To Evolve and Optimise Colour Sets for Website Templates Using Genetic Algorithms

Balázs Pete

09771417

Abstract

In this project, the idea of evolving colours will be investigated. Some background on the possible need of such technique will be presented, detailing the methods used in order to thest the validity of such idea. Results of the eperiments will be stated, showing the change in the colour-set population over the experimental process. Finally, some of the issues with the results will be highlighted, and a proposition to resolve these problems will be given.

1 Introduction

One of the most important aspect of a website are the look and feel, how the website is presented, what the design looks like, as it is one of the main factors that determine the usability of a website.

As it can be seen in most current websites, the styling relies on a set of colours which when chosen with care, can greatly uplift the quality of the overall look, however when chosen poorly it can repel users. A typical colour set would concentrate on one or two main colours, these being in most cases for background and foreground, such as text colour. The set would contain different shades of these main colours, usually chosen to fit the different roles they would take. As an example, most websites use plain white as their background, however a container within that site would have slight shade of grey as its background and have a dark colour, such as black, for fonts. The creation of an optimal colour set, one where the colours match nicely together, is sometimes a long and tedious process, as even a minor adjustment to one colour may offset the overall niceness, the measure of how much the colours fit well together in a human perspective, of the colour set.

The aim of this project is to see whether Genetic Algorithms (GAs) are an effective tool to develop and optimise colour sets for website templates, and to analyse how different combinations of GA methods can be applied to find an optimal algorithm to solve the problem.

The outline of this document is the following: In the next section, the experiment will be presented with a detailed report on the steps taken. The third section will present the results of the experiment, and their interpretation. Finally the conclusion will sum up the report and will suggest where improvements can be taken to further experiment in this area.

2 Theory and Experimental Process

The aim of the project is to generate a nice colour set from a starting set of randomly generated colours. Due to the computers' inability to tell what colours go nice together by default, human interaction is needed. However evaluating colours for each iteration of a genetic algorithm is inconvenient hence another solution needs to be found.

Artificial neural networks (NN) are a type of mathematical model, which try to simulate the functionality of a human brain. A NN consists of several nodes, called neurons, interconnected using weights, in order to allow signals to travel from one end of the network to the other. Upon receiving signals, a node calculates a weighted sum of all of its inputs, and generates an output if the result of the operation reached a predefined threshold. A single node is not capable of powerful calculations, however the power of the neural network lies in the combined result of a vast number of such nodes [4].

Neural Networks are effective tools of classification in machine learning, when the possible relationship between the samples is unknown, or if it is very hard to detect or encode. In theory, if a large amount of sample data is available, in our case a large amount of colour-sets that are defined as nice or not nice, the NN can be trained to classify these sets, and in addition it should be able to learn the relationship between these sets in order to be able to classify other unknown colour-sets afterwards.

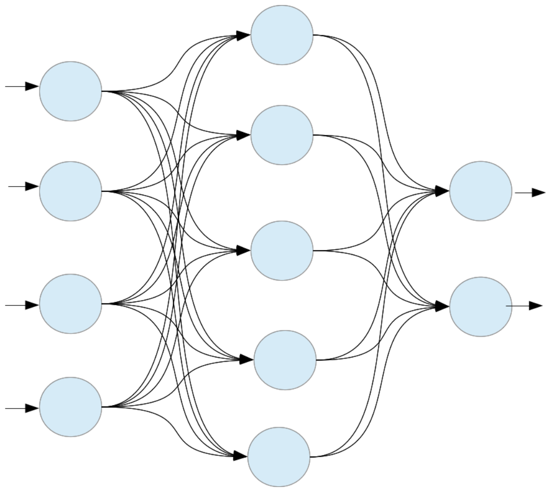


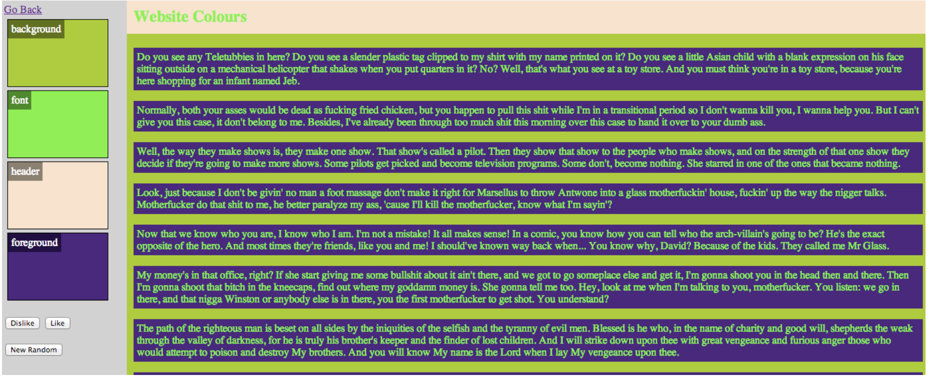
Figure 1. A diagram showing the structure of a Neural Network

To be able to train the neural network, a domain of pre classified samples are required, however this amount is too big for a single individual to collect within a reasonable amount of time, and in addition the evaluation of these colour-sets would be biased towards the classifier, as in the person collecting the data, which is not a side effect that we desire.

An effective technique in artificial intelligence is the idea of crowd sourcing. The idea is that if faced with a difficult problem, or a problem with a large domain, breaking the problem down into small problems and issuing these chunks to a lot of individuals, then combining the result together will give an accurate result more efficiently, than requiring one well trained person to solve the whole, bigger, problem. In our case, crowd sourcing had to be applied to the collection of the sample, pre evaluated data. If small sets of colour-sets are evaluated by a lot of people, it should result in a large domain of available samples. In addition the evaluation results of the samples in this domain should be well averaged across people, and reflect the average evaluation of colour-sets of the people, hence avoiding the issue of a to biased sample domain.

In order to collect such information, an application had to be created. It was built using web technologies, which allowed for quick prototyping of the system. This was an optimal solution, since we only needed a prototype to collect enough information, and had to allow for an easy and very convenient user experience, in order to facilitate the process of collecting data. Due to the properties of the technology, users experienced a quick and easy setup time, as their web browser took care of all of the tedious process.

Upon opening the application, the users were presented with a very simple user interface, with a control panel and a dummy-page viewer. The page viewer was customised by the colours that have been selected on the control panel, where users could alter the four main colours: the background of the page, the foreground of the page sections, the colour of the page header, and the font of the text. Initially, the colours were set to a random configuration, and the users have had the choice to make some modification to these colours. To be able to choose whether the current colour-set is likeable or not, and rate it, the user had to select one of the available actions, like or dislike. Once a colour-set has been rated, the configuration data was submitted to a simple web application server, written in Node.js [5], which then stored the data in a MongoDB [6] instance. When the web application received confirmation from the server that the data had been accepted, a new random configuration was presented to the user.



Website Colours

Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it? Do you see a little Asian child with a blank expression on his face sitting outside on a mechanical helicopter that shakes when you put quarters in it? No? Well, that's what you see at a toy store. And you must think you're in a toy store, because you're here shopping for an infant named Jeb.

Normally, both your asses would be dead as fucking fried chicken, but you happen to pull this shit while I'm in a transitional period so I don't wanna kill you, I wanna help you. But I can't give you this case, it don't belong to me. Besides, I've already been through too much shit this morning over this case to hand it over to your dumb ass.

Well, the way they make shows is, they make one show. That show's called a pilot. Then they show that show to the people who make shows, and on the strength of that one show they decide if they're going to make more shows. Some pilots get picked and become television programs. Some don't, become nothing. She starred in one of the ones that became nothing.

Look, just because I don't be givin' no man a foot massage don't make it right for Marsellus to throw Antwone into a glass motherfuckin' house, fuckin' up the way the nigger talks. Motherfucker do that shit to me, he better paralyze my ass, 'cause I'll kill the motherfucker, know what I'm sayin'?

Now that we know who you are, I know who I am. I'm not a mistake! It all makes sense! In a comic, you know how you can tell who the arch-villain's going to be? He's the exact opposite of the hero. And most times they're friends, like you and me! I should've known way back when... You know why, David? Because of the kids. They called me Mr Glass.

My money's in that office, right? If she start giving me some bullshit about it ain't there, and we got to go someplace else and get it, I'm gonna shoot you in the head then and there. Then I'm gonna shoot that bitch in the kneecaps, find out where my goddamn money is. She gonna tell me too. Hey, look at me when I'm talking to you, motherfucker. You listen: we go in there, and that nigga Winston or anybody else is in there, you the first motherfucker to get shot. You understand?

The path of the righteous man is beset on all sides by the iniquities of the selfish and the tyranny of evil men. Blessed is he who, in the name of charity and good will, shepherds the weak through the valley of darkness, for he is truly his brother's keeper and the finder of lost children. And I will strike down upon thee with great vengeance and furious anger those who would attempt to poison and destroy My brothers. And you will know My name is the Lord when I lay My vengeance upon thee.

Figure 2. The interface of the colour gatherer web application

To approach the required amount of people to collect enough information, the application was advertised on social networking sites, asking for volunteers. Instructions to users were given on a welcome screen before directing them to the application. In total, 146 people have given some input to the system, resulting in the collection of some 1804 samples in 3 hours. For the purposes of this experiment, this size for the sample data set should be acceptable.

The next stage in the experiment was to extract all the information from the database, and convert the data into a binary string, the appropriate input type for an evolutionary algorithm. This process was executed using a short script written in Ruby. The colour-set consisted of four colours, encoded in the RGB format. Each RGB code contained three integers, which have been converted to their 8-bit binary representation. All 8-bit representations then were concatenated, in order, into the final 96-digit representation. Each representation has a corresponding bit, signalling whether the colour set is likeable or not.

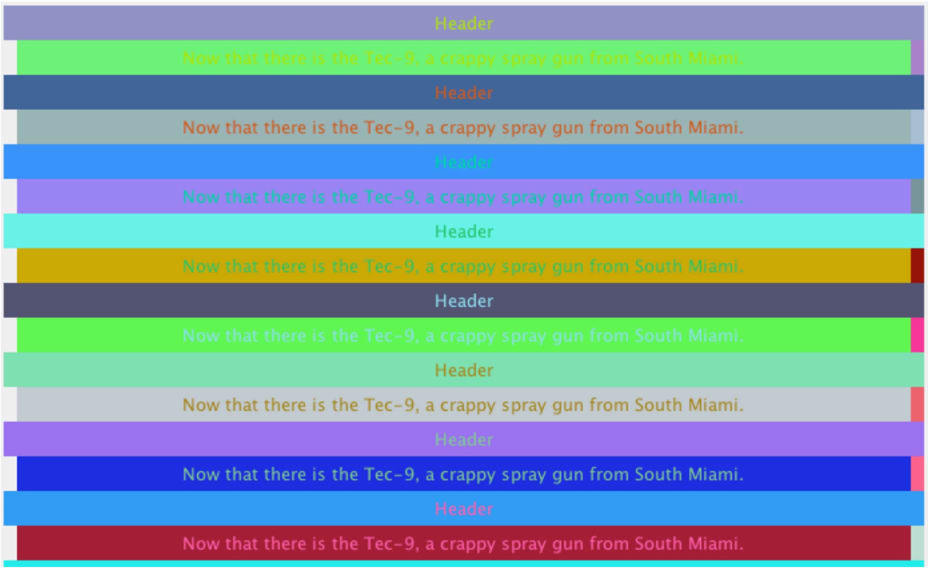
Once the sample data was available, the experiment could be started. However since the algorithm used a neural network as an evaluation function, the network had to be trained in order for it to be useful.

Neuroph [7], a Java based Neural Network package, was used for this purpose. A small java application was written to train a NN with the sample data. The NN had 96 input nodes, and 1 output node. The 96 inputs corresponded to the bit sequence of an input binary string, and the output was set to 1 if the encoded colour set was liked by a user, and to 0 if it was disliked.

Initially, the whole set of data was used for training, however the process took a very long time to execute, and after 24 hours of continuous training, the process was halted. The reason why the process took so long to execute was due to several reasons: as more and more data is used to train the NN, the complexity of adapting the NN for the new data increases, and since the NN package used only took advantage of a single core, the calculations involved in the training were, relatively, too slow. To resolve this issue, a subset of the sample data was used to train the NN. A function was introduced, which selected 500 samples at random from the domain, and then the previous technique was used again for the training, which was successful after running for two hours. The training program saved the NN configuration in a file, allowing the import and use of the NN in other places.

To achieve the aim of this project, a set of random colours was generated. Since colours had a binary string representation, the random generator simply generated a string of length 96 with random amount of 1's and 0's. This representation could be converted into RGB data and displayed visually to a user. To evolve the colour-sets, the GEVA environment was to be used, however interfacing the environment with the neural network package did not work as expected, and the issue could not be resolved within a reasonable amount of time. Therefore the required generic algorithms, selection, mutation and crossover, were additionally implemented.

The experiment was run multiple times using a population of size of 50 for 100 generations. As user evaluation was desired, the algorithm prompted the user to select the best 10 colour-sets at the 50th generation, increasing the likelihood of selection of these sets significantly. In addition, the experiment was repeated with using just the selection and the mutation algorithms, to see how this change has affected the overall result.



Header

Now that there is the Tec-9, a crappy spray gun from South Miami.

Header

Now that there is the Tec-9, a crappy spray gun from South Miami.

Header

Now that there is the Tec-9, a crappy spray gun from South Miami.

Header

Now that there is the Tec-9, a crappy spray gun from South Miami.

Header

Now that there is the Tec-9, a crappy spray gun from South Miami.

Header

Now that there is the Tec-9, a crappy spray gun from South Miami.

Header

Now that there is the Tec-9, a crappy spray gun from South Miami.

Header

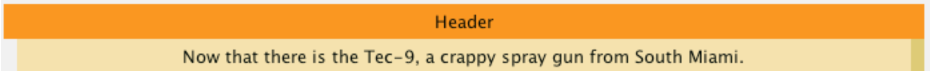
Now that there is the Tec-9, a crappy spray gun from South Miami.

Figure 3. A sub section of a set of randomly generated colour-sets displayed visually. (A bug is present in the visualisation code, however it can be omitted as no mass user validation has taken place)

3 Experimental Results

The experiment gave some interesting results, which are detailed in this section.

As explained previously, for each run of the experiment, the initial population consisted of 50 randomly generated colour-sets. These colour sets then were evaluated by the Neural Network, which assigned a score for each set between 0 and 1. A close score to “1” means that the NN has determined the colour-set to be nice, or likeable. The selection function favoured colour-sets, which had a score of over 0.5, with a probability of selecting the highest scoring set more frequently. Each experiment consisted of evolving the population for 100 generations, with an input from a user mid way the evolution. When the evolution reaches the 50th generation, the user has been presented with a popup where the more favourable colour-sets have been selected. The selected colour-sets' evaluation score has been altered to "1", as they were considered as likeable sets of colours. For the remaining generations, these colour sets have dominated and determined the overall evolution of the colour-sets. A possible improvement to the system would have been the increase of the frequency of user input.



Header

Now that there is the Tec-9, a crappy spray gun from South Miami.

Figure 4. A final colour-set, the result of the evolution

It has been observed that the variation between colours decreased, as the number of generations has been increased. At the end of the experiment the population consisted of identical colour-sets for all runs of the experiment.

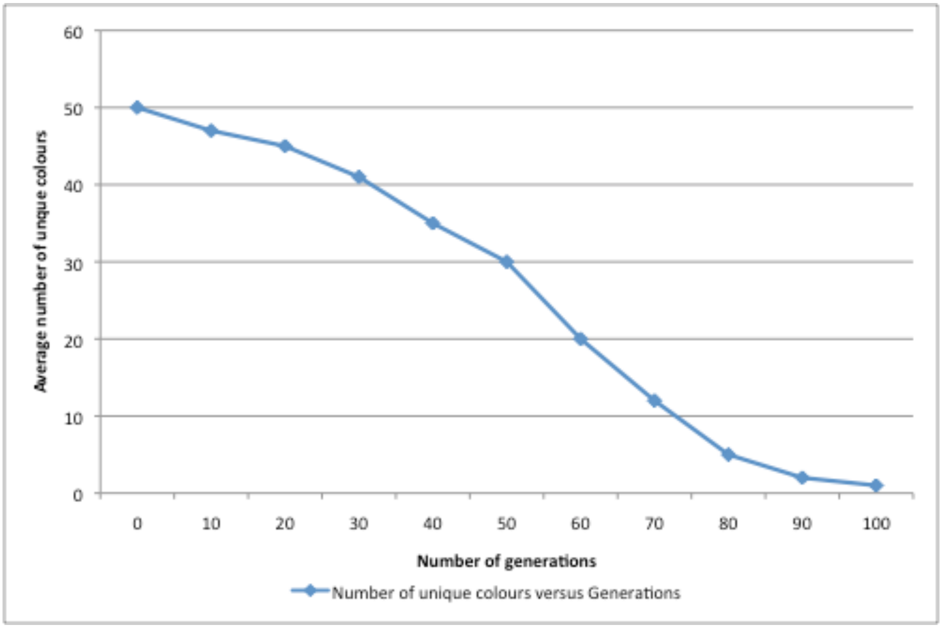


Figure 5. An average of the number of unique colour at key number of generations

When running the experiments with only the selection and mutation algorithms, the same behavious as with the full fledged evolutionary method has been observed. Variation in the population has been decreased as the number of generations has been increased. However, on average, the evolved colour-sets have been less likeable, with some sets evaluated as not likeable at all. This observation was done by a user, not by a Neural Network, placing a heavier weight on the probablility of correctness of the result. The figure below shows the overall likeness result of the evolved colours on a 5 point evaluation scale ranging from 0 to 1.

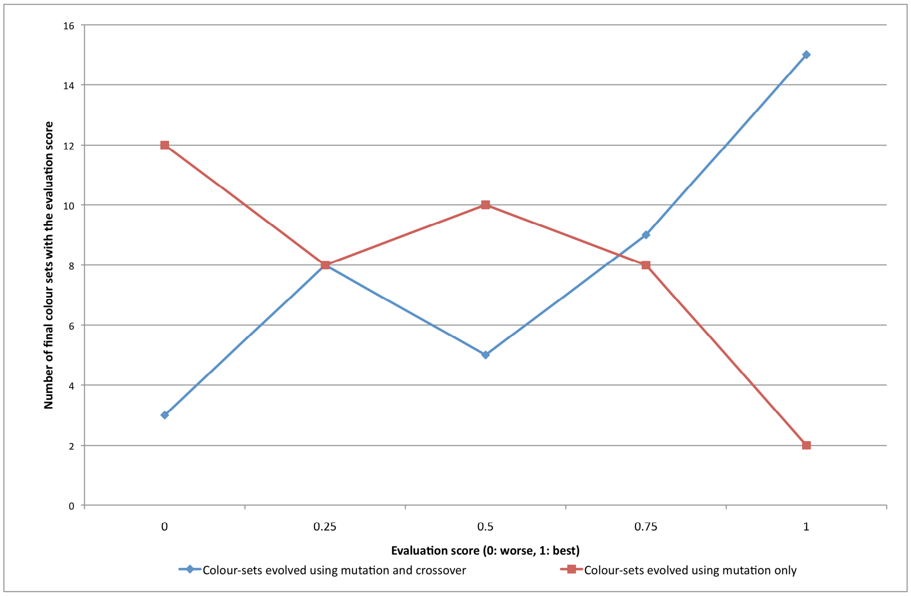


Figure 6. Results of the human evaluation of evolved colour-sets, 30 runs for each evolution method

4 Conclusions

Throughout the experiment, several steps were taken, which have altered the final result to be less reliable. These issues all are the result on one main issue, the unavailability a large base of human evaluators, or the availability of large processing power. One of the sources of was the training of the NN with a limited number of samples. A possible way to test the classification correctness of the trained Neural Network is to contrast compare the colour-sets' classification value with the result given by the NN. A short script was written for this purpose, and run three times. On average, the NN correctly classified 79% of the samples. However the actual correctness value of the Neural Network might be much lower, as the network has been trained with only a limited number of samples.

The NN shows some promise in replacing individuals for the evaluation of colours, however a possible solution to this problem is to apply the idea of crowd sourcing to the evaluation process. The idea is that multiple evaluation experiments would be run concurrently, however these experiments could be frozen until the evaluation function has a result for the selection. A user interface and experience similar to the data collection application could be designed, where users can evaluate the colours of a generation. Whenever a result is available for an evaluation function, the evolution is resumed until the next generation requires evaluation, or the end is reached.

The results of these experiments show, that the concept envisaged in the introduction promises to be viable. However, further experiments are required in order to fully determine likelihood and correctness of this proposition.

5 References

[1] Jean-Charles Pomerol, 1996, "Artificial intelligence and human decision making", *European Journal of Operational Research, Volume 99, Issue 1, 16 May 1997, Pages 3–25*

[2] John R. Koza, 1992, "Genetic Programming: On The Programming Of Computers By Means Of Natural Selection", *Massachusetts Institute of technology*

[3] Sung-Bae Cho, 2002, "Towards Creative Evolutionary Systems with Interactive Genetic Algorithm", Kluwer Academic Publishers

[4] Cross, Simon S., Robert F. Harrison, and R. Lee Kennedy. "Introduction to neural networks." *The Lancet* 346.8982 (1995): 1075-1079.

[5] Node.js – "Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications." - http://nodejs.org/

[6] MongoDB – “MongoDB […] is a scalable, high-performance, open source NoSQL database. ” - http://www.mongodb.org/

[7] Neuroph – “Neuroph is lightweight Java neural network framework to develop common neural network architectures.” - http://neuroph.sourceforge.net/