

# Table of contents

[I. Introduction 1](#_Toc533094495)

[II. Approaches to creativity in psychology and computational science 2](#_Toc533094496)

[II. 1 What is creativity? 2](#_Toc533094497)

[II. 2 Creative association in computational models 3](#_Toc533094498)

[II. 3 Creativity by evolution 4](#_Toc533094499)

[III. *POEVOLVE* – Levy’s limerick generator 5](#_Toc533094500)

[IV. Evolving Titles Model 6](#_Toc533094501)

[IV. 1. *Idea* - Random text generator 6](#_Toc533094502)

[IV. 2. *Experience and Reflection* - Evaluation with neural network 7](#_Toc533094503)

[IV. 3. *Evolution* – Improving text with evolutionary algorithms 8](#_Toc533094504)

[V. Results 9](#_Toc533094505)

[VI. Conclusion 11](#_Toc533094506)

[References 13](#_Toc533094507)

[Figures 15](#_Toc533094508)

# I. Introduction

The following model was inspired by the work of Robert Levy, Cognitive Science and Linguistics student at Oswego State University. In his thesis [Levy, 2000] he introduced *POEVOLVE*, a limerick generator program based on evolutionary computation and recurrent neural networks. I designed my model according to the basic architecture of *POEVOLVE*, altering specific details.

The major motivation behind my choice was to explore the psychological aspects of creativity in the frame of a computational project. Trying to grasp the essence of the human mind’s creative facility through algorithms is a very impressive objective, which may bring us closer to understanding the fundamental nature of human intelligence. Not to mention what creative AI may achieve with respect to innovation, marketing, problem solving and even scientific breakthroughs.

My version of the model, *Evolving Titles* is designed to create book titles that are similar in creative value to the ones invented by human authors. I distinguished four structural units of the program by the names *idea*, *experience*, *reflection* and *evolution*, which represent a random text generator, the collection of input data for the neural network, the training of the network and the evolutionary selection algorithms, respectively. In the following sections, after a brief introduction to the concept of creativity and its computational and evolutionary approaches, I will elaborate on the precise structure of the program, evaluate the results and highlight some aspects for further research.

# II. Approaches to creativity in psychology and computational science

### II. 1 What is creativity?

There are several definitions of creativity to be found in the literature of psychology, however, most of these agree on two essential factors: originality and functionality.

According to Sternberg and Lubart creativity is “the ability to produce work that is both novel (i.e., original, unexpected) and appropriate (i.e., useful, adaptive concerning task constraints).” [Lubart and Sternberg, 1999, p 3.]

In an early research paper about creativity, psychologist Sarnoff A. Mednick describes the process of creative thinking as the “forming of associative elements into new combinations which either meet specified requirements or are in some way useful.” [Mednick, 1962, p 221.] He highlights the difficulty of assessing usefulness, but resolves this problem by restricting to cases when it can be arbitrarily defined by the researcher. As for the other important criterion of creativity, he describes the *originality* of a response to a certain stimulus as being inversely proportional to the probability of its occurrence in a given population. He also introduces the concept of *associative hierarchy*: the probability and speed (or *associative strength*) of making a connection between two concepts as a function of the originality of the connection. The flatter the associative hierarchyis of an individual, the greater their creative ability – meaning their associative strength only shrinks moderately with respect to more remote associations. [Mednick, 1962, p 222.]

Mednick describes three ways of producing creative solutions. The association of elements may be evoked by *serendipity* (accidental contiguity of stimuli in the environment), *similarity* of the stimuli, and *mediation* between different stimuli through their common elements. [Mednick, 1962, pp 221-222.]

The aspects of creativity described above form the basis for designing creative algorithms, which I elaborate on in the next section.

### II. 2 Creative association in computational models

In one of his case studies, artist and software engineer Tom De Smedt [2013] models creative idea flow by interpreting the mind as a semantic graph. The nodes of the graph stand for the individual concepts, while edges represent the different types of relations between them, such as *is-a, is-part-of, is-opposite-of, is-property-of, is-related-to, is-same-as* and *is-effect-of***.** The graph he uses is a manually constructed database that can be thought of as the equivalent of all the cultivated knowledge and experience within a human mind. The model can also broaden its knowledge base by mining the Internet for the properties of a yet unknown concept.

To create ideas, the program uses the *concept halo*: the network surrounding a specific node to a given depth. By a process called *spreading activation*, all the related concepts can be retrieved for a node. The deeper is the halo, the broader circumscription we get. With the help of *similarity* and *mediation* between concepts through their common properties, the program is able to make associations, such as the slogan “*Brussels, the toad*” or the inference that bats are creepy animals.

Smedt evaluates the originality of ideas generated by the model based on the number of their Google Search results. As for the assessment of appropriateness, the model lacks a powerful feature: a conscious feedback loop to reflect on its own ideas. (In my model, a trained neural network is meant to function as such a feedback loop, hopefully making up for some of the simplifications I made in the association process, due to the limited scope of the project.)

Francisco et al [2018] experiment with the artificial creation of riddles and rhetorical figures. They emphasize the importance of the combination of *lateral* and *vertical* thinking. Vertical thinking is the way of understanding conceptual similarity in a hierarchical semantic structure. Lateral thinking requires a much more creative approach that makes abstract connections such as the analogy between a *lawyer* and a *shark* possible, based on their common properties. To perform such associations, unconventional usage of words needs to be included in the knowledge base.

### II. 3 Creativity by evolution

Evolution is a blind process, meaning it has no predefined goal. Certain accidental genetic and behavioral patterns enhance reproductive success, while others work against it. Thus, self-reinforcing features will proliferate in a given population, leading to the refinement of organic structures.

In computational science, evolutionary algorithms mime the strategy of nature in order to converge to optimal solutions. Such an approach is especially helpful when there is no way to express the problem to be solved in a closed form and creativity is required. A constructed measure of fitness is selects the best candidates to ground the genetic base for the following generations.

An important aspect is to maintain diversity by introducing random mutation, in order not to converge too fast and getting stuck in a local optimum, emphasizes De Smedt [2013] in his case study called *EVOLUTION*. Later on he presents how creative behavior can emerge as a result of genetic algorithms, by having virtual creatures adapt to their encoded environmental circumstances. Due to competition, weaker features fade away and creative problem solving is reinforced. The model is hierarchically fair, meaning that highly evolved creatures compete with newcomers as well, who may occasionally beat them due to a surprisingly effective feature. Such hierarchical fairness is excluded from my model (where the evolving “creatures’ are pieces of text), as only successors of earlier generations compete in the genetic pool. However, random mutations and the survival of some weaker specimens are included to ensure diversity.

### III. *POEVOLVE* – Levy’s limerick generator

In his thesis, Robert Levy [2000] describes a limerick generator program based on evolutionary computation with neural network as fitness evaluator. This is a short introduction to the architecture of his program.

The Limerick Generator Toolkit includes all lexical tools necessary to generate text in the appropriate limerick form, involving words, phonemic properties and grammatical structure. Levy applied a recurrent neural network to evaluate the poems generated randomly with the help of the toolkit. To gather input data, he conducted a survey with university students, asking them to evaluate a set of limericks, consisting of both human- and machine generated samples and used the results to train the network.

The highest layer of the model, *POEVOLVE*, is a set of genetic algorithms. *POEVOLVE* sorts out the ‘fittest‘ samples of limericks based on the output of the neural network, and performs several mutation and crossover algorithms on them to create the next generation of text. The process is repeated a sufficient number of times with the objective to produce limericks advanced enough to compare to human poetry.

Levy evaluated the performance of *POEVOLVE* by testing whether there is a significant increase in the score of consecutive limerick generations given by the neural network. Both Pearson’s test for correlation and the T-test confirmed improvement in the final ratings compared to the original ones. However, contradictory results have also been found by other tests. A robust success of the program therefore cannot be established, which calls for further improvements and testing. See below an example of the limericks generated by *POEVOLVE.* [Figure 1.] In what follows, I describe my version of the model.

“drawl hiding efficient plan pity

idea remind whale life pretty

George lover stiles fast

bore naples shoe passed

alas known aspiring path city”

1. Figure - Limerick generated by POEVOLVE [Source: Levy, 2000, <https://github.com/rplevy/poevolve/blob/master/thesis/compile/examples.txt>]

### IV. Evolving Titles Model

### IV. 1. *Idea* - Random text generator

As a lexical base for text generation, I use WordNet® (2010) of Princeton University, an openly available database of English words, organized into a semantic graph structure. Operations with WordNet® are supported by the Python package NLTK. Each word in WordNet® is stored in the form of *synsets*, by which all concepts semantically related to the word are accessible. In the model I use the relations *hypernym*, *hyponym*, *member/part/substance holonym*, *member/part/substance meronym* and *similar to*. WordNet® does not contain prepositions, so I downloaded a list of the most common English prepositions and their frequency in the average use of the language.

The *idea* layer includes a random association function that accepts a word as argument and returns another word: the endpoint of a randomly activated semantic path. How far the path reaches is a stochastic variable following an exponential distribution (with scale parameter arbitrarily set to 2). This distribution is basically meant to represent Mednick’s *association hierarchy*. Choosing a different distribution form or changing its parameter allows for more, or less remote associations.

The text generator function picks randomly from the set of all alphabetic-only words of WordNet® and calls the association function to create a sequence of semantically related words. The sequence is occasionally completed with other randomly chosen words and prepositions sampled according to their frequency. So far, this process mainly builds on *serendipity* and *mediation*, and does not follow any strict grammatical or semantic rule, as the program is expected to adapt to such rules during the evolution process. See some randomly generated samples below. [Figure 2.]

‘whitening in bleach at’

‘vergil plate of horseshoe’

‘seamed join in bob’

‘vigilantly remarry on solemnize dress’

‘collectivist coherence quality incomprehensibility’

2. Figure - Randomly generated pieces of text [Source: own]

### IV. 2. *Experience and Reflection* - Evaluation with neural network

To train the fitness evaluator, I collected input data in the form of actual book titles and pieces of text generated randomly by the algorithm I described above.

The *experience* layer searches for book titles in the WorldCat database [worldcat.org], by a keyword chosen randomly from WordNet®’s list of words. Altogether, I collected 4000 real book titles and generated another 4000 randomly. I gave each piece of text a single dummy value that equals one if the text is from the WorldCat database, zero otherwise. The sample texts are encoded as arrays, where *n* is the length of the longest text. Each value of the array is the index of the corresponding letter in the alphabet (plus space).

Using the Python package Scikit-Learn, I ran a Grid Search with several model types and hyper parameters, including *Linear Logistic Regression, Multi-layer Perceptron Regressor, Lasso Regression, Elastic Net, K-Neighbors Regressor, Decision Tree Regressor* and *Random Forest Regressor,* to achieve better model performance. Eventually, *Linear Logistic Regression* proved to be the best fit for the data, classifying 81% of both the training and the test set correctly. I saved the model for later use during the evolution process. [Figure 3.]

Model name: LogisticRegression

Fitting 8 folds for each of 1 candidates, totalling 8 fits

[Parallel(n\_jobs=2)]: Done 8 out of 8 | elapsed: 5.0s finished

Test set score: 0.81

Best parameters: {'max\_iter': 10000}

Best cross-validation score: 0.81

Best estimator:

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,

intercept\_scaling=1, max\_iter=10000, multi\_class='ovr', n\_jobs=1,

penalty='l2', random\_state=None, solver='liblinear', tol=0.0001,

verbose=0, warm\_start=False)

3. Figure - Output of Scikit-Learn's Grid Search

### IV. 3. *Evolution* – Improving text with evolutionary algorithms

In this layer, the first generation of text is produced by the random text generator and evaluated by the logistic model. The top fifty percent of the text is selected as a genetic pool for the next generation, which is produced by mutation and copying processes. A fragment of survivors and some randomly sampled weaker specimens are directly copied. The remaining samples’ properties are slightly altered in order to ensure genetic diversity. The mutation algorithm chooses some of the words of a piece of text and replaces them with semantically related or randomly chosen new ones. The same pieces of text are also processed by another version of the algorithm, which keeps all the component words but shuffles their order. This way, different types of potentially beneficial properties may be reinforced in the next generation. The process is repeated a sufficiently large number of times. The overall fitness of the successive generations is stored for later evaluation of the model.

# V. Results

After running the program several times, I obtained mixed results. An unequivocal improvement in the average score given by the logistic model is present in the successive generations: after a steep upgrade it slowly converges to one. [Figure 4.]

4. Figure - Average model score of successive title generations [Source: own]

However, looking at the qualitative aspect of the results, we can witness a classic case of evolution going wrong. Repetitive sequences of rare and useless words start to proliferate. [Figure 5.] This may indicate that the model is stuck in a local optimum, or the evaluator regression is not powerful enough to extract the most important features of human-generated text. Also, it is a weakness of Logistic Regression that its output is binary and therefore cannot differentiate precisely enough between the samples.

‘shine pusher baryta’

‘rodent air nailer’

‘wage downhill barium’

‘caff knock jump’

‘ale barium body’

[Generation: 100]

‘abohm memsahib abohm’

‘ohm sahib ohm’

‘ohm ohm memsahib’

‘sahib ohm ohm’

‘ohm ohm sahib’

[Generation: 500]

5. Figure – Samples from generations 100 and 500 [Source: own]

# VI. Conclusion

As I described in the first chapter, there are two important components of creativity: appropriateness and originality, which are difficult to assess. In our case the fitness evaluator did not prove to be a proper objective measure to rely on. Browsing the titles generated by the model manually can give us an intuition about its performance, but is not sufficient for an overall evaluation). For further improvement of the model, the followings are to be considered.

*Idea*. Evolution may be more effective, if certain grammatical rules are fixed for the random text generator. This way, basic linguistic criteria are established from the beginning and so completely incoherent constructions are excluded. In my version of the random text generator, association is a mere listing of conceptually related words, which is a poor exploitation of WordNet®’s informational content. A more sophisticated association algorithm could collect more data about the individual concepts and activate semantic paths in a lateral thinking style (which I described in the first section). To illustrate my point, the program could be taught to purposefully connect the words *mind* and *knife* by the concept of *sharpness*.

*Experience and Reflection.* There’s plenty of space for improvements regarding the collection of the data, the encoding of the characters and the structure of the model applied to evaluate the text. Extracting the properties of good quality human-generated text is a very difficult task, given the complexity of language. However, recurrent neural networks perform this task surprisingly well [Karpathy, 2015] and may significantly enhance the performance of Evolving Titles.

*Evolution*. The evolution process can also be refined by introducing other types of mutation and crossover algorithms and optimizing over the parameter space (where parameters are the proportion of survivors in a generation and the scale of diversity).

After these improvements are carried out, the model can again be evaluated using the objective measure of the model score and the more subjective criteria of originality and functionality.

# References

De Smedt, T. (2013): *Modeling Creativity. Case studies in Python,* Antwerpen University. Downloaded: https://www.clips.uantwerpen.be/sites/default/files/modeling-creativity.pdf, 17 December 2018

Francisco, V. et al [2018]: *Exploring the Potential of Concept Associations for the Creative Generation of Linguistic Artifacts: A Case Study With Riddles and Rhetorical Figures,* Frontiers in Psychology, Volume 9, Issue 1792. Downloaded: <https://www.frontiersin.org/articles/10.3389/fpsyg.2018.01792/full>, 19 December 2018

Karpathy, A. [2015]: *The Unreasonable Effectiveness of Recurrent Neural Networks.* Downloaded: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>, 19 December 2018

Levy, R. (2000): *Limerick Generation by Evolutionary Computation with Neural Network as Fitness Evaluator: A Simulation of Human Creativity,* Oswego State University, College Honors Program. Downloaded: <https://github.com/rplevy/poevolve/blob/master/thesis/doc/levythesis2000.pdf>, 17 December 2018

Lubart, T. I. & Sternberg, R. J. (1999): *The concept of creativity: prospects and paradigms,* Cambridge University Press. Downloaded: <https://books.google.hu/books?hl=hu&lr=&id=d1KTEQpQ6vsC&oi=fnd&pg=PA3&dq=The+concept+of+creativity:+prospects+and+paradigms&ots=Ft-0YhumsW&sig=TtOauJtuQYviYZCF_C6kbBBUfa0&redir_esc=y#v=onepage&q=The%20concept%20of%20creativity%3A%20prospects%20and%20paradigms&f=false>, 17 December 2018

Mednick, S. A. (1962): *The associative basis of the creative process*, Psychological Review, Volume 69, Issue 3, pp 220-232. Downloaded: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.170.572&rep=rep1&type=pdf>, 17 December 2018

*Top 50 Prepositions*, on TalkEnglish.com. Downloaded: <https://www.talkenglish.com/vocabulary/top-50-prepositions.aspx>, 18 December 2018

WordNet® (2010): *An Electronic Lexical Database*, Princeton University. Downloaded: [https://WordNet®.princeton.edu/](https://wordnet.princeton.edu/), 17 December 2018

Scripts and output files are available on GitHub: <https://github.com/dormanh/Evolving-Titles>

# Figures

[1. Figure - Limerick generated by POEVOLVE 5](#_Toc533094271)

[2. Figure - Randomly generated pieces of text 6](#_Toc533094272)

[3. Figure - Output of Scikit-Learn's Grid Search 7](#_Toc533094273)

[4. Figure - Average model score of successive title generations 9](#_Toc533094274)

[5. Figure – Samples from generations 100 and 500 10](#_Toc533094275)