COLOR COMPOSITION

BY

JOSHUA E. GANG

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Computer Science

Written under the direction of

Matthew Stone

and approved by

_____

_____

New Brunswick, New Jersey

May, 2016

ABSTRACT OF THE THESIS

Color Composition

by Joshua E. Gang

Thesis Director: Matthew Stone

Recent research has used crowd sourced corpora of language to learn grounded meanings that associate color descriptions with uncertain regions in hue-saturation-value color space. In this paper, we explore the degree to which the interpretation of syntactically-complex color terms can be predicted compositionally from their constituents. Using both Elastic Net Regressors and Random Forest Regressors, we build models to predict the composed colors present in both Lux and in the tail data that was unused during the learning of Lux. We evaluate the performance of the models by assessing the learned parameters against the Lux parameters. We additionally look at novel human-generated descriptions and build a system that names colors productively.

Acknowledgements

I'd like to thank my adviser, Matthew Stone, and my friend, Brian McMahan, who kept on helping me when I got stuck. I'd also like to thank Hepburn Best who kept me up at night while I was working.

Dedication

My parents, Ira Gang and Gail Alterman, and my best friend, Ira Herniter.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

We use language to communicate with other people what we are thinking. A speaker formulates an idea using the appropriate words in a language, which are grounded to ideas and structures, and the recipient uses the same knowledge and reconstructs the idea in their own mind. This is a creative process; adjectives, whose syntactic role is to modify and make more explicit a noun phrase, so too modify a base idea. Multiple words also be mixed together to form an amalgamation of the ideas, in a process known as Conceptual Blending [1].

This effect, known as compositionality, can be observed in color in two ways: by combining adjectives such as *light*, *dark*, and *deep* with a base color, and also by compounding the words to generate a blending effect, such as *blue-green*. By exploring how these compositions can occur in color, we create a method to generalize these compositional functions to apply to any grounded color system, with the potential to expand to any grounded system.

1.2 Color

Color is the visual perception property corresponding in humans to different named categories, such as *red, blue, yellow*. These categories derive from the spectrum of visible light interacting in the eye with the light receptors present, the rods and cones. Color labels are also associated with objects and materials ground in the real world based off of their physical properties, such as *pea green* or *brick red*.

There are many ways that we can represent color values, in what is known as a

color space, so that we may describe it using a coordinate system, allowing each color to be uniquely identified and located. The most common representation is RGB, as seen in Figure 1.1. RGB consists of a 3-set tuple of numbers ranging from 0-255, which represent how much each of the primary colors, *red*, *green*, and *blue*, is present in the color. A RGB value of (0,0,0) means that there is no *red*, *green*, or *blue* in the color, and this color is *black*. A RGB value of (255, 255, 255) means that *red*, *green*, or *blue* are maximally present, and this color is *white*.

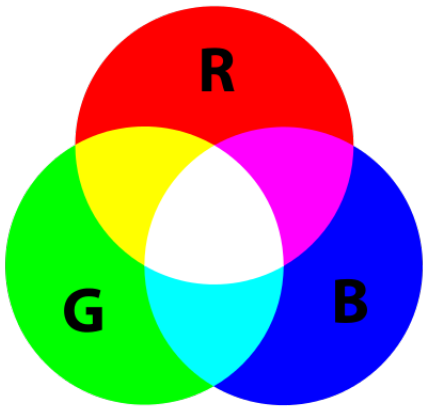

Figure 1.1: A representation of RGB color space [2]. As you mix together different quantities of *red*, *blue*, and *green*, you end up with different resulting colors.

The representation that we use in this paper for color is HSV, or Hue, Saturation, and Value, as see in Figure 1.2. HSV gives us an understanding of what elements make up the color, rather than just what other colors go into it, such as in RGB. Hue is a 360^o wheel, expressing the color base, as seen in Figure 1.4. We also make use of an adjusted-hue space, which maps the 180 to 360 part of the hue space into -180 to 0, in order for ease of calculations when a color crosses the 360/0 boundary, as seen in Figure 1.5. Value is a 0 to 100 greyscale. When the value is 0, the color will be completely *black*. Saturation is a 0 to 100 scale of the how much the base color in the Hue is mixed with the greyscale provided by the Value.

1.2.1 Data

All of the data comes from a survey done by Randal Monroe, of xkcd, in his Color Survey [4], in which he polled over 222,500 users who visited his website, gathering over

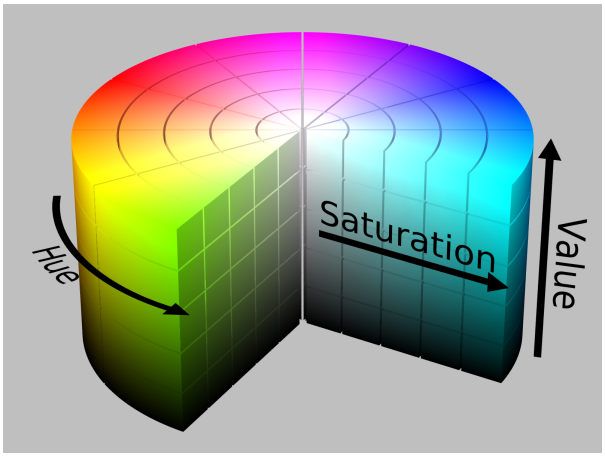

Figure 1.2: A representation of HSV color space [3]. Hue positions us on the color wheel, while saturation and value change around the greyscale components of the color.

5 million color identifications on a broad range of RBG patches. Crowd sourcing such NLP data is significant for several reasons. Instead of running a traditional experiment, which is tedious, expensive and can suffer from sample bias, crowd sourcing the data can allow for a more diverse sample, and one gains the ability to gain many times more data, which can make your data less skewed. The data gathered still suffers from some amount of sample bias, as all of the participants had to be readers of xkcd, but it contains much less than an in-house paid study would.

This crowd sourcing, especially for language, can be very important because we wish to model how the general populace uses language, and the sheer amount of responses gathered help us get an overarching view for not only how words in a language are used to represent certain ideas, but also a measure of what words and phrases are the most popular.

1.2.2 Previous Work

The Lexicon of Uncertain Standards, also known as LUX, is the primary model that we employ. It was developed by McMahan and Stone [5]. LUX captures the vagueness and flexibility of grounded meaning, as uncertainties over the grounded color space, as well as the popularity of the labels. Similar work has also been done in Andreas and Klein [6], where they treat each color as a singular point in space, and then use lexical

analysis in order to determine the vector transformation functions for a color.

Lux

The Lux model for representing color consists of 829 color labels. Each label has an associated model, which in turn is made up of three sub-models, one for each hue, saturation, and value, as well as an availability, α, which both normalizes the distributions and places them on a scale based on how popular the color label is, formally defined as

$$c_\alpha = \frac{\epsilon * |c|}{\int \phi_{Hue} * \int \phi_{Sat} * \int \phi_{Val}}, \tag{1.1}$$

where ϵ[1] is an empirically calculated normalizing constant.

Each of the sub-models individually consists of two Gamma distributions, along with a case function: to the left or right of the centers, the μs, of each Gamma distribution we use the Gamma's survival functions, while between the two μs we consider the probability to be 1. The parameters for each of the individual Gamma are represented by the location μ, the shape β, and the scale γ. The probability, ϕ, of a point, given a sub-model, is calculated by

$$\phi(x) = \begin{cases} \int^x \Gamma(\gamma_1, \beta_1) \text{ If } x < \mu_1, \\ \int_x \Gamma(\gamma_2, \beta_2) \text{ If } x > \mu_2, \\ 1 \text{ Otherwise} \end{cases} . \tag{1.2}$$

This can be seen in Figure 1.3.

The likelihood of any HSV color point, (h, s, v) being labeled as color c, is

$$\theta_c(h, s, v) = c_\alpha * \phi_{c_h}(h) * \phi(c_s)(s) * \phi(c_v)(v). \tag{1.3}$$

[1] .68

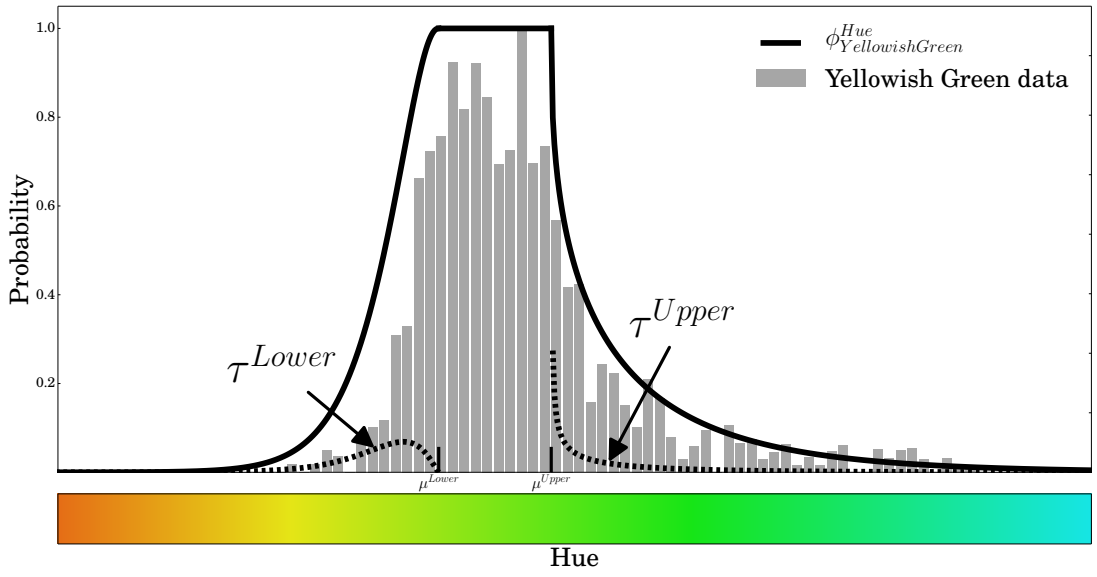

Figure 1.3: The Hue model for Yellowish Green, separated into its separate components [5].

Andreas and Klein

In Andreas and Klein [6], they map a color label to a vector in HSV color space, as opposed to lux which grounds a color label into a set of distributions in color space. They then learn a transformation vector based off of a parse of the color label, and use that vector to transform a base color into a composed color.

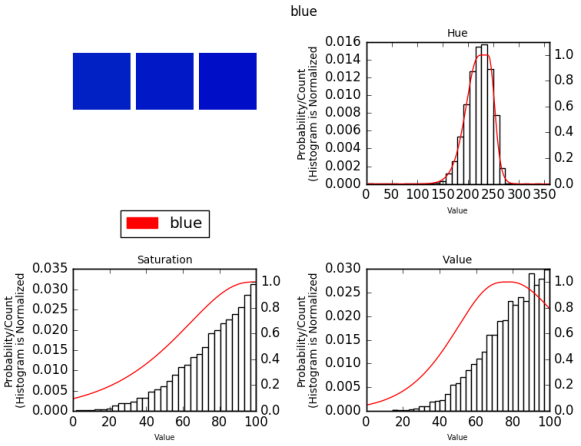

Figure 1.4: The color Blue, according to Lux.

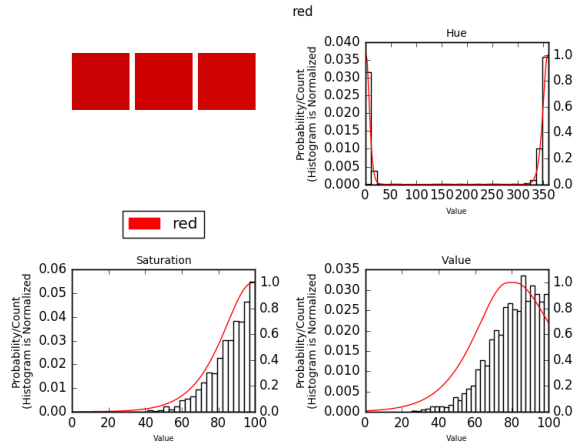

Figure 1.5: The color Red, according to Lux. Note how, on the Hue graph, it crosses through the 0 point on the x-axis to continue at the 360 mark. This is an example of hue-adjusted space.

Chapter 2

Compositions and Experiments

2.1 Compositions

Our aim is to be able to learn composition functions for every type of composition, so that we can take one or more base colors and compose them into a different color, to allow for a more simplified base model. Being able to learn such compositions based off of distributions also generalizes to other types of uncertain distributions outside of color, which provides us a better understanding of the world. We use several regression techniques in order to learn the composition functions.

There are three distinct categories of compositions that we observe in Lux: *adjectives*, *compounds*, and *noun objects*.

2.1.1 Adjective

In Linguistics, an *adjective* is a describing word, whose syntactic purpose is to qualify a *noun phrase*, making it more specific and giving more information about the *noun phrase* that it is modifying. These adjectives also exist in color labels; they do not exist as their own color, their sole purpose is to change other colors, to modify them into a more specific color. A full list of the adjectives found in Lux can be found in Table 2.1. Any adjective that was present less than 4 times was not considered for learning. There are 36 distinct adjectives, composing a total of 296 colors over $19,046$ datapoints in the test dataset. Of these, we cover 97% of them, for a total of $18,472$ datapoints.

2.1.2 Compounds

The second type of composition are the *compounds*. These are all compositions where both of the components are colors, and not just adjectives. The resulting color is a mixing of the two input colors, and is usually referred to as both of the colors at the same time. For example, *yellow green* is a mixing of *yellow* and *green*. Oftentimes these compounds are used instead of a unique label for the associated color, as seen in Figure 2.1. *Yellow Green* and *chartreuse* are extremely similar colors to one another, and yet *yellow green* is approximately 3 times as popular as *chartreuse*.

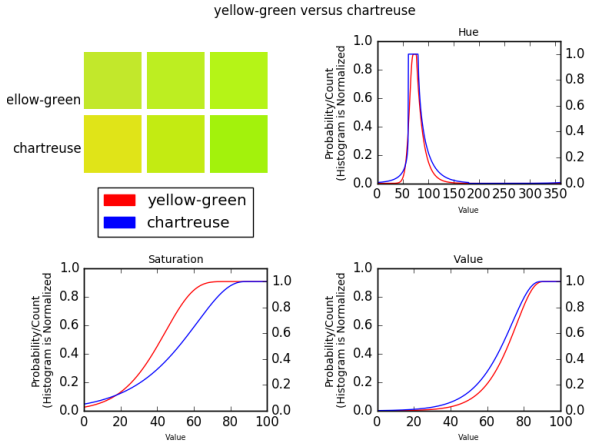

Figure 2.1: *Yellow Green* vs *Chartreuse*. The two colors occupy the same color space and are extremely similar, and yet the Availability for Yellow Green is approximately 3 times as large as Chartreuse, .092 vs .032, meaning that *yellow green* is about 3 times as popular a color label as *chartreuse*

Of this type of composition, we propose that there are three subtypes: *simple compounds*, *y-compounds*, and *ish-compounds*. *Simple compounds* consists of just the two colors mixed together. *y* and *ish* compounds both have linguistic modifications on the first color in the composition, which imply a difference in the mixing of the two colors, as can be seen in Figures 2.3 and 2.4. There is a noticeable difference in the spread of the distributions in each of the different types of compositions. The *simple compositions* tends to be the tightest out of the three, while the *ish-composition* is wider, while the *y-composition* is the largest out of the three. This falls in line with the linguistic interpretations of the suffixes {ish} and {y}. {ish} denotes an approximation around the absolute value of a *noun phrase*, while {y} approximates and abstracts even

further. As such, attaching these suffixes to the first color label in the composition denotes an approximation over the entire composed color label. A look at the data distribution can be found in Table 2.2.

2.1.3 Noun Objects

The final type of composition are the *noun objects*, where the label is composed of a color and a real world object, which ends up in a description of the typical appearance of the object. An example of this is *brick red*, where the *red* that is being described is one typical of bricks. As shown in Figure 2.2, *brick red* is a specific shade of *red*, and this shading is not consistent amongst all noun compositions, and is not consistent with compounds as stated above. These labels look like they should be treated like a normal composition, however they aren't compositions at all, their description is just that of a real world object. As such, we do not do anything with these labels.

2.2 Experiment

By examining the base colors and their composed counterparts in each of the composition types, we are able to derive a function for that composition that we can then apply to any base color in order to modify it into a different color.

We make use of two machine learning methods.

2.2.1 Elastic Net

The Lasso [7], or the Least Absolute Shrinkage and Selection Operator, is a regression method that performs both variable selection and variable regularization, in an attempt to create a simpler model. It does so by forcing all of the coefficients to be less than a certain sum, which forces some of the coefficients to become 0, upon which it chooses the subset of variables that minimizes the objective function, [8]

$$\min_{w} \frac{1}{2n_{samples}} ||Xw - y||_2^2 + \alpha ||w||_1, \tag{2.1}$$

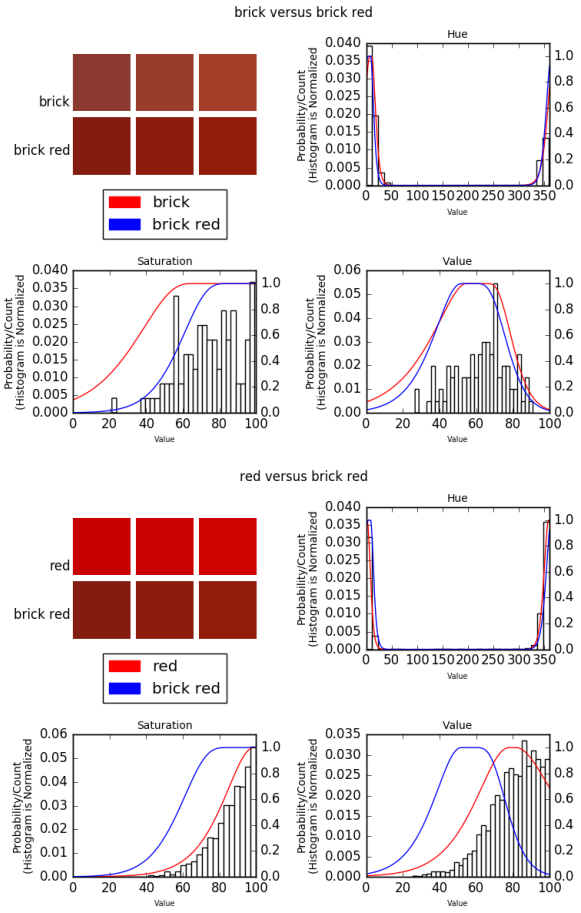

Figure 2.2: *Brick* compared to *brick red*, and *brick red* compared to *red*. *Brick red* is a specific shade of *red*, that relies on real-world knowledge in order to determine what color it actually is.

where α is a constant and $||w||_1$ is the ℓ_1 norm of the parameter vector. However, Lasso runs into issues when the number of covariates p in the regression is greater than the number of samples n. As Lasso can only pick a maximum of n covariates to optimize, it will tend to pick only one covariate from any set of highly correlated covariates in the equation. Even when $p > n$, this can still happen, and ridge regression can also perform better in this instance.

Ridge Regression is a regression technique used to attempt to solve some of the problems of Ordinary Least Squares by imposing a penalty on the side of the coefficients. The ridge regression [8] minimizes

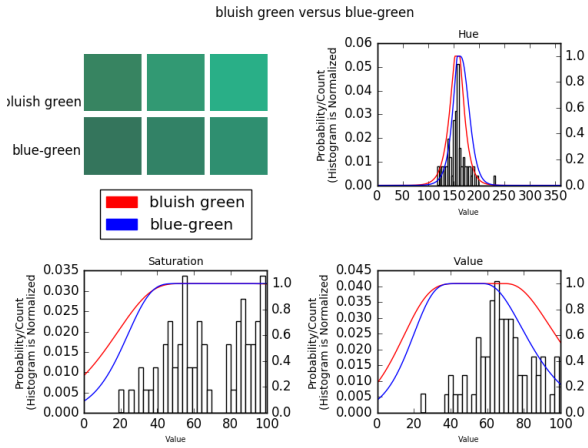

Figure 2.3: *Bluish Green* vs *Blue Green*. While the Hue axis for both colors are almost identical, *bluish green* is wider than *blue green* is, which would imply that the "ish" appended to the first color means that the color is less constrained in its definition.

$$\min_{w} ||Xw - y||_2^2 + \alpha||w||_2^2, \tag{2.2}$$

which is a penalized residual sum of squares, where α controls the amount of shrinkage, with a larger value of α corresponding to more shrinkage, and $||w||_2$ is the ℓ_2 norm of the parameter vector.

The Elastic Net [9] is a regularized regression method that combines the ℓ_1 penalty of the Lasso method, as seen in second part of Equation 2.1 with the ℓ_2 penalty of the Ridge method, as seen in the second part of Equation 2.2, in order to offset the limitations of the base Lasso Method. In order to avoid the issues stated above, the Elastic Net uses the Ridge regression to find the ridge regression coefficients, and then does a Lasso shrinkage on those coefficients.

The objective function for the Elastic Net [8] is

$$\min_{w} \frac{1}{2n_{samples}}||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2, \tag{2.3}$$

We use internal cross-validation on each of the folds in order to determine the values of α and ρ. In essence, the algorithm takes the input fold and uses k-fold on that in order to determine the best values for α, ρ.

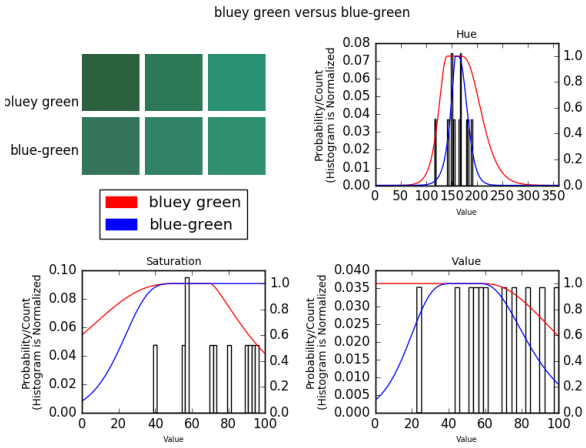

Figure 2.4: *Bluey Green* vs *Blue Green*. Much like *bluish green*, they are both roughly in the same area of color space, however unlike *bluish green*, *bluey green* has a much much wider distribution, which implies that adding a "y" to the first means that you become extremely lose with what you consider to be that color.

2.2.2 Random Forest

Decision trees are a white-box model resembling a flowchart, in which each node in the tree represents a boolean test, and each branch represents the outcome of the boolean test, pointing to more nodes which have their own tests, until an end leaf is reached. In order to evaluate a piece of data, the datum is passed through each node's boolean test, following the tree until it hits a leaf, upon which the leaf's classification or regression is applied to the datum. During training, the tests are determined by what action would create the largest split in the remaining data. Decision trees are cheap to both construct ($(O(n_{features}n^2_{samples}\log(n_{samples})))$ and use ($(O(\log(n_{samples})))$, but have a tendency to overfit, which can happen if they end up creating too many rules, which would not generalize very well. As such, we made use of Random Forests.

Random Forest [10] is an ensemble learning method that uses many decision trees in order to do regressions and classifications. Decision trees tend towards overfitting, and by using multiple trees we can avoid that tendency. Decision trees are trained on a bootstrap aggregation of the input data. In each tree, whenever there would be a split, rather than choosing the best split amongst all possible features, the best split amongst a random sub-sample of remaining features is chosen. This increases the bias of the individual tree, but since we are training multiple trees, the variance will decrease. We

perform predictions by giving the forest the new data, and returning either the mode or the average of all of the tree's predictions.

2.2.3 Methods

We train a set of functions for each of Hue, Saturation, and Value. For each composition type, our training data becomes all of the base lux color models for the composition whose function we are trying to learn, while our target data are all of the composed lux color models for the same composition. For example, for the *light* composition, our input would be the lux models for *blue*, *yellow*, and *brown*, while our target would be *light blue*, *light yellow*, and *light brown*.

For adjectives, for each dimension, we train 6 functions, with the input being the respective variables for that dimension's ϕ curve, $\mu_1, \log\beta_1, \log\gamma_1, \mu_2, \log\beta_2, \log\gamma_2$, while the output is one of the stated variables. We transform those 4 variables into logspace in order to guarantee that they never go below 0. For each of the mus, we train an ElasticNetCV regressor, with 10,000 iterations and 100 αs, while for the other 4 variables, we trained a Random Forest Regressor, with 100 trees. This gives us, for each composition, a total of 18 functions that we learn.

For compounds, we perform a similar process, with a slight difference. Unlike for adjectives, where the label structure is always the same, the adjective followed by the base color label, there exist replicas in the lux vocabulary of what we call twins, colors who appear to be the same and yet have the colors swapped, such as *green blue* and *blue green*. As such, we split our training into two: one set of 18 functions where the μs of each dimension are lower ordered first, and another set of 18 where the μs of each dimension are higher ordered first. Then, to predict what a composed color would look like, we look at the order of colors in the label. If the first color in the composed label has a higher μ than the second color, then we use the set of functions that were trained in the higher-to-lower order first, and vice versa.

We chose to use a Elastic Net Regressor for two reasons: the scales of the μs, relative to the other variables, are much, much greater, so ideally we would want a sparse function, if most of the variables would have little to no effect on the μ's end

result. Additionally, both the μ_1 and μ_2 are highly correlated with each other, so the Elastic Net Regressor should identify both of them and make use of them, due to the nature of its optimization function.

We chose to use a Random Forest regressor for the other variables because what the other variables represent are essentially how wide the model becomes. There appears to be no correlation from the original models to the composed models, but there might be information based off the rest of the variables. Thus, we used a Random Forest Decision Tree to hope to capture this information.

For training, we generate a list of labels that correspond to the composition. We then use k-fold cross validation in order to train and test the models. We then recalculate the availability of the model, using the new ϕ curves.

2.3 Evaluation

During the initial training of Lux, the survey data was split into a training set and a test set. We make use of this test set in order to test our models.

For each pair of composed model and predicted composed model, we calculate the negative log likelihood of all of the datapoints with respect to each model. We then sum all of the likelihoods and divide by the number of datapoints, to get the average negative log likelihood of each datapoint for the entire composition. This is called the Empirical Entropy of the models (Equation 2.4), which tells us how probable, for any datapoint, that it falls under the datapoint's labeled distribution.

$$EmpiricalEntropy = \frac{1}{n} * \Sigma_{i=1}^{n} - \log \theta_i(i). \tag{2.4}$$

During evaluation, we compared the predicted color models that we generated, using the base color models and our composition functions, against the actual color models present in Lux.

We also do a statistical significance test, to determine if the differences between the two models is significant or whether or not the difference was just due to random chance. In order to do this, we first calculate the scores given to all datapoints for

the color by both lux's color model and our predicted color model. We then bootstrap 1,000 random trials of selecting with replacement from all of the scores, and for each trial we calculate the difference between the two sets of scores. We then compare the actual difference between the two sets of scores and calculate the percentage of times that this result occurred. If the result occurred less than 5% of the time, we rejected the null hypothesis that the two models approximate the test datapoints in the same way.

2.4 Results

Our model performed very well when compared against Lux. In general, the colors created by our model tend to fall into one of two categories: they either fit the data extremely well on all dimensions, or the phi-curve generated flat-lines on one of the of the dimensions, giving either no probability or equal probability amongst the entire dimension, which usually occurred in Hue. This second result often occurs when there are few data points in the composition. There are instances in which the generated composed distributions fit the data better than the original Lux distributions do; a reason for this may be that some of the initial fits in Lux were not as best as they could be, while the generated models used generalized knowledge about what the color should look like, and the data matches.

A summary of results is found in Table 2.3 and in Figure 2.5. We now discuss a selection of the functions that compose the compositions. Each subsection has an associated function parameter table, which tells us the coefficients or the important features for each of the functions. We chose Aqua since it was the first color alphabetically that appeared in all of the selected compositions.

2.4.1 Light

Light is the composition that has the most number of occurrences in Lux. There is almost no change in the Hue, aside from a slight widening and a shift left in the spectrum. For Saturation, there is a drastic shift downwards, telling us that the color

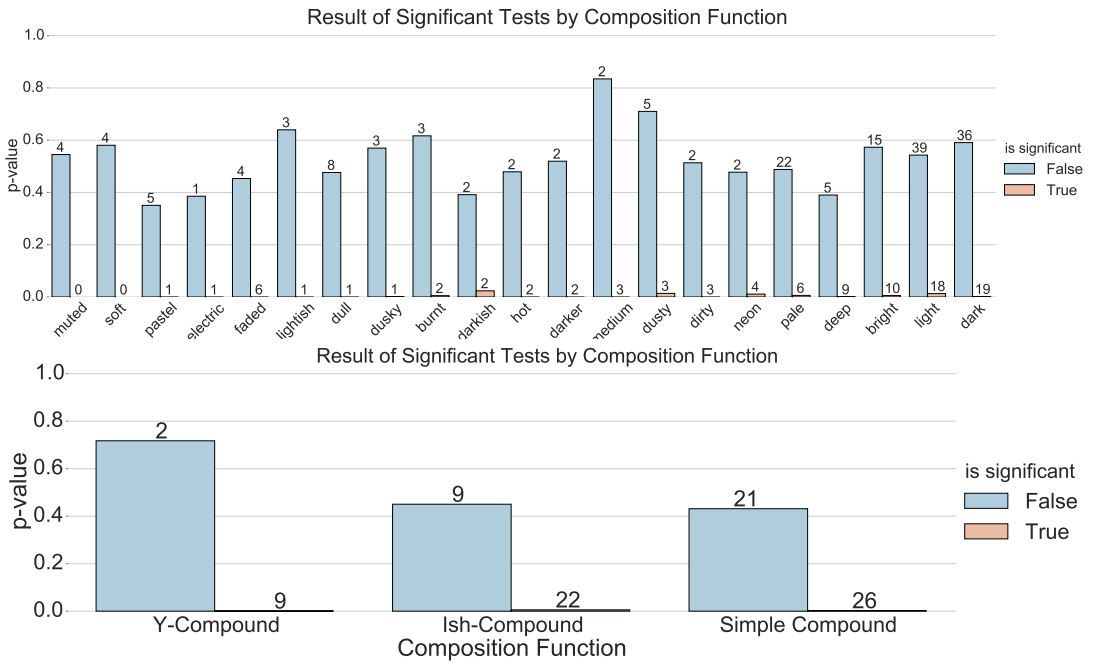

Figure 2.5: Summary of results found in Table 2.3. The bars represent the average p-val for each category, while the numbers represent the number of colors that fell into that category.

is greyer and less intense than its original color. For Value, there is a drastic shift upward, from both the high coefficient but also from the high intercept; the lowest that any non-adjusted light color can be is a value of approximately 40. Thus the light composition tells us that while the primary color does not change, it becomes significantly less pure, less intense, while it becomes much brighter.

The Function Parameters are found in Tables 2.4-2.10, while an example comparing the predicted distribution to the actual composed distribution can be found in Figure 2.6. An example of the Light function present in Lux is in Figure 2.7

2.4.2 Dark

Dark is the composition that appears the second most number of times. Similar to Light, there is almost no change in the Hue values, except for a slight widening of the distribution. The Saturation does not have a drastic shift either, but it does widen a non-insignificant amount, extending further into the grey area. Meanwhile, the Value shifts dramatically downwards to less than 80% of its original value, telling us that it

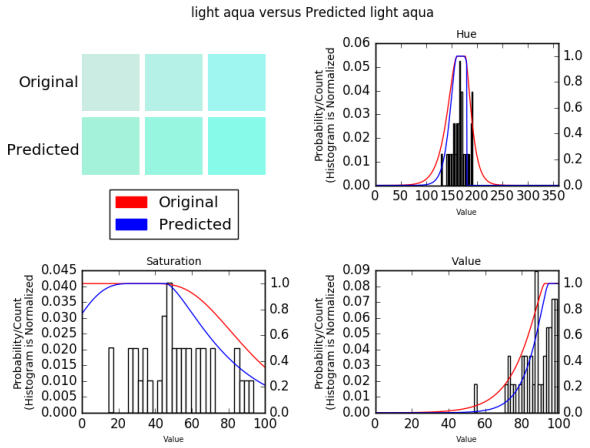

Figure 2.6: Light Aqua, and the Predicted Light Aqua

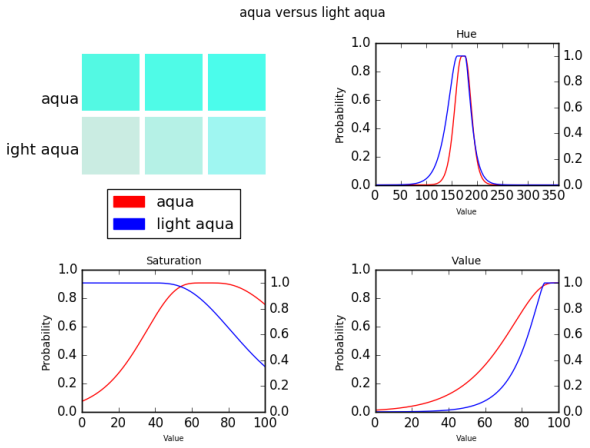

Figure 2.7: Aqua and Light Aqua as found in Lux.

becomes darker. Thus, the dark composition tells us that while the primary color does not change, it becomes slightly less pure, while darkening immensely.

The Function Parameters are in Tables 2.11-2.17, while an example comparing the predicted distribution to the actual composed distribution is in Figure 2.8. An example of the Dark function present in Lux can be found in Figure 2.9.

2.4.3 Pale

Pale appears to be a much more extreme version of light, with the same patterns in its functions, only with even higher values. For Light, the lowest that a Value could be was 40, while for Pale, it's 55.

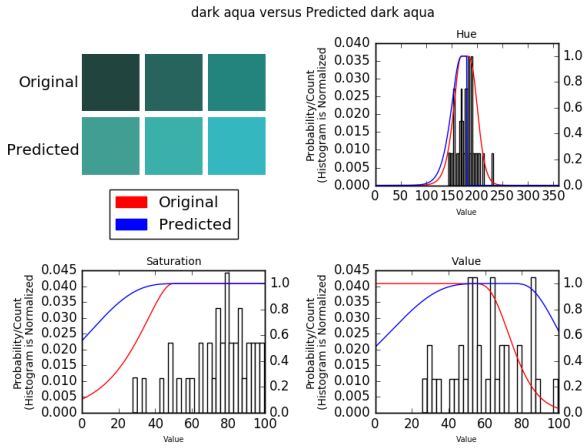

Figure 2.8: Dark Aqua, and the Predicted Dark Aqua

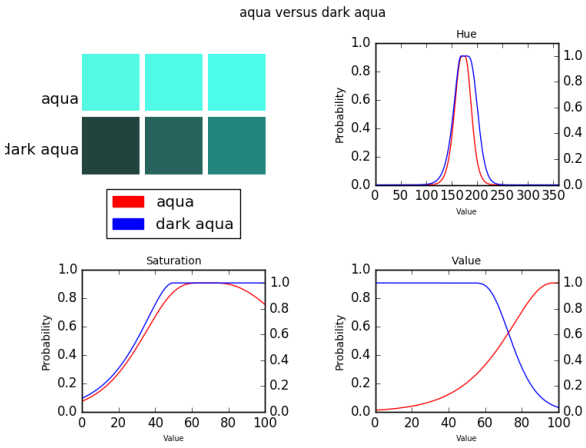

Figure 2.9: Aqua and Dark Aqua as found in Lux.

The Function Parameters are in Tables 2.18-2.24, while an example comparing the predicted distribution to the actual composed distribution is found in Figure 2.10. An example of the Pale function present in Lux is in Figure 2.11

2.4.4 Deep

Much like the previous three categories, Deep has almost no change in its Hue values. Deep's Saturation expands upwards, with its lower bound remaining mostly the same, while it's upper limit increases towards 100. It's Value shifts downwards, becoming darker. Thus, Deep is characterised by its primary color remaining the same, while its becomes darker. The widening of the Saturation means that with regards to Deep, the

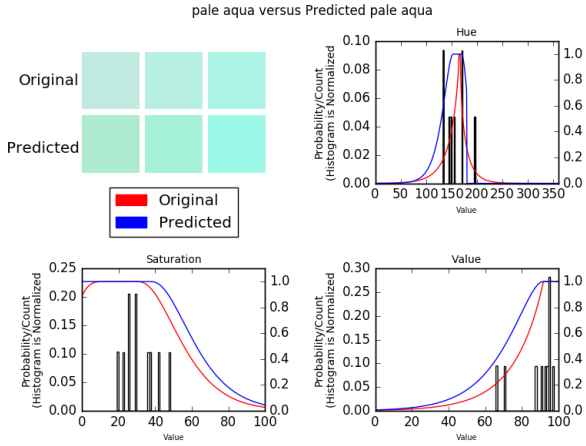

Figure 2.10: Pale Aqua, and the Predicted Pale Aqua

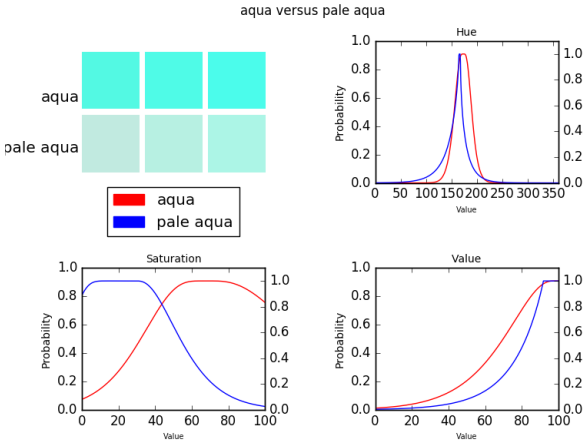

Figure 2.11: Aqua and Pale Aqua as found in Lux.

changes in how much grey is in it is not as important as how much darker it becomes, while it generally becomes purer, it is not required to be considered Deep.

The Function Parameters are found in Tables 2.25-2.31, while an example comparing the predicted distribution to the actual composed distribution ise found in Figure 2.12. An example of the Pale function present in Lux is in Figure 2.13.

2.4.5 Compounds

Compounds consists of mixing two colors together, with no further linguistic modifications.

We used a slightly different method for the two color compositions than was used

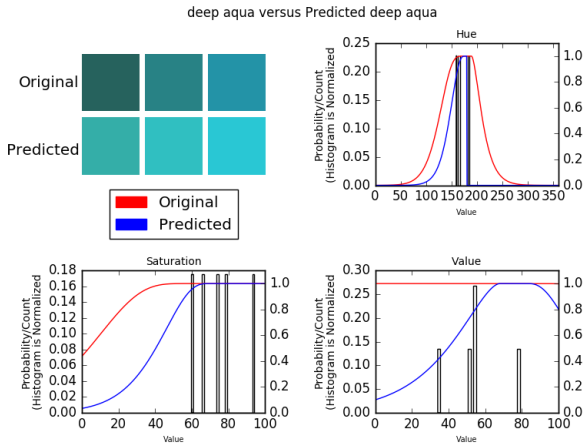

Figure 2.12: Deep Aqua, and the Predicted Deep Aqua

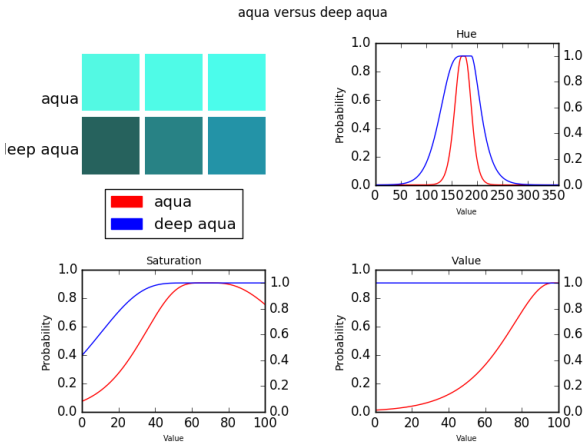

Figure 2.13: Aqua and Deep Aqua as found in Lux.

earlier. Present in Lux are colors which can be considered duplicates of each other, where the colors are almost the same and yet the ordering of the words are different, such as 'green-blue' and 'blue-green', as seen in Figure 2.14. As such, the ordering of the inputs might have an effect on the end color created. Thus, the inputs for each color composition were limited to color compositions which had the Hues in the same order as the color composition that we are trying to make. For instance, if the Hue of the first color is closer to less than the Hue of the second color, then we only trained and used a composition function where that fact was true for all of the inputs as well.

The Hue ends up between the two colors, with the second color having much more importance than the first color. The lower bound for the composed color ends up

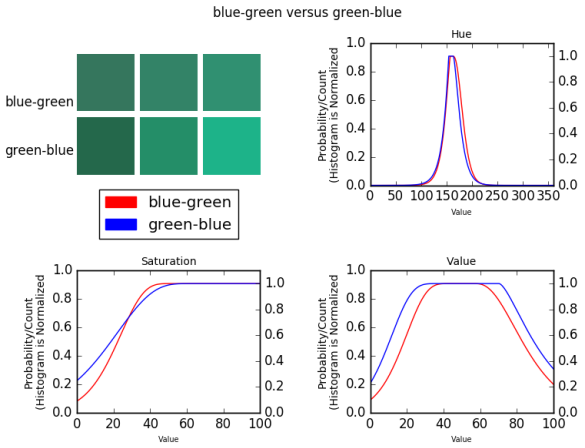

Figure 2.14: Green Blue vs Blue Green. While there is not a large difference between the two, there is the chance that there is a difference that is hidden within the word ordering.

shifted towards 0 from the position of the second color, between the two colors, while the upper bound remains close to the lower bound of the second color. The Saturation ends up with roughly 70% of the first color and 30% of the second color, for both of the bounds, while the Value is roughly 50-80% of the second color and 30% of the first color for both bounds. What this tells us is that what people consider to be a mix of two colors varies widely, depending on where in the spectrum the original colors lie. Colors that are more intense and more vibrant have a larger impact on the resulting color combination.

Adjective	Times Present	# of Datapoints	% of Datapoints
light	57	7,049	37.01%
dark	55	4,513	23.70%
pale	28	1,439	7.56%
bright	25	1,843	9.68%
deep	14	452	2.37%
dull	9	233	1.22%
dusty	8	222	1.17%
faded	7	116	0.61%
neon	6	492	2.58%
pastel	6	304	1.60%
burnt	5	490	2.57%
dirty	5	79	0.41%
medium	5	171	0.90%
electric	5	136	0.71%
soft	4	58	0.30%
muted	4	37	0.19%
hot	4	631	3.31%
darkish	4	41	0.22%
dusky	4	63	0.33%
lightish	4	44	0.23%
darker	4	59	0.31%
off	3	21	0.11%
vivid	3	18	0.09%
warm	3	16	0.08%
lighter	3	62	0.33%
baby	3	293	1.54%
vibrant	3	20	0.11%
raw	2	11	0.06%
mid	2	29	0.15%
rich	2	10	0.05%
flat	2	11	0.06%
true	2	33	0.17%
cool	2	14	0.07%
ugly	2	14	0.22%
drab	1	22	0.12%
TOTAL	296	19,046	100.00%

Table 2.1: Adjectives in Lux, and number of times that they are found in the Lux vocabulary, as well as the number of datapoints in the test set for each adjective. We discarded any adjectives that appeared less than 4 times, which only discarded approximately 3% of the data.

Type	Times Present	# of Datapoints	% of Datapoints
Simple Compound	47	3,151	69.09%
Ish-Compound	31	1,212	26.57%
Y-Compound	11	198	4.34%
TOTAL	89	4,561	100.00%

Table 2.2: Datapoint information for the compounds in lux, similar to Table 2.1.

Composition	Num Datapoints	Lux Entropy	Predicted Model Entropy	% of no significant difference without availability	% of no significant difference with availability
soft	58	10.822190	10.370445	100%	100%
neon	492	6.121584	6.828746	66.7%	50%
deep	452	8.031913	7.909304	85.7%	35.7%
bright	1843	5.765445	5.708914	60%	60%
pastel	304	7.808696	8.192886	83.3%	83.3%
muted	37	11.476829	10.812793	75%	100%
burnt	490	5.801682	5.817992	60%	60%
hot	631	4.813669	4.837247	50%	50%
faded	116	10.263318	10.161881	100%	85.7%
dirty	79	10.066384	10.468254	40%	40%
medium	171	9.088845	8.730067	40%	40%
electric	136	8.904705	8.895868	60%	80%
darkish	41	11.670583	11.175465	75%	50%
dark	4513	5.676472	5.542899	74.5%	65.5%
dull	233	9.334435	9.177858	88.9%	88.9%
dusty	222	8.300571	8.269519	50%	62.5%
light	7049	4.843748	4.907457	80.7%	70.2%
dusky	63	10.390892	10.494882	75%	75%
lightish	44	11.570441	11.493112	50%	75%
darker	59	10.856628	10.512069	50%	50%
pale	1439	6.601283	6.636638	85.7%	78.6%
simple compound	3151	6.964963	8.917363	55.3%	44.7%
ish compound	1212	8.238537	12.731032	61.3%	32.3%
y compound	198	9.763376	13.759155	63.6%	18.2%

Table 2.3: Results for the compositions. The lower the Entropy the better fit the model is on the data. For the last two columns, this is the percentage of occurrences within the composition that our bootstrap test returned True over all of the composition pairs.

Table 2.4: Light Function Parameters

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.80	-0.00	-0.00	0.14	-0.00	-0.00	-0.87
μ_2	0.42	-0.00	0.00	0.52	-0.00	0.00	8.79

Table 2.5: Coefficients and Intercept for the Elastic Net Regressors for Hue

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.13	0.28	0.19	0.18	0.11	0.12
γ_1	0.09	0.17	0.50	0.07	0.07	0.09
β_2	0.07	0.24	0.14	0.06	0.14	0.35
γ_2	0.08	0.09	0.12	0.07	0.15	0.49

Table 2.6: Feature Importances for the Random Forest Regressors for Hue. The closer to 1, the more important.

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.33	-0.00	-0.00	0.15	0.00	0.00	-1.14
μ_2	0.22	-0.00	-0.00	0.24	0.00	0.00	13.92

Table 2.7: Coefficients and Intercept for the Elastic Net Regressors for Saturation

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.33	0.09	0.09	0.28	0.10	0.11
γ_1	0.13	0.13	0.15	0.13	0.26	0.19
β_2	0.09	0.16	0.11	0.17	0.24	0.23
γ_2	0.07	0.09	0.11	0.16	0.15	0.42

Table 2.8: Feature Importances for the Random Forest Regressors for Saturation. The closer to 1, the more important.

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.00	0.00	0.00	0.55	-0.00	0.00	40.18
μ_2	-0.13	0.00	0.82	0.63	-1.54	1.34	47.30

Table 2.9: Coefficients and Intercept for the Elastic Net Regressors for Value

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.27	0.12	0.09	0.22	0.12	0.17
γ_1	0.19	0.11	0.10	0.29	0.15	0.17
β_2	0.05	0.07	0.05	0.10	0.11	0.62
γ_2	0.07	0.08	0.31	0.21	0.07	0.26

Table 2.10: Feature Importances for the Random Forest Regressors for Value. The closer to 1, the more important.

Table 2.11: Dark Function Parameters

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.62	-0.00	0.00	0.40	0.00	-0.00	-4.76
μ_2	0.50	-0.00	0.00	0.53	0.00	0.00	4.20

Table 2.12: Coefficients and Intercept for the Elastic Net Regressors for Hue

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.08	0.23	0.34	0.06	0.16	0.14
γ_1	0.10	0.13	0.52	0.07	0.08	0.10
β_2	0.10	0.16	0.08	0.07	0.40	0.19
γ_2	0.09	0.12	0.07	0.08	0.12	0.51

Table 2.13: Feature Importances for the Random Forest Regressors for Hue. The closer to 1, the more important.

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.71	-0.00	0.00	0.00	0.00	-0.00	11.33
μ_2	0.41	-0.00	0.00	0.19	0.00	0.00	32.19

Table 2.14: Coefficients and Intercept for the Elastic Net Regressors for Saturation

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.17	0.15	0.14	0.06	0.39	0.08
γ_1	0.10	0.17	0.14	0.11	0.29	0.20
β_2	0.14	0.06	0.07	0.39	0.11	0.23
γ_2	0.07	0.10	0.10	0.07	0.15	0.50

Table 2.15: Feature Importances for the Random Forest Regressors for Saturation. The closer to 1, the more important.

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.83	-0.00	-4.48	0.00	-0.00	0.00	-8.38
μ_2	0.28	0.00	-2.96	0.61	0.01	-0.00	-6.15

Table 2.16: Coefficients and Intercept for the Elastic Net Regressors for Value

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.30	0.09	0.20	0.20	0.09	0.11
γ_1	0.09	0.23	0.31	0.12	0.12	0.14
β_2	0.09	0.36	0.18	0.10	0.18	0.10
γ_2	0.34	0.07	0.11	0.19	0.14	0.15

Table 2.17: Feature Importances for the Random Forest Regressors for Value. The closer to 1, the more important.

Table 2.18: Pale Function Parameters

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.53	-0.00	-0.00	0.38	-0.00	-0.00	-2.96
μ_2	0.47	-0.00	-0.00	0.45	-0.00	-0.00	9.62

Table 2.19: Coefficients and Intercept for the Elastic Net Regressors for Hue

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.17	0.09	0.20	0.21	0.17	0.17
γ_1	0.05	0.13	0.60	0.08	0.09	0.05
β_2	0.07	0.22	0.43	0.07	0.10	0.12
γ_2	0.08	0.11	0.13	0.05	0.08	0.56

Table 2.20: Feature Importances for the Random Forest Regressors for Hue. The closer to 1, the more important.

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.07	-0.00	-0.00	0.35	0.00	0.00	-3.57
μ_2	0.06	-0.00	-0.00	0.31	0.00	0.00	10.72

Table 2.21: Coefficients and Intercept for the Elastic Net Regressors for Saturation

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.07	0.47	0.09	0.10	0.20	0.07
γ_1	0.16	0.17	0.11	0.06	0.36	0.14
β_2	0.14	0.20	0.12	0.12	0.11	0.30
γ_2	0.07	0.18	0.13	0.06	0.16	0.41

Table 2.22: Feature Importances for the Random Forest Regressors for Saturation. The closer to 1, the more important.

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.38	0.00	-0.00	0.00	0.00	0.00	54.93
μ_2	0.63	0.55	0.07	-0.42	-0.55	0.28	76.61

Table 2.23: Coefficients and Intercept for the Elastic Net Regressors for Value

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.11	0.28	0.25	0.14	0.11	0.11
γ_1	0.13	0.13	0.16	0.09	0.31	0.19
β_2	0.17	0.12	0.23	0.19	0.12	0.16
γ_2	0.19	0.10	0.17	0.19	0.26	0.08

Table 2.24: Feature Importances for the Random Forest Regressors for Value. The closer to 1, the more important.

Table 2.25: Deep Function Parameters

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.38	-0.00	0.00	0.66	0.00	-0.00	-3.91
μ_2	0.46	-0.00	0.00	0.55	0.00	0.00	11.83

Table 2.26: Coefficients and Intercept for the Elastic Net Regressors for Hue

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.20	0.06	0.30	0.13	0.05	0.27
γ_1	0.05	0.22	0.15	0.08	0.24	0.26
β_2	0.20	0.22	0.06	0.29	0.07	0.15
γ_2	0.07	0.09	0.21	0.06	0.10	0.47

Table 2.27: Feature Importances for the Random Forest Regressors for Hue. The closer to 1, the more important.

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	2.30	-0.00	0.08	-1.72	0.00	0.00	36.59
μ_2	1.98	-0.00	0.54	-1.48	-0.11	0.60	50.14

Table 2.28: Coefficients and Intercept for the Elastic Net Regressors for Saturation

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.05	0.19	0.07	0.07	0.42	0.20
γ_1	0.02	0.14	0.08	0.03	0.15	0.59
β_2	0.22	0.09	0.12	0.26	0.15	0.17
γ_2	0.10	0.30	0.26	0.08	0.10	0.16

Table 2.29: Feature Importances for the Random Forest Regressors for Saturation. The closer to 1, the more important.

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2	Intercept
μ_1	0.60	-0.00	-0.00	0.52	0.00	0.00	-47.16
μ_2	0.58	-0.00	-0.00	0.63	0.00	0.00	-38.89

Table 2.30: Coefficients and Intercept for the Elastic Net Regressors for Value

Name	μ_1	β_1	γ_1	μ_2	β_2	γ_2
β_1	0.24	0.10	0.18	0.34	0.08	0.06
γ_1	0.19	0.03	0.12	0.12	0.12	0.42
β_2	0.19	0.45	0.09	0.18	0.03	0.05
γ_2	0.07	0.12	0.31	0.09	0.26	0.14

Table 2.31: Feature Importances for the Random Forest Regressors for Value. The closer to 1, the more important.

Table 2.32: Simple Composition Function Parameters

Name	$1:\mu_1$	$1{:}\beta_1$	$1{:}\gamma_1$	$1{:}\mu_2$	$1{:}\beta_2$	$1{:}\gamma_2$	$2:\mu_1$	$2{:}\beta_1$	$2{:}\gamma_1$	$2{:}\mu_2$	$2{:}\beta_2$	$2{:}\gamma_2$	Intercept
μ_1	-0.03	0.00	-0.00	-0.06	-0.00	0.00	0.29	-0.00	-0.00	0.74	-0.00	-0.00	-26.32
μ_2	-0.00	-0.00	0.00	0.00	0.00	0.00	0.52	-0.00	-0.00	0.40	-0.00	-0.00	-6.20

Table 2.33: Coefficients and Intercept for the Elastic Net Regressors for Hue

Name	$1:\mu_1$	$1{:}\beta_1$	$1{:}\gamma_1$	$1{:}\mu_2$	$1{:}\beta_2$	$1{:}\gamma_2$	$2:\mu_1$	$2{:}\beta_1$	$2{:}\gamma_1$	$2{:}\mu_2$	$2{:}\beta_2$	$2{:}\gamma_2$
β_1	0.07	0.06	0.12	0.06	0.07	0.06	0.07	0.22	0.06	0.07	0.11	0.04
γ_1	0.04	0.04	0.12	0.03	0.06	0.09	0.05	0.19	0.23	0.08	0.03	0.04
β_2	0.03	0.03	0.04	0.04	0.06	0.36	0.04	0.10	0.06	0.03	0.06	0.15
γ_2	0.04	0.04	0.14	0.04	0.10	0.18	0.03	0.12	0.09	0.05	0.08	0.07

Table 2.34: Feature Importances for the Random Forest Regressors for Hue. The closer to 1, the more important.

Name	$1:\mu_1$	$1{:}\beta_1$	$1{:}\gamma_1$	$1{:}\mu_2$	$1{:}\beta_2$	$1{:}\gamma_2$	$2:\mu_1$	$2{:}\beta_1$	$2{:}\gamma_1$	$2{:}\mu_2$	$2{:}\beta_2$	$2{:}\gamma_2$	Intercept
μ_1	0.70	-0.73	-0.94	-0.12	-0.00	1.86	-1.02	-0.32	-1.34	1.34	-0.55	0.31	-4.54
μ_2	0.63	-0.28	-2.02	-0.00	-0.37	2.04	-0.93	-0.00	-1.18	1.22	-1.04	0.38	9.52

Table 2.35: Coefficients and Intercept for the Elastic Net Regressors for Saturation

Name	$1:\mu_1$	$1{:}\beta_1$	$1{:}\gamma_1$	$1{:}\mu_2$	$1{:}\beta_2$	$1{:}\gamma_2$	$2:\mu_1$	$2{:}\beta_1$	$2{:}\gamma_1$	$2{:}\mu_2$	$2{:}\beta_2$	$2{:}\gamma_2$
β_1	0.08	0.14	0.08	0.05	0.05	0.08	0.07	0.25	0.07	0.07	0.01	0.05
γ_1	0.09	0.06	0.09	0.08	0.04	0.16	0.04	0.08	0.05	0.04	0.09	0.18
β_2	0.08	0.35	0.06	0.17	0.02	0.04	0.05	0.03	0.08	0.05	0.03	0.05
γ_2	0.46	0.11	0.04	0.15	0.03	0.03	0.02	0.03	0.03	0.02	0.03	0.06

Table 2.36: Feature Importances for the Random Forest Regressors for Saturation. The closer to 1, the more important.

Name	$1:\mu_1$	$1{:}\beta_1$	$1{:}\gamma_1$	$1{:}\mu_2$	$1{:}\beta_2$	$1{:}\gamma_2$	$2:\mu_1$	$2{:}\beta_1$	$2{:}\gamma_1$	$2{:}\mu_2$	$2{:}\beta_2$	$2{:}\gamma_2$	Intercept
μ_1	0.20	2.34	1.51	0.13	0.00	0.13	0.03	-0.00	-0.01	0.76	3.79	-0.00	-39.50
μ_2	0.08	1.42	2.50	0.28	-0.00	0.25	0.25	-0.09	-0.00	0.36	2.24	-0.00	-12.31

Table 2.37: Coefficients and Intercept for the Elastic Net Regressors for Value

Name	$1:\mu_1$	$1{:}\beta_1$	$1{:}\gamma_1$	$1{:}\mu_2$	$1{:}\beta_2$	$1{:}\gamma_2$	$2:\mu_1$	$2{:}\beta_1$	$2{:}\gamma_1$	$2{:}\mu_2$	$2{:}\beta_2$	$2{:}\gamma_2$
β_1	0.03	0.06	0.12	0.05	0.04	0.04	0.24	0.05	0.03	0.21	0.11	0.02
γ_1	0.07	0.10	0.07	0.05	0.13	0.12	0.06	0.07	0.05	0.05	0.15	0.07
β_2	0.10	0.09	0.05	0.20	0.09	0.10	0.04	0.06	0.06	0.03	0.07	0.12
γ_2	0.07	0.25	0.03	0.34	0.02	0.05	0.00	0.01	0.02	0.01	0.18	0.01

Table 2.38: Feature Importances for the Random Forest Regressors for Value. The closer to 1, the more important.

Chapter 3

Grammar and Tail Data

3.1 Grammar

We constructed a context free grammar in order to be able to create and identify the models associated with any label that can be generated from the grammar. Most of the grammar was written by Brian McMahan, while we made further adjustments as needed in order to touch on all of the cases we wished to cover. We also constructed a parser that links Lux base models and composition functions with different rules in the grammar, so that by running a color label through the parser, we can generate the associated model.

The grammar consists of 350 hand-cultivated rules, which can be seen in Figure 3.1. S is the sentence, the color label. N are the base color words, while OBJ are the color words that are tied to real-world objects. $NISH, NY$ are the "ish" productions and the "y" productions of some of the base color words. ADJ are the color modifiers. NP are the different types of compositions that we can have, while $NP2$ exists in order to properly construct some parse trees.

The base terminals in the grammar account for 300 color labels in Lux. The rest of the 529 labels can be constructed from this grammar. Out of the $133,515$ labels representing $421,151$ datapoints that make up the tail, we can generate a parse tree for $12,827$ of them, covering of $107,322$ datapoints. A large portion of the labels in the tail consist of 1-or-2 of joke responses. A larger coverage of datapoints is likely possible through some more processing, such as spellchecking or the removing extraneous parts of labels, meant to describe the color in more detail, such as *peach or light red*, or even through the identification of the color mentioned in the label, such as *salmon* from the label *sheesh, how many variations of salmony colours can there be?*.

3.2 Tail

One useful application of this method is that we can extend it to colors that are present outside of Lux. Lux was originally trained on all of the colors that had more than 100 entries. The rest of the data is located in the tail, and had too few datapoints to reasonably work with. However, given a base model, we can determine what the composed model is. This allows us to create a model for a color that occurs in the tail, which we can test using those points.

Unlike the original Lux models, we cannot calculate availability of the colors in the tail. As such, the results only give us an idea of how well the proposed tail-models fit the data points, and it is not balanced around how popular the label is. Additionally, labels that are rarely mentioned can be heavily skewed, since they have a low number of users who have called them that label, the chance of purposefully misleading data cannot be balanced out as easily as the models in Lux.

In order to evaluate how well our model does on the tail, we perform a similar bootstrap test as mentioned above. Unlike before, due to the scarcity of datapoints, we evaluate the entire function at once, using all of the datapoints in the tail that fall under first-level compositions at once, rather than comparing each color individually. As a baseline, we use a backoff model, which uses the base color found in lux. Additionally, we discarded the availability of the baseline colors, as we cannot construct an availability for the tail colors. We only evaluated on the adjectives, as there is no defined backoff for any of the color compounds.

To evaluate each composition, we first constructed a list of all base color labels paired with a list of all composed base color labels found in the tail. Then for each pair, we then used the tail datapoints and found the likelihood of that datapoint given both our predicted model color and lux's backoff color model. We then took all of the likelihoods for each dataset and performed the same bootstrap method in order to determine if the differences in likelihoods were due to random chance or if there was a significant difference between them, using a two-tailed test with $p = .05$. The results can be seen in Table 3.1. In 16 out of 21 cases, our models had significantly

lower entropy than the baseline, while in the other 5 there was no significant difference between the two.

Composition	Predicted Models Entropy	Lux Backoff Entropy	Significant Difference	P-val	# of Labels	# of Datapoints in Tail
neon	1.9437	1.7870	False	0.336	150	1115
deep	1.3743	1.9498	True	0	319	2749
bright	1.1950	2.0683	True	0	383	3486
pale	1.6829	2.3152	True	0	463	4060
pastel	2.7063	2.5463	False	0.473	171	1142
muted	1.3876	2.0820	True	0	181	1035
hot	1.5643	2.0092	True	0.018	80	511
dirty	1.7695	2.8301	True	0	184	1194
medium	1.3590	1.8932	Truee	0	270	1675
electric	1.0977	1.925	True	0	113	547
darkish	1.3079	1.9354	False	0.071	90	526
dark	1.6847	1.9868	True	0	564	8270
burnt	2.7156	2.9491	False	0.558	110	786
dull	1.6647	2.1157	True	0	276	1790
lightish	1.4944	1.9750	False	0.417	71	219
dusty	1.8321	2.2300	True	0.001	200	1356
light	1.5320	1.9926	True	0	563	8910
dusky	1.5431	2.3740	True	0	82	467
faded	1.3932	2.4980	True	0	229	1250
soft	1.2021	1.98411	True	0	165	834
darker	0.9351	2.1836	True	0	202	1239
TOTAL			16/21		4804	43161

Table 3.1: Our predicted models vs the lux backoff model for the color labels found in the tail. The entropy is as stated in Equation 2.4. If the difference was significant, then we put True. Otherwise, we put False. The null hypothesis was that there was no difference in the likelihoods given by both models, and the p-vals are the likelihood that the null hypothesis is true.

3.3 Demo

We constructed a demo program, so that a user can easily use these composition functions in order to see what a color they want looks like. If the color label passed in is one that can be parsed, and has all of the associated composition functions allowed, then the program outputs an image showing both the colors and each dimension's probability distributions, as seen in Figures 3.2 and 3.3.

The demo program operates in two steps. In the first step, we use the grammar

portrayed in Figure 3.1 in order to parse an input color label using a basic chart parser. We then take the resulting parse trees and do a post-order traversal, applying the precomputed composition function referenced by the parent node for each branch to the resulting models supplied by the child nodes. The leaves of the trees return the original lux model. An example of the demo can be found in Figure 3.4.

```
S → NISH | NP | OBJ | NP NP | ADJ NP2
NP → ADJ OBJ | N | NISH N | NY N | OBJ N | ADJ NP
NP2 → NP NP
ADJ → 'baby' | 'bluish' | 'bright' | 'burnt' | 'cool' | 'dark' |
'darker' | 'darkish' | 'deep' | 'dirty' | 'dull' | 'dusky' | 'dusty' |
'easter' | 'electric' | 'faded' | 'flat' | 'hot' | 'light' | 'lighter'
| 'lightish' | 'medium' | 'mid' | 'muddy' | 'muted' | 'neon' | 'off' |
'pale' | 'pastel' | 'raw' | 'rich' | 'rusty' | 'soft' | 'true' | 'very'
| 'vibrant' | 'vivid' | 'warm'
NY → 'bluey' | 'greeny' | 'orangey' | 'pinky' | 'purpley'
NISH → 'blueish' | 'brownish' | 'greenish' | 'greyish' | 'orangish' |
'pinkish' | 'purplish' | 'reddish' | 'tealish' | 'yellowish'
N → 'aqua' | 'aquamarine' | 'aubergine' | 'auburn' | 'azure' | 'beige' |
'biege' | 'black' | 'blue' | 'blurple' | 'brown' | 'buff' | 'celadon' |
'cerise' | 'cerulean' | 'chartreuse' | 'claret' | 'cobalt' | 'cornflower' |
'cream' | 'crimson' | 'cyan' | 'ecru' | 'fawn' | 'fuchsia' | 'golden' |
'green' | 'grey' | 'gross' | 'hazel' | 'heather' | 'heliotrope' | 'hiccup'
| 'horrible' | 'indigo' | 'iris' | 'magenta' | 'magneta' | 'maroon' |
'mauve' | 'muave' | 'navy' | 'ochre' | 'orange' | 'pale' | 'periwinkle'
| 'pink' | 'pretty' | 'puce' | 'purple' | 'red' | 'rouge' | 'royal'
| 'russet' | 'saffron' | 'scarlet' | 'sepia' | 'sienna' | 'slate' |
'spam' | 'tan' | 'taupe' | 'teal' | 'terracotta' | 'turquoise' | 'ugly'
| 'umber' | 'velvet' | 'vermillion' | 'violet' | 'viridian' | 'weird' |
'white' | 'wisteria' | 'yellow' | 'yuck'
OBJ → 'acid', 'green' | 'adobe' | 'algae' | 'almost', 'black' | 'amber' |
'amethyst' | 'apple' | 'apricot' | 'army', 'green' | 'asparagus' | 'avocado'
| 'banana' | 'barbie', 'pink' | 'barf' | 'barney' | 'battleship', 'grey'
| 'berry' | 'bile' | 'blood' | 'blueberry' | 'blush' | 'booger', 'green'
| 'bottle', 'green' | 'brick' | 'british', 'racing', 'green' | 'bronze' |
'bubblegum' | 'burgundy' | 'butter' | 'butterscotch' | 'cadet', 'blue' |
'camel' | 'camo', 'green' | 'camouflage', 'green' | 'canary' | 'caramel'
| 'carnation' | 'carolina', 'blue' | 'caucasian' | 'celery' | 'charcoal'
| 'cherry' | 'chestnut' | 'chocolate' | 'clay' | 'cocoa' | 'coffee' |
'copper' | 'coral' | 'cranberry' | 'dandelion' | 'denim' | 'diarrhea' |
'dirt' | 'dodger', 'blue' | 'drab' | 'dried', 'blood' | 'duck', 'egg', 'blue'
| 'dusk' | 'earth' | 'eggplant' | 'eggshell' | 'emerald' | 'evergreen'
| 'fern' | 'fire', 'engine', 'red' | 'flesh' | 'flesh', 'tone' |
'flourescent', 'green' | 'fluorescent', 'green' | 'fluro', 'green' |
'foam', 'green' | 'forest' | 'french', 'blue' | 'frog', 'green' | 'gold'
| 'goldenrod' | 'grape' | 'grapefruit' | 'grass' | 'grassy', 'green' |
'gunmetal' | 'highlighter', 'green' | 'hospital', 'green' | 'hunter', 'green'
| 'ice' | 'icky', 'green' | 'indian', 'red' | 'irish', 'green' | 'ivory' |
'jade' | 'jungle', 'green' | 'kahki' | 'kelly', 'green' | 'kermit', 'green'
| 'key', 'lime' | 'khaki' | 'kiwi' | 'lavender' | 'lawn', 'green' |
'leaf' | 'leafy', 'green' | 'leather' | 'lemon' | 'lilac' | 'lime'
| 'lipstick' | 'macaroni', 'and', 'cheese' | 'mahogany' | 'maize' |
'mango' | 'marigold' | 'marine' | 'melon' | 'merlot' | 'metallic', 'blue'
| 'midnight' | 'military', 'green' | 'milk', 'chocolate' | 'mint' |
'minty', 'green' | 'mocha' | 'moss' | 'mud' | 'mulberry' | 'murky', 'green'
| 'mushroom' | 'mustard' | 'night', 'blue' | 'nude' | 'ocean' | 'olive' |
'olive', 'drab' | 'orchid' | 'parchment' | 'pea' | 'pea', 'soup' | 'peach'
| 'peachy', 'pink' | 'peacock', 'blue' | 'pear' | 'pine' | 'piss', 'yellow'
| 'pistachio' | 'plum' | 'poop' | 'powder', 'blue' | 'primary', 'blue'
| 'prussian', 'blue' | 'puke' | 'pumpkin' | 'pure', 'blue' | 'putty' |
'raspberry' | 'reddy', 'brown' | "robin's", 'egg' | "robin's", 'egg', 'blue'
| 'rose' | 'rosy', 'pink' | 'ruby' | 'rust' | 'sage' | 'salmon' | 'sand'
| 'sandstone' | 'sandy' | 'sap', 'green' | 'sapphire' | 'sea' | 'seafoam'
| 'seaweed' | 'shamrock' | 'shit' | 'shocking', 'pink' | 'sick', 'green'
| 'sickly', 'green' | 'silver' | 'skin' | 'skin', 'colour' | 'skin', 'tone'
| 'sky' | 'slime', 'green' | 'snot' | 'spearmint' | 'spring', 'green' |
'squash' | 'steel' | 'stone' | 'stormy', 'blue' | 'strawberry' | 'sunflower'
| 'sunshine', 'yellow' | 'swamp', 'green' | 'tangerine' | 'tea', 'green' |
'tiffany', 'blue' | 'tomato' | 'topaz' | 'tree', 'green' | 'turtle', 'green'
| 'twilight' | 'ultramarine' | 'vomit' | 'watermelon' | 'wheat' | 'wine' |
'wintergreen' | 'lemon', 'lime' | 'green', 'apple'
```

Figure 3.1: The production rules for the color grammar.

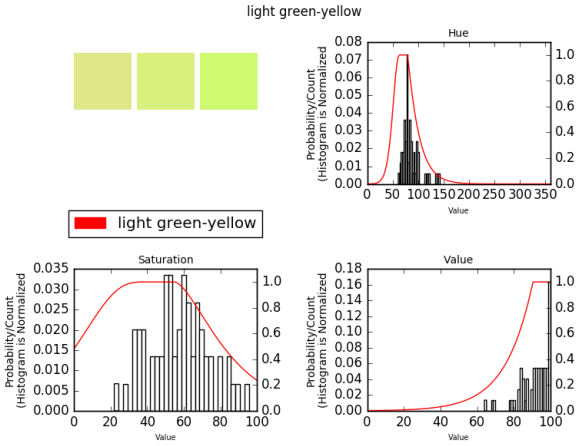

Figure 3.2: Light Green-Yellow, a color that does not appear in Lux, created by applying the "Light" composition to "Green-Yellow"

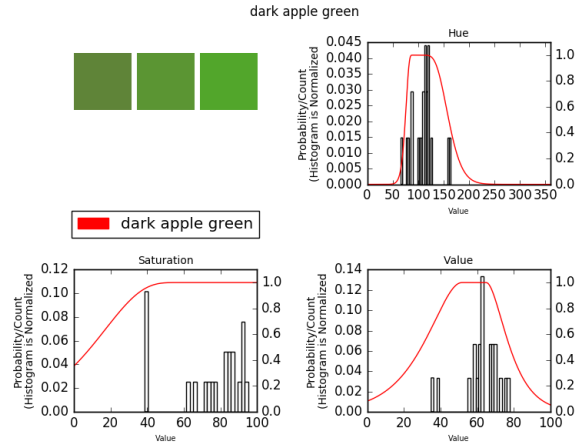

Figure 3.3: Dark Apple Green, a color that does not appear in Lux. created by applying the "Dark" composition to "Apple Green"

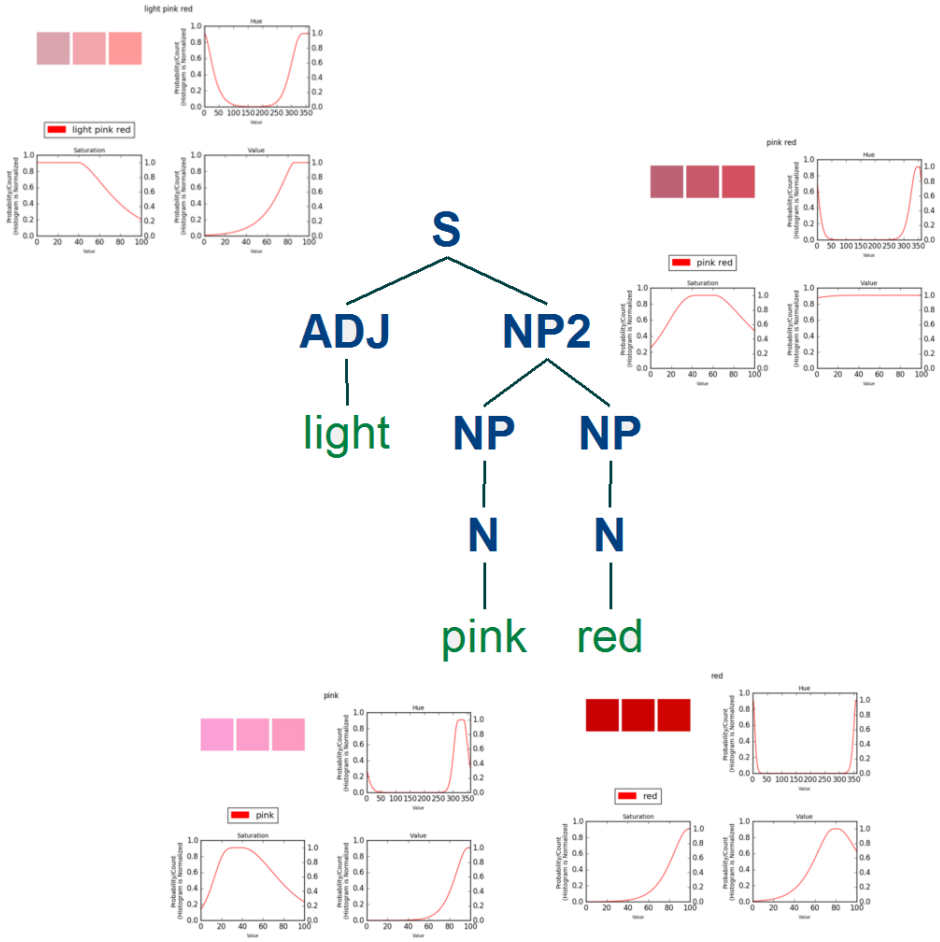

Figure 3.4: Light Pink Red, as assembled by our demo program. By computing the parse tree and performing a post order traversal, we can piecewise create each sub-tree until we get our desired result.

Chapter 4

Conclusion

4.1 Conclusion

In conclusion, color composition can be codified, and we can generalize from colors that we do know into colors that we do not know. Adjectives rarely have any effect on hue, for they modify only the Saturation and Values of the stated color, while the other types of compositions modify the Hue as well, as they are creating more of a mixture of colors. These types of composition functions, applied to models of colors rather than colors themselves, can be further expanded upon to general models that represent other types of objects in the world, and can allow for a more expansive model of the world.

4.2 Future Work

The methodology used in this paper can most likely be extended to multiple types of grounded systems, however they may run into problems. We were able to split our problem into 18 problems, train and predict with them separately, and combine the end results into a model that both worked and made sense when compared to the rest of the models present in Lux. Such assumptions might not hold true for other types of modelings of grounded systems, as they might not be able to be subdivided into smaller problems, and the end results might not make logical sense. For instance, to get around some of the problems that arose, we transformed some of the inputs into logspace, so as to enforce a certain type of behavior on the outputs, but that might be applicable to other types of systems. However, when dealing with transformations in other areas such as image processing or 3-d model transformations, it might still be possible to use these methods.

Further work could be done with the grammar, such as changing it from a context-free-grammar into a feature-based-grammar, as the current grammar is not fully extensible to all types of compositions, such as changing any color category into an "ish" version of itself.

Additionally, exploration using the demo yielded interesting results when dealing with multiple compositions used at the same time. When applying many adjectives to the same color, the end results using our models were rather poor. Our intuition is that when using multiple adjectives or many multiples of color in the same label, the effect of each individual adjective is lessened, unlike our dataset, where each one is present as the only modifier. As such, in the case of *light light blue*, rather than applying our *light* function twice, once to *blue*, and once to *light blue*, we should rather be applying a different type of composition function, either a *light light* function or a lesser version of *light* twice. However, except for *very*, which appears to be a function on a composition function, these types of compositions are not present in lux.

References

[1] Mark Tunner and Gilles Fauconnier. Conceptual integration and formal expression. *Metaphor and Symbol*, 10(3):183–204, 1995.

[2] Wikipedia. Rgb color model — wikipedia, the free encyclopedia, 2016. [Online; accessed 14-April-2016].

[3] Wikipedia. Hsl and hsv — wikipedia, the free encyclopedia, 2016. [Online; accessed 14-April-2016].

[4] Color survey results — xkcd. `http://blog.xkcd.com/2010/05/03/color-survey-results/`. Accessed: 1/5/2016.

[5] Brian McMahan and Matthew Stone. A bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics*, 3:103–115, 2015.

[6] Jacob Andreas and Dan Klein. Grounding language with points and paths in continuous spaces. In *CoNLL*, pages 58–67, 2014.

[7] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.