Emotion Driven Literary Bindings

by Tiffany France

Date: May 15, 2021
Thesis Advisor: Alec Barrett
URL: https://tiffanyfrance.com/ml-books/
This thesis is submitted in partial fulfillment of the requirements for the degree of Master of Science in Data Visualization at Parsons School of Design.

Table of Content

Abstract

This thesis uses machine learning emotion classification processes in order to create emotionally representative book covers. Both proprietary and hand crafted APIs are used to analyze the corpuses of 26 Project Gutenberg novels. Multiple emotion recognition models are evaluated for accuracy and quality of output. Results indicate that machine learning models are able to adequately predict the emotions in novels and generate supporting data which can be visualized into book cover art.

The final outcome of this project will be printable manuscripts with custom generated covers. The visualizations are built using D3js and data storytelling techniques. The charts will be imported into InDesign for final production and typesetting. In order to avoid copyright issues, all books used are in the public domain, with original publish dates between 1800 and 1925. The physical book will contain the text of the original author, as well as accompanying text and diagrams provided from this project.

Introduction

> *“Why, what book is it the wench has got hold on?” he burst out at last.*
>
> *“‘The History of the Devil,’ by Daniel Defoe,—not quite the right book for a little girl,” said Mr Riley. “How came it among your books, Mr Tulliver?”*
>
> *Maggie looked hurt and discouraged, while her father said,—*
>
> *“Why, it’s one o’ the books I bought at Partridge’s sale. They was all bound alike,—it’s a good binding, you see,—and I thought they’d be all good books. There’s Jeremy Taylor’s ‘Holy Living and Dying’ among ’em. I read in it often of a Sunday” (Mr Tulliver felt somehow a familiarity with that great writer, because his name was Jeremy); “and there’s a lot more of ’em,—sermons mostly, I think,—but they’ve all got the same covers, and I thought they were all o’ one sample, as you may say. But it seems one mustn’t judge by th’ outside. This is a puzzlin’ world.” - The Mill on the Floss, George Eliot*

George Eliot’s novel *The Mill on the Floss*, published in 1860, contains an early reference to the English idiom “Don’t judge a book by it’s cover.” The phrase is meant to differentiate the inside content from the outside appearance, both literally as in the case of books, and metaphorically.

It is commonly accepted in publishing that a book's cover is the "foremost aspect" of the publishing process, and the component which has the most influence on the sale of a book[1]. Making a cover visually appealing is an important part of the process. It is also important that the cover reflects the contents inside.

Literature is an expression of the human emotional range through stories. In his paper "Feelings in Literature", Johansen notes that feelings have the most important role in literature, giving us the space for empathy and emotional engagement with the characters' endeavours. In addition to empathy and emotional engagement, feelings allow us to "understand and engage ourselves in fictional worlds", with emotions being used as a vehicle of the imagination to transport us into an imaginary space with fictional dilemmas and resolutions[2].

Book exteriors can be expected to capture the emotional journey or some aspect of that journey within the cover art. This thesis attempts to create a computed process for collecting emotions of text-based data and expressing the results through the visualization techniques. The output of the process will be interpretive, generative book covers based on the emotional landscape of the words in each corpus combined with an artistic representation of that information.

In order to evaluate the emotional range of the text, machine learning techniques will be used, specifically emotion recognition and sentiment analysis. The data will focus on the "Top 100 EBooks" list from Project Gutenberg[3]. The text of these public domain books is for use with almost no restrictions as outlined in the Project Gutenberg License[4] and U.S. Copyright Laws. The content from these books will be collected through a script in Jupyter Notebooks, then evaluated by a machine learning model.

IBM Watson provides an emotion recognition tool within the Alchemy Language Service known as Tone Analyzer. This product will be used in order to evaluate the emotions of the text for 26 Project Gutenberg works. The product returns seven emotions: Anger, Fear, Joy, Sadness, Analytical, Confident, and Tentative. The API returns JSON which will evaluate the results based on each sentence or chapter level as desired. Values are returned between 0 and 1.

---

[1] Yampbell
[2] Johansen
[3] Project Gutenberg, https://www.gutenberg.org/browse/scores/top
[4] Project Gutenberg, https://www.gutenberg.org/policy/license.html

Other APIs will be used as detailed below. These APIs are a combination of proprietary, open source, and hand crafted machine learning tasks. All outputs will be evaluated for merit, then visualized for the project.

Additionally, the project will include an interactive piece for users where they can evaluate their interpretations of the passages. Using a custom designed interface, users can tweak their understanding of the emotions contained in a piece, then compare that against the results of the machine learning model.

This project hopes to find out if book covers can accurately and distinctly be created using a generative process with the actual text of the book as the data. The project will also evaluate multiple emotion recognition models for accuracy and value within the literary space. Additionally, it will provide the user a platform with which to "test" their interpretation against the machine driven models.

Sentiment Analysis vs Emotion Recognition

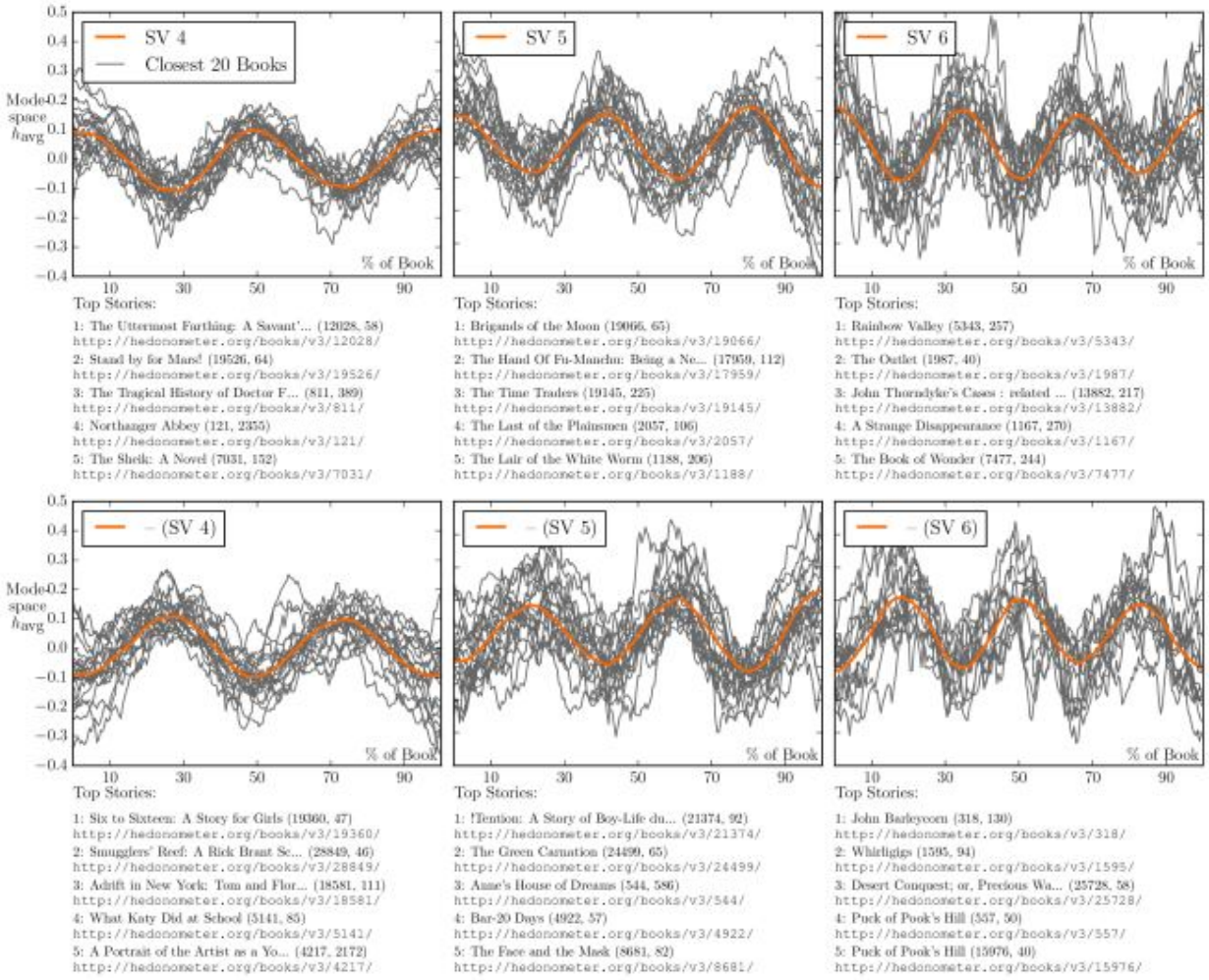
Kurt Vonnegut, the American author of *Slaughterhouse-five* and 14 other novels, attended the University of Chicago between 1945 and 1947 in hopes of obtaining a master's degree in Anthropology. His proposed idea for his thesis was that "stories have shapes which can be drawn on graph paper." He noted "the shape of a given society's stories is at least as interesting as the shape of its pots or spearheads"[5]. His thesis proposal suggested using statistical methods to show the recurrence of master plots within a community's literary space. By evaluating the emotional patterns within storylines, one could potentially trace an arc representing the "shape of stories". The Anthropology department denied his thesis, because, as Kurt Vonnegat states: "it was so simple and looked like too much fun"[6]. But the idea didn't leave him. He continued to espouse it's virtues, including suggesting ways that computers could aid in understanding of master story plot arcs.

The concept was taken up years later by a team of researchers at the University of Vermont. In their 2016 paper "The Emotional Arcs of Stories are Dominated by Six Basic Shapes" the authors state that "advances in computing power, natural language processing, and digitization of text now make it possible to study a culture's evolution through its text using a big data lens"[7].

---

[5] Vonnegut
[6] Open Culture, https://www.openculture.com/2019/12/why-the-university-of-chicago-rejected-kurt-vonneguts-masters-thesis.html
[7] Reagan

The team identified 10,000 keywords with “meaningful” sentiment scores, collected in a crowdsourcing effort using Mechanical Turk. For their dataset, the researchers filtered more than 50,000 Project Gutenberg books through various requirements, including limiting data to English speaking books between 20,000 and 100,000 pages and removing duplicate titles. The books were run through a process of matching the weighted keywords against the preprocessed texts in the dataset. They used a combination of statistical and machine learning methods (Singular Value Decomposition (SVD), Hierarchical Clustering, Self Organizing Map (SOM)) to match features and apply classification labels. The conclusion of the study noted that the process identified and returned six master emotional arcs[8].



To determine a story’s emotional arc, the researchers used labMT’s dataset, hedonometer, which is a list of words with happiness scores[9]. The scores ranged from “terrorist” at the bottom

[8] Reagan
[9] labMT-Hedonometer, http://hedonometer.org/words/labMT-en-v1/

with a score of 1.3 to laughter at the top with a score 8.5. Because of the single scale nature of the hedonometer, this type of scoring is more sentiment-based, than emotion (which would consist of specific feeling based words or classes). However, because of a combination of labels and scoring, it could be suggested that the study lands somewhere between sentiment analysis and emotion detection.

Sentiment analysis is the understanding of opinion based on a positive/neutral/negative scale. Emotion recognition, however, would not use the three categories of positive, neutral, and negative, rather it would provide some sort of score for specific emotions (anger, happiness, joy, frustration, boredom). Often, these labels contain a magnitude score which relates the intensity of the class.

Emotion detection is a natural language processing task which falls into the realm of "affective computing" in machine learning under the artificial intelligence umbrella. It is a branch of sentiment analysis, which is another natural language processing task. In general, the term "sentiment analysis" implies an identification of positive or negative returns. Emotion recognition explores the specific classification of that feeling. For instance, we can have negative sentiment ratings for the sentences "I am bored" and "I am angry", however the feelings behind each statement differ greatly.

Emotion detection can be useful for showing a detailed picture of the sentiment. In the study "Emotion Recognition from Text Based on Automatically Generated Rules", the authors point out that emotion detection can have many benefits for applications such as calendar, email, and social networking. For instance, understanding the emotions of a meeting or email may help the receiver prepare psychologically for an upcoming interaction[10].

Emotion detection can also be useful for the customer service industry. Not only can it assist in better understanding patrons' needs, but it can also be useful as a performance metric for customer service agents. The assessment can help provide metrics on why a customer is dissatisfied, and not simply that the customer is unhappy.

In the case of literature, understanding the range of emotions may be helpful in a number of situations. On a large scale, emotion detection within stories may find patterns of civilizations, as proposed by Kurt Vonnegut, in a more meaningful way than just understanding positive or negative sentiment. If, as the saying goes, we are the stories we tell, what can we learn about humanity by looking at the unfolding of emotions in beloved books? Further, how do these themes repeat or diverge amongst societies?

[10] Shaheen

On an individual scale, emotion detection may be helpful for book recommendation algorithms. If a reader or section of readers can identify the emotional landscape of a book they enjoy, perhaps a similar one will be of interest to them. Likewise, if they dislike books with a certain emotional range, this output could be useful in steering them away from certain pieces. Visualizing the output of emotion detection can be useful as a map to guide in understanding how we view and experience story arcs both within the printed page and as the human race.

Emotion detection in literature may show if there is a trend of emotional distribution amongst genres or authors. While the Vonnegut-inspired study returns six emotional arcs, more answers can be found by digging into the text and identifying the specific emotional drivers in a selected literary corpus.

Categorizing Emotions

Emotional labels vary amongst studies when classifying text. Popular terms include words such as happy, sad, angry, and surprise. The psychoanalytic field lacks an official consensus on how to organize and define human emotions[11]. Machine learning scientists often use the schemas based on the work of four prominent researchers: Plutchik, Ekman, Parrot, and Tomkins.

Plutchik

Robert Plutchik's work is a popular preference for data scientists interested in classifying emotions. He proposed eight primary emotions: anger, fear, sadness, disgust, surprise, anticipation, trust, and joy. He placed these emotions on a color wheel, assigning a hue to each term. Colors are used to highlight both opposite emotions and intensity within the vertical dimension of the flower cones[12].

The wheel of emotions was later expanded into the "hourglass of emotions". The hourglass model uses the same terms but categorizes them into four sentic dimensions. This variation is "biologically inspired, psychologically motivated, and based in the idea that emotional states result from the selective activation/deactivation of different resources in the brain"[13].

[11] Gu
[12] 6 seconds, https://www.6seconds.org/2020/08/11/plutchik-wheel-emotions/
[13] Shaheen

Ekman

In their paper, “Semantic-Emotion Neural Network for Emotion Recognition from Text”, the authors adopt Paul Ekman’s six basic emotions for their system. Ekman initially proposed seven basic emotions: fear, anger, joy, sad, contempt, disgust, and surprise; but later he dropped contempt as a fundamental emotion, landing on six basic emotions: fear, anger, joy, sadness, disgust, and surprise[14].

Parrot

Shivhare proposed using the emotion system detailed in W. Gerrod Parrot’s book titled “Emotions in Social Psychology”. The emotion system “formally classified the human emotions through an emotion hierarchy in six classes at primary level which are Love, Joy, Anger, Sadness, Fear and Surprise”[15]. Shivhare notes that additional emotions fall into secondary and tertiary levels. Noting the complexity of emotional hierarchy, Shivhare creates a proposed architecture that takes into account emotion word ontology and leveled schema.

Tomkins

Silvan Tonkins is a psychologist and social theorist who worked on a theory of emotion classification. His work yielded eight basic emotions: enjoyment/joy, interest/excitement, surprise, anger/rage, disgust, distress/anguish, fear, shame/humiliation. In the paper, “A New Feature Selection Scheme for Emotion Recognition from Text”, the authors point out that “several of these emotions are created based on the character or personality of the individual whereas the rest of them are formed essentially related to the social interactions”[16].

Wilcox

Gloria Wilcox’s “Feeling Wheel” was designed to help people navigate through their emotions. The wheel initially contained four major emotions: scared, sad, mad, and glad. In order to balance the diagram, Wilcox expanded “glad” into three emotions: joyful, powerful, and peaceful[17].

---

[14] Gu
[15] Shivhare
[16] Erenel
[17] The Feeling Wheel, https://allthefeelz.app/feeling-wheel/

Existing Research in Emotion Recognition

The field of Emotion Recognition began in the mid-1990s with the work of Rosalind Picard on "affective computing". Emotion detection is split into several techniques including face, image, speech, video, sound, and text recognition. Batabaatar notes that text is becoming increasingly important due to the large amount of textual data on the web including blogs, tweets, forums, and comment sections. Even though the concepts have been around for several decades, not much work has been committed to the area of emotion detection (even less on text-based input)[18].

Preprocessing

Because of the infancy of the field of research, there are still several emerging approaches. All techniques begin with a varying level of preprocessing.

Preprocessing is the act of retrieving the text from the document and applying certain cleaning mechanisms for better results. All examples go through some type of tokenization or word parsing. The BOW model (bag of words) is an NLP process for parsing words as strings, while keeping a count of the occurrence of each word. This process is frequently used within text classification systems, including emotion recognition.

Preprocessing methods vary across studies. They include techniques such as removing stop words, identifying non-content sentences, disambugation, tense setting, identifying parts of speech, casing, negation, and others. One key concept in preprocessing is lemmatization which is the grouping of words to be evaluated as a single concept. This is important for phrases such as "school bus" or even longer phrases or euphemisms. Researchers often note the importance of clean data and the effect on both quality of final output and relieving the stress of computational effort.

Five Approaches to Emotion Detection

Within the field, there are five primary approaches: Keyword, Lexical, Learning Model, Deep Learning, and Hybrid.

Keyword-based Approach

The keyword-based approach finds "occurrences of keywords in a given text and assigns an emotion label based on the detected keyword"[19]. For this approach, a list of emotion-based

[18] Batabaara
[19] Alswaidan

keywords are identified (possibly from a source like WordNet). Once preprocessing is complete on the dataset, the keyword list is used to "spot" keywords in the data. Once a match is found, the intensity of the emotion is weighted. Negation is sometimes checked after this stage. Finally emotions are labeled. This process requires the researcher to identify a list of words, subjecting the process to bias. The process is manual and not easily updated or transferred. Even though this process can prove somewhat successful, several projects using this approach have found significant levels of mislabeled emotions[20].

Lexical-based Approach

A lexical affinity approach expands on the keyword technique. In addition to spotting and matching keywords, this approach "assigns a probabilistic affinity for a particular emotion to arbitrary words apart from picking up emotional keywords"[21]. Like the prior approach, this technique has a low barrier of entry for technical knowledge, however it requires manual work to set up. Additionally, it is possible to attribute emotional assessment to a word incorrectly. Shivhare provides the example of assessing the word "accident" with a negative emotion. This would not be correct if the sentence evaluated was "I avoided an accident" or "I met my girlfriend by accident"[22].

Learning Approach

The learning-based approach attempts to classify processed input with an emotion label using machine learning models. A common approach is to use SVM (Support Vector Machines), which is a supervised learning model based on research performed in the late 20th century by developer Vladimir Vapnik and his colleagues. The model analyzes data for classification, known as a prediction model. It works by creating examples as points in space, with new points mapped on top of the hyper-planes. The new points are classified with whichever hyper-plane they land in. Other machine learning techniques for emotion detection classification include the use of k-Nearest Neighbor, Naive Bayes classifier, Maximum Entropy, and statistical methods, such as Chi-Square.

Though many scenarios exist, one such classical learning model can be seen in the research completed by Diman Ghazi and fellow researchers. Their experiment used a combination of SVM and BOW. The steps included (1) separating emotion sentences and non-emotion sentences (2) identifying the polarity of the emotion sentences (positive includes happiness,

[20] Alswaidan
[21] Shivhare
[22] Shivhare

negative includes sadness, fear, anger, etc) (3) then finally classifying the negative emotions with an SVM classification system[23].

One drawback of the learning based approach is that it still depends on keyword extraction to provide classification. Both machine learning approaches (learning approach and deep learning) have the challenge of needing to train large datasets in order to get accurate results[24].

Deep Learning Approach

A deep learning approach moves past simpler learning-based approaches. Deep learning programs learn and understand input from concepts and their relations to other concepts. They are able to learn complicated systems by basing the learning on simpler established systems. There are several popular deep learning approaches used for emotion detection from text. Singular Value Decomposition (SVD) is a technique for decomposing matrices into smaller chunks. This is often followed by clustering the output using an unsupervised method, such as a Self Organized Map (SOM)[25].

Another frequently used deep learning technique is Long Short-Term Memory (LSTM). In their study "Semantic-Emotion Neural Network for Emotion Recognition from Text", the researchers used bidirectional LSTM to "derive the hidden state of each word" and gather contextual and semantic information from a words in an experimental dataset of microblogs[26].

Other deep learning algorithms for emotion detection include Convolutional Neural Network (CNN), a type of Artificial Neural Network (ANN). CNN and Extreme Learning Machine (ELM) are "feedforward" neural networks that work well for emotion recognition classification, clustering, and regression. Feedforward systems move information in one direction, from input, to hidden layers, to output layers, as opposed to recurrent neural networks which can work in a cycle. In the aforementioned study, a CNN network was used to "extract emotion features from emotion-based word embeddings" using filtered layers on local features[27]. The combination of BiLSTM and CNN proved to show exceptional returns for emotion recognition within emotional word embeddings.

Hybrid Approach

In the paper "A Survey of State-of-the-Art Approaches for Emotion Recognition in Text", the authors mention several hybrid approaches to emotion detection. These included research

[23] Ghazi
[24] Cambria
[25] Reagan
[26] Batbaatar
[27] Batbaatar

combining keyword based approaches with learning based approaches which showed a marked improvement in recognition accuracy over other existing singular approaches[28]. Other researchers found greater accuracy by combining traditional learning with deep learning, lexical methods with deep learning, and additional various combinations[29].

Limitations of Emotion Detection

Though the field of emotion detection has been around since the mid-1990s, there is a notable lack of APIs available for analyzing and classifying data. This could be the result of criticism for emotion detection as a pseudoscience. It is considered exceedingly difficult to interpret certain emotions like irony, sarcasm, and euphemisms from text. The sentence "You really hurt my feelings, I am so sad," can indicate true intent or mockery. It is impossible to understand the sentiment without gathering more information about the context.

Not only is it difficult to interpret the meaning of textual data, but at certain times humans don't understand their own emotions. "Machines need to have accurate ground truth for emotion modeling"[30]. It can be challenging to combine the malleable nature of emotions with the hard logic of computers. The question is a psychological one: Who defines what it means to be angry or sad? And do all humans articulate this in the same way?

In 2019, Batbaatar (et all) notes that many machine learning methods used for emotion recognition "overly rely on handcrafted features which require lots of manual design and adjustments"[31]. This manual work can be expensive and take a long time to run. (The paper proposes to combat that problem with deep learning solutions.)

Most emotion detection systems are created for social media or comments sections (Twitter, YouTube comments, etc). Therefore, they cater to components such as URLs and emojis. For instance, the API Text2Emotion dedicates a large part of the computational process to running an emoji_extractor in order to decode the meaning of the icons used in text. If unneeded, this feature adds unnecessary overhead to the codebase. Additionally, certain linguistic choices may be used in casual chat, as opposed to formal complaints, as opposed to literary work, that might not be accounted for in existing APIs.

[28] Seol
[29] Alswaidan
[30] Shaheen
[31] Batbaatar

The field of emotion detection covers not just text but images, speech, and video. A large portion of APIs are dedicated specifically to images and facial recognition. While combing through the open source and enterprise libraries, it is sometimes difficult to know which media the code uses as input unless the authors have explicitly documented the use case.

Because it is a new field, libraries come and go. This could be because the APIs were experimental, out of date, too difficult to manage, or because the technology was put behind a pay wall. The web is littered with broken links to repositories that no longer exist. The APIs included in this study were selected with these factors in mind. They are described in detail in the APIs/Models section of this paper.

About the Source Data

Project Gutenberg was founded in 1971 by author Michael S. Hart with the goal of creating and offering a digital library. The first digitized "book" added to the collection was the *Declaration of Independence*. The project is run by volunteers who participate in creating and maintaining the archive. The project contains about 62,000 free eBooks. Most of these items are free for use and distribution under the Gutenberg Licence[32].

The data for this project was collected both manually and programmatically. A few small APIs are available for Project Gutenberg, but they were unreliable and bug-prone. There is no official API for Project Gutenberg, though Gutenberg 0.8.1 Python library was used for collecting clean portions of the texts after other book information was identified.

132 unique book titles were selected from two list sources:

- Project Gutenberg Top 100 EBooks Last 30 Days, https://www.gutenberg.org/browse/scores/top
- Best Books of the 19th Century, Goodreads: https://www.goodreads.com/list/show/16.Best_Books_of_the_19th_Century

This list was filtered by removing duplicates and keeping only novels published after 1800. The final dataset consists of 90 novels with publication dates ranging from 1808 (Goethe's Faust) to 1925 (Fitzgerald's Great Gatsby). The date range was chosen in order to maximize the potential for being interpreted correctly by the code. This project visualizes 26 of the 90 novels on the accompanying web application.

[32] Project Gutenberg Licence

Thoughts on 19th Century Writing

In order to better understand the output of the emotion APIs, it may be useful to understand the political, social, and cultural behaviors of the time period when these books were written. The majority of novels used were published during the Victorian era (1837-1901). This era appears after the end of romanticism and Age of Enlightenment, and before the beginning of modernist literary styles.

The 1800s represented great change throughout the world. The Industrial Revolution created high productivity, mass production, and factory working environments. Railroads were laid to ship goods for long distances. The first telephone call was placed in 1876. Light bulbs were invented in 1878. Advances were made in Mathematics and Science, including the publication of Charles Darwin's *The Origin of Species by Means of Natural Selection*. Better medicine lead to longer, healthier lives. Many countries saw a strong push for social change. In Germany, Karl Marx published the *Communist Manifesto* in 1848, considered to be one of the most influential pamphlets ever published. The British empire controlled a fifth of the world's land including Canada, Australia, India, and parts of Africa.

Greater literacy rates lead to the consumption of more literary works. In England alone, over 60,000 works were published during Victoria's reign. Books became more accessible to the masses with the invention of the rotary press in 1847.

19th century literature captures the social climate, personal struggles, and industrial advancements of the time. Based on his own personal experiences, Charles Dickens writing often captures the toil of the everyday working man. His writing has a fondness for the moral upright nature of ambition and self improvement, evident in his portrayal of Pip in the novel *Great Expectations*. Dickens and other writers of the time also shed light on class issues.

In Knut Hamsun's novel *Hunger*, we see the cruelness of poverty from the eyes of a starving protagonist in Norway. In Upton Sinclair's *The Jungle*, we are revealed the working conditions of factories of the time. These novels lead to greater compassion pushing forward social change.

In the United States, slavery was abolished in 1865. This project includes several slave narratives of the 19th century including *Uncle Tom's Cabin*, *Incidents in the Life of a Slave Girl*, and *Narrative of the Life of Frederick Douglass*. Additionally, we see attitudes on race reflected in novels such as *Heart of Darkness*. In *Wuthering Heights*, Emily Bronte highlights the ill treatment of the Romani orphan Heathcliff who is described from the first meeting as "a dirty, ragged, black-haired child", with maltreatment that forms the character's hardened personality.

Literature created during this time often idealised the protagonist. For instance, in the *Bleak House*, Charles Dickens packs the main character, Esther, with a selfless and nurturing manner, the upstanding model for 19th century women. Jane Austen's novels often highlight the role of women in the society, including the pressures of finding a suitable husband, as a central theme in novels such as *Pride and Prejudice*, *Emma*, and *Sense and Sensibility*.

The industrial revolution combined with advances in technology lead to a rise of science fiction novels. H.G. Wells published *The Time Machine* in 1895 and *The War of the Worlds* in 1898. French novelist Jules Verne chronicled fantastic adventures in *Twenty Thousand Leagues Under the Sea* (1870) and *Around the World in Eighty Days* (1873). The detective novel became popular with characters such as Sherlock Holmes, published in 1892.

In addition to industry and technological advancements, gothic writing also became popular. The gothic novel is also known today as terror or horror novels. We see the combination of current day sentiment mixed gothic themes in Mary Shelley Wollstonecraft's *Frankenstein* (1818), Bram Stoker's *Dracula* (1897), Robert Louis Stevenson's *Dr. Jekyll and Mr. Hyde* (1886), and Victor Hugo's *The Hunchback of Notre-Dame* (1831).

With the advancements in literacy, children's tales were also popular. Several now famous books published in the 19th century include *Grimm's Fairy Tales* (1812), *Alice's Adventures in Wonderland* (1865), *Treasure Island* (1883) and *The Jungle Book* (1894).

Book lengths were long, often broken up and released as multipart series. Small publications were sold for a penny, called Penny Dreadfuls. A few of the longest publications in this study include *Middlemarch* by George Eliot containing 316,059 words, *The Brothers Karamazov* by Fyodor Dostoyevsky containing 364,153 words, and *War and Peace* by Leo Tolstoy containing 587,287 words. By means of comparison, the average novel today is about 90,000 words; for example, Harry Potter and the Sorcerer's Stone has 76,944 words.

APIs / Models

Emotion recognition systems vary greatly in their evaluation metrics and scoring results. Therefore, I wanted to evaluate the corpus using a multitude of approaches in order to detect consistency and/or interesting patterns among the available tools in the field. I am utilizing three proprietary APIs, three open source APIs, and code from three code examples. The APIs/codebases were specifically built for emotion detection, though their labeling schema and output is not consistent. The three enterprise APIs are pay models, as listed below. Exact code

architecture and emotion detection models are not revealed for the enterprise solutions, presumably because of the proprietary nature of the product.

Proprietary APIs

MeaningCloud

MeaningCloud[33] recently added a Predefined Deep Categorization model for Emotion Detection specifically for evaluating text. The product uses Plutchik's Wheel of Emotion, with eight primary bipolar emotion labels: joy/sadness, anger/fear, trust/disgust, surprise/anticipation. This model can be run using node, returning JSON with an absolute value, or count for the number of instances of a label, and a relative count for the percent magnitude.

Parallel Dots

ShelfWatch by ParallelDots[34] offers an Emotion Analysis API. The product evaluates six primary emotions: happy, angry, excited, sad, fear, and bored. The API is based on deep learning algorithms which are used to extract features from text. The features are then used to appropriately classify the emotion of the data. This product uses CNN (Convolutional Neural Networks) on an in-house tagged dataset. This was the most expensive and limited of the enterprise options. The free tier was limited to a certain number of API "credits" per day, which wasn't enough to run a single Gutenberg novel. However, the credits reset daily so I could split a novel over multiple days.

IBM Watson Tone Analyzer

IBM Watson Developer Cloud offers a Tone Analyzer service that "uses linguistic analysis to detect joy, fear, sadness, anger, analytical, confident and tentative tones found in text"[35]. The API works in various formats, including CURL and Python which were both used for this project. The analyzer returns information both at the document level and at the sentence level. The API is robust with more options than the other two enterprise APIs, including user-set properties for configuring the level of output.

[33] https://www.meaningcloud.com/products/deep-categorization
[34] https://www.paralleldots.com/emotion-analysis
[35] https://tone-analyzer-demo.ng.bluemix.net/

Open Source APIs

Text2Emotion

Text2Emotion[36] is a simple python package with the goal of extracting emotions from content. The system returns five emotion labels: Happy, Angry, Sad, Surprise, and Fear[37]. The API performs three steps: preprocessing, keyword matching, and then applying a classification score. A large part of the library is dedicated to preprocessing work for cleaning URLs and emojis, indicating that the library was likely made with social media in mind. While the package is simple in nature, it is one of the more publicized and available APIs in that it has a repository, a website, a PyPi package, and a handful of blog posts on Medium. The "marketing" of this service causes it to show high and often when searching for "Emotion Recognition APIs". Though the library is not the most robust option, there is something to learn from the detailed documentation of this product. A service, whether open source or not, cannot be used or appreciated if it cannot be found.

LIMBIC

Limbic is a lexicon-based emotion recognition library trained on ~90 books. The tool is intended to be used for plain text, books, and subtitles. The name "limbic" comes from the "limbic system", a part of the brain which controls emotions and behavior. The API evaluates the following emotions: sadness, joy, fear, and anger. These emotions are referred to as the "Affection Emotions". The API also returns an intensity score for each label. One major drawback of Limbic, as well as most services used in this paper, is that it does not handle negation, i.e. "not happy" is evaluated as "happy". As of writing this paper, the author is working on a PyPi package but a user can still run the library by downloading the repository.

IEMOCAP

IEMOCAP (Interactive Emotional Dyadic Motion Capture) is a database with video, speech, motion capture of faces, and text. This database is annotated into the following categories: anger, happiness, sadness, and neutrality. These emotions also contain "dimensional labels" including valence, activation, and dominance. The database was created as a corpus for emotion related studies seeking to understand "expressive human communication"[38]. In 2019, a group of researchers published the 3rd version of a paper on Artificial Intelligence using this database entitled "Multi-Modal Emotion Recognition on IEMOCAP Dataset using Deep

[36] https://pypi.org/project/text2emotion/
[37] API documentation sometimes uppercases emotion lists. The casing in this paper follows the original reference.
[38] https://sail.usc.edu/iemocap/

Learning"[39]. The authors also produced a code base available in GitHub[40]. This study uses neural networks to perform emotion recognition on all parts of the IEMOCAP database, including speech, facial recognition, and text. The goal of the study was to use multiple forms of input (not solely text or video) in order to provide a multi-pronged analysis of the subject's emotions. While the study went far into depth outside of the goals of this thesis, there were several helpful takeaways found in the python code on the repository[41].

Code Examples/Datasets

Proprietary codebases were locked down. I could only understand how they were built through the limited information provided in the documentation of the product. The open source APIs were sparse, and were built with specific goals in mind. Therefore, in order to have full control over the intention of the code use, I wanted to create my own evaluation systems. For that I referenced the following APIs and datasets.

Lexis

Lexis is a React based application running a Flask backend[42]. The work is the result of a project created by six developers at a hackathon (HackFMI 8). Lexis evaluates text for six categories: anger, disgust, fear, happiness, irony, neutral, sadness, sarcasm, and surprise. The product relies on Scikit Learn, a popular Python machine learning library used throughout this project. The model was trained on 75,000 tweets. It analyzes and visualizes sentence level data.

Emotion-Analysis-On-Text

Emotion-Analysis-On-Text is a project created by Bhagyashree[43], while attending IIIT as a graduate student of Machine Learning and Intelligence System[44]. Her code is a bit undeveloped, however, there were a lot of good nuggets of information to learn. Bhagya uses supervised learning models, specifically SVM, in order to detect emotions from text. Her dependencies include the typical Machine Learning Python libraries: Numpy, Pandas, NLTK, and Scikit Learn. Her research follows one of the papers mentioned in this thesis[45].

---

[39] https://arxiv.org/abs/1804.05788
[40] https://github.com/Samarth-Tripathi/IEMOCAP-Emotion-Detection
[41] https://github.com/Samarth-Tripathi/IEMOCAP-Emotion-Detection/tree/master/code/python_files
[42] Lexis, https://github.com/frisibeli/lexis-text-analysis
[43] This is her GitHub username as her full name was not listed with her work. Her first name, Bhagya, was mentioned, and is referenced here.
[44] Emotion Analysis On Text, https://github.com/Bhagya4347/Emotion-Analysis-On-Text
[45] Bhagyashree, https://airccj.org/CSCP/vol2/csit2237.pdf

Hedonometer

Intrigued by the study based on Kurt Vonnegut's theories, I wanted to better understand the study from University of Vermont which used a hedonometer. The labMT hedonometer contains 10,222 words with happiness scores[46]. I used this table for building a keyword match described in detail below.

Process

Acquiring the Corpus

Project Gutenberg works are labeled with a unique identifier. For example, *Wuthering Heights* has a book ID of 768. This ID is searchable on the website. It also can be used to list all the files associated with that work by appending the ID to the URL, for example, "https://www.gutenberg.org/files/**768**/". This link contains all the files (text, epub, html) available for the novel. Ideally, I would have liked this ID to play a more pivotal role in gathering the data, however several problems arised.

The first problem is that not all the URLs are the same. Some are formatted with just the number. Others have a number and a prefix, for example 768-h. Without looking at the file, I was unable to detect a pattern for when these prefixes were applied. This meant I was limited in my ability to search by URL pattern.

Additionally, at the top of every file there is Project Gutenberg text. After that text there is sometimes an image, a table of contents, the title, etc. Removing this information became complicated because some titles have "chapters" (*Wuthering Heights*), others have "books" (*Tale of Two Cities*), and still others have "letters" (*Frankenstein*).

Project Gutenberg provides HTML files for most books, however I quickly learned these were not reliable. Many of the books had a great div structure with "chapter" and "section" classes, however many of them did not. Some of them had no markup at all, rather just a jumble of content. It became clear that scraping was going to be a hands-on task.

I found a Gutenberg API[47] which seemed hopeful. I was able to use it to get the stripped text of the novels by ID. Even though the API is capable of providing other information, I was unable to

[46] Hedonometer, https://hedonometer.org/words/labMT-en-v1/
[47] Gutenberg API, https://github.com/c-w/gutenberg/

install a dependency (BSD-DB) due to it conflicting with existing Python packages needed for other projects.

My solution was to use Gutenberg API to retrieve the document text in Jupyter Notebooks. Then I used Python to break up the text into chapter lists, checking behind each book to make sure the array values looked proper. Then I further broke the chapter lists into lists of sentences. I exported these arrays from Jupyter to a local file using Python's json dependency.

Because the process was more labor intensive than initially expected, I pulled the data on an as needed basis. This allowed me to experiment fully with one or two books while building my visualizations, without the overhead of a ton of data. I didn't want to have to rerun scripts again in case they didn't work with later code. Further in the development process, once other code was in place, I quickly ran the script with an array of ids, pulling 26 titles from the top 100 list.

Evaluating the Corpus

With data in hand, I was ready to start using my models. My initial intention was to use 100 Project Gutenberg books. However, as previously mentioned, I ended up only using 26.

The first API I used was MeaningCloud. The configuration was fairly simple to set up in Python. I created environment variables for the API call and credentials. The documentation is brief, but not much was needed to access the existing "Emotion" model. For each item in my chapter arrays for each book I ran the following code and received a response:

```
for i in range(len(frankenstein)):
    payload={
        'key': license_key,
        'txt': frankenstein[i],
        'lang': 'IAB_2.0_en',  # like IAB_2.0_en
        'model': 'Emotion',
    }

    response = requests.post(url, data=payload)

    print('Status code:', response.status_code, i)
    print(response.json())
```

```
Status code: 200 0
{'status': {'code': '0', 'msg': 'OK', 'credits': '10', 'remaining_credi
ts': '18782'}, 'category_list': [{'code': 'Joy', 'label': 'Joy', 'abs_r
elevance': '20', 'relevance': '100'}, {'code': 'Sadness', 'label': 'Sad
ness', 'abs_relevance': '11', 'relevance': '55'}, {'code': 'Trust', 'la
bel': 'Trust', 'abs_relevance': '11', 'relevance': '55'}, {'code': 'Ant
icipation', 'label': 'Anticipation', 'abs_relevance': '6', 'relevance':
'30'}, {'code': 'Surprise', 'label': 'Surprise', 'abs_relevance': '5',
'relevance': '25'}, {'code': 'Disgust', 'label': 'Disgust', 'abs_releva
nce': '3', 'relevance': '15'}, {'code': 'Fear', 'label': 'Fear', 'abs_r
elevance': '3', 'relevance': '15'}, {'code': 'Anger', 'label': 'Anger',
'abs_relevance': '2', 'relevance': '10'}]}
```

The response required more conversion as it came through with unnecessary fields about status and credits which weren't applicable to the visualizations. Also, the results needed to be condensed. I created a custom Python script pieced together to clean the files and store the results in a json file by book name.

MeaningCloud was the slowest of all the APIs used. Additionally, it often failed to return results. Where time allowed, I went, line by line, through the text in order to find the breaking

sentence. There was no apparent reason for the breakage, and my best guess is that it usually failed on short sentences with strong emotions, for example, “How dare you!”.

Next, I ran the arrays through ParallelDots. My hope was to be able to compare the two APIs since they were similar in their approach and output. ParallelDots has it’s own Python library. This library was quite easy to use. After importing the namespace and adding my API key, I passed the data in a pd.emotion() function. This returned the list of emotions in a clean format. Unfortunately, the API call was quite expensive. I was only able to run a few chapters of a book each day before running out of credits. I tried to trick the API by tokenizing the chapters before sending them, however this returned an error. Because of the limitations on the business end of this API, I was only able to run six titles in the two months I used the service.

IBM Tone Analyzer was much more generous with their trial account. IBM has a Python library which I installed, then imported the “ToneAnalyzerV3” package. The library required authentication (stored in my environment variables). In the configuration, I set which model and service I was using. Then I ran a tone() function with my content in order to get chapter level emotions and document level emotions. Once I knew the json structure I would need, I wrote a Python script to automatically format the results and place them into an appropriate folder as part of the call.

I wanted to experiment with as many APIs as possible. I also experimented with the above mentioned APIs which returned results not used on the accompanying web application. I chose not to use Text2Emotion because of its simplicity in nature. I chose not to use Limbic because the negation was not fully flushed out. I chose not to visualize the results from IEMOCAP since the code was initially created for input other than text.

After trying all the existing APIs that I could find, and doing research on how a model is created, I wanted to try creating my own Emotion Recognition service. I started out attempting to use a Bag of Words (BOW) model to achieve keyword matching. Because of the quantity of sentences, the resulting vector matrices were not meaningful in a way I could visualize.

Next, I chose to see what I could do with the hedonometer list. I used the chapter arrays, tokenized the words using NLTK, removed the stop words, and set the results to lower case. Then I compared this new list of words to the hedonometer, keeping only the words that had a hedonometer rating of greater than seven or less than three. The hedonometer is a huge list of words. A large majority of the words fall into the 4 to 7 range. These words, such as “kitchen”, “boat”, and “walking” have little importance to understanding the sentiment of a story. Therefore, I eliminated the middle range, keeping only the lowest and highest values (about

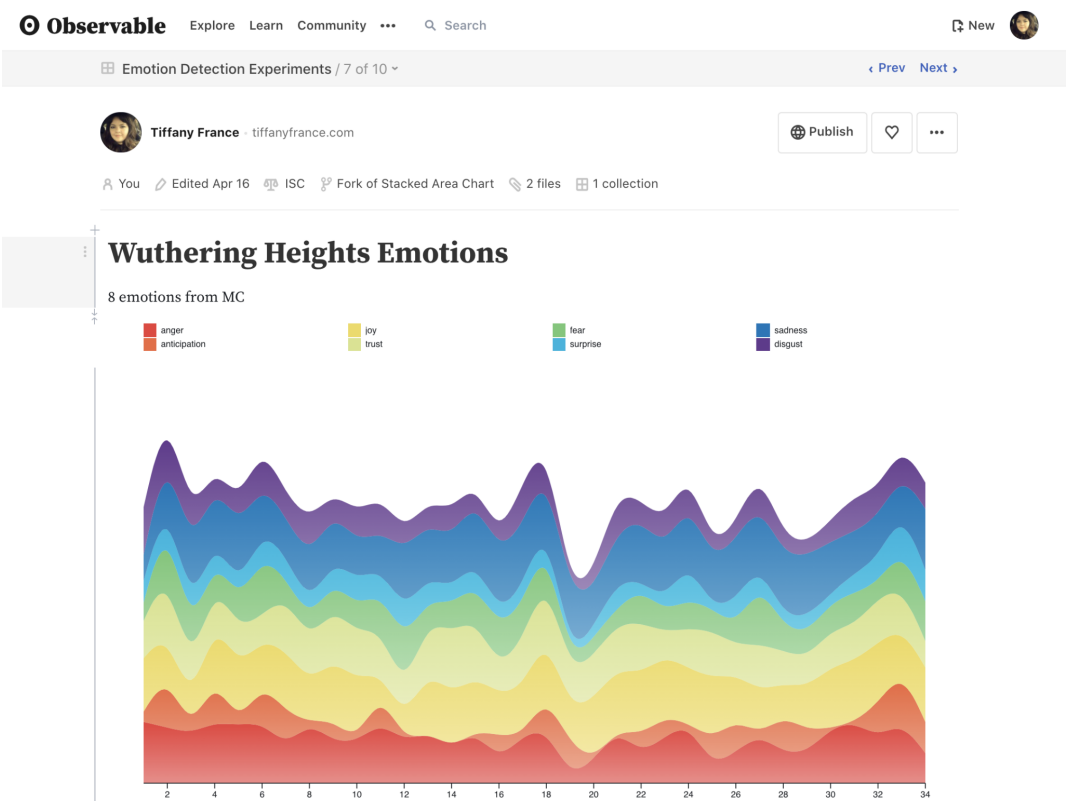
10% of the words) in order to get a feel for the negative and positive landscape by chapter. While “greater than seven” and “less than three” seem like magic numbers, there was a constant exponential increase in every novel surveyed when words scored in the four or six range were included. Because of this pattern, I determined the threshold as a split range throughout all pieces.

Visualizing the Data

Data in hand, the next step was to try various visualization layouts. My goal was to iterate on a visualization with the hopes of best displaying the meaning of the output. I knew I wanted to use nonstandard charts. But I also wanted to ensure the integrity of the data and make clear the message in the information.

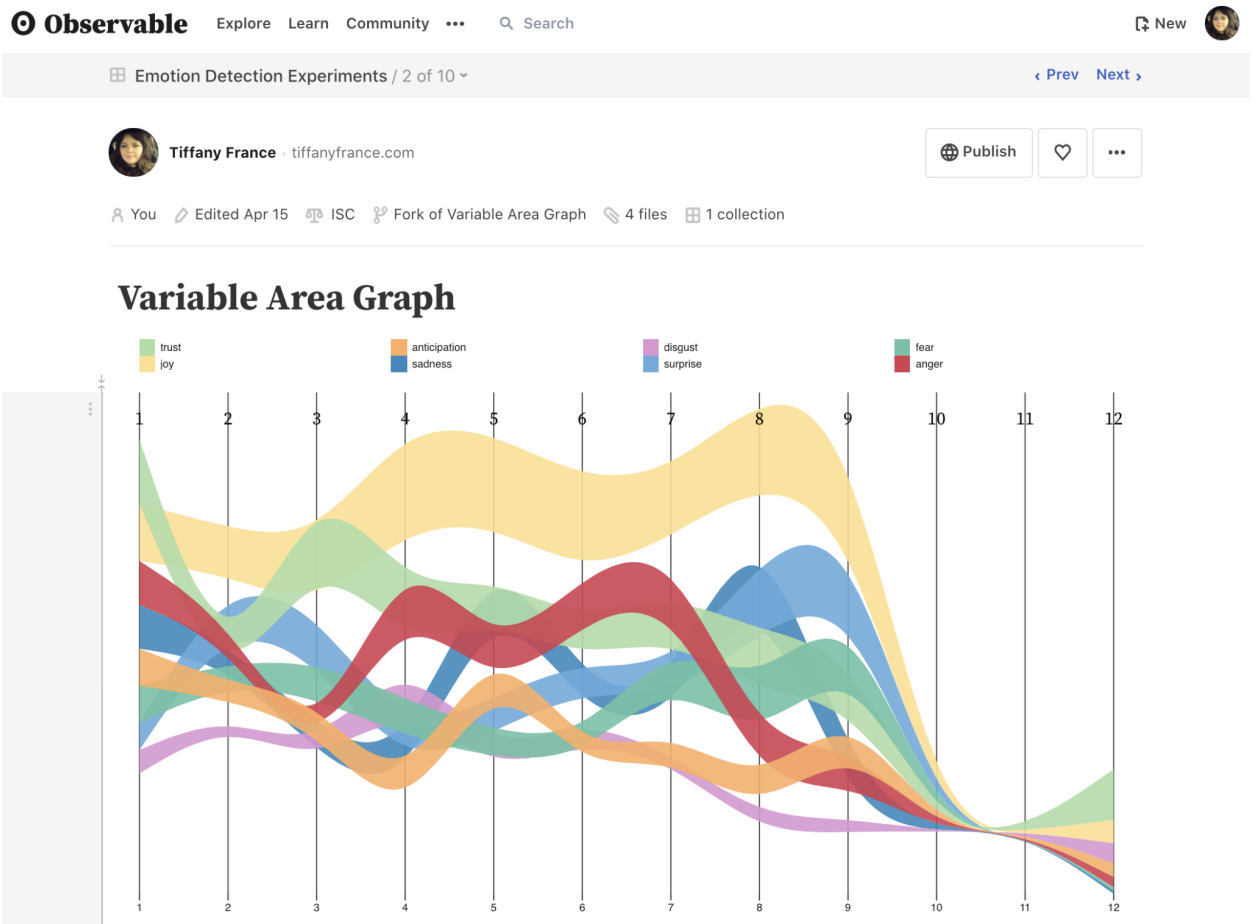
Visualization #1: Variable Line Chart

Using the MeaningCloud results, I started by visualizing the data using simple Observable charts. First, I wanted to visualize the breakdown of emotions by chapter, understanding the intensity of each emotion. I created a stacked area graph, forked from Mike Bostock's Stacked Area Chart [link]. At the time, I only had testing data from one book, *Wuthering Heights*. I didn’t see any interesting insights in this layout, however, creating it allowed me to solidify a color scheme based off Plutchik’s Emotion Model.

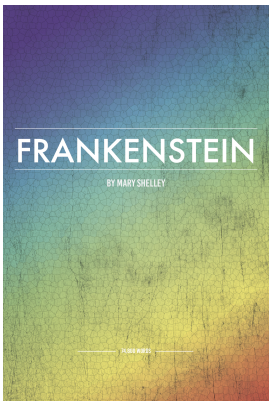


Because of the relative boringness of this visualization, I started to experiment on how to bring meaning forward in the chart. I converted this visualization to a streamgraph, essentially moving the x axis from the baseline to the center. The result showed more variation in each of the sections, however it still didn't tell a story. I kept experimenting with the design, trying to identify what was missing. In this case, I was missing the insight - I was failing to answer the question "what information matters most from this data?"

After a week of sketching and researching, I realized what told the story: (1) emotion prominence as compared with other emotions in the chapter and (2) the intensity of emotions. With those two items as my goal, I started envisioning a new way to show the data. In the end, I decided to use my own take on a variable line chart. The variable lines showed ranking, pushing the strongest emotion to the top of the chart and the weakest emotion to the bottom. The graph also showed intensity by stretching the line at each chapter to show the weight of the scored result. Additionally, the interpolation between each chapter helps the eye to see where emotions change position or size, i.e. where one emotion decreases and another increases.



Visualization #2: Gradient



In the beginning of the prototyping phase, I mocked up a cover idea using a gradient. While most designs deviate from the original, in this case, the final output of the gradient visualization looks remarkably similar to the original concept.

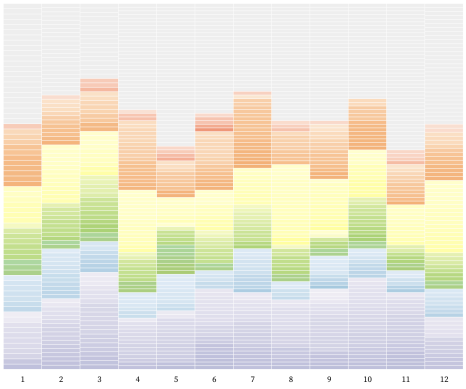
The goal was to see the emotional palette of an entire novel portrayed using color on the cover, so that with a single glance you could anticipate the "feeling" landscape of the plot contained within. IBM ToneAnalyzer provided document level analysis perfect for this type of visualization.

With the document level classification scores, I averaged the numbers to equal 100%. Then I created stop points for each emotion, ordering the emotions in a typical rainbow order to simulate nature. I used these numbers to create a linear-gradient in css, which I visualized using the browser on the web application.

Next, I created a custom Adobe script for automating the gradient in Illustrator. The script took the data values and imported them using the percentages as "ramp points" which created a swatch with the proper rainbow gradient in Illustrator.

Visualization #3: Stacked Bars

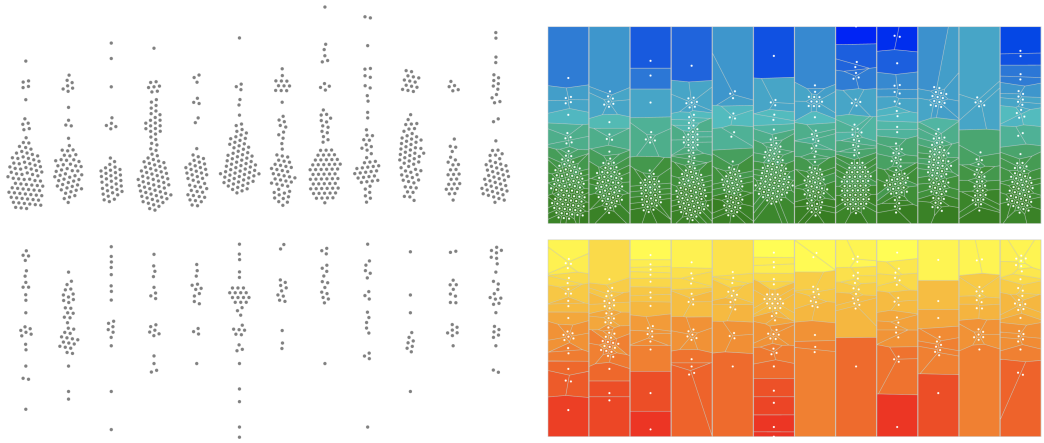
Using sentence level scores from IBM Watson, I wanted to show the percentage of sentences for each emotion by chapter. The original chart also shows sentences which had no emotion in a gray shade. The purpose of stacks of blocks was to understand that this was incremental units which made up the whole. The colors of the blocks also vary in opacity based on the intensity of that emotion for each block, for instance, the darker the purple the higher the value for the emotion "analytical" for that sentence.



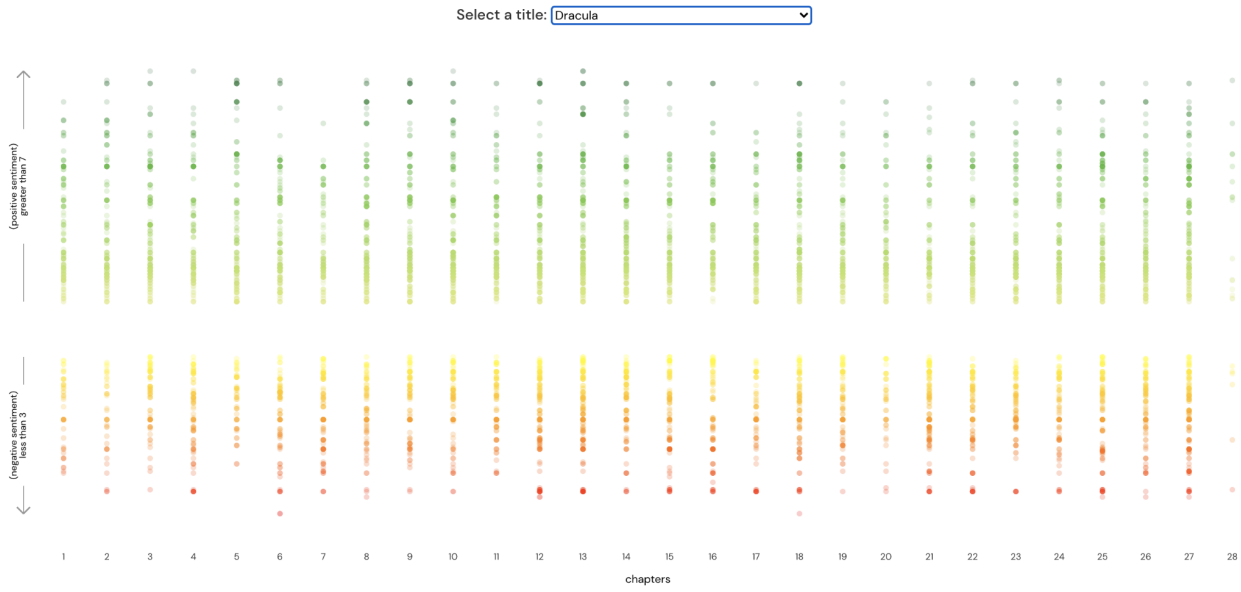
Visualization #4: Hedonometer

The hedonometer returned values that were somewhat sentiment based and somewhat emotion based. Because of the keyword nature, I wanted to make the keywords discoverable, so that through a user interaction (hovering), the visualization could be explored.

First, I visualized these values as a beeswarm. But the clustering patterns seemed to be telling a story that was not accurate. Then I tried adding a voronoi behind the clusters to visualize the space of emotion, but that was highlighting the space rather than the actual data point. Where there was a single data point in the upper region, the entire section turned blue and indicated a high presence of the emotion, when it was actually a low single presence. Though visually intriguing, this was definitely not an accurate representation.



From here I reverted back to a simpler version of the code, before I added the clustering and force mechanics to the data. This gave me solid lines of circles for each chapter. By adding an opacity to the circles, I could see where the values stacked up. Initially, I wanted to put something in the gap between 3 and 7 on the chart, but instead chose to label the y axis with arrows to indicate the portion of data intentionally kept out.



Implementing Visualizations as Book Covers

The next step in the process was to design the book covers using the visualized results. To bring in the visualizations, I inspected the source in Chrome's development tools and copied the entire svg element. Then I could simply paste it into Illustrator. In Illustrator I was able to make color adjustments and remove unwanted components. Once I had the visualizations looking good, I exported them into InDesign. I had to use Illustrator as a middleman in this process because InDesign does not work nicely with pasted svg code.

The publisher had specific dimensions for the margins, gutter, and bleed of book covers. Once these were set up in InDesign, I added the visualization and adjusted the position. Typography was an essential component to the success of the book covers. I matched the typeface to the genre: the geometric firmness of Futura for The War of the Worlds, the flourishing serif of Playfair Display for Pride & Prejudice, and the literary small cap version of Mrs. Eaves to match the whimsy of Alice in Wonderland.

The novel copy was laid out in a different file using appropriate margins and bleeds for book interiors. I created several master templates in InDesign to add formatting and page numbers. I referenced several paperbacks from my bookshelf in order to understand how title pages, paragraphs, and chapter headers are formatted. I used Baskerville font on all the books.

Much work had to be done to the ProjectGutenberg files in order to prepare them for print. Punctuation was corrected. Line breaks and paragraph spacing was removed. Quotes had to be modified to match typographic standards.

I created specific paragraph styles for chapter headers, first line of chapters, all paragraphs, and captioned text. Then I custom designed each table of contents and title pages per book. I added notation about the project and the applicable visualization within the book with annotations to help the user understand the graphs. Finally, I addressed orphans and widows within the book, typesetting the contents. Each book took several days to design and arrange.

Design Choices

In this section, I want to discuss the design choices made for the project starting with color. Color was tricky in this process because it can convey much meaning. In the end, I chose to use Robert Plutchik's Emotion Wheel for my color for two reasons: other researchers in the field use it, and it was a solidly accepted model for defining emotions. I am aware that the rainbow color scheme also carries LGBTQ symbolism. For that reason, I tried to craft my covers carefully as to not mislead a potential reader about the contents within.

The rainbow palette uses colors high in saturation and contrast. These are two important qualities for displaying data. The colors work nicely as a group, as well, but I noticed they work best when kept in rainbow order. For instance, on the gradient visualization, reordering the emotions to change the position of colors interferes with the pleasantness of the scheme.

For the website, I wanted to mimic a slideshow. This approach would give me the maximum amount of space to spread out my visualizations. This had two benefits. The first is that many of the visualizations have a lot of data. Maxing out the horizontal space allowed the data to be fully expressed within a larger area. The second benefit is that it provided simplicity, so that there was only one focusable item on the screen at a time. This ensured that the user would have a better chance of being immersed in the content, and avoid distraction with multiple messages. Also, it adds a minimalist aesthetic to the project which I enjoy.

For the most part I avoided using legends, though I did include one on the comparison of MeaningCloud and ParallelDots API results. I chose to include a legend there because the differences of emotion labels are nuanced, and could be easily missed. The other labels are included in the descriptive text as a color coded array. The text is typeset in a code-based font to remind users that this is inherently a technical project.

Finally, I included an interactive portion because I wanted the user to engage with the content. As I was going through the research, I kept having a nagging feeling that the results were not accurate. Therefore, the “Try it out!” section is meant to address those concerns, both for myself and for the user. I was surprised to find that the IBM ToneAnalyzer did a very good job detecting appropriate emotions on the sentence level

Building a Vue Application

Because of the design nature of my project, I decided to keep each “slide” as a separate component. This made it easier to rearrange the slides in App.vue, adding and removing as needed. Within the components folder, I also included charts to be used throughout the application. This allowed me to use the VariableArea.vue chart in multiple different components.

The trickiest part of setting up the application was updating the data via a select menu. For this I added a v-model to the select menu, then passed the filename for each book as a value in the options. The component watched for a change in the file path, then reran a draw function each time it detected a change. Because I had already written the code in Observable, I simply modified it to work within the Vue components as the draw function.

Implementing RWD

I wanted to make sure that the project was accessible to mobile users. Therefore, I spent some time at the end of the project working on responsive web design.

Before drawing each graph, the JavaScript code gets the width of the page and sets it as the width of the visualization. This ensures, on page load, that the visualization will not exceed the given space.

CSS considerations were made using media queries. The contents of the headers and paragraphs were set to resize to 100%. Floats collapsed down to one column. Padding was adjusted for maximum viewing experience, whether on a small or large screen.

Working with the Publisher

In order to receive printed books, I worked with Barnes and Nobles publishing. First, I had to verify the right to print public domain work. I researched multiple articles which stated public domain books could be reprinted, but I wanted to be sure that I wasn't infringing on any copyrights. I worked with Barnes and Nobles publishing and Amazon, which both have a special process specifically for reprints of works within the public domain.

Barnes and Noble interface is easy to use, however there is a good deal of waiting between each step. I uploaded my cover pdf and interior pdf according to their specifications. The files are examined through an automatic process, then sent for manual review. The manual review took about one week for completion. Once the review was complete, I could order a copy. With rushed shipping, the book took three weeks to arrive. Once it arrived I noted any errors, fixed my document, reuploaded the files, then resubmitted the package. I had to wait again for the manual review. After that was approved, I was assigned an ISBN and the book was set to "preorder" status. When you put a book on preorder you cannot purchase a copy to review for a week. So after a week I ordered another copy, but as of writing this, the second iteration has yet to arrive. Four books are available for preorder on Barnes and Nobles.

I also submitted Frankenstein to Amazon's KDP. This interface was much smoother however more restrictive. The system allowed me to use Barnes and Noble ISBN numbers. Amazon would not approve any book with text in the gutter or a certain number of blank pages. I asked for an override to this restriction, but was informed that there is no way to override the process.

Conclusion

The field of Emotion Recognition is still quite young, lacking in formal systems or solid examples. IBM Watson stands out above the other solutions as the best for interpreting a literary corpus. ParallelDots and IBM both had easy to use Python packages. One major takeaway is the value of documentation when sharing open source code. Repositories containing clear information on the process and implementation were highly informative and enjoyable to use.

Can machines classify emotions in literature? With a good model and clean data, it appears there is some success. Can this data be converted into book art? Absolutely. The output was heavily designed, though a more generative approach could be taken using Processing or p5.js.

In the digital space, one can go their entire career without having a single physical manifestation of their work. These books were a method to permanently capture a visualization in a consumable way. Because the analyzed text is included, people can use the data visualization in real time as they are reading the novel, providing a highly practical implementation of analytic work.

References

6 Seconds. Plutchik's Wheel of Emotion. 6 Seconds. https://www.6seconds.org/2020/08/11/plutchik-wheel-emotions/.

Alswaidan, Nourah. A Survey of State-of-Art Approaches for Emotion Recognition in Text. 2020.

Batbaatar, Erdenebileg. Semantic-Emotion Neural Network for Emotion Recognition from Text. 2019.

Cambria, Erik. Affective Computing and Sentiment Analysis. 2016.

Erenel, Zafer. A New Feature Selection Scheme for Emotion Recognition from Text. 2020.

Ghazi, Diman. Prior and contextual emotion of words in sentential context. 2013.

Gu, Simeng. A Model for Basic Emotions Using Observations of Behavior in Drosphila. 2019. https://doi.org/10.3389/fpsyg.2019.00781.

Johansen, Jørgen Dines. "Feelings in Literature." Integrative Psychological and Behavioral Science. Springer-Verlag. February 17, 2010. https://link.springer.com/article/10.1007/s12124-009-9112-0.

labMT Hedonometer. http://hedonometer.org/words/labMT-en-v1/.

Open Culture. Why the University of Chicago Rejected Kurt Vonnegut's Master's Thesis. 2019. https://www.openculture.com/2019/12/why-the-university-of-chicago-rejected-kurt-vonneguts-masters-thesis.html.

Project Gutenberg. Frequently Viewed or Downloaded. Project Gutenberg. https://www.gutenberg.org/browse/scores/top.

Project Gutenberg. The Project Gutenberg License. Project Gutenberg. https://www.gutenberg.org/policy/license.html.

Reagan, Andrew. The Emotional Arcs of Stories are Dominated by Six Basic Shapes. 2016.

Seol YS, Kim DJ. Emotion Recognition from Text Using Knowledge-based ANN. 2008.

Shaheen, Shadi. Emotion Recognition from Text Based on Automatically Generated Rules. 2014.

Shivhare, Shiv Naresh. Emotion Detection from Text. 2019.

Vonnegut, Kurt. The Shape of Stories. https://www.youtube.com/watch?v=oP3c1h8v2ZQ&ab_channel=DavidComberg.

Yampbell, C. (2005). Judging a Book by Its Cover: Publishing Trends in Young Adult Literature. The Lion and the Unicorn. 29(3). 348–372. doi:10.1353/uni.2005.0049.