Automated Color Selection Using Semantic Knowledge

Catherine Havasi
MIT Media Lab
havasi@media.mit.edu

Robert Speer
MIT Media Lab
rspeer@mit.edu

Justin Holmgren
MIT
holmgren@mit.edu

Abstract

Colorizer is a program that hypothesizes color values that represent a given word or sentence, taking into account both physical descriptions of objects and their emotional connotations. This new application of common sense reasoning uses background knowledge about the world to build a model of the connections between everyday things, and uses this model to guess an appropriate color for a word. Colorizer can run over either static text or real time input, such as a speech recognition stream. It has applications in games, the arts, and webpage design.

Introduction

When people think about objects they encounter in the world, the object's properties, such as color, play a major role. It isn't coincidence that many color names contain the names of nouns such as "midnight blue", "lemon yellow", and "sky blue". Although we do not have precise color words for every object we encounter, the colors of objects are a big part of our common sense knowledge.

If we are to create a common sense model which works and reasons in the "real", non-textual world, we need ways to understand and incorporate color as well as other types of non-textual information like sound, touch, and vision.

Here, we describe a system which builds a model of the way people think about colors, and uses this model to associate colors with words, phrases, or larger body of text. This system, Colorizer, uses common sense knowledge from the Open Mind Common Sense project (OMCS) and a combination of color data from several sources to build its model using a technique called spectral association.

Grounding Common Sense

When John McCarthy introduced the notion of "common sense" in his seminal 1959 paper (McCarthy 1959), he discussed how much information, from high-level mental processes such as theory of mind, cause and effect, and simple physics, goes into understanding natural language beyond the words themselves. Traditionally, the OMCS project has focused on textual, affective, and word-based common sense knowledge, but common sense also involves spatial, visual, temporal and physics-related knowledge (Davis 1990).

As a step beyond reasoning about text, then, we wish to begin to incorporate into common sense systems other forms of ground knowledge, such as images or sound samples with descriptions. This combined data would represent an even wider range of representations, which could be used to solve new problems and allow the system to reason about and get input from an even broader portion of the real world.

The Domain of Colors

As a first step in this process, we have created Colorizer, a program that guesses a color to represent a given input word or sentence by taking into account both physical descriptions of objects and emotional connotations. Colorizer provides a link between discrete textual input and continuous feature values, and begins to provide a link between linguistic and other types of common sense.

As we work to ground our primarily linguistic knowledge in different kinds of data, we have chosen to first explore the domain of color. We consider color an interesting domain for a few reasons.

Much of the way people think about common sense is visual, and examining color lets us work with one kind of visual information without the normal difficulties associated with computer vision projects. Additionally, with the rise of computational creativity and computer aided design, it seems important to give computers a sense of objects' colors and the meanings behind color choices.

The question of color prediction from background knowledge isn't well researched, possibly because of the amount of knowledge it requires. NodeBox's PRISM (Smedt and Bleser 2010) tool attempts to come up with a color palette associated with a word or concept by selecting colors from the webpages of top search hits, but it is not intended to select an individual, representative color for a word, and is mostly intended as a visualization or creative tool.

Using Common Sense

To create a system which automatically chooses the color which best matches incoming text, we must build on a system which models the way people think about the world. For this purpose, we use Open Mind Common Sense (OMCS).

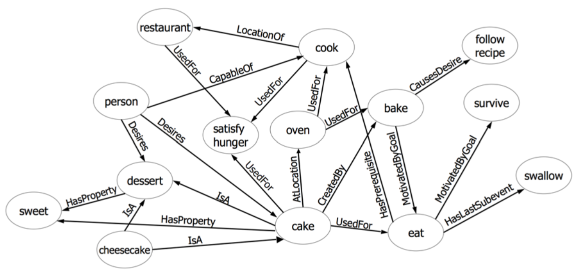


Figure 1: Some of the nodes and links in ConceptNet.

OMCS has been compiling a corpus of common sense knowledge from volunteers on the Internet since 1999. It currently has a corpus of over one million simple English statements which tend to describe how objects relate to one another, the goals and desires people have, and what events and objects cause which emotions. The data in OMCS has a reliablity score derived from how many people have approved an assertion or independently asserted the knowledge.

These statements are then automatically transformed into a semantic network representation, called ConceptNet (Havasi, Speer, and Alonso 2007)(Liu and Singh 2004). ConceptNet represents concepts as nodes which are connected by a set of 21 different relationships such as *PartOf* and *CreatedBy*. These nodes represent individual words or longer noun or verb phrases that appear in English, disregarding stopwords and suffixes. A portion of ConceptNet appears in Figure 1.

Collecting Crowd-sourced Color Data

In order to guess colors, even using common sense knowlege, we must start out with some data about what colors people associate with various real-world objects and abstract concepts.

We acquire our color information from three sources: ConceptNet, NodeBox, and the XKCD Web survey of color names.

NodeBox

One source of *a priori* color information is from the color database of NodeBox (Experimental Media Group 2010), a Python library for graphics programming. As part of an included color visualization toolkit, NodeBox provides a mapping from a small number of words to eleven pre-defined colors: blue, brown, green, grey, orange, pink, purple, red, white, and yellow.

An average of 90 concepts are paired with each color by designers at NodeBox. Some objects are paired with multiple colors. Examples of pairings range from the standard (*Christmas* is red and green, while *melancholy* is blue) to the more abstract (*peace* is pink and *loyalty* is yellow).

We build an association matrix with the colors and words pairs, normalized so the words match ConceptNet concepts, if possible.

ConceptNet

Within ConceptNet, there is already some color information provided as assertions about the colors of objects. These are particularly useful to us because they already use ConceptNet's representation of concepts, which we map all our other data onto.

Most of these assertions fall under the *HasProperty* relation, which connects objects to properties that might describe them, such as *small*, *round*, or *blue*. Whenever the target of a HasProperty relation is a color in NodeBox's list of basic colors, we include it as a data point describing the color of that object.

XKCD

In March 2010, Randall Munroe, author of a popular Web comic called XKCD, asked his readers to participate in a survey about color names. The setup of the survey was simple: it would generate a random color, show it to the user, and ask the user to name that color. The survey instructions encouraged both straightforward and creative color names, so a color might be described as "purple", "eggplant", "dusty prune", or "painful bruise purple". After compiling the survey results, Munroe made the raw data available on his Web site (Munroe 2010).

There were two forms of the survey: one chose colors at random from the entire RGB space, while the other asked only about fully-saturated colors. We made use of the data from the first version, which contains over 2.3 million associations between colors and names collected from over 150,000 users.

Matching colors with concepts

When encountering a word for which we must provide a color, the first thing we do is check to see if we already have a color (or colors) for this concept in our knowledgebase. These are the colors provided by the resources which we incorporated, or colors already found as properties in ConceptNet.

We do not rely on matching the exact name of a color, such as "lemon yellow" or "light minty green". We want to associate the color "lemon yellow" with the individual concepts "lemon" and "yellow", because it is a useful data point about what they look like. For this, we make use of ConceptNet's existing ability to extract concept names from phrases. Beyond simple tokenization, this allows discovery of multiple-word concepts (extracting "swimming pool" from "chlorinated swimming pool") and normalization of word endings (extracting "cloud" from "ocean under dark clouds").

We consider any concept that is matched to three or more colors in the color survey this way to have a color, in addition to all the concepts that were assigned colors from ConceptNet and NodeBox. Because this gives us multiple color values for most concepts, we next need a way to identify a consensus color value for each concept.

Identifying color centroids

The data we have compiled, particularly from Munroe's color survey, give us a scattered set of color values for each color name. For many color names, one can assume that there is

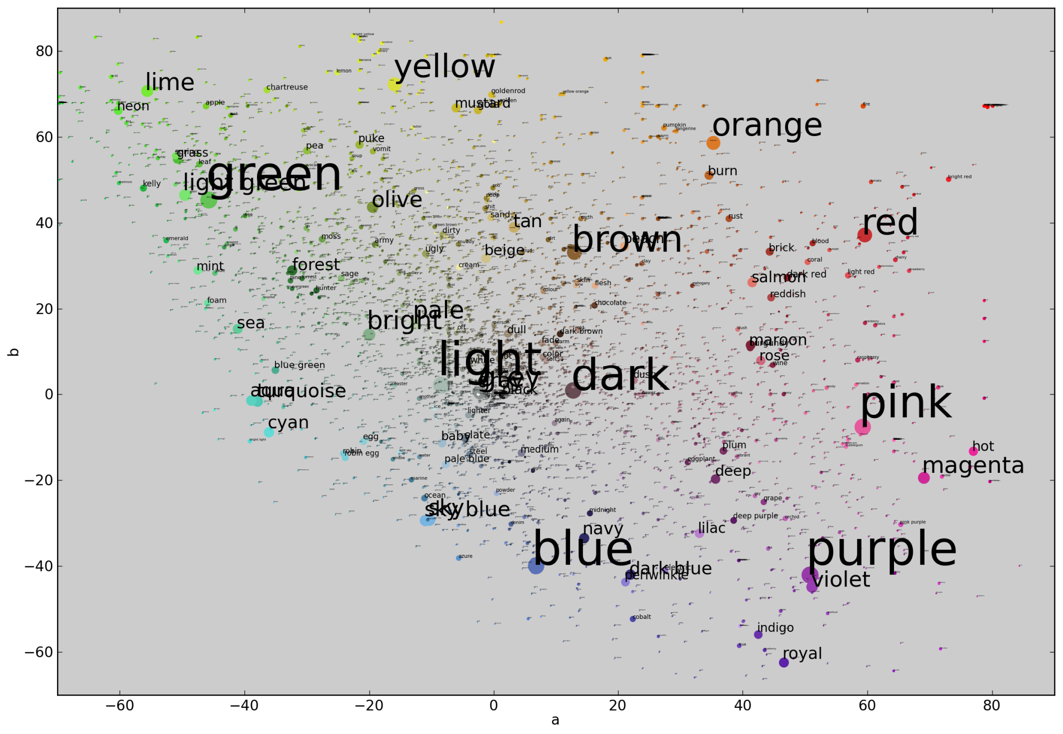


Figure 2: A chromaticity diagram (a vs. b in Lab coordinates) of the color centroids of all concepts. Larger concept names represent more information.

a "true" color value that many people roughly agree on, and that the color values we have are approximate observations to that true color value. Similarly, we can treat these as observations about ConceptNet concepts: a color identified as "lemon yellow" gives us an observation for the concept *lemon* and the concept *yellow*.

Munroe identified consensus color values from his data using the "average of a bunch of runs of a stochastic hill-climbing algorithm" (Munroe 2010), with the details of the algorithm left unspecified. We choose instead to find a point that we call the *color centroid*, which is approximately the observation with the smallest total distance to all other observations.

Finding the color centroid requires measuring distances between colors. RGB values are not ideal for this, because they do not correspond to the way people perceive color differences. For example, two shades that people identify as "bright green" can be very far apart in RGB, and green and blue are as far apart as red and blue even though people perceive them as being closer. Instead, whenever we measure differences between colors, we choose to do so in the CIE Lab color space, which is designed so that equal Euclidean distances correspond to equal perceived differences (Schand 2007).

We convert all the RGB observations from Munroe's color survey to Lab coordinates using the sRGB standard, and we also represent color descriptions that appear in ConceptNet and NodeBox as observations at specific points in Lab space. A statement that "money is green", for example, would produce an observation for the concept "money" at the CIE Lab point representing saturated green.

We define the color centroid of a concept as follows: first find the component-wise median of all its observations (the point with the median L value, median a value, and median b value), and then return the actual observation that is closest to that point in Euclidean distance. Using a median (which minimizes the mean distance) instead of a mean (which minimizes the mean-square distance) ensures that concepts with a wide range of color values do not simply appear gray and washed-out.

Interpolating unknown colors

What we have described so far allows Colorizer to extract color values from a large database of observations about colors. It can identify colors for concepts such as "eggplant", "clay", or "bright" just by looking up their color centroid. The interesting task that we will now describe is to synthesize a reasonable color for an *unknown* concept.

When Colorizer is asked for the color of a concept for which it has color data, it can simply return the centroid of that color data, as described above. When it is asked for the color of an unknown concept, it uses a nearest-neighbor smoother to interpolate the color using the known colors of concepts that are semantically related to it. This measure of semantic relatedness, which takes advantage of the large amounts of background knowledge in ConceptNet, is described in the next section.

The nearest-neighbor smoother takes in a list of concepts and their respective "relatedness" to the target concept. It chooses the ten most related concepts that have known colors, and returns a weighted average of those colors, using the relatedness score as the weight. What remains is to define the measure of relatedness, which we do using the background knowledge that ConceptNet provides.

Spectral association

Interpolating a color for an unknown concept depends on a measure of semantic relatedness. For example, if the color of "inlet" is unknown, we want to use facts such as "an inlet is part of a river" and "a river contains water" to infer that it should take its color from concepts such as "river" and "water".

One way to measure the connectedness between a pair of concepts numerically is to perform a number of steps of spreading activation. One node begins with an activation value of 1.0, and at each step, each node will distribute some of its activation to its neighbors. The weighted average of a number of steps of spreading activation will give a measure of relatedness, or "association", between all nodes and a target node.

This can be time-consuming to calculate, so instead we take advantage of a dimensionality-reduced representation of the concept graph to calculate these association values all at once. This technique, which we call "spectral association", is a variant of the AnalogySpace representation that OMCS uses to reason over ConceptNet (Speer, Havasi, and Lieberman 2008).

We can represent the concept graph as a square, symmetrical association matrix, C, whose rows and columns are both labeled with concepts. When two concepts are connected by an assertion in ConceptNet, we put the weight of that assertion in ConceptNet in the two appropriate entries of C. All remaining entries are 0. After constructing the matrix, we scale it so that its rows and columns are unit vectors.

Applying C to a vector containing a single concept spreads that concept's value to its connected concepts. Applying C^2 spreads that value to concepts connected by two links (including back to the concept itself). To spread the activation through any number of links, with diminishing returns, the operator we want is:

$$1 + C + \frac{C^2}{2!} + \frac{C^3}{3!} + ... = e^C$$

We can calculate this odd operator, e^C, because we can factor C using the spectral decomposition. As a square symmetric matrix, C can be represented as $V\Lambda V^T$, where V is an orthogonal real matrix of its eigenvectors and Λ is a diagonal matrix of its eigenvalues. At this step, we can ease computation by performing dimensionality reduction, keeping only the largest eigenvalues and their corresponding eigenvectors.

We can raise this expression to any power and cancel everything but the power of Λ. Therefore, $e^C \approx Ve^{\Lambda}V^T$. This expression lets us calculate spreading activation from any concept or combination of concepts quickly. While its result is a very large, dense matrix, the relevant entries or rows can be computed lazily by multiplying by each of the factors separately.

As one final step, we rescale this expression to put the association values on a consistent scale. Let $V' = Ve^{\Lambda/2}$, so that $e^C = V'V'^T$. If we normalize the rows of V' to unit vectors, yielding V'', then $V''V''^T$ (a matrix whose values we can still compute lazily from these two factors) will give us association values for all concepts that have a maximum of 1.0, instead of having much larger values for highly-connected concepts such as "person".

Some examples of the most-activated concepts starting from particular concepts are:

- table $\rightarrow$ table, salt shaker, dining room, chair, furniture store, carpenter, table cloth, glass front cupboard, bar stool, tablecloth ...
- keyboard $\rightarrow$ keyboard, send email, laptop, PC, personal computer, access Internet, CPU, motherboard, accomplish task, search information ...
- sad $\rightarrow$ sad, sob, weep, wail, sadness, upset, shy, water in eye, deep sadness, tear ...

We call this technique "spectral association", not because it inherently has anything to do with colors, but because it uses the spectral decomposition to relate entries to each other in a semantic network.

To find candidate concepts for interpolating the color of an unknown word or phrase, then, Colorizer will start from a vector of all concepts that appear in it, similarly to the way the concepts were extracted in the first place – so, for example, "orange peel" will activate the concepts "orange peel", "orange", and "peel". It then applies spectral association to that vector, and choose the most associated concepts that have known colors from the result. Each concept's color is not guaranteed to be unique — it's possible for two concepts to have the same RGB color.

Some examples of the results of spectral association and interpolation appear in the table of evaluation results, Table 1.

Evaluation

In order to evaluate Colorizer, we can divide up the large amounts of data that Munroe's color survey provides into training and test data. We do this in the form of a "leave N out" test, where we hold out all the information about certain colorful concepts from the training data, and evaluate the predicted color values against those held-out color values.

Because Colorizer assigns colors to concepts that can appear in many color names, we need to be careful to completely separate the training and test data. We randomly designate a number of concepts from ConceptNet to be "test concepts", and then place any color name that contains any of those concepts into the test set. If "lemon" is a test concept, for example, then the training set will contain no color survey answers including the word "lemon". We can then test whether Colorizer can infer the color associated with "lemon" according to its semantically related concepts.

Test setup

We randomly selected 200 concepts that appeared in both ConceptNet and the color survey. We held out color names

Concept	Related concepts	Test color	Colorizer
teddy bear	ferret, bear, small dog, sloth…	#8a3e0c	#988168
bruise	discomfort, trauma, unpleasant, open sore…	#72435f	#a57e70
slime	relish, vegetable, plant leaf, jade…	#80b638	#3cc243
paper	output, pencil, publish, report, page…	#e2daa2	#ac9c95
azalea	tulip, bloom, lilac, daffodil…	#d3458b	#cd9a94
ocean water	Pacific Ocean, sea, Indian Ocean, seawater…	#66dad2	#717aa6

Table 1: Some examples of how Colorizer reconstructs colors in the test data.

including those concepts from the training data set, and included them in the test data set if there were at least three data points for that concept, giving us 98 usable items of test data.

For each test concept, we considered the color centroid of all its data points to be the "correct" color to test against. Then, we compared various distances in the Lab color space:

- The distance from the correct color to 50% gray, as a baseline
- The distance from the correct color to the color that Colorizer predicts for the concept
- The distance from the correct color to an arbitrary other color for the same concept in the test data, to determine "human-level performance"

Results

The baseline distance over our test data was 49.0 – that is, a trivial system could predict that everything was gray and get an average distance of 49.0. At the other end of the scale, the average "human-level" distance that came from looking up points in the actual test data was 39.9. Colorizer's performance, at an average distance of 42.3, was noticeably closer to human-level than to the baseline.

A one-way ANOVA shows that these methods differ significantly in performance ($F(2, 194) = 5.47$, $p < .01$). A further Tukey HSD test shows that Colorizer performs significantly better than the baseline at the $p < .05$ level ($HSD[.05] = 6.75$), while the difference between Colorizer and human-level agreement is in fact nonsignificant. These results are shown in Figure 3.

Using Colorizer

Colorizer can be used to add color to static free text such as poetry or speeches. In addition to artistic applications, automatic color choice can show the changes in emotion in a document and help to analyze its flow of ideas. Colorizer can

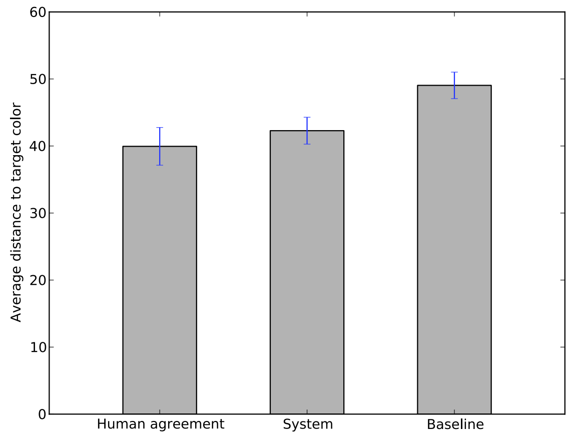


Figure 3: An evaluation of Colorizer's accuracy in predicting colors.

also be used in more free-form games to help automatically design and color elements that users create.

To demonstrate, we have created a web application which uses Colorizer to create a color visualization of text and poems, whose output appears in Figure 4. The application creates the visualization by applying Colorizer on multiple levels. First, individual words are colored, then lines of text, then paragraphs, and finally a color is selected for the entire document. These colors are then layered with partial transparency to form the visualization.

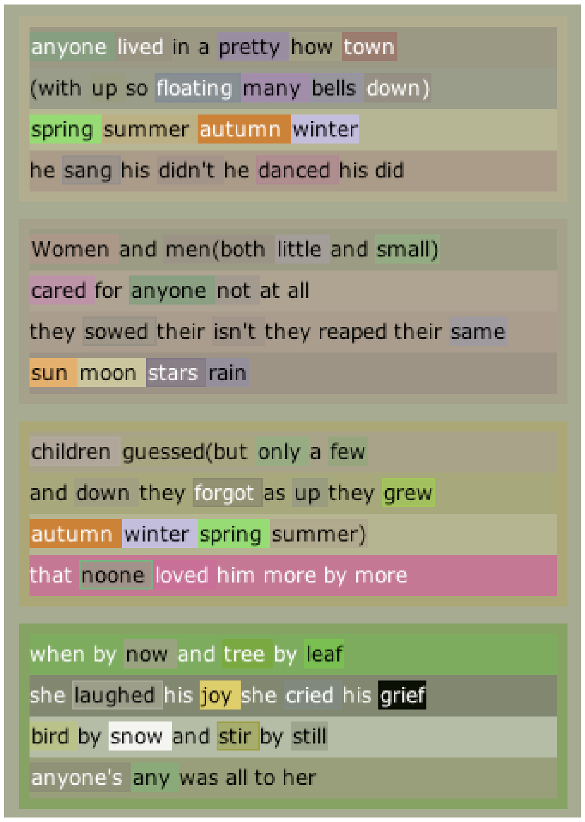
For this application, we created a measure of "colorfulness",which is a numerical representation of how much color is part of an object's representation. Concepts were defined to be more colorful when they were used more times in our input data or were inferred to be related to many of the "colorful" training concepts. Words with lower colorfulness have less of an effect on the color of their surrounding text, and are shown with a lower alpha value (more transparency) than colorful concepts.

The Colorizer process can be performed quickly over data that arrives in a stream, making it suitable for a wide variety of applications. We have combined Colorizer with streaming speech recognition to create an application which smoothly picks a color most appropriate to what is being said at the time. This has applications in storytelling, art, theater, improvisation, and children's play.

Conclusion

We believe our results are an encouraging step in the process of linking textual common sense information with other modalities of knowledge. Because our color selector performs close to humans' level of agreement in matching colors to concepts, it has the potential to be a useful component in art and design applications.

A limitation of this work so far is that it only generates single colors for single meanings. Some concepts are best expressed by sets of colors, or color schemes. An extension that we plan to explore is to find multiple representative color


anyone lived in a pretty how town
(with up so floating many bells down)
spring summer autumn winter
he sang his didn't he danced his did

Women and men(both little and small)
cared for anyone not at all
they sowed their isn't they reaped their same
sun moon stars rain

children guessed(but only a few
and down they forgot as up they grew
autumn winter spring summer)
that noone loved him more by more

when by now and tree by leaf
she laughed his joy she cried his grief
bird by snow and stir by still
anyone's any was all to her

Figure 4: Colorizer matches colors to the images and emotions of an E. E. Cummings poem.

centroids, instead of a single overall centroid, in order to generate color schemes. These could be used to automatically suggest colors for graphic designs based on the text that is being conveyed.

In the future, we will continue to explore applications which can benefit from automatically created colors and color schemes. We also plan to apply similar techniques to another sensory modality, creating a mapping between concepts and audio information such as ambient sounds and music.

References

Davis, E. 1990. *Representations of Commonsense Knowledge*. Morgan Kaufmann.

Experimental Media Group. 2010. Nodebox. NodeBox web site. `http://nodebox.net/code/index.php/Home`.

Havasi, C.; Speer, R.; and Alonso, J. 2007. ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent Advances in Natural Language Processing*.

Liu, H., and Singh, P. 2004. ConceptNet: A practical commonsense reasoning toolkit. *BT Technology Journal* 22(4):211–226.

McCarthy, J. 1959. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, 75–91. London: Her Majesty's Stationery Office.

Munroe, R. 2010. Color survey results. XKCD blog. `http://blog.xkcd.com/2010/05/03/color-survey-results/`.

Schand, J. 2007. *Colorimetry*. Wiley-Interscience.

Smedt, T. D., and Bleser, F. D. 2010. Nodebox Prism. NodeBox web site. `http://nodebox.net/code/index.php/Prism`.

Speer, R.; Havasi, C.; and Lieberman, H. 2008. AnalogySpace: Reducing the dimensionality of common sense knowledge. *Proceedings of AAAI 2008*.