Aalborg University Copenhagen

Semester: 10th

Title: Exploring emotion detection in textual data, using knowledge based methods and web scraping, and communication of the results through affective visualizations created by generative algorithm

Project Period: 01/02 /19– 28/05/19

Semester Theme: Master's Thesis

Supervisor(s):

George Palamas

Project group no.:

Members:

Anna Lunterova
Ondrej Spetko

Aalborg University Copenhagen

Frederikskaj 12,

DK-2450 Copenhagen SV

Semester Coordinator:

Secretary:

Abstract:

The following exploratory research investigate approaches to natural language processing, and propose alternative way to approach visualization and communication of multivariate data. The aim is to 1. improve the traditional recommender systems by adding emotional category to the data organization, and 2. to communicate the subsequent metadata further through an affective visualization. Respectively, the design process is divided into two phases of natural language processing (NLP) phase and visualization phase.

The NLP phase consists of extracting experimental features of arousal, entropy and color, in addition to the more traditional features of valence and subjectivity from the IMDB reviews dataset. Afterwards the machine learning algorithm, the t-Distributed Stochastic Neighbor Embedding (tSNE), is used for reduction of the feature set dimensionality, resulting in a spatial organization representing the textual data of IMDB reviews. The extracted features are observed and judged by basic statistics, value distributions and visualizations. The visualizations are further supported by clustering analysis for purely experimental purposes.

The visualization phase consists of exploring the affective capabilities of computationally generated visuals, with an intention to communicate data qualities through an affected state. From the NLP phases metadata, specific data qualities are chosen and then mapped into preattentive elements of the final affective visualizations. Each visual represents the data qualities of reviews of one movie. Together 20 movies signatures are visualized. Afterwards, from the perspective of communicativeness, the created data-informed generative art visuals are evaluated additionally with the effectiveness of the generative algorithm.

Although further investigation are needed, results from the NLP phase of feature extraction support the significance of the influence of the experimental features on the organization of the textual data. The evaluation of the visuals through a quantitative online survey done by 79 participants, confirms the possibility to communicate data characteristics through an affective visualization.
We believe the immersion of the two phases brings new perspective into understanding and communicating textual data.

Exploring emotion detection in textual data, using knowledge based methods and web scraping, and communication of the results through affective visualizations created by generative algorithm.

Anna Lunterova[1], Ondrej Spetko[1], and George Palamas[1]

Aalborg University, A. C. Meyers Vaenge 15, 2450 Copenhagen, Denmark

Abstract. The following exploratory research investigate approaches to natural language processing, and propose alternative way to approach visualization and communication of multivariate data. The aim is to 1. improve the traditional recommender systems by adding emotional category to the data organization, and 2. to communicate the subsequent metadata further through an affective visualization. Respectively, the design process is divided into two phases of natural language processing (NLP) phase and visualization phase.

The NLP phase consists of extracting experimental features of arousal, entropy and color, in addition to the more traditional features of valence and subjectivity from the IMDB reviews dataset. Afterwards the machine learning algorithm, the t-Distributed Stochastic Neighbor Embedding (t-SNE), is used for reduction of the feature set dimensionality, resulting in a spatial organization representing the textual data of IMDB reviews. The extracted features are observed and judged by basic statistics, value distributions and visualizations. The visualizations are further supported by clustering analysis for purely experimental purposes.

The visualization phase consists of exploring the affective capabilities of computationally generated visuals, with an intention to communicate data qualities through an affected state. From the NLP phases metadata, specific data qualities are chosen and then mapped into preattentive elements of the final affective visualizations. Each visual represents the data qualities of reviews of one movie. Together 20 movies signatures are visualized. Afterwards, from the perspective of communicativeness, the created data-informed generative art visuals are evaluated additionally with the effectiveness of the generative algorithm.

Although further investigation are needed, results from the NLP phase of feature extraction support the significance of the influence of the experimental features on the organization of the textual data. The evaluation of the visuals through a quantitative online survey done by 79 participants, confirms the possibility to communicate data characteristics through an affective visualization. We believe the immersion of the two phases brings new perspective into understanding and communicating textual data.

Keywords: Natural Language Processing · Sentiment Analysis · Generative Art · Affective Visualization · Emotional-context aware system ·

Dimensionality reduction · Affective Intent· Affective Computing · Data-Driven Visual · Visualizing Metadata Qualities · Recommender systems · Emotion detection

1 Introduction

With the fast development of social media and ever increasing possibilities to create short content through various networking services or recommendation systems, more and more textual data are generated by users. Whether it is users stories, posts, comments from social media, or reviews of publications, products or services. This text data is stored across multiple platforms in vast quantities. Users turned from being primarily passive recipients of information on the web, and became active contributors to the amount of data stored. This phenomenal accumulation of textual content has attracted many researchers and necessitates further automated analysis, extraction, recognition or adaptations of the system. Processing of the text, Natural Language Processing, allows e.g, automatic translations [28], information retrieval [26, 2], document classification[65], question answering[73], entity recognition[77], sentiment analysis[53], etc. Particularly famous use of users behavioural data nowadays, recommender systems seek to predict users preference when browsing through various contents. Recommender systems are used for variety of areas, including e-commerces (Amazon, Netflix) music or videos recommendations (such as Youtube), search queries, social tags, online dating etc., providing a relief to the users information overload problem. The algorithm is based on analyzing previously searched items or similar users characteristics. As a more specific example, Social Tagging Recommender Systems categorize and retrieve content using open-ended tags. Tagging data are user-provided[20], and it is today a popular method to organize and retrieve items of interest in the Social Web. Tags are used as either main or additional source of content for the recommender system, but tag-augmented recommender are only slightly more accurate than the recommendations of the pure content-based one[13]. Additionally, it has been shown that the average precision of user tags is about 0.5 and the average recall (completeness rate) of the user tags is 0.5 as well. This means half of the tags created by users are noise and half of the true labels are missing[11].

Beginning of emotion recognition and analysis has sparked a lot of researches in computer sciences and human computer interaction in the past decade. The idea and importance of fusing emotions and affect into technology, was presented for the first time in 1997, by Rosalind Piccard[58]. The emotional communication enriches and facilitate a deeper understanding of the content through the expressiveness of verbal (spoken words, prosody) cues. Automatic extraction of emotions from the text can provide additional information about the content that has been in recent years used for various purposes such as suicide prevention, intelligent tutoring systems, online communication to build smart robots, product review, emotions application development, social studies, etc[30, 14, 52, 53].

Processing and analyzing content, based additionally on the emotional category, adds an important attribute for the traditional recommender systems. Organizing and retrieving data points with additional source of content introduces a novel way to enhance the understanding of the textual data content. We believe that creating emotion based recommender systems has potential to both: -address the issue of imprecise sorting based primarily on tags, users or date, by suggesting alternative categorization of items, -bring deeper understanding of the content to the system and then potentially to the user.

Textual data processing and analysis form a new subset of the same data, called metadata. This metadata gives an opportunity to explore and communicate the newly found characteristics of the original data. A graphical representation of data and their attributes, or data-driven design, serves as a method of displaying sorted data items, and relationship between them, to not only facilitate their understanding[75, 44], but also brings focus on the creative and aesthetic potential of the data uses[34]. Emphasizing engagement and interest with the data, the visual appearance of data in form of data visualization, or data art, forms data-driven visualizations that communicate not only information but also and affect[79]. These problem areas and fields of interests lead to the following research motivation:

How can we extract and communicate emotional signature of textual data to potentially enhance the traditional recommender systems?

This research is divided into two distinct phases, first part focused on the analysis and categorization of textual data for the purpose of extraction of emotion based features from textual data. The second part will take the opportunity to investigate the ways to communicate and visualize the subsequent metadata obtained in first phase from the original textual data.

2 Background research

2.1 Recommender systems

Generally recommender systems are divided into two categories: content based and collaborative-filtering based [61].

Content based systems are recommending products or elements based on features that defines them. For example in case of music the content based system would recommend songs and other music products based on the sound properties of the songs.

Collaborative-filtering based recommender systems are suggesting content based on the history of the users choices and comparing them with similar users and their choices. Collaborative-filtering recommender systems are not interested in the features defining the content items themselves but rather build on statistical probability of similar users liking the similar products. It is not a rule that a recommender system should fall exclusively under just one of these categories.

In regards to this project the interest incline towards the content based approach of recommender systems as the aim is ti extract features representing content of a textual data.

2.2 Natural language processing (NLP)

Natural language processing is relatively novel concept in linguistic science and allows processing and analyzing large amounts of natural language data [39]. NLP field is very famous in the word of data science nowadays, and there is access to various tools and libraries for this purpose. Among the most famous Python libraries belong CoreNLP [40] from Stanford University, SpaCy [24], NLTK [70] or TextBlob [37]. Some of them differ in implementation, performance and capabilities. However, when it comes to the baseline functions like part-of-sentence tagging (POS) [1], phrase isolation or basic sentiment analysis, they usually produce almost the same results.

For example TextBlob is a python library that offers a simple functionality to access its methods and perform basic NLP tasks. Including property of sentiment analysis returning both sentiment and subjectivity values of the processed textual data. Sentiment analysis can be used to add informative value to the emotional signature of the processed textual data. On the other hand CoreNLP is more robust library that provides stand alone support for developers. CoreNLP is developed in Java and has more complex functionalities like dependency structuring of the part-of-sentence tokens and many more. The implementation works through API from the developers environment. Natural language processing usually follows three basic steps: preprocessing of data, feature extraction and inspection/evaluation. These three steps are usually done in repetitive manner by means of exploration before jumping to any conclusions.

In the preprocessing phase the researchers usually try to eliminate most of the noise from the data and, generally speaking, prepare the data for the phase of feature extraction. The phase of preprocessing contains operations like tokenization (represent text as group of tokens=words), removal of stopwords (over common words e.g. the, a, so etc.), lemmatization (transforming words into their basic form e.g. flew to flying), special character elimination (also removal of punctuation and digits), lower-case transformation (WORD to word), removal of white spaces and other. In the preprocessing phase the choice of operations applied to the textual data often depend on the purpose of the research. For example in case of emotion-based feature extraction, the punctuation and generally text format, is considered to be a big clue that can indicate some emotion attributes. For example - WHY? - may indicate negative sentiment thanks to the capital letters used as opposed to its lowercase version - why?. Many NLP libraries already have these preprocessing operations included and therefore making the whole process of NLP easier for those using them [40]. In the feature extraction phase the main NLP methods (e.g. topic modelling, sentiment analysis etc.) are applied on the processed text from the previous stage of preprocessing. This phase is responsible for extraction of features that would serve the purpose of the research or experiment. In the last phase the distribution and significance of the features and results is inspected, evaluated and reconsidered. In case of potential improvements the three steps can be repeated in any order needed to ensure improved feature extraction and generally better or more significant results.

2.3 Word to vector

For the purpose of representing a textual data in a computational friendly way, textual data needs to be represented as numbers. Word2Vec is a class of machine learning models that is responsible for representing words of a text in embedded space (vectors) according to their relative meaning derived from co-occurrence in text [42]. These word embeddings are highly useful for Natural Language Processing tasks. Recently, the technique has also been applied to more general machine learning problems including product recommendations. Representing words in vector space is more and more commonly used for locating similar words that share common contexts in the analyzed content, mostly for the purposes of deep learning, semantic analysis, sentiment analysis, classification, recommender systems, etc. Word2vec algorithm takes a textual data, and map each of the words on one vector space. This space is composed likely of several hundred dimensions (size depends on the amount of processed text), with each word being assigned a corresponding vector in the space. High-quality distributed vector representations of words can grasp quite precise syntactic and semantic word relationships. Word embeddings are further usable for topic modeling known as Doc2Vec model.

2.4 GloVe

GloVe is a new global log-bilinear regression model for the unsupervised learning of word representations performs comparable to the word2vec and other models on word analogy, word similarity, and named entity recognition tasks [56]. It is developed as an open-source project at Stanford, with a large downloadable database of pre-trained vectors. Advantage of using the GloVe is the big pre-trained map of word vectors that was trained on immense amount of textual data providing general mapping of words without having to train your own model of word2vec.

2.5 Sentiment analysis and arousal

In NLP for emotion detection it is common to follow Paul Ekmans model of six basic emotions which are Anger, Disgust, Fear, Happiness, Sadness and Surprise [47, 15]. These main groups of emotions contain subcategories of emotions forming emotion ontology [68, 71].

Emotion ontology is important part of the emotion detection because it maps the relationship of emotions and enables for main emotion assumption based on minor emotions that are present. Emotions have also other representation. This representation is located in 3D space of valence arousal and dominance (VAD) where valence refers to sentiment (positive, negative) of the text, arousal refers to activation (activation, deactivation) and dominance. Domain of dominance does not carry significant value as it highly correlates with valence (0.97) [82]. The resulting space of valence and arousal is called Circumplex model of emotions and allows for emotion mapping based on these two domains.

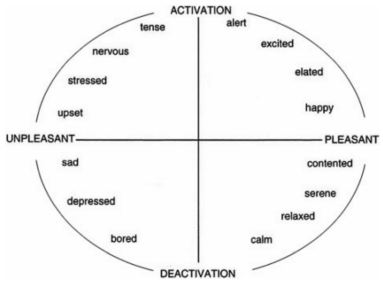


Fig. 1. Circumplex of emotions

In NLP another dimension is often present aside from the circumplex model. This dimension is subjectivity and provides information about the personal stance of the text (subjective, objective). Together sentiment (valence) and subjectivity form basic feature for most of the NLP implementations. The arousal domain is only an experimental feature that is usually deducted from the arousal lexicons created by mechanic turks (people labeling data).

With an aim to propose alternative ranking method for facebook fan page, in the research *The Lexicon-based Sentiment Analysis for Fan Page Ranking in Facebook* [53] the focus was on crawling textual data from users posts and comments. For analyzing the user engagement and comment polarity, lexicon AFFIN was used to add users opinion into the analysis. Including users opinion in the account when evaluating a fan page was found to be more accurate, compared to evaluation based only on number of comments, posts, and likes. Although the aim of this research was different and purely for evaluating success rate of fan pages on Facebook, the insights were regarding the approach towards the sentiment analysis and web crawling.

2.6 Emotion detection

There is quite bit of research and work done in the field of emotion detection through text. In general the approaches to detect emotion from the text can be summed into 3 main categories [67]:

- Knowledge (or Keyword) based approach
- Learning based approach
- Hybrid approach

Knowledge based approach focuses on assigning words with emotional affect levels (anger:60%, happiness:40% etc.) or directly assigning the words with the main emotion category (anger) based on existing lexicons like WordNet-Affect [71] or NRC-VAD [46].

In data science the lexicons and datasets are the core component of learning for models and predictions. In the field of NLP lexicons play major role. Main

reason being the lack of emotional information encoded in the textual data itself. For this purpose some NLP lexicons provide at least partial answer to this problem with their emotional rating of words. Among the top used emotion lexicons are WordNet and NRC.

In the research *Can word embeddings help to find latent emotions in text?* [66] it has been shown that standart word embeddings algorithms as word2vec and GloVe do not predict main emotion from the component of emotions, resulting in an imprecise emotional distribution of words. Lexicons serve as knowledge based approach for sentiment analysis and classification technique, and improve the accuracy of embeddings in representing emotions.

The NRC Valence, Arousal, and Dominance (VAD) Lexicon (2018) includes list of more than 20,000 English words and their valence, arousal, and dominance scores. Other lexicons (such as Affective Norms for English Words (ANEW) lexicon (1999), the WordNet Affect (2004), the SentiWordNet 3.0 (2010)) exists but having lower number of words and some are lacking dimensions of valence, arousal or dominance, they have been measured as having lower reliability.

WordNet lexicon is emotion ontology lexicon and it is specifically designed around the ontology of emotions and maps affective words to this hierarchy where a word pissed leads to main category anger thanks to the hierarchical structure of the emotion links in the lexicon. WordNet-Affect lexicon is extension of the original lexicon by means of adding value of valence for each of the affective words in the lexicon. This lexicon is, however, limited to only affective words in the first place. NRC-VAD lexicon has different approach. First of all the lexicon work with common words as well as affective words which is advantage as opposed to WordNet lexicon and it has much more words listed. The lexicon is build by people (mechanical turks [12]) rating common words based on their affective value in dimensions of VAD (valence, arousal, dominance). NRC-VAD lexicon can assist the traditional sentiment analysis by providing mapping of arousal dimension of words completing the coordinate system of Russels circumplex of affection model of valence (sentiment) and arousal.

After mapping the words in the text with affective values or basic emotion, the final emotion can be then assumed as the dominant one (ex. anger) or displayed as probability distribution of all main emotions spotted (ex. anger:60%, happiness:40%). The scope of keyword based approach can differ from keyword level through sentence level all the way to the document level evaluation.

This approach lacks context judgement and therefore different meanings (I met her by accident; I had an accident) may produce similar results if not supported by further semantic and syntactic analysis of the sentence instead of selective word-based evaluation only.

Learning based approach uses supervised and unsupervised machine learning algorithms to determine probability distribution of the emotion values in the text and classify the final emotion. Supervised machine learning requires big datasets of samples that are divided into training and testing samples. Disadvantage of this approach lies in the need of labeling data that allows for training and testing

of the model and eventually validation and accuracy measures. Unsupervised learning uses statistical measures to represent semantic relations of words within sentences and their relevance to target emotion.

Hybrid based approach uses combination of the two previous approaches. Hybrid solutions tend to have higher accuracy compared to exclusive use of knowledge or learning approach.

In this project the knowledge approach for emotion detection is used because the lexicons provide faster implementation compared to learning and hybrid models.

2.7 Experimental features from textual data

When dealing with textual data it is often the task to conclude the emotion from the text itself because we dont have a way to account for external data like mindset of the author. Majority of approaches for emotion detection are focusing on the syntactic and semantic relations and modeling of the textual data, supported by lexicons and other labeled datasets that cant reach levels of certainty for the reason of disagreement between the labeling judges.

When reading, people tend to visualize what they read [51]. Depending on the type of text they read or conditions in which they happen to read, the perception may differ but the base idea prevails: reading as well as expressing ourselves in textual format communicate certain form of taste or color that is locked within the words, phrases or the whole text as one. This perception usually goes hand in hand with emotion felt when reading the same text. Visualizing and perceiving certain words may be different for each person but there could be a way to generalize this perception. If we could use internet to provide us general representation of a textual data that would be beyond traditional methods of NLP (e.g. topic modelling, sentiment, subjectivity) in a reasonable data form, we could use this as an experimental feature for potential emotion representation of the text. In other words perception or visualization (imagination) of a textual data could be reflected on the internet using the search engines and the textual data as input. This idea originates from the fact that people label and structure content they create on the internet in a way that the images (and other medias) are enclosed in relevant context linking the textual information closely to the media themselves.

Internet search engines are specifically designed to link textual data from search bars to results of various media types (pictures, videos, music) as well as text. Images compared to videos and music are easier data form to process as they are simply bunch of pixels with color values. The idea would be to represent a textual data in pictures from a search engine and further apply image processing methods to find a way to summarize the findings into a feature representing the textual data. It is generally known that perception of colors is related to the emotions [27, 74]. Therefore, the colors of the images that represent a textual data could provide potential insight into the emotional print of the data that

they represent. Additionally entropy, known as average information of an image, could be used along with the color to enhance the representation value of the textual data as a visual information.

There exist few datasets that provide mapping of common words with colors. In his work Saif M. Mohammed developed a lexicon based on the people labeling common words with colors [45]. This lexicon was later combined with similar dataset of word-emotion labeled lexicon resulting in mapping of word-color-emotion. The dataset is relatively small and limited in terms of number of judges per term to be used for this project.

2.8 Visual representation of data

Analysis and processing of the textual data serves for enhancing deeper or more precise understanding of the textual data, such as its content and emotional value. The result of this process is metadata describing the data in depth, providing space for next directions such as novel organization for recommender systems, answering analysts specific questions regarding values, or communicating the results further to the public. For the case of further communication and representation of the obtained metadata, data visualizations are often used to transform a table with raw data values, into more easily comprehensible and aesthetically pleasing visuals of different forms[6, 18, 44, 48, 50]. From broad perspective, the approaches for visually encoding the information visually varies across research, depending on its purpose, from being purely functional with focus towards the visualization maximum informativeness, to more artistic and exploratory approaches with more attention towards its aesthetics and creative expression. This chapter will describe the investigation of the two approaches.

2.9 Functional apporach

Form follows function is known principle associated with industrial design and modern architecture. Especially among scientists or engineers, the communication of complex and inter-related datasets requires high informativeness and often is the only focus. An example of most commonly used tools are default excel tables, predefined Tableau templates, default settings for plotting tables through Matplotlib library for Python, or online tools for making graphs that pay rather little or no attention to the aesthetics of the graph. Graphics that are designed with high functional purpose, focus on their simplicity and short cognitive interpretation time of the data. Well known data artist A. Cairo in his book, The Functional Art, explains functionality as an accurate depiction of data, that help the viewer to think about the shown information rather than the design of it[5]. Functionality is the foundation for the next four principles of a good DV, that are as follows: truthful, aesthetically pleasing, insightful, and enlightening[6]. Similarly, Tufte (1997) in the book Visual Explanations: Images and Quantities, Evidence [76] makes a distinction between a friendly data graphic, which is one that helps readers to understand the data, and an unfriendly data graphic which will not enable readers to efficiently grasp the

data. He argues that the functionality is found in the form of graphical elegance of simplicity of the visual representation of complex data. He invented term data-ink ratio, what is the ratio between how much ink is used for the graphics, versus how much data is being communicated. Here he argues that the beautifulness of data visualizations equals simplicity. Primary aspect of the visual is the data itself, and the graphics is mainly and often only considered as the medium for effective communication. The functionality of the visualization is measured quantitatively, most commonly such as time needed for interpretation and making sense of data, accuracy of interpretation, its readability, efficient learning rate, etc[30]. An effective data visualization can be altogether characterized as a visual representation of accurately depicted data, that is easy and fast to understand cognitively.

However, although Tuftes work connects functionality with the minimalism of the graphics, the recent growth of applying computing technology to data visualization and the problem of high dimensional datasets open a new discussion in terms of the before perceived necessity for lowest data-ink ratio. In the digital age, large amount of complex data and the simple use of computer technology to create generated visuals, brings further challenges and calls for new possibilities, when approaching data communication[25].

2.10 Aesthetic approach

Although the functionality being the main and unquestionable focus when evaluating data visualization, recently more and more research goes into the effects and importance of aesthetics. The need for aesthetics is gaining attention for promoting positive effect on perception of data, sharpening focus, reducing boredom, for enhancing the experience and engagement with the data, improving problem solving skills and also amplifying the ability to obtain knowledge[7, 33, 60, 16]. Aesthetics are the investigation of the reasons that assess the value of the visual, depending on the sensations or emotion it produces[63]. Based on the theory of cognitive-affective processing system model[43], while the functional approach involves the cognitive processing system that interprets the content, aesthetics of the stimuli engage the affective processes. These affective processes involve persons feelings or emotions towards the visual, and the experience of these felt evaluations serves as perceived information. The functionality of affect can be directly observed in the way people react to certain stimuli in the environment, and indirectly in the way people cognitively process information about the environment[64]. In comparison with the engineering or scientific discipline where the value depends on the functionality and utilization, here the assessment of value is in a form of the viewers emotional response.

What exactly is that emotion that drives human behaviours is an open and rather philosophical question. Throughout the history various dimensional models of emotions (e.g Ekman emotion model, the OrtonyCloreCollins emotional model, Plutchiks model of emotions, Parrots categorical model, etc.[57]) were created with an aim to better understand emotion and the processes behind

them. One particular already mentioned model, the circumplex model of affect[62, 59], proposes that all affective states arise from cognitive interpretations of core neural sensations. Compared to the categorical models where the emotions are rather considered separate and emerging from independent neural systems, here the emotions are placed on a continuum of valence (pleasant vs unpleasant dimension) and arousal (activation or also called intensity) dimension, and considered arising from common neurophysiological systems.

The x axis with valence represents the polarity of emotion, from negative to positive,and the y axis with arousal is the emotions activation, from low to high. This suggests, that when perceiving and evaluating the aesthetics, the affective system judges based on the perception of valence and arousal resulting in a perceived feeling. This preattentive processing is triggered automatically with a speed of 250-500 ms compared to much slower attentive (cognitive) top-down processing[32, 23].

Returning to the previous chapter, the same model is also used within the NRC lexicon of words used for sentiment analysis, where the words are categorized by their value based on their valence, arousal and dominance. The uses of this model from 1980 are spreaded across various fields, and help to understand and define different emotional states as being related rather than separate. When creating or evaluating aesthetics, this can serve for placing the affected state within the model and its corresponding emotion.

2.11 Neuroaeshtetics

Often criticized challenge of the aesthetic approach, is that the assessment of artistic value of the visual is considered as highly subjective and in lack of an adequate way to provide quantitative measurements[8]. However, emotions were proved at various researches to have influence on the performance (functionality, utility and usability) elements of the visual[54, 72], and quite new field of neuroaesthetics bring an empirical approach of understanding why. Merging design and cognition, neuroaesthetics attempt to explain and understand the aesthetic experiences at the neurological level based on the knowledge from neuroscience. The study of brain scans informs that aesthetic experiences are a product of interactions between sensorymotor, emotionvaluation, and meaningknowledge neural systems (figure 2).

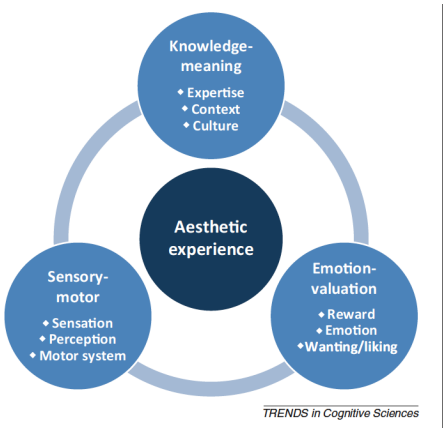


Fig. 2. Three neural systems contributing to aesthetic experience[9]

Although there is still needed further study of the significance of the individual differences in neural structures, there are common pathways activations and aesthetic preferences across individuals[9]. That means taking into consideration found characteristics for perceiving a visual, can help to understand the creation of affected state. Aesthetic processing, defined as appraisal of the positivity (valence in the circumplex model) of not only artworks but also objects and shapes[4] seems to be happening across different brain areas for different sensory modalities[78]. This supports the studies of affective motion textures, creating specific affected states across groups of people by choosing specific art attributes and their characteristics, e.g. path curvature, shape, direction, trajectory, smoothness, acceleration, linear vs radial shapes etc[17, 36]. Although these studies had only around 20 participants, this suggests there is a link between specific attributes and affected state it creates.

To help understand different art attributes and make clearer distinctions between what characteristics an artwork has or can have, the Assessment of Art Attributes (AAA)[10] questionnaire was looked into. The AAA is a commonly used tool to quantitatively assess attributes of visual artwork. These attributes are divided into two. -The formal perceptual attributes: balance, color saturation, color temperature, depth, complexity, and stroke style. -The content representational attributes: abstractness, animacy, emotionality, realism, representational accuracy, and symbolism.

Similar division of art attributes were found when researching preattentive features of visuals[81], or visual impact of elements of art in older books about graphics[21]. This distinction of attributes is used both during the assessment and also creation process of a visual, while the questionnaire is used for determining and evaluating the similarities across individual perceptions of an artwork[49].

During this chapter, the focus shifted from the cognitively processed informativeness called the visual functionality, to the datas creative potential of artistic expression to evoke affected states through an aesthetic experience. The aesthetic approach evaluates the created data art in terms of the produced feelings and sensations. The preceding research suggests, that visual representation and communication of data could be more effective if the artistic side of the visualization is strengthened. This facilitate engaging users attention, encouraging the user to react and attend to the stimulus. Additionally it suggests, that desired characteristics of the data could be mapped, and communicated through an affective state by choosing specific artworks attributes.

Now we will look into novel technique of visualizing and possible way of merging rules for creating a visual with the data.

2.12 Generative art

As creation is related to the creator, so is the work of art related to the law inherent in it. The work grows in its own way, on the basis of common, universal rules, but it is not the rule, not universal a priori. The work is not law, it is above the law. Paul Klee, 1961[31]

A novel art form called generative art, defined by artist and professor Philip Galanter, as referring to an art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art[19]. Merging philosophy, data, aesthetics, and experimentation between quite distinct fields of computer science and art, it is an algorithmic way of creating a visual from data values. Quite different from the traditional art, here the use of principles of numerical representations and modularity through algorithm is aiding or leading the automated process of creation. Various applications of generative art exist, such as live performances, music visualizations, visual art, literature, architecture etc. Exploration of the affective capabilities of generative art[16] can be looked as a new tool for affecting audiences through the use of computing technology. In the paper from P. Galanter, it is suggested that generative art systems, depending on the degree of automation, are considered from the perspective of being on a continuum between high disordered (random, low comprehensibility, high information content) and highly ordered (orderly, high comprehensibility, low information content). The organizing principles in the comparison of various generative art systems are the effective complexity, order, and disorder. The goal of the system and the author determines where on the continuum the produced generative art algorithm will belong, which helps with defining rules for the development approach. The non-specific continuity (or an open interpretative form) that the high disorder offers leads the audience to accept illogical, irrational and multiple continuities, and make their own meaning out of the visual. Various commonly used generative algorithms exists such as Perlin noise, cellular automata, fractals, oscillation figures, formulated bodies, etc[55, 3]. To access

and use these algorithms, Processing, is a flexible open source software sketchbook that allows for simple prototyping and learning to code within the field of visual arts. With over 100 libraries, good documentation, and large community of enthusiastic users, it is a simple way to start experimenting with generative art.

2.13 State of the art for visual representation of data

The aim of researching the state of the art, was to find and understand previous similar works in the field and their techniques. Although the field of visual representation of data was researched and described in the previous chapters, the state of the art chapter gives better understanding of how these fields were connected in previous studies. This chapter describes some of the studies, dividing and categorizing them from the point of view depending on their focus.

A paper focused on analyzing social media content and creating a tool called HarVis[2]. HarVis is a framework with an aim to assist analysis of YouTubes textual content, facilitating acquisition, storage, processing and visualization of content. The visualization offers exploratory analysis of relevant informations and extensive data about videos, authors and comments, based on the searched tags, topics, or names. Research is divided into two particular aspects: data acquisition method, and data visualization. On the figure below is the exploratory visualization of organized data.

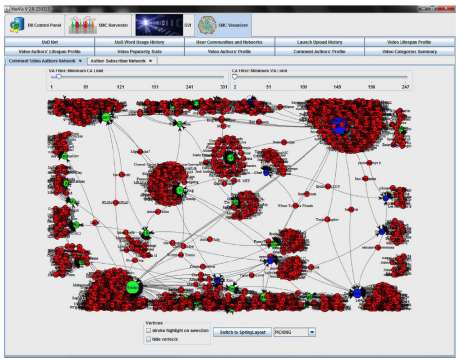


Fig. 3. HarVis[2], organized map of YouTube content and networking communities. Authors of videos are green, authors for comments red, and blue for people with dual roles.

Compared to the fan page ranking research, here the central focus was on the organization and visualization of the already existing data about the content, without further processing or analysis of the textual data. Information presented in the visual was organized to facilitate answering of five main topics regarding the content: temporal evolution of the network, vocabulary network, authors

popularity and influence, existing categories, user communities and influencers. The evaluation of the tool was based on quantitatively evaluating the precision of produced visuals, compared to the previous data retrieval systems. However, the produced data visualization has high complexity and informational content, communicating all different categories and their connections with only few annotations and without legend. As this was not evaluated during the research, the visuals usability and effectiveness in communicating can be questioned. Yet, this served as an insight into alternatives regarding organizing content visually.

In the research *Design and Development of Visualization Tool for Movie Review and Sentiment Analysis (2016)* the focus was on the problem of the usual display of reviews and recommends chronologically in a list format. To make it easier to grasp information and categorize the reviews based on the most helpful, the authors developed a visualization tool that visualizes sentimental analysis of the reviews on pre-made Sentiment Dictionary. More information regarding the used dictionary and its name are missing. The final visual (figure below) shows only a small number of highlighted emotionally rich comments on the movie, extracted from the analyzed reviews and linked by clicking to the original review.

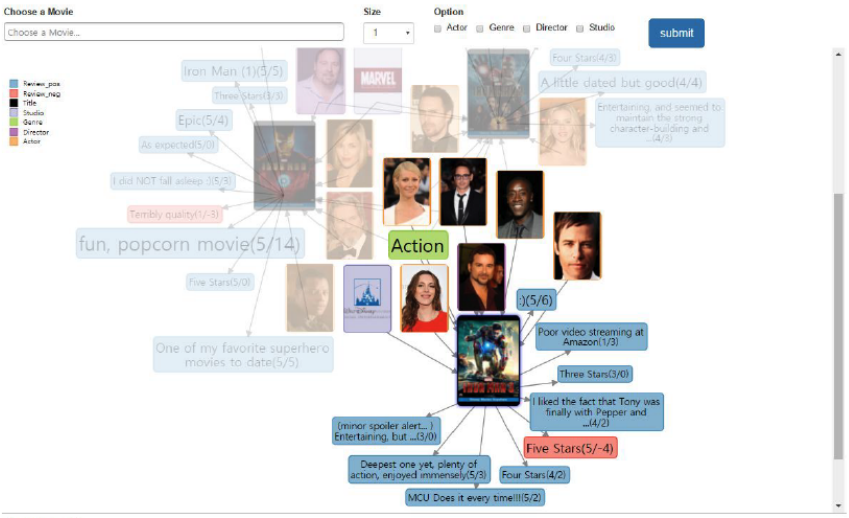


Fig. 4. Proposed system of Movie Reviews Visualization tool[83]

This is another functional approach towards navigating through the movie ratings or product reviews in an interactive data visualization. User evaluation of the developed visualization tool effectiveness was noted as an interest of future investigations.

Cosmovis - visualization of heat map sentiments, projected onto star signs[22] falls in the same category where the focus was to visualize reviews based on the sentiment. With slightly different and more aesthetical approach, the authors firstly collected data from Korean movie service NAVER, then established their own korean sentiment word dictionary with consultation with linguist. Each of seven main emotions was classified by category, that included its kinds of synonyms as sentiment words in the same emotional category. Then the reviews were organized by their main sentiment, and projected onto a heatmap visualization. To facilitate the cognitive understanding an interesting approach of projecting heatmap into constellations was used. The asterism name, and the symbol on which the heatmap was presented, was chosen based on the movies main emotion or emotions. The final visual for one movie can be seen on the following Figure 5.

Fig. 5. CosMovis constellation map of sentiment words from movie Paranormal Activity[22]

The evaluation consisted of pilot testing the reaction and learning time of the visual between two groups of 10 participants. The drawback of the study was that participants used the location of the emotion nodes to asses the sentiment rather than the symbol of the constellation. Although the informativeness of the two-dimensional model where the emotions were placed can be considered as an helpful information that led better results in understanding the visual, the

visual communication of the chosen symbols is being questioned at the end of the research.

From the more aesthetic and affective visualizations focused studies, *Beyond Data: Abstract Motionscapes as Affective Visualization (2017)*[16] was chosen as an example for the approach focused on communicating through sentiment. The paper summarizes findings on the affective expressiveness of abstract motionscapes and set guidelines for extending the use of affective visuals from mainly artistic circles also to HCI studies. Authors create different interactive motionscape in VR, where the motionscape varies in shape (compositional layout), motion factors (speed,direction,path curvature) and presentational aspects as viewpoint and display condition, since those are considered as key contributors for creating the intended motion affects. Instead of communicating information as static visual, here the focus goes towards communicating intended feelings of the environment, such as in the case of movies, games, or performances. Although such scenarios are created with quite different goal, the findings of the study are interesting for also other various uses of communicating through affective intent and support the idea that there is a link between embedded visual properties and perceived emotional value. The only research was found that connects the information visualization and affective visualization with a focus on communicating emotions through a static data visualization. In *Affective Data Visualization: A Preliminary Study (2018)*[41], authors explore how data visualizations can be used to get a targeted emotional response. Twenty visuals with different fonts, colours and font sizes were tested by 41 people in regards to the emotions evoked. The color was found as the only element affecting the evoked state. However, the tested visuals were in the form of graphs, where only the used elements to form the same graph varied. The research lacks further investigations, such as adding more preattentive visual properties for testing the emotion manipulation effect, or trying different kinds of data visualizations.

This chapter summarizes the state of the art studies in the fields of sentiment analysis and visual representations approaches. The studies varies and were described starting from the ones purely focused on textual data analysis, towards the ones focused on visualizing the analyzed informations with different approaches. Finally, recent studies regarding affective visualizations and their guidelines on communicating with affective intent were mentioned. This acknowledges the direction of this paper investigation.

2.14 Dataset

For the purpose of extracting emotional information from a textual data it is highly recommended to select data that revolves around stories and complete idea formulations with ideally randomized topic of the content.

It is very common nowadays to use social media API to retrieve big datasets especially for data science purposes. However, these are usually platforms that users use to express themselves in limited form (comments, reactions, opinions), lacking the core of the story (subject often not present in the text) since the focus of the content is aimed towards an external medium (video, tweet, picture) that

it refers to. It is possible to filter out the data that hold value of a complete story from these social platforms but it would not be time efficient for this projects timeline. Another major problem with social medias textual data is frequently used (even desired) sarcasm. Sarcasm is almost impossible to detect from just the textual expression if the clue/s lie in an external medium (previous tweet/comment, general subject, footage etc.).

Famous datasets for sentiment analysis and other NLP tasks are coming from Tweeter or YouTube in form of textual data from user tweets and comments respectively. This textual data is very rich on emotional print in various forms like emoticons, gifs, excessive punctuation or text-formatting expressions (Capital letters, spaces, spam etc.). Such data can serve well for sentiment analysis and possibly for deeper emotional print as is desired in this project. However, these same features that provide potential value for generation of an emotional print of the data are as well very computationally expensive to distinguish and classify mainly thanks to overuse of the sarcasm which is often undetectable knowing nothing about the general subject of the content or previous tweets in the same chain (chain of tweets as people react to other tweets).

For this reason the dataset of IMDB reviews seems much more convenient. IMDB reviews are another very famous source of data (in data science). IMDB reviews are almost guaranteed to provide complete point of view (potential emotional print) while mentioning the subject (movie). Disadvantages of using such dataset are the length of the text data that is very large on average causing potential loss of definition (sharpness) of the extracted information. For example the authors stance may shift along the review as the focus switches between the sub-subjects (actor performances, plot, CGI, etc.) creating noisy data that (without advanced processing) produce insignificant results (middle ground, averages).

A new platform is also emerging called Duckling. This platform hosts environment for users to write stories in form similar to a slideshow containing both text and other medias. This dataset was believed to suit the purpose of this project but the serious size and language (often in danish) limitations were crucial in deciding not to use it further.

Considering the options in terms of pre-processing needed, potential value carrying, amount of data points and access to the datasets the dataset of IMDB reviews was chosen. Convenient enough the dataset of IMDB reviews with additional sentiment labeling (positive=1, negative =0) feature was found [38]. This labeling can serve as test for the sentiment feature extraction-method validation.

2.15 Final problem statement

This chapter summarizes key findings from the background research. Afterwards the final problem statement and reasoning of the interest for this thesis will be stated and explained.

There are several already known methods to address processing of textual data for the purpose such as analysing the sentiment, subjectivity or semantic meaning. Natural language processing algorithms such as word2vec and different

lexicons, as GloVe, NRC or SentiWordNet are already existing toolkits for text processing. Analyzing and processing textual data facilitate deeper understanding of the content, and can suggest a more meaningful categorization of the data compared to the traditional recommender systems.

The ways for visually representing the information were divided into two distinct approaches, functional and aesthetical. While they could be further divided into sub-categories depending on what is being communicated or methods of communication, the aim was to provide an understanding of both approaches and provide reasoning of their use. The functional approach is measured in terms of effectiveness of the visuals communicativeness, while the aesthetic approach is focused rather on communicating affective states and measuring the perceived sentiments. There is ongoing research in the topic areas of evaluating affective intent in motionscapes, or influence of different shapes and their characteristics on the affective state. From the research of neuroesthetics, it is known there are common neural pathways activations for different preattentive attributes that influence our perception, and also common aesthetic preferences among most people. The intentional influence of users sensations had become one of the central parts of the new media age. This has been mainly targeted within the development of narrative experiences, for example cinema storytelling or game industry, however the use of affective intent is slowly rising and being extended to the HCI technology as well.

Although both approaches, functional and aesthetic, have their common and reasonable use between quite distinct groups of enthusiasts, there was not found much research in regards of combining them. Apart of adding aesthetics into data visualization, or evaluating perception of different kinds of data visualizations, the area of communicatively mapping data onto affective visualizations is yet to be explored. This lead to a question whether extracted data values, usually communicated in form of informational data visualizations, can be visualized with an aesthetic approach, yet without loosing their communicativeness. Generative art opens a possibility to define data-driven set of rules for creating metaphors from the data values, that would be communicated through an affective state evoked by the computer drawn artwork. This extends the traditional uses of the data, and combines it with an added experimental and artistic value.

From the analysis the current problem area can be divided into two smaller fields of interest: emotion-content aware recommender systems and affective visual representations of metadata. Both fields could be rather separate directions of the research, since each has their own distinct purposes that needs to be designed, implemented and evaluated rather separately. However, our aim was to connect both directions and take the results of the recommender systems textual data analysis and translate their form onto an affective visualization. This was for the purpose of combining our different fields of interest and additionally the combination allowed extending the researched area. For this reason, the following chapters of design, implementation, evaluation and results will be divided respectively.

The final problem statements for each of the problem areas are following:

- *How can image scraping and arousal lexicons influence results of traditional NLP methods for the purpose of improving recommender systems?*
- *How can characteristics of complex metadata from multivariate analysis of movie reviews be mapped and communicated through an affective visualization?*

3 Methods

Having established an understanding of the field and defined the final problem statements, the next step is to define the methodology for attaining the goal of the study. The end of the analysis suggests division into two smaller problem areas, which can be understood also as separate phases complementing each other, but with their own aims and measurements. First phase will be called NLP phase and the second phase is the visualization phase. For both phases the fundamental methods for evaluating will be introduced separately.

In regards to the FPS, the goal of the NLP phase is to analyze textual data with the use of NLP tools and lexicons, to create more meaningful, emotional-content aware categorization of the data. The researched tools such as different lexicons, GloVe model and other NLP methods, will serve as exploratory tools for the data analysis, and the findings will form data features representing textual data. This will result in a high-dimensional dataset with the features of the original textual data. The high dimensional dataset then needs to be further analyzed and simplified in a meaningful way, to facilitate the understanding of the results. The convenience of the results then needs to be evaluated, to asses the meaningfulness of the created organization.

The goal of the visualization phase is to use the created metadata from the resulted organization, understand their characteristics, and map the characteristics into visual attributes of the affective visualization. The affective visualization will be created with the use of different generative art algorithms. For this phase, IMDb reviews will be used as the textual data. The researched values of different visual elements, and the principles from generative art chapter, will serve as guidelines for the phases of mapping, and shaping the final form of the visual. The art assessment tool will be extended for evaluating affective visualization of data and its communicativeness of different values. Additionally, the algorithm will be evaluated based on the effectiveness and aesthetical value of the generated results. For evaluating the final visuals, the measurements are divided into two following categories.

3.1 Topic modelling, sentiment analysis and emotion

Since the big emphasis of the project is on recommender systems and potential application of the system in the social media, the topic modelling is accepted as a feature for the further NLP analysis as it is one of the key factors that distinguish between textual data. In other words topic of the textual data matters even if the intention of the project is to experiment with potential emotion indicating

features. For topic modelling the implementation of the GloVe word embedding models provide fast and precise solution to the word to vector operation. These vectors can be then used to assume the general topic of the textual data similar to the Doc2Vec model.

In order to represent textual data based on the potential emotional signature it possess, the basic sentiment analysis methods such as sentiment, subjectivity and topic needs to be performed on the data and the features needs to be extracted for the purpose of further NLP analysis. Many libraries promise relatively accurate sentiment analysis such as NLTK, CoreNLP, Spacy, Gensim [80], TextBlob or custom libraries like Vader [29] that also accounts for emoticons. The final choice depends on the implementation difficulty and the accuracy achieved.

To complete the circumplex of emotions space for emotion mapping, the arousal needs to be extracted from the textual data as well. This can be done using NRC-VAD lexicon. This lexicon covers 20 000 common words including affective words. Other lexicons lack behind with size and validity of the results (too few judges for labeling).

The accuracy of the sentiment domain could be tested against the sentiment labels obtained from the dataset like IMDB sentiment dataset. Unlike sentiment other features like subjectivity and arousal are not labeled in the IMDB sentiment dataset and therefore the accuracy and validity of the results can be only evaluated as manual observation using visual cues like charts, plots and statistic cues.

It is expected that lexicon based extraction of arousal should provide limited results in terms of accuracy using IMDB sentiment dataset. This is partially because even with the size of 20 000 words NRC-VAD lexicon cannot cover majority of the natural language of english. The other factor contributing to the lack of value distribution (spread) for arousal, as well as other features extracted from the dataset, could be the length of the reviews in the dataset that (for earlier mentioned reasons) creates noisy data reducing the significance of the results. However, for the experimental purposes of this project these warnings are accepted and the methods are implemented.

3.2 Image scraping

To evaluate the potential of information delivered by the approach of representing textual data as images it is necessary to develop a system that will scrape internet for pictures through use of one of the biggest search engines available (Google, Yahoo, Bing). After retrieving the image representation of the text the images need to be represented in a convenient way so the potential information they carry can be distinguished easier. For this purpose the color and entropy features of the images will be the extracted and used as features for the further NLP analysis of the textual data together with the sentiment analysis and topic modelling features obtained in the previous step.

Just like it was with unlabeled features subjectivity and arousal, the scraping method providing features of entropy and color cannot be tested directly

(true,false) and the validity of the information these features carry will therefore need to be concluded from visual observations and possibly basic statistical measures.

3.3 Evaluation through vizualization

Considering successful extraction of the features (topic, valence (sentiment), arousal, subjectivity, entropy and color) from the textual data, the relevance or validity of the features (or their experimental combinations) could be observed by means of basic statistics like correlation of the features, mean and variation values. Following the initial statistics the feature set could be observed in visualizations allowed by dimensionality reduction algorithms (t-SNE, PCA). Judging of the visualizations could be aided by clustering algorithms like K-means or Gaussian Mixture. Thanks to the lost relevancy of the distances between visual clusters formed by t-SNE, it is not recommended to use clustering algorithms or and cluster distance measures on the visualizations produced by t-SNE. However, the purpose of the evaluation is to visually judge the results and therefore an aid in means of clustering measures, even if not definite, could be tolerated. Clustering would allow for further measures of the validity of the clustering like Silhouette score that could therefore represent the final score of the visualization (combination of features). This score would only serve as a clue because its validity is compromised by the dimensionality reduction algorithms in the previous step. Among the cluster visualizations could also belong distribution measures like Gaussian Mixture to display the distribution and densities of data points in the clusters.

3.4 Communicativeness of the visual

Measuring the correlation between the perception of the visual and data characteristics, and assessing the range of words used to describe the same visual.

Generative algorithm effectiveness Evaluating the generated results from the perspective of effective complexity, in regards to the continuum between order and chaos. This means the visual results for the same dataset needs to be similar enough in terms of the communicated values but without having the same form.

4 Design and implementation of NLP phase

In this chapter the process of experimentation and tuning of the NLP phase will be described. The intention of the NLP phase is to extract meaningful information, with emphasis on the emotional tone, from the textual data of IMDB sentiment dataset. The idea is that this information will be encoded in the features of topic, sentiment, arousal, subjectivity, entropy and color. The initial

analysis showed that features of sentiment, subjectivity and topic labeling are very common and their implementation is relatively easy thanks to wide range of libraries. The rest of the features are considered experimental and their implementation will therefore be documented better. The structure of this chapter will start with description of data pre-processing followed by description of extraction of features starting with topic modelling, sentiment and subjectivity. The following feature extraction described will be arousal followed by image scraping method for entropy and color feature extraction. The chapter will be enclosed by description of the feature selection and evaluation method.

All the implementation was done in Python programming language in Jupyter Notebooks environment. Jupyter Notebooks environment allows for code development in manner of a notebook-like structure where the output of each code section can be displayed creating an ideal environment for scientific Python-based implementation documentations. The design and implementation are described as precisely as possible in the following sections but the complete code can be seen in Appendix B for image scraping implementation, Appendix C for entropy extraction implementation, Appendix D for preprocessing implementation and Appendix E for NLP implementation and observation.

4.1 Preprocessing

The original IMDB sentiment dataset has 25 000 data points (reviews). For the experimental purposes only a small chunk of 1000 data points was cut out. The preprocessing consisted of punctuation, white spaces and digitals removal, lowercase reformatting, tokenization, length threshold (minimal length of 4 words) and finally rebuilding of the dataset (from tokens). The library used for most of these features was NLTK (natural language toolkit). Despite the significant effort to make each operation do hundred percent of its work, the data required manual operations in order to prevent some information losses. Even with this the manual intervention the data contains impurities. However, these can be tolerated as their presence in the text is not significant. For the purpose of the improved sentiment feature extraction results, the original IMDB dataset is preserved as a second instance of the same data set that will be used specifically for the purpose of sentiment feature extraction thanks to the importance of punctuation and text formatting for sentiment detection.

4.2 Topic modelling

Going into the phase of feature extraction from preprocessing the two datasets are available. Main dataset are preprocessed IMDB reviews with sentiment labels. The secondary dataset consist of the original text data from the IMDB reviews and this dataset will be used only for sentiment feature extraction later on. For the rest of the features the serving dataset will be main (preprocessed) dataset. The preprocessed reviews are tokenized (split into words) and this token representation of each review is added to the main dataset. The frequency of each

token within each review is counted and the resulting array of term frequencies is also added .

	sentibin	text	tokens	words	counts	wordsT
0	1	With stuff going moment started listening musi...	[With, stuff, going, moment, started, listenin...	[people, hate, drug, know, going, message, lik...	[5, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, ...	152
1	1	Classic Worlds Timothy Hines entertaining film...	[Classic, Worlds, Timothy, Hines, entertaining...	[Hines, critic, effort, look, entertaining, We...	[3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, ...	62
2	0	film start manager Nicholas Bell giving welcom...	[film, start, manager, Nicholas, Bell, giving,...	[animal, Sabretooths, film, pack, Park, Miller...	[4, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...	200
3	0	must assumed praised film greatest filmed oper...	[must, assumed, praised, film, greatest, filme...	[film, care, dont, opera, theatre, another, do...	[4, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...	164
4	1	Superbly trashy wondrously unpretentious explo...	[Superbly, trashy, wondrously, unpretentious, ...	[plot, Ingrid, even, Fire, sense, need, time, ...	[3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, ...	187

Fig. 6. Dataset - tokens , roken frequency (counts) and word count (wordT)

To be able to perform any topic modeling approach the tokens (words) first need to be translated into vectors using word2vec principal. Instead of creating a custom word2vec model from the dataset the pretrained GloVe model of 300-dimensional vectors (biggest available) was used. As mentioned already the GloVe models are pretrained on immense amount of text and map around 400 000 english words. Dimensions in these models indicate complexity and accuracy of the final model (bigger = better). A custom method for topic extraction of each review was then assembled. This method takes most frequent words from the review, translates the words to the vectors based on the GloVe model and finally determines final topic vector based on the average of the cumulative value of those words. Each word vector is weighted by the frequency of the occurrence within the review. The final topic of a review is therefore defined as average vector (word) of most frequent words within the review with the frequency of each word being accounted for.

	sentibin	text	tokens	words	counts	wordsT	topicV
0	1	With stuff going moment started listening musi...	[With, stuff, going, moment, started, listenin...	[people, hate, drug, know, going, message, lik...	[5, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, ...	152	[-0.1943876946290188, 0.1390375972670674, 0.1...
1	1	Classic Worlds Timothy Hines entertaining film...	[Classic, Worlds, Timothy, Hines, entertaining...	[Hines, critic, effort, look, entertaining, We...	[3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, ...	62	[0.13889166629976696, 0.07754777661628193, -0....
2	0	film start manager Nicholas Bell giving welcom...	[film, start, manager, Nicholas, Bell, giving,...	[animal, Sabretooths, film, pack, Park, Miller...	[4, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...	200	[0.03761188271972868, 0.31459445423550075, -0....
3	0	must assumed praised film greatest filmed oper...	[must, assumed, praised, film, greatest, filme...	[film, care, dont, opera, theatre, another, do...	[4, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...	164	[-0.2937451899051666, -0.1867117385069529, -0....
4	1	Superbly trashy wondrously unpretentious explo...	[Superbly, trashy, wondrously, unpretentious, ...	[plot, Ingrid, even, Fire, sense, need, time, ...	[3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, ...	187	[-0.24887001141905785, -0.07671374827623367, -...

Fig. 7. Dataset - topic vector (topicV)

It seems fair to mention here that initially the topic feature had dimensions of the GloVe model (300 dimensions) creating uneven balance in feature set with other features (300 + 7[eventually]). This issue was resolved by reducing the dimensionality of the topic vectors to just two using dimensionality reduction algorithm.

An experimental measure of the topic distribution across the whole dataset was performed by affinity propagation clustering algorithm. This algorithm allows for clustering without number of clusters being predefined. Affinity propagation therefore, gives a glance of the distribution of topics of the reviews for purely experimental purposes. The results were not very interesting, showing great number of clusters. The separation of data based on topic can be only indicated by t-SNE visualization of the distribution.

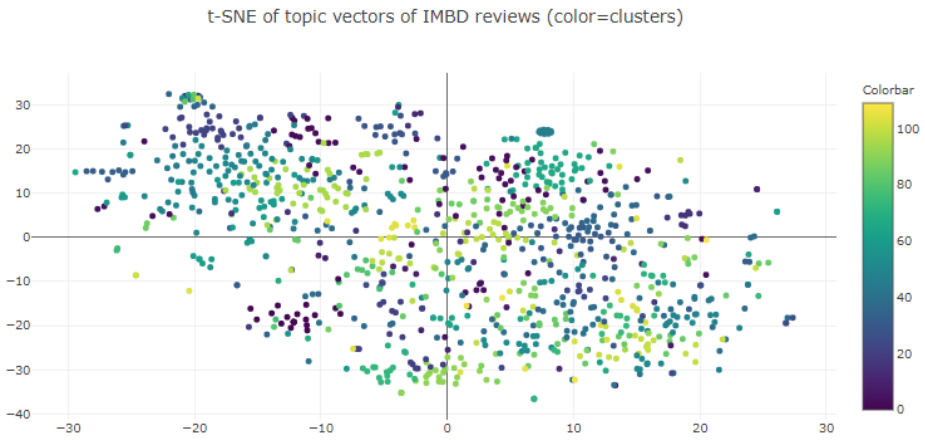


Fig. 8. t-SNE visualization of topic vectors

4.3 Sentiment feature

Sentiment can be extracted using almost any NLP library out there. The differences in terms of results are usually minimal. However, these methods usually requires training and therefore, for the time management purposes, the lexical and rule based approaches were researched instead. One idea was to use NRC lexicon for sentiment analysis. When researching the possible options a less famous library for sentiment analysis was found on GitHub called Vader (Valence Aware Dictionary and sEntiment Reasoner). This sentiment analysis library is based on lexicons and syntactic rules (rule-based) that allows for sentiment extraction specifically designed for social media originated textual data. In the earlier phase this project development the dataset used was different than the IMDB that was eventually decided to be used. Therefore, the Vader sentiment was considered and tested with impressive results vastly because Vader considers punctuation, emoticons and some slangs resulting in better judgement than many famous sentiment analysis toolkits. Once the IMDB review dataset was chosen the sentiment feature extraction method was reconsidered but its performance was still comparable to the conventional methods (70% success rate) and outperformed the pure lexicon based approach using NRC lexicon.

Fig. 9. Statistical comparison of valence rating accuracy using NRC-VAD lexicon and Vader

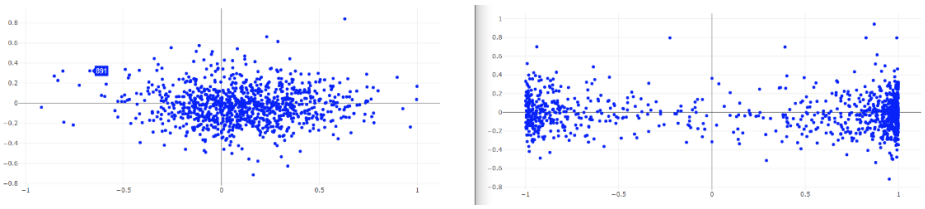


Fig. 10. Comparison of valence distribution using NRC-VAD lexicon (left) and Vader (right)

The sentiment of a review was performed on the dataset of unprocessed (original) dataset of IMDB reviews. When deciding the level on which the Vader should operate (sentence or text as whole) it was decided to go with top level (text as whole) as the results shown despite the lower accuracy (-3% compared to sentence level), the distribution seemed more defined.

The precise testing of the success rate was possible thanks to the labeled data of the IMDB dataset used for this project. The Vader library was chosen to extract the sentiment values for each review and the values were added to the dataset.

4.4 Subjectivity feature

To complete the set of conventional features of NLP (topic, sentiment, subjectivity) the subjectivity was added to the feature set of the dataset. The library used for this operation was TextBlob thanks to its convenient implementation.

The subjectivity was rated on the sentence level as the distribution of values appeared better as opposed to the top level (text as whole). Dataset used for the subjectivity was also original (unprocessed) dataset of IMDB reviews that was used in sentiment extraction phase. The distribution of subjectivity values resulting from TextBlobs subjectivity extraction method was displayed as the original t-SNE visualization of topics ordered by the subjectivity of the data points (X axis).

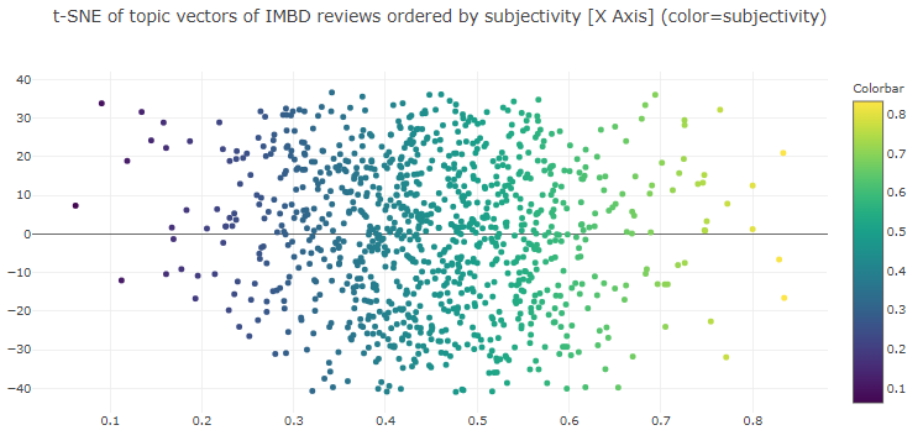


Fig. 11. t-SNE map of topic vectors of IMBD reviews ordered by subjectivity values (X-axis)

When comparing the t-SNE visualizations of features (without and with subjectivity feature) the subjectivity brings visible separation.

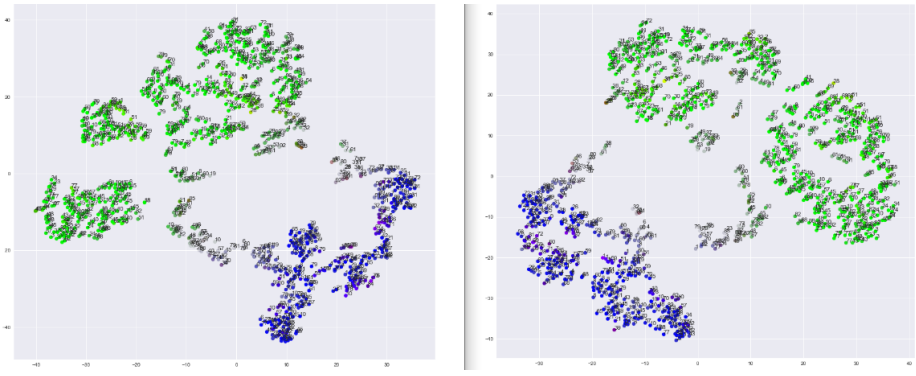


Fig. 12. Comparison of t-SNE visualizations of features of valence and topic with (right) and without (left) subjectivity feature included

4.5 Arousal feature

In order to implement circumplex of emotions model for emotion mapping the arousal was assumed by using NRC lexicon of valence, arousal and dominance (VAD). When exploring the resulting values the weak spread of arousal was indicated by means of a visual plot.

Fig. 13. Plot of IMDB reviews in space of valence(X) and arousal(Y) where arousal is extracted from every token (word)

To enhance this spread a method of frequent-token (word) selection (previously used for topic modelling) was used. This selective representation of each review shown much better spread of the arousal values across the dataset.

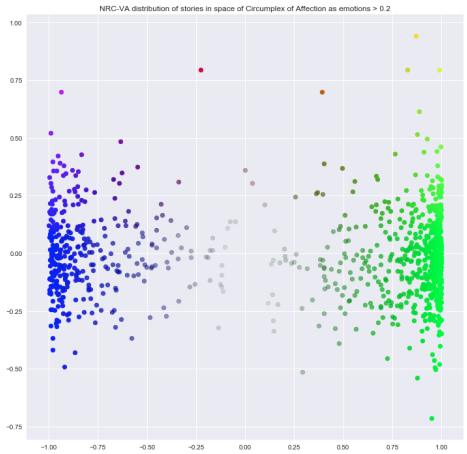


Fig. 14. Plot of IMDB reviews in space of valence(X) and arousal(Y) where arousal is extracted from only top 5 of the most frequent tokens (words)

4.6 Image scraping for entropy and color features

Originally inspired by a github work of Nelson-Liu [35] the web scraper was build to search Yahoo image results based on an input phrase and extract dominant colors and entropy of the images. Entropy is feature that represents informative potential of a data including picture pixels [69]. Google search engine would be preferred thanks to its robustness and popularity but since Google requires paid API for scraping the Bing and Yahoo needed to considered instead. Comparing the two Yahoo provides bigger database of pictures and resulting images from Yahoo were experimentally more accurate than those from Bing in terms of relevancy to the searched phrase. The scraper was set to gather only top 5 resulting images as consequence of high cost in terms of computational power discovered later on. Images that the scraper returns were rescaled and combined into one picture for more effective color clustering. The original images were preserved and used for entropy calculation extraction.

The dominant color extraction works as set of k-means clusterings. First k-means clustering is performed on the combined image retrieving 5 main colors (clusters) including gray scale results. These main colors are then filtered, excluding gray scale results using standard deviation threshold filtering performed on the color channels of each of the 5 main colors. If the phrase of the scraping returns no colors or only grayscale results, the resulting color for that phrase will be black. Entropy is implemented using sklearn.metrics library for entropy. The entropy is calculated as average of entropies of all images obtained from the scraping. The entropy for each image is performed on the grayscale version of the image.

The process of phrase selection from the text is done using library TextBlob that has functionality to isolate phrases that revolves around a noun (noun-

phrases). The noun phrase is expected to provide potential center of focus (noun) around which the shaping words are located (adjectives etc.) [84]. This structure of phrases should enhance and define the informative value of the selection and therefore improve the scraping results of the searching engine.

Resulting values of the image scraping method are entropy and color features where color is represented by three color channels of hue, saturation and value in HSV color system.

4.7 Feature selection and evaluation

Having wider range of features: topic, sentiment, arousal, subjectivity, entropy and h, s, v channels defining each review, the big task at this point was to observe and evaluate the correlations and behavior of the features.

For each feature selection (combination of features) the 4 steps of observation were performed as follows. First the statistics and clustering was performed on the dataset (of the selected features). Then came dimensional reduction of the dataset using PCA and t-SNE followed by comparison of the t-SNE and PCA in terms of clustering potential using Gaussian Mixture and K-means clustering algorithm. And finally visualization of the population density and distribution of the data points in the t-SNE and PCA visualizations.

5 Design and implementation of visualization phase

The following chapter describes and explains the design choices for the final affective visualizations. It is divided into smaller parts, starting with specifying design requirements, then metadata characteristics, chosen initial shape, choice of initial generative algorithm, mapping choices for visual attributes, and then examples of resulting visuals and description of the iteration process. Aim is to provide an overview on the process leading to the resulting images.

5.1 Design requirements

The methods chapter has defined the methodology for evaluating the final visual in terms of the prospective measurements. The specific design requirements of the visualization are defined as follows, based on the analysis and methods chapter. Additionally, necessary mid-stones in between the requirements derived from the analysis were chosen, and their rational will be explained later on.

- specific data characteristics of the metadata from analyzed reviews needs to be chosen, and prioritized upon their informativeness about the reviews
- the design needs to feature abstraction of this characteristics
- the approximate shape and style of the visual needs to be chosen
- the visual should be of non-specific continuity and have an open interpretive form

- the abstracted characteristic needs to be communicatively mapped onto basic visual elements of the shape, while taking into account the hierarchy of perception
- the appropriate algorithm should define the interactions and transformation of the elements, based on the mapping principles
- the resulting images needs to communicate the mapped metaphors sufficiently, while keeping the effective complexity

5.2 Metadata

The obtained metadata from the processing phase were in form of an csv files of 150 values with 11 dimensions. For each movie there was one file with the resulting data describing the 150 reviews. Which different movies to analyze was decided, and they were picked by hand. The aim was to represent movies from different genres, at least 2 from same genre, with different ratings and preferably from different origins. This was to have various samples for analysis, that would result in more different visuals, although whether the genre, rating or origin of the movie would have impact on the results of the analysis was only supposed. The dimensions of the data values, as mentioned in the processing phase, were: t-SNE map x,y position in the 2D map, word2vec x,y positions, value of valence and arousal, entropy features, and mean values for saturation, hue and value of the assigned color. The decision was to continue using only some of them, and those were the t-SNE x,y position, valence and arousal. This was in order to keep only the most informative features of the dataset by considering what is most essential about the reviews to the audience and taking inspiration from the state of the art work on movie reviews analysis. To further understand the values such as the standard deviation and mean for the valence and arousal simple statistics were applied for further analysis through Python, using statistical library numPy. Additionally the t-SNE points were visualized to see their distribution, and clustering algorithm DBSCAN was applied to cluster similar points in the map together. This gave an understanding of the diversity of clusters for different movies, their size, number of points, and mean. For the valence and arousal (the mood from negative to positive, and the intensity of the emotion) the decision was to communicate one value for each movie. Therefore mean valence and arousal was calculated, which resulted in normalizing the values of arousal, and being around 0 for all the movies. However, since the standard deviation was also low and there were only few outliers with extreme values, the mean value was considered as representative. While the need to prioritize minimum number of communicated characteristics, this at the end resulted in not using the arousal value in the visual. Valence, number of clusters, number of noise points, cluster size and mean were considered as the essential values to communicate. The figure below shows comparison of the DBSCAN clusters of t-SNE points for 3 different movies (Bruno(2017), Life of Pi(2012) and Free Solo(2018)).

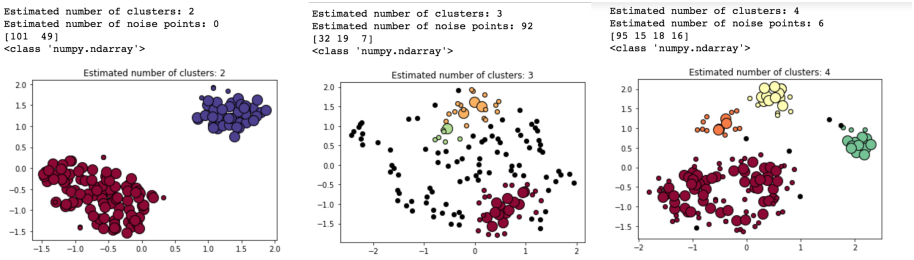


Fig. 15. Results of clustering on the reviews t-SNE map

The mean value of valence ranging from negative to positive was positive for most of the movies, even the ones with low ratings, however the range of the values was much higher and therefore considered as representative characteristic of the overall sentiment of the reviews. The diversity of clusters represents the the diversity of reviews, some with more similar characteristics than others, and some with diverse opinions resulting in a map of only noise points. The values of word2vec, entropy, or HSV values were not considered as very informative about the reviews, while they were already included in the dimensionality reduction process of the t-SNE algorithm.

5.3 Visual form

Although the metadata values, such as the mean valence, number of clusters, clusters mean and size will be somewhat shaping the form of the visual, initial form and style of the visual needs to be chosen. Inspiration was taken from the TV color test signal circle, and TV noise lines. Although other shapes were considered, the minimalist circular shape with horizontal lines was preferred for its simplicity and commonly perceived mildness and neutrality of the circular shape. The figure below shows the noise lines, color test circle, and the abstracted final form of the visual before applying the algorithm.

Fig. 16. The inspiration for the initial visual form

Although more variables could be an option and perhaps more reasonable style exists, the resulted shape achieved the desired simplicity. The purpose was to make the transformations and interactions of the line be the main attention

catching features and not its form or colors. The result of the experimental visualization should be aesthetic and non-narrative, so that the audience need to interpret the work themselves and make their own meaning out of it. This refers to the requirement of the non-specific continuity.

5.4 Generative algorithms

Mainly two generative algorithms, Perlin noise, and Attractors were tested and used. The algorithms for experimentations were chosen based on their interestingness, and ability to use multiple values that would influence their effect on the shape.

Perlin noise is a gradient type of noise, meaning the output values are changing gradually, with less randomness compared to the traditional random function. It is commonly used to achieve less machine like look, (as its developer Ken Perlin said), such as for procedural terrain, fire effects, water or clouds applying texture or shaping environment in the fields of games or other visual media.

Attractors is more complex algorithm, refined for processing by generative design community[3], that works as virtual magnets. They can be considered as points in shape, that either attract or repel other objects in shape of nodes. Each time the program is run, they have some effect on the environment what makes them time based. Effect of attractors resembles stretching of a fabric, or pattern creation, depending on the strength applied, attractors position, and other factors. Further explanation of an attractor class and its functions will be in the implementation chapter.

Firstly, the Perlin noise was added to the height of the line splitted into nodes, which resulted in curves, going from the left to the right side of the circle. Another values were added to the y position, which resulted in creating certain pattern that was followed by all the lines. By adjusting noise range and range of values added to the function calculating the final y position, different complexity and noisiness of the visual was achieved. An example of the Perlin noise curves applied on the initial form can be seen on two figures below.

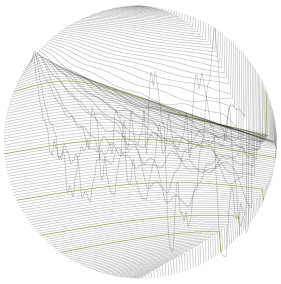


Fig. 17. Perlin noise based visual example 1

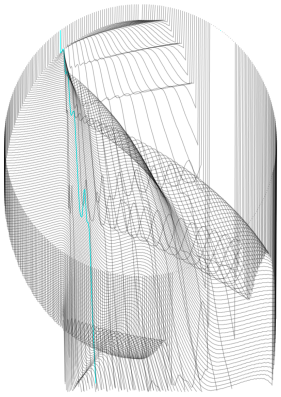


Fig. 18. Perlin noise based visual example 2

Although there were more variables to control such as number of lines, the noise of the circle radius or the colors, apart of the noisiness other variables such as position of the noise and its variance were not found to be controllable. This resulted in very random shapes, with little in common. This would make it more difficult to make communicative changes on the shape.

Attractors were more easy to be shaped, with functions as strength, ramp, radius, and direction. There are various ways to work with attractors, and effects

range from different kinds of stretching, repulsing and visually creating holes, twirling, etc. Twirling was decided as the only effect to maintain unity across visuals. Different types of twirling can be achieved, depending on the passed values to the functions. The example of range of effects for twirling can be seen on figure below.

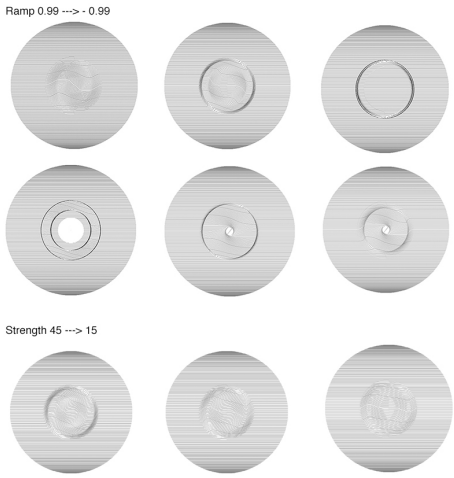


Fig. 19. The range of effects for different values of ramp and strength

On the pictures there is only one attractor in the middle, and is creating an effect for the time of approximately 3 seconds. The values changed are for the ramp function in the first six upper pictures, and with different strength value in the lower three pictures. Additionally radius affects the radius of the attractors effect, and by choosing negative values for strength, the direction of the twirl is changing. To experiment with the algorithm and see how it affects the overall shape, the initial position for the attractor was following the position of the mouse. The resulted images from the experimentation were expressive, with having quite distinct affect, and the decision was to continue with the attractors as the main algorithm.

5.5 Mapping

The chosen values from the metadata, valence, number of clusters and their mean and size, needed to be translated into metaphors, that would communicate their value in a meaningful way. Taking inspiration from the art assessment tool, and commonly mentioned qualities of data, following characteristics were chosen: random, diverse, separate, noisy. Additionally, the word positiveness was added for communicating the valence. All five characteristics, with their

antonyms added to the opposite side of the scale, creates large range of possibilities for the data expressiveness, such as positive,structured, uniform, separate and noisy, etc. While the valence has its own characteristic translated into the mood because it is the main communicated value, the other metaphors are for the representation of the clusters characteristics.

For visually encoding the data metaphors, the hierarchy of elementary perceptual tasks was followed as a rule for assigning most relevant form that would visualise the characteristics by their importance[6]. On the figure 20 the accuracy of how precisely we are able to perceive different visual attributes is shown.

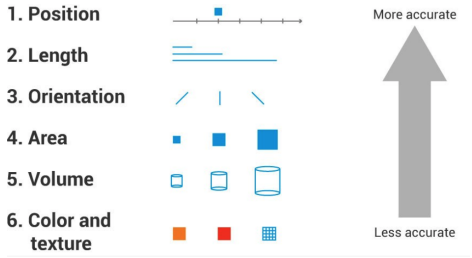


Fig. 20. Hierarchy of elementary perceptual tasks

Position on the axes has the highest accuracy, length of line second, orientation of the line or object as third, then area of the shape, next is the 3D volume of the shape, and finally the color or the texture indicators. This suggests how to choose the correct attributes by their importance in being perceived. Before choosing how each metaphor will be mapped, the visual elements of the chosen shape were defined (figure below).

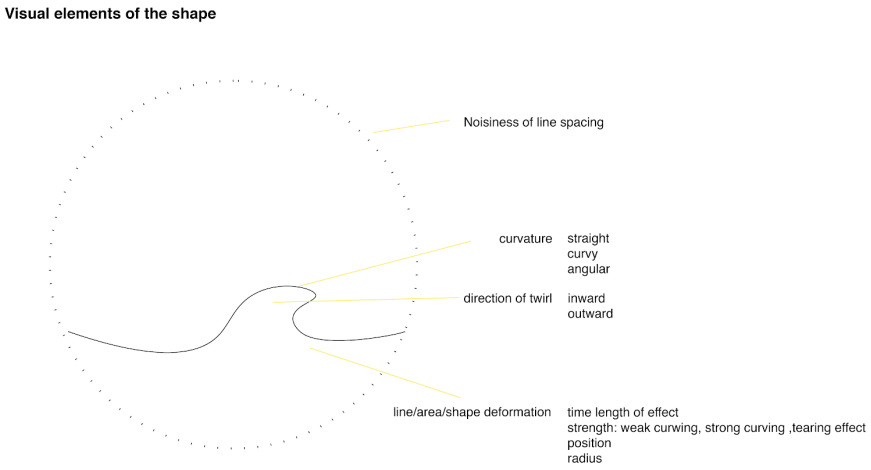


Fig. 21. Visual elements of the shape

Firstly, to get a better understanding of the look of the main elements, the position of the twirl was linked to the position of the attractors. The attractors had the positions of the cluster mean, and radius depending on the size of the cluster. This created direct translation of the clusters onto the shape. Number of attractors was depending on the number of clusters. Noisiness of the line spacing refers to uniformness of the circle radius. On the picture the circle radius is smooth, and below is the explanatory picture for noisy circle radius (figure 22).

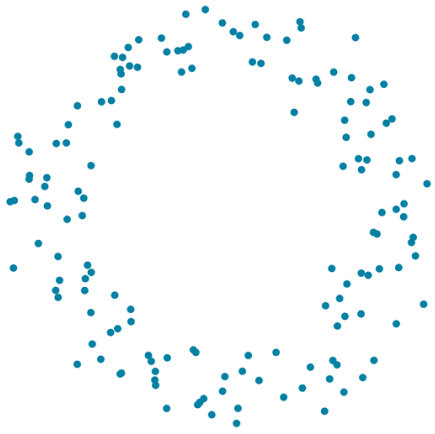


Fig. 22. Noisy circle radius

It is important to remind that the horizontal lines are connecting dots describing a circle. If the circle radius is noisy, the lines are going to start and finish at more random positions in the circle. Noisiness of the circle was translated to the metaphor of noise. Next attribute is the curvature of the line, which can be straight, curvy and angular. While straight is the passive form of the initial shape, curvyness and angularity refers to the line after it is being transformed by the attractors. This is partially achieved by the ramp function, and refers to the mood, as the rounded objects are generally perceived as positive and angular as more rough. Additionally the ramp function in negative values creates an effect of hole, which adds to the perceived disturbance. Direction on twirl was decided to be changing during the effect, and different effects were explored. When changing only once, more expressive and more affected by change results would be obtained, while often changing the direction of the twirl would lower the effect of transformation. The decision was to keep the change in twirl at random times and number of changes would be based on the number of clusters. More clusters would create effect of less visible transformation, which is in parallel with the perception of less strong cluster of opinions. Length of deformation was also decided to be linked to the number of clusters. The more clusters, the lower the effect of the destruction since the low radius, but the longer time would the effect last, more noticeable but smaller effects would be created. Strength of the effect was chosen to be same for all visuals, since it would only lower the speed by which the effects are happening, yet longer time frame with low strength creates same effect as short time frame with strong strength applied.

Additionally the valence was mapped into a colour range from pink to dark blue, resulting in three coloured lines, that would be placed accordingly on the visual. For high valence and this positiveness, the lines would be in the upper part of the circle. This was to support the communicativeness of the valence value, and make it more precise by adding the direct metaphor of scale from up do down.

This chapter provided an overview on how the metadata were mapped into the visual. It needs to be noted, there is yet too little research on the perception of the specific elements. While there is still large space for experimentations, various assumptions were tested only by intuition and trying out the different mapping combinations and their resulting visual . The accuracy of the chosen results will be evaluated during the experiment.

5.6 Results

After mapping of the values into the algorithm and the shape, for each movie various visuals were created to evaluate their randomness, similarity of communicativeness and aesthetics. Multiple movie signatures were compared, the mapping and the range of values updated during various iterations until the desired result was obtained. Below are the initial more expressive visuals.

Fig. 23. Results with more expressive forms

To be less expressive, the time length of drawing and changes to the strength were applied. This resulted in more neutral but with still defined characteristics, as can be seen on final picture below. The picture shows comparison of two visuals for the same movie, with positive mood on the top and negative on the bottom.

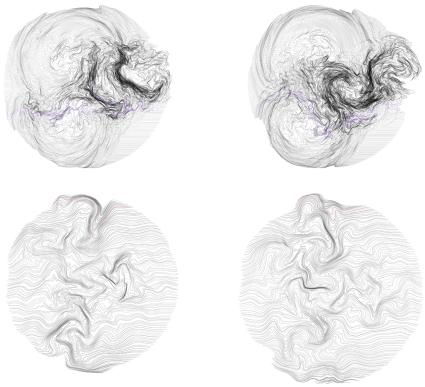


Fig. 24. Final visuals for movies *Wrinkle In Time (2018)*, and *Toy Story (1995)*

After the adjustment, the differences in the final drawings for the same movie were more distinct. Movies with negative mean valence would be more disturbed and edgy, and positive ones more curvy. Reviews with many clusters would have many smaller area disturbances, and noisiness would be visible on the imprecise line ordering and more spreaded effects across the shape. On the example above, the Wrinkle In Time had only 8 points (from 150 total) of noise and two bigger and two smaller clusters, while Toy Story had 66 points of noise and five small groups. The complete list of all visuals for 20 movies, together with the name of the movie and mapped results from the clustering of t-SNE points, can be found in the appendix A1.

6 Implementation of the visualization phase

The implementation and design were done in parallel, since the ideas needed to be tested, to see the results visually and iterate over while applying changes. Also the programming skills needed to be updated, so every idea needed to be tried out to see if and how it can be accomplished. This chapter will explain the tools used, with main focus to explain the parts of the drawing code of the visuals, and how it works. For the implementation decision was to go with the tools and languages that are well supported with large community, to ensure easier problem solving and faster learning and implementation.

6.1 Data analysis and formatting

The entire data analysis of the metadata from the processing phase, was written in Python language and executed in Jupyter Notebook environment. Scikit and NumPy libraries were used for the clustering, by using the DBSCAN machine learning algorithm. DBSCAN, or Density-based spatial clustering of applications with noise, is one of the most common clustering algorithms. It is a non-parametric algorithm, that takes a set of points in space and groups together points with many nearby neighbors and marks as outliers (noise) points that are in a low-density regions, with nearest neighbours too far. As input it takes the array of points, parameters of epsilon, (eps) which is the maximum distance between two points to be considered as neighbours, also called local radius for expanding clusters, and number of minimum samples (min_sample) which is the total weight, (number of samples) in a neighborhood for a point to be considered as a core point. From the NumPy library, only the mean and standard deviation functions were used. For each movie, the exported file included the number of clusters, their size and their mean, and screenshot of the clusters. The mean and standard deviation was calculated again in the code, so it was not necessary to export. Output formatting was in a form of a csv file.

6.2 Drawing the visual

As mentioned in analysis, Processing was chosen for its simple environment consisting of setup and draw function. Before the setup function only the variables are defined, and then in setup the before drawing needed functions are called and definitions such as the size of the canvas and its colour are defined. Before loading the file with data values, the shape needed to be drawn and functions created. The figure belows shows the functions called in the setup function.

```
void setup() {
  size(680, 680);
  smooth();
  pixelDensity(2);
  strokeCap(ROUND);
  strokeJoin(ROUND);

  makepoints(); //calculates the positions of points around a circle, and stores them in the 2D arrays.
  sortMe(left); //sorting them to be sorted by their y value
  sortMe(right);
  initGrid(); // setup node grid. This is where I save the position of each node (from all the lines) into the array.
  loadData();
  setupClusterPoints();
```

Fig. 25. The setup functions inside setup

To briefly explain, the makepoints function calculates the coordinates of each point for the right and left side of the circle and stores them in an array. The idea was to have them created around particular circle radius with added noise value to create less perfect circle. To achieve that, the nodes x and y positions were calculated based on the noise value of the circle radius, by using sinus and cosinus of the angle, the angle and radius. The sortMe functions was created to sort the points from the created array (for both left and right side) based on their y value. The initgrid function is creating the grid of nodes. Each node is an object with x and y value, stored in a list array. The nodes were created on the line connecting left points with the right points, where after each pixel on the line one node is created. The number of nodes for a line depends on its length. Since later on the decision was to make some line different color, this was achieved by having arraylist of all the nodes in an arraylist of all the lines. Once this was done, the only thing was to connect all the nodes on one line by drawing a line between them. This resulted in the initial shape consisting of a circle with pure horizontal lines, without the added attractors and data. The loaddata function loads the data and calculated the valence and arousal mean and standard deviation, from which at the end only valence mean was used. Last function called in setup, setupClusterPoint setups the data about the clusters, to be later on assigned as number of attractors, attractors position and radius.

6.3 Applying the attractors

Attractor class has 5 variables, x and y position of the attractor, strength ramp and radius value. It has one function, called attract, which takes the node object (which has only x,y position) and apply changes to its velocity values, and thus changing its position. The force of the attractor depends on the distance between the node and the attractor. Below on the figure 26 the attract function calculating the force is shown.

```
void attract(Node theNode) {
  // calculate distance
  float dx = x - theNode.x;
  float dy = y - theNode.y;
  float d = mag(dx, dy);

  if (d > 0 && d < radius) {
    // calculate force
    float s = pow(d / radius, 1 / ramp);
    float f = s * 9 * strength * (1 / (s + 1) + ((s - 3) / 4)) / d;

    // apply force to node velocity
    // swapping of dx and dy makes the twirl
    theNode.velocity.x += dy * f;
    theNode.velocity.y -= dx * f;
  }
}
```

Fig. 26. The attractor function calculating the force

The attractor function for calculating the force was setup with a help of the book Generative Design[3], where one section is dedicated to attractors. Attractor objects are also initiated in the setup at the end, with their initial position, strength, radius and ramp value. At the next phase, the draw function, is calling the attractor over and over with different values, changing the values of nodes that are drawn as lines, and thus changing the visual in real time. Each cycle results in slightly different visual, and it is changing gradually, resulting as an animation. However, this would result in few static attractors, depending on the number of clusters and their position. To add more interesting and gradual effects of the attractors, movement was needed, such as in the case when the attractors were following the mouse position. This was done by adding random walk, which is a walk based on the Perlin Noise, that would be around the attractors original position, respectively the clusters position. This was done by iterating through the clusters original positions, and adding the noise in both positive and negative values, and different value for x and y, to move the position around the main cluster. As a resulting code, that can be seen as a whole in the appendix A4, the initial attractors are the clusters points, and then for each cycle, the position of attractor is iterating through the nodes and through their added noise. Additionally, the range of added noise was linked to the number of noise points, to create more randomized look when high noisiness of data. After the new lines connecting changed nodes, there is a condition for recording, saving the frames at certain times, and condition to stop the loop when the time runs off. This briefly summarizes how the attractors works, and how the visual is drawn.

Lastly, the mapping of the data values is happening in the setup. Processing map function works perfectly for that, as it takes 5 floats, first is the value that

is being mapped, then is the initial range where the value belong, and then is the range values for the output. The loaded data were just passed through the map function. Additionally the clusters position needed to be mapped from cartesian to polar coordinates, since the initial t-SNE map is a square, whereas the map on the circle is a circle. This was done by calculating the angle and the radius from the original x and y, then the size of the radius was decreased, and the x and y coordinates were recalculated. However, since this resulted in the same effect as reducing mapping values to be all inside a square described by the circle, at the end in regards of simplifying the code only the mapping was used. This issue could be solved differently, by changing the values of radius accordingly to their position. The values in the corners would have the radius more decreased as compared to the values more in the center. However because of a lack of time resources, this could not be prioritized and only noted for future investigations.

This chapter summarized the creation process of the visual representations. Although there were aspects that could be further looked into, such experimenting with the metaphors mapping more or more precise mapping of the clusters, the results were considered meeting the minimum design requirements. The next phase consisted of evaluating the solution.

7 Evaluation of NLP phase

7.1 All features

As the first part of the evaluation all of the features were selected.

The correlation shows very little connection between the features which is good indicator meaning the features are unique.

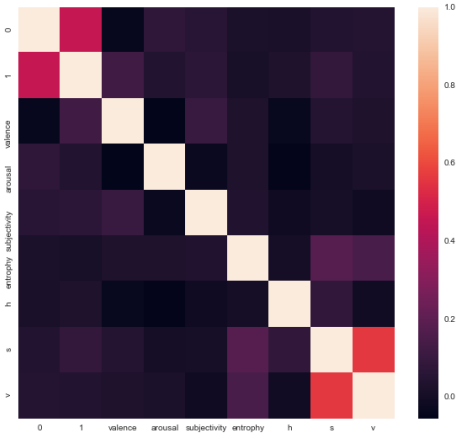


Fig. 27. Heatmap of the correlation values of the features

Basic statistics show that value distribution of features of entropy and hue are tilted on a side (high/low mean).

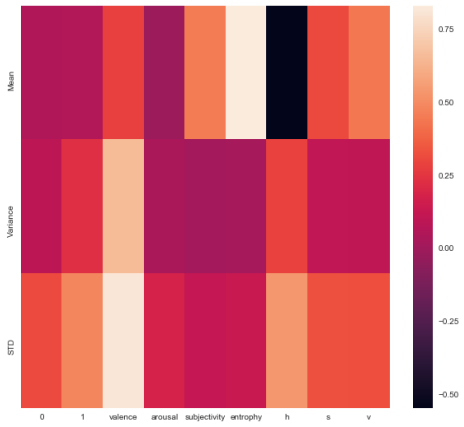


Fig. 28. Mean, Variance and Standard deviation of the features

The case of low mean for hue could be caused by filtering out gray scale results. Very high mean for entropy could be caused by noisy data thanks to the way the entropy is calculated. The minimum value of entropy is not 0 but negative number which created big gap that cause uneven distribution and the subsequent rescaling (for purpose of even feature value scales) will preserve the problem. Another noticeable outlier is high standard deviation of valence (sentiment). This can be caused by binary-like distribution of sentiment values thanks to nature of the IMDB sentiment dataset. The dataset is specifically built to distinguish the sentiment poles (negative, positive) and therefore the distribution of the valence values are expected to be almost binary. Since the labels on the dataset sentiment are binary this theory is not verifiable.

The Silhouette score performed on the raw feature set indicate ideal cluster range within 3 to 5 clusters. These scores are still very low meaning that the identity of the clusters is not very noticable.

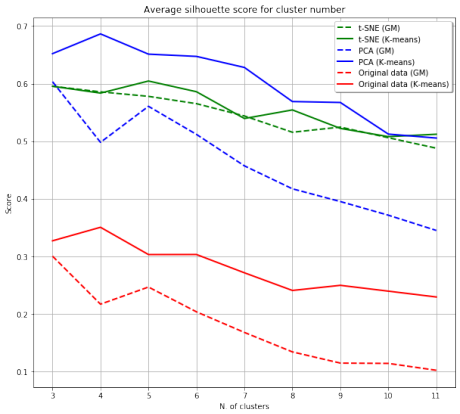


Fig. 29. Silhouette score of the clustering on raw feature set (red) and dimensionally reduced feature set (blue, green)

When visualizing the feature set with t-SNE and PCA we can instantly notice that the color feature is systematically distributed (color of data points = color feature) and clusters are clearly separated.

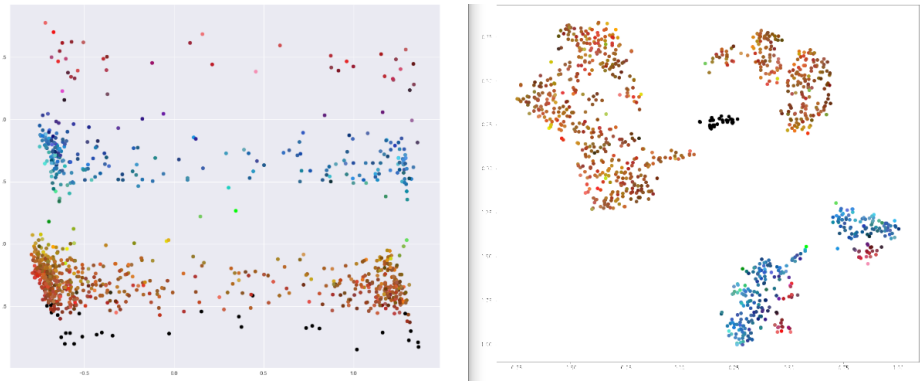


Fig. 30. PCA and t-SNE (left to right respectively) visualizations of complete (all) feature set

Further clustering analysis performed on these visualizations confirms the clear visual separation of data points both visually (Figure 31) and reflected in terms of Silhouette score (Firgure 29). Suggested number of clusters match the initial predictions and the visualizations allowed for more confident Silhouette scores indicating stronger identities of the separation clusters.

Fig. 31. Silhouette scores of clustering and respective visualizations using Gaussian Mixture on t-SNE (top left) and PCA (top right) and using K-means on t-SNE (bottom left) and PCA (bottom right)

Finally the density distribution in clusters shows that the density of data is relatively evenly distributed in both PCA and t-SNE.

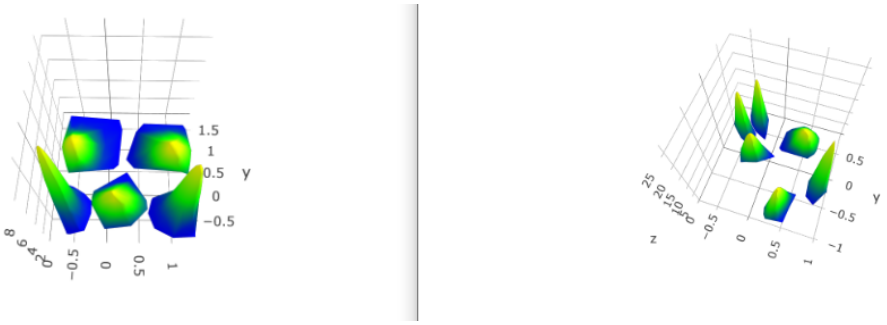


Fig. 32. Probability distribution function of PCA and t-SNE (left to right respectively) performed on complete feature set using K-means algorithm with 5 clusters for clustering

From the evaluation of the selection of all features available it can be observed that the separation is well defined indicating the importance of the features. However, when looking at the explained variance ratio for PCA reduction operation of these features, it is clear that topic (0,1) feature has substantially higher importance than other features.

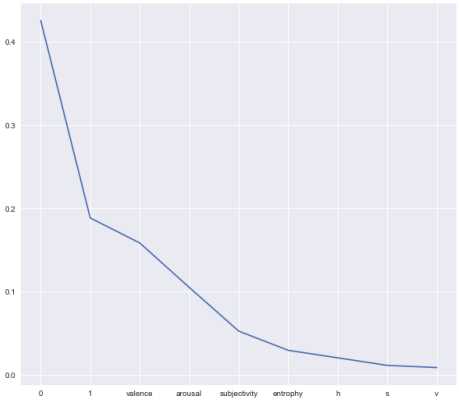


Fig. 33. Explained variance ratio of PCA performed on complete feature set

From the same PCA variance ratio we can also see that image scraping is ranked to be not that important in the visualization. The color feature is clearly influencing the results despite the low variance ratio shown from PCA.

When further inspecting image scraping features of entropy and HSV color channels, it was shown that feature of color value (V) is not very informative compared to other features and was decided to exclude it from further observations.

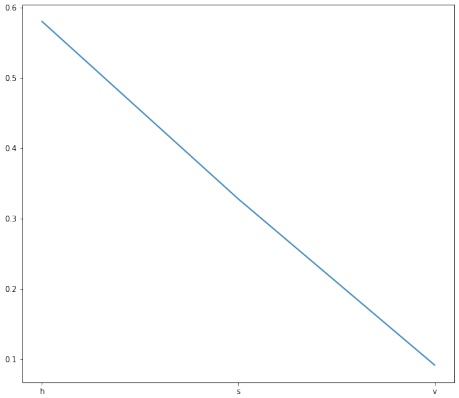


Fig. 34. Explained variance ratio of PCA performed on features of color channels (H, S, V)

7.2 No image scraping features

In order to explore the informative potential of the scraping features, as was initial purpose of this project, the scraping features were excluded from the

feature selection and the visualizations and observations were repeated in the similar manner.

From the PCA and t-SNE visualizations it is obvious that the color separation of the visualizations is not present anymore.

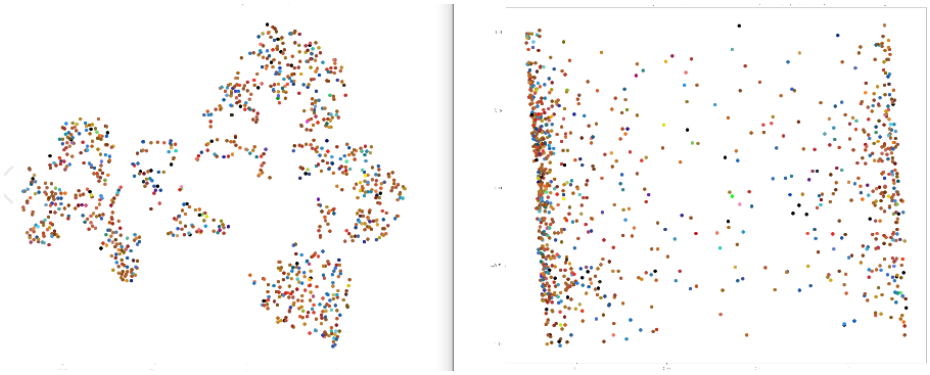


Fig. 35. t-SNE and PCA visualization of the complete feature set without image scraping features of entropy, h, s and v

The visualization of t-SNE also take upon more diverse separation at cost of lower definition. PCA explained variance ratio also shown the weakest feature in this feature selection being subjectivity which would be possible to conclude from the previous feature-selection measures as well. The information obtained from the observation of this feature selection appear to support the strong influence of image scraping color that was excluded from this selection.

7.3 Valence, arousal, subjectivity features

So far we observed full feature set and compared it to the absence of the scraping features which provided us clues that the scraping in fact has a dramatic impact on the data separation.

Next step in the exploration of the feature set was observing the isolated behavior of valence, arousal and subjectivity. In this feature selection the expected dominant feature in terms of variance ratio value was sentiment (valence) thanks to its more separated (binary like) distribution of values as opposed to subjectivity and arousal that are relatively normally distributed. From the observed results this expectation is obvious from the values of variance ratio for PCA performed on this feature selection.

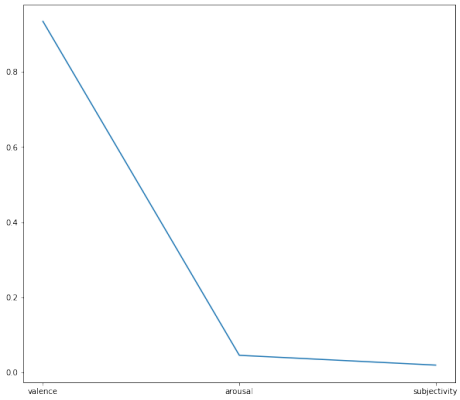


Fig. 36. Explained variance ratio of PCA performed on features of valence, arousal and subjectivity

Compared to previous feature selection (no scraping) where the topic feature was present, the visualization of this selection shows even less diversity which was expected thanks to reduction in features.

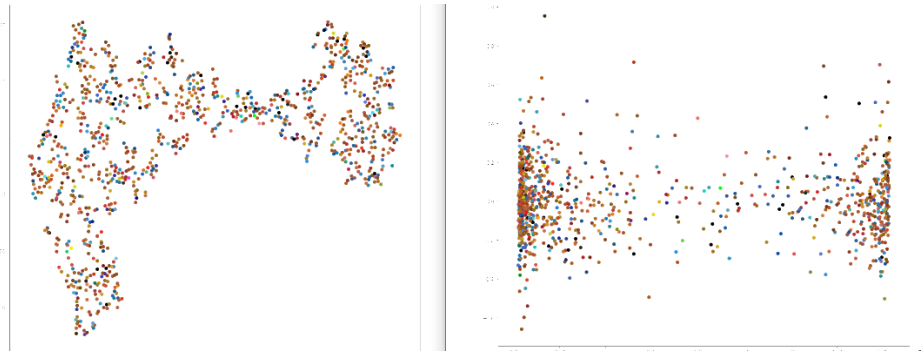


Fig. 37. t-SNE and PCA visualizations of feature set of valence, arousal and subjectivity

7.4 No topic feature

The next selection tries to explore the distribution data without features of topic. As mentioned earlier the value (V) was excluded from the image processing features.

The initial clustering analysis on the raw feature set for this selection shows that the separation of data is less diverse (fewer visual clusters) with similar

definition compared to the complete feature set (all feature selection). This could be explained by exclusion of topic and absence of value-of-color (V) feature.

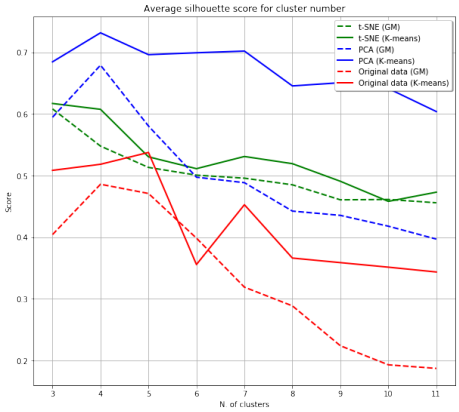


Fig. 38. Plot of silhouette score for feature selection of complete feature set without topic features

Density distribution of the data points in the both t-SNE and PCA map appears to be even with one or two dominating clusters in terms of density.

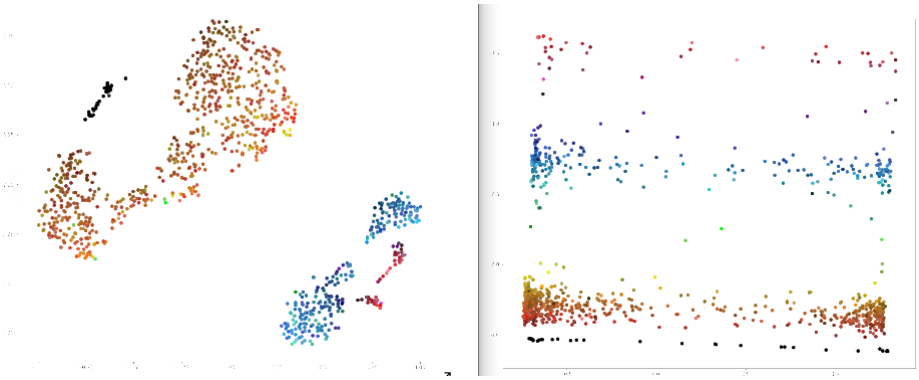


Fig. 39. t-SNE and PCA visualizations of complete feature set without topic features

7.5 Topic and image scraping features

The initial clustering measures indicate that this selection has less defined separation of data points indicated by less radical fallout of the silhouette scores as

the number of clusters increase and overall smoother appearance of the silhouette score curves.

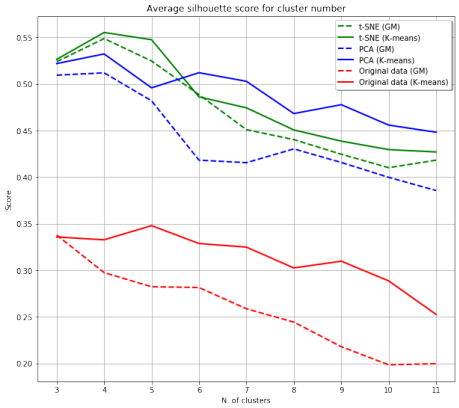


Fig. 40. Plot of silhouette scores for feature set of image scraping and topic

This suggestion is proved in the t-SNE visual map as well.

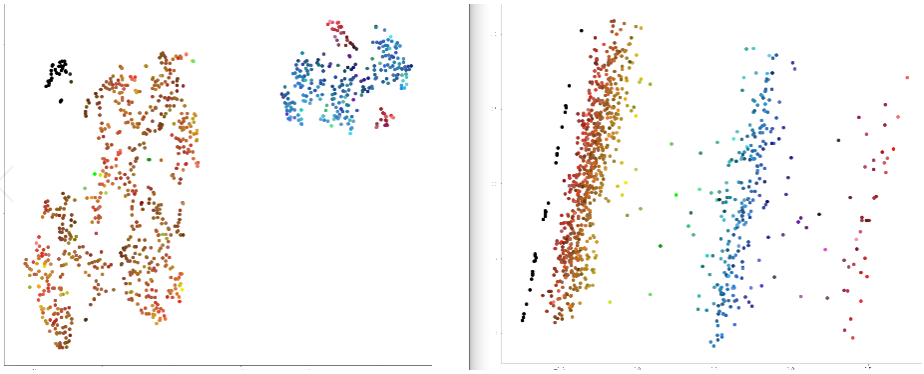


Fig. 41. t-SNE and PCA visualizations of feature set of image scraping and topic

On the other hand the loss of definition is traded for gain in diversity indicating more variance in topic features compared to previous selection of valence, arousal and subjectivity. This fact can be observed in the variance ratio of features from the first feature selection (complete feature set). Similarly to the previous selection of features the PCA map preserves pretty defined separation. However, the density distribution is inclined towards one cluster more than oth-

ers in a very significant manner. This uneven density distribution of PCA map can be also observed in the probability distribution map as well.

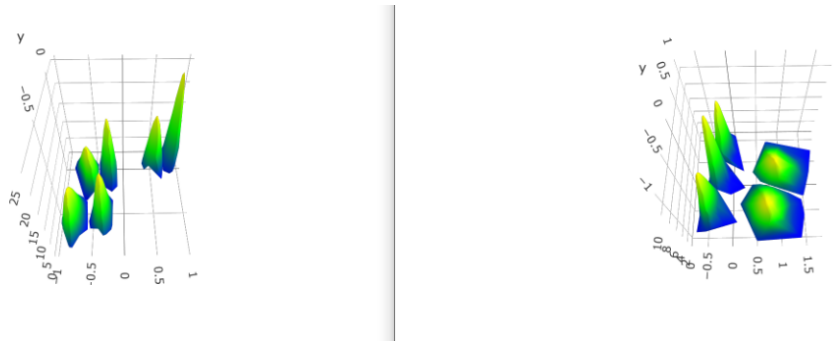


Fig. 42. Visualization of probability distribution function of t-SNE and PCA (respectively) performed on feature set of image scraping and topic using K-means clustering with 6 and 5 clusters (respectively)

8 Evaluation of visualization phase

Based on the final problem statement following null hypothesis was chosen:

H0: The affective visualization of mapped data values does not communicate the data characteristics.

The aim of the evaluation chapter is to use the defined methods and outline an experiment that will seek to reject the above hypotheses. Although there is only one experimental group, the idea of using the null hypothesis is to have clearly stated sentence of the opposite of the aim of the experiment, that could be refused using the combination of the measured results.

The previously defined measured variables were the communicativeness of the visual, and the effectiveness of the algorithm. The communicativeness is divided into communicativeness of the valence, and the communicativeness of the metaphors.

Valence : For measuring the communicatived of the perceived valence, a likert scale question from 1-9, negative to positive, will be given, asking about the perceived mood of the visual. This will be compared with the original valence value, mapped into the scale from 1-9. Measurements will include: Precision - how far from the real value the assigned value was. Correct overall mood - how many from the total number of movies were assigned correct for the positive ones, and how were assigned correct for the negative movies.

Metaphors : Metaphors can be divided into two kinds. Four of them were connected to the data qualities (random, diverse, separate, noisy) and the rest (harmonious, deep, calm, simple, loose, cold, strong, movement) were connected to

the visual perception of the artwork, and taken from the art attributes assessment questionnaire from the background research. This was to evaluate the consistency in perception between people.

For the measurements, each metaphor and its antonym (e.g. deep vs flat, harmonious vs dissonant, etc.) will be given, and the task will be to choose the appropriate word that the participant would use to characterize the visual.

- Data qualities. For the data qualities Krippendorff's alpha (coefficient between 0-1) will be calculated, answering for how many visuals right answer was chosen
- Visual qualities. How much in agreement are the chosen perceived qualities of the visual between the participants.

Additionally, qualitative data in form of keywords that the participant would assign to the visuals will be asked and then compared. This is to approximately compare the consistency in perception when translated freely into words, compared to the consistency when the specific words are given. Although this question might result in quite large range of different words that might be hardly compared as intended, this question also served as an investigative question of how people assign words cognitively to the perceived visuals.

Apart of the overall communicativeness, the second defined measurement was the effectiveness of the created generative art tool. This will be evaluated by asking people to pair the visualized movie with its pair, in a form of another visual, and the correctness of answers will be measured. This is to see if people could guess which visual describes the reviews of the same movie, where the value will inform about the scale of the algorithms effective complexity.

Starting from the final problem statement, this part of the evaluation chapter served for further defining the aimed outcomes of the experiment. In the next chapter, the experiment procedure will be described.

8.1 Experiment procedure

The testing was in a form of an online questionnaire, shared on the social media, mainly through testing groups on Facebook. This allowed to have higher number of participants in a shorter time. Although five experiments were done also in person as a pre-test evaluations, the rest of the participants went through the experiment online. Total number of 79 participants filled the questionnaire. The questionnaire was created as an interactive website, with the answers linked to the google form. The complete questionnaire can be found in the appendix A2. The test consisted of 8 questions together. After the introduction, first question was the pairing exercise, to get people accustomed to how the different visuals might look like. The participant was asked to assign the most similar visual from the six pictures. This question was asked twice for different visual. Next part consisted of looking at a specific picture, and answering three questions related to it. First question was regarding the perceived mood, and the second one was an open question, to assign which first words comes to the mind describing the

visual. The third question consisted of the twelve metaphors and their antonyms, and the task was to choose which characteristics belongs to the visual. The complete list of characteristics can be seen on the figure below.

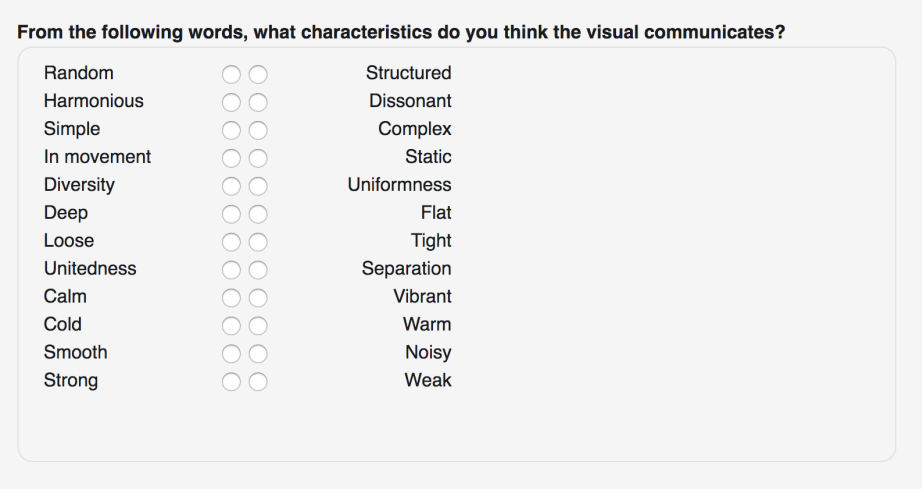


Fig. 43. List of possible characteristics to choose from to describe the visual

The same three questions were repeated for another visual. Doubling the number of same questions allowed to get double the number of visuals evaluated. The given visuals for each participant were randomized, resulting in different visuals given at each experiment. The total number of visuals equals the number of visualized movies which is twenty. The movie signature siblings of the visuals were used only for pairing. After the eight questions, the participant was again thanked for participating, and as a gift the participant could choose a movie which he wishes to have visualized and add his email address. To not affect the results, this was done by surprise and was not shared with the participant beforehand the experiment.

8.2 Communicativeness results

Results of the visual representation phase were divided into two categories, communicativeness and effectiveness of the algorithm. For the communicativeness, the communicated valence, metaphors and keywords were analyzed separately. The raw data from the results can be found in appendix A3. The results were analyzed using Excel and Jupyter Notebook. Since the rating questions were doubled for each participant, together 158 evaluations of visuals was collected for the analysis. One participant filled only half of the questionnaire.

Valence. From answers regarding the scale for rating the mood of the visual, the answers were saved in a number ranging from 0 to 8. This was compared with

the movies original value of valence, which ranged from -1 to 1. The valence values for each movie was mapped into the scale of 0-8. The difference between the two values was calculated, and then the mean difference was obtained, which resulted in 1.9 point difference on the scale. Therefore the precision of the assigned valence and original valence is fluctuating approximately 2 points.

To evaluate how well the overall positiveness and negativeness was mapped, the results were further divided into two groups- for positive movies and negative movies. The results showed that 75 were assigned correctly as positive, from the total number of 135 positive visuals, which is around 57% of correctness. From the negative visuals 22 out of 22 were assigned right, with 100% correctness. That is together, 97 out of 157 correct guesses, approximately 62% of respondents perceived correct value. Interesting to note, there were together 135 positive visuals, but only 76 visuals were perceived as positive, while 81 visuals were perceived as negative, although there were only 22 negative visuals altogether.

Metaphors. Metaphors describing the data qualities and visual qualities were analyzed separately. For all visuals, the correct, or rather the intended quality to be communicated, was assigned in the table to compare with the obtained values. All answers were separated by movies, and analyzed separately for each movie before calculating the overall coefficient. Firstly, the data qualities were analyzed. For the characteristic of random vs structured, the result is coefficient of 0.57. This means 57% of answers assigned the aimed communicated characteristic of random or structured. For diversity vs uniformness, it is coefficient of 0.65, for unitedness vs separation = 0.54 and finally for smoothness vs noisiness coefficient of 0.62 was obtained. The total coefficient of success for assigning all four data qualities is 0.6. The table for the above calculations is on the figure below. The movies with negative valence are marked as dark red.

movie	Total guesses for the pic	Correct random/ structured	Correct diversity/ uniformness	Correct unitedness/ separation	Correct smooth/ noisy
0	9	4	2	3	4
1	10	2	8	2	7
2	9	5	6	5	2
3	12	9	7	5	10
4	5	4	5	3	5
5	7	3	7	3	5
6	6	1	5	3	3
7	6	5	6	5	5
8	*6*	*5*	*5*	*4*	*5*
9	7	3	3	3	5
10	*11*	*5*	*3*	*1*	*2*
11	10	8	7	6	9
12	*7*	*5*	*7*	*7*	*7*
13	2	2	2	2	2
14	4	1	0	2	1
15	8	5	3	6	4
16	12	5	7	8	6
17	6	5	5	4	5
18	8	5	5	6	6
19	9	5	8	4	2
20	3	3	1	3	3
	157	57.32484076	64.96815287	54.14012739	62.42038217

Fig. 44. Correct assigned values of data qualities for each movie

For the visual qualities, the aim was to measure the correlation between the obtained answers from the participants. This was done by firstly calculating the total amount of chosen words for each pair of words, again divided by the movie. Then percentage of mostly chosen word was obtained, for each pair of words, and which was the winning word was marked down. The mean percentage of mostly chosen word for all movies together, and for each pair of words was taken as the final value. The final value representing the agreement rate from the 8 different visual qualities (Harmonious/Dissonant, Deep/Flat, Calm/ Vibrant, Simple/ Complex, Loose/Tight, Cold/ Warm, Strong/Weak, In movement/ Static), was averaged resulting in coefficient of 0.74, with standard deviation being 0.03. The final values for agreement of each pair of words, as well as the mostly chosen column marked as L for left and R for right can be seen on the table below.

	Harmonious/ Dissonant	win	Simple/ Complex	win	Movement /static	win	Deep/ Flat	win	Loose/ Tight	win	Calm/ Vibrant	win	Cold/ Warm	win	Strong/ Weak	win
0	0.555556	R	0.5	R	0.75	L	0.875	L	0.625	R	0.625	R	0.5	R	0.555556	L
1	0.75	L	0.888889	L	0.888889	L	0.75	L	0.5	R	0.9	L	0.625	L	0.777778	L
2	0.5	R	0.571429	L	0.777778	L	0.857143	L	0.5	R	0.555556	L	0.571429	L	0.714286	L
3	0.727273	R	0.7	R	0.909091	L	0.818182	L	0.5	R	0.818182	R	0.7	L	0.8	L
4	1	R	0.8	R	0.8	L	0.8	L	0.6	R	1	R	0.6	R	0.8	L
5	1	L	0.666667	L	1	L	0.6	L	0.8	L	0.6	R	0.6	R	0.75	L
6	0.666667	L	1	L	0.666667	R	0.6	L	0.6	L	0.6	L	0.6	L	0.8	L
7	1	L	0.833333	L	0.6	R	0.6	R	0.6	L	0.666667	L	0.6	L	0.6	L
8	0.833333	L	0.833333	L	0.6	L	0.666667	L	0.833333	L	0.8	L	0.6	L	0.6	R
9	0.6	L	0.6	L	0.666667	L	0.6	L	0.8	R	0.6	R	0.8	R	1	L
10	0.9	R	0.875	R	0.727273	L	0.545455	L	0.636364	R	0.8	R	0.818182	L	0.818182	L
11	0.8	L	0.7	L	0.6	R	0.7	R	0.888889	L	0.8	L	0.555556	R	0.6	L
12	1	R	1	R	0.6	L	0.5	R	0.666667	R	1	R	1	L	0.857143	L
13	1	R	0.5	R	1	L	1	L	0.5	R	1	R	1	L	1	L
14	1	R	1	R	0.75	L	0.75	L	0.75	L	1	R	0.5	R	0.5	R
15	0.625	R	0.625	L	0.625	L	0.5	R	0.625	R	0.625	L	1	L	0.571429	R
16	0.5	R	0.5	R	0.666667	L	0.833333	L	0.75	L	0.583333	L	0.5	R	0.583333	L
17	1	R	0.5	R	1	L	0.5	R	1	L	1	R	0.666667	L	0.666667	R
18	0.666667	L	0.625	R	0.875	L	0.571429	L	0.857143	L	0.571429	L	0.857143	L	0.714286	L
19	0.555556	L	0.625	R	1	L	0.875	L	0.5	R	0.625	R	0.625	R	0.875	L
20	0.5	R	0.5	R	1	L	1	L	1	L	1	L	1	R	1	L
	0.770479		0.706841		0.785859		0.711534		0.692019		0.770008		0.700904		0.742079	

Fig. 45. Agreement on visual qualities for each movie. L or R means which column was the mainly selected one. Left is the first word in the column, and right is the second. The number is the agreement coefficient, representing how much percentage from the total the same word was selected.

On the table it can be observed that whereas for some movies the agreement on particular quality was between all the participants, for some movies the 0.5 value stays for the same amount of answers for both two opposite words.

Keywords. Analysis of the total of 373 assigned words to the visuals was done by calculating the words density. This was done for all movies together. The figure below shows the visualization of the mostly used words, describing the visual.

Fig. 46. The keyword density visualization

8.3 Effectiveness of the algorithm results

The aim of the pairing exercise was to evaluate how many correct movies visuals were assigned. From the total of 158 pairings completed, 78.8% were well matched and 22.2% of the visuals were mismatched. This informs about the effective complexity of the algorithm.

8.4 Summary of results

Multiple observations can be noted from the results of evaluations. Firstly, the measurement of the communicativeness of the visuals mood, which was compared to the original mean valence of the total reviews for each movie will be summarized. Although the average difference between the assigned value and original value is 2 points, which is not too high, the standard deviation from the mean is also 2. This marks quite high variance between the answers correctness. This suggests that the mood was only somewhat communicated. However from the further investigation of the answers, where they were divided only into group of positive and negative, more correctly assigned answers were marked, resulting in 0.57 correctness rate for positive visuals and 1.0 for the negative visuals, meaning all negative visuals were marked as negative. It is important to note that the number of negative and positive visuals were not the same. Since only three analyzed movies had negative valence, and all the visuals were shown in random order, this resulted in only 22 negative visuals shown from the total of 157 questionnaires. That means the sample size for the positive visuals

and negative visuals is quite unequal. Additionally, since many more answers were marked as negative than the total number of negative visuals, this shows that big number of the positive visuals were still perceived as negative, and this influenced the total coefficient of connectedness of both positive and negative, which is 0.62. This suggests that the mapping of valence for the visuals might have been correctly assessed, but adjustment in the range of the ramping needs to be lowered for the positive visuals, to result in higher number of positively affected states. This might affect also the overall precision in correctness of visuals mood on the scale, however this adjustment will need to be only part of the next iteration and evaluation.

Secondly, the communicativeness of the data qualities translated into the visuals metaphors is to be looked into. From the results in the included previous figure (44) it can be seen that while for some movies the correctness is quite high, for some is even lower than half of the answers, meaning that the opposite quality than intended was more chosen. Additionally, some qualities were better mapped than others. For example diversity vs uniformness, compared to unitedness and separation, had much higher success rate of correctly assessed value. Further investigations are needed into which visuals are generally better at communicating the data qualities and which had the lowest ranking, to see if there is any correlation between their form, and to assess needed calibration perhaps of both, the algorithm and the representative elements of the visual. This will be part of the next iteration. For now, the overall rate of correctly assessed words representing the mapped data quality, 0.6 is considered as quite low. However, the results from analyzed data answers are still positively affirmative towards the possibility of the data communication, and although the success rate is low, the qualities can be considered as partially communicated.

From the evaluation of communicativeness of the visual qualities, more positive results were obtained. This confirms that the perception of visual qualities is common among people, and the overall agreement rate is 0.74. However, regarding the previous figure (45) from the results chapter, interesting pattern can be observed, that also for these results, some movies visuals had much higher agreement rate than others. Further investigation into which movies had higher and which lower agreement rate needs to be done, to see if there are any correlations. The results from this part of the evaluation affirms that since there are general patterns in perceiving given visual, the possibility to map the right data qualities into the visual elements is confirmed, although the process of mapping needs further research.

From the analyzed keywords, it can be seen that quite various words were used, with only few repetitions of words such as calm, smooth, chaos, waves, messy, broken, etc. The words for all movies were analyzed together, since the answers divided by movies would result in too low number of words for analysis. From the total of 373 words there was high number of unique words, together 257. The analysis of the keyword density, in comparison to the results of communicativeness of visual qualities, shows, that although from the suggested words there are patterns in perceiving the same qualities across visuals, when people

are asked to write their own words to describe the visual, these answers vary greatly. This can be to numerous of reasons, such as the type of visual attribute that is more obvious for each person is very different, and other differences in perceiving visual qualities. However, when particular visual quality would be asked about, the focus is shifted towards perceiving and evaluating that specific quality of the picture. This was observed also during the in person observations of going through the questionnaire, where the participants would scroll up to re-check the visual and back down to rate the particular visual quality, multiple times for each quality, suggesting that the remembering of the feeling from the visual might not be enough. To conclude the keywords analysis, although there are few more common words, the open questions about the perceived visual quality yielded very different results compared to the intended visual qualities that were communicated. This suggests differences in hierarchy of perception between individuals, and it opens many further questions regarding perception, such as the difficulty to transcribe feelings into words. To further investigate this assumptions, as well as re-question the results from the keyword analysis, analysis using dictionary of synonyms should be done, to see if the amount of unique clusters of synonymous words would be lower.

To evaluate the effectiveness of the algorithm, the pairing exercise served as a measurement of effective complexity, (balance between order and chaos), to see if the visuals for each movie were similar enough to be paired together. This was to see if the visuals are not too random to be assessed. The results of 78.8 % success rate in matching shows that most visuals are well matched. Low number of unsuccessful rates might mean various conclusions. Firstly, that there are few visuals that are difficult to match because of too high similarity and too little distinction between multiple movies, or that the visuals are too different to be matched together. After evaluating which movies were less successful to be paired, it was seen that it was the movies with very similar clusters, see figure 47 below.

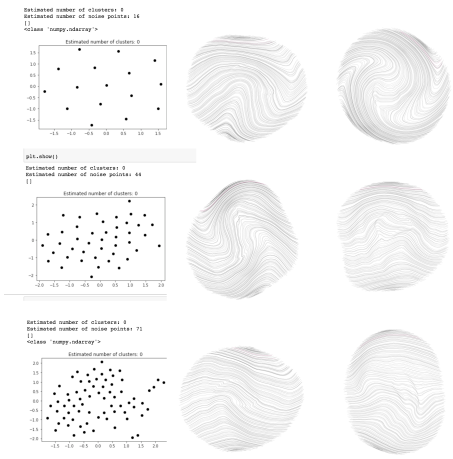


Fig. 47. The issue of few very similar visualization for the pairing exercise. The picture shows three different movies visualized twice each. The results of the reviews t-SNE points clustering on the left shows that this was the issue with the movies that had only noise and no clusters.

This means that it was rather a problem of having too similar movies in the analysis, and the similarity was most probably high enough for matching. However, the evaluation did not include the assessment of whether the results are not too structured and still different enough. Since this was only guessed by the authors and then forgotten about during evaluation, this brings a bias into the results, and the effective complexity can not be properly evaluated from both sides of the scale. Whether or not more randomness is needed in the visuals should be investigated further.

Taking into account the findings from the various parts of the experiment, the results are positive towards the possibility to communicate data characteristics through an affective visualization, and both for the valence and for the data qualities, the intended characteristics were somewhat communicated. This informs about the null hypothesis, and the findings are positive towards rejects it. Although the success rate varied for different movies, and particularly for the movies with positive overall valence the success rate was only below 60%, when considering also findings from the agreement rate of visual qualities, it seems that higher success rates might be achieved by calibrating the mapping in the algorithm. Further iterations and evaluations are necessary to test this assumption, and additional research into mapping needs to be done.

Especially valuable might be to: 1. consider the most common keywords as the main visual elements for mapping, 2. Use the findings and success rates from the data and visual qualities questions, which could be used to inform about the perceived characteristics, and only then connect characteristics to the specific data qualities, that would be only afterwards used for the mapping in

the algorithm. These two suggestions propose an opposite bottom up approach to data mapping, starting from the user evaluating the visual example to inform and test the perception of visual rather then assumings based on the background research and studies of different visuals. Additionally, testing equal number of negatively and positively reviewed movies, including only movies with cluster characteristics distinct enough, and only movies with enough ratings will most likely bring clearer results.

This concludes the evaluation chapter, the procedure, measurements, results and summary of the results together with few insights towards the improvements of the next iteration. The final problem statement, which was, *How can characteristics of complex metadata from multivariate analysis of movie reviews be mapped and communicated through an affective visualization?*, was answered and the tool was evaluated.

9 Discussion

The discussion includes perspectives on the reliability and validity of the overall process and its phases, together with the needed improvements for the future perspective.

The analysis provided a better understanding of quite various fields such as natural language processing algorithms for sentiment and semantic analysis, visual representation approaches (functional and aesthetic), and their state of the art techniques, uses and advantages. Additionally, the most closely related papers from the state of the art were looked into. This had taken the research from the initial problem statement, focused on enhancing the traditional recommender systems and communicating the emotional qualities of the textual data towards two different final problem statements. One was focused on bringing deeper understanding of emotion-content in textual data, and other on communicating resulted metadata through an affective visualization. The aim and preliminary research requirements of each phase were different, and their fields of interest were not necessarily connected, however the choice was to connect into two phases of one research, merging two fields and using results of one phase for another phase. The choice of two separate problem statements influenced the bandwidth of the whole research and the fields that needed to be considered, which might have restricted the possible achievable depth of understanding. Yet it allowed us to combine two distinct phases and their distinct approaches, resulting in a new experimental approach of emotion-content aware textual data communication, through affection. While looking back at the analysis, and what could have been improved, would be rather earlier interest in the visual representation aesthetic approach, since the field of neuroaesthetics and the relationship between design and cognition needs to be further explored. Even though the scope of analysis was large, deficit considered as primary was that the time taken for analysis was longer than planned, resulting in a shorter time for design iterations.

9.1 NLP phase

From the observations of the NLP phase it is apparent that the image scraping had drastic influence on the organisation of the clustering. This can have good and bad implications depending on the validity and identity of the informative value carried by the image processing features.

The image scraping features are extracted using experimental approach and the informative value (meaning) of the features extracted is not proven nor tested. It is only assumed that the resulting color and entropy features representing a review carry some relevant information about the text. The low color diversity may also indicate that the method of color extraction from the scraped images is biased or inaccurate. Other explanation of low color diversity could be present under assumption that the information the color and entropy represent is relevant. In that case the reviews are simply distributed with low diversity in the domain of the color and entropy just like the sentiment feature was distributed (almost binary) in this project. Going into the experimental phase, due to the tight time frame of the project, the latter was assumed.

Similarly to the image scraping features the lexicon based arousal feature is of experimental origin and the informative validity (does the feature really describe arousal of the text) of the feature is not proven nor tested. The relevancy is however assumed while relying mostly on the informative validity of the NRC-VAD lexicon that was used to measure the feature.

9.2 Visualization phase

For the design of the visual representation, although clear design requirements were stated, further investigations of the needs for each requirements would bring higher reliability. From the summarizing results of the design representation evaluation, it is suggested that including one smaller test during the design of metaphors mapping process might have bring more clarity, and increasing the later success rates of communicativeness. Additionally, because of the lack of time resources, multiple smaller choices regarding the overall design aesthetics were based on assumptions and not evaluated from the general publics point of view. Evaluation of the design tool consisted only of self reported measures through an online questionnaire and five in-person questionnaires, without any data source triangulation. Adding further quantitative data gatherings, such as measuring pupil dilation, or measuring brain activity through MRI could bring further insights about the focus of the participants during the questionnaire, and further understand and correlate the effects of the visuals between the people. Also including qualitative data, in form of semi-structured interviews, focused on the perception of the visual could be triangulated with the results from the questionnaire. This opens door into possibly many specific further investigations of each experimental variable, that were in this research all combined into one evaluation, which might yielded imprecise and skewed results. For example, comparison of different communicativeness strategies with different mapping trials, between control and multiple experimental groups, would help to better clarify

and evaluate the potential of this approach and the created tool. To conclude, suggestions for the most important future work regarding affective visual representation are: further research into neuroaesthetics, pre-evaluations regarding visual qualities perception, using multiple experimental groups to evaluate various types of mapping, include data triangulation, compare the perception of qualities from the raw textual data and from the visual, changing the amount of randomness in the algorithm, and more equal sample of movies and rather less movies. Other investigations of the possibility to communicate data through an affect intent could be done for different types of high dimensional data. Although there are many suggestions for further investigations, we believe there is a big potential in merging the functional and aesthetic approach through generative art. The goal of mapping specific data qualities, intended to be communicated in form of feelings through an affective visualization was partially achieved.

10 Conclusion

The aim of the research was to answer two distinct final problem statements: *How can image scraping and arousal lexicon help improve information about emotion-content in textual data for purpose of recommender systems?* and *How can characteristics of complex metadata from multivariate analysis of movie reviews be mapped and communicated through an affective visualization?*. The background research consisting of various fields such as natural language processing algorithms for sentiment and semantic analysis, visual representation approaches (functional and aesthetic), and their state of the art techniques, uses and advantages. The design consisted of two separate phases, NLP phase and visual representation.

10.1 NLP phase

In this phase it was attempted to retrieve experimental features of arousal, entropy and color of textual data that could hold an informative value about emotional print of the text along with the traditional features of NLP. The observations supported the significance of the influence of the experimental features on the dataset but user tests of the informative relevancy of the experimental features together with further technical adjustments is highly suggested to ensure potential practical implementation into recommender systems. It is believed that this approach could have interesting impact on the current recommender systems.

10.2 Visualization phase

Results from the processing of textual data of the movie reviews dataset that was chosen, were then passed into the next phase as metadata. The visualization phase focused on creation of affective visualizations, where generative algorithms were taken as a tool for making affective visuals based on data-driven rules. The

passed metadata from the previous phase were further analyzed, and specific data qualities were chosen to be communicated. Those were valence, representing the overall mood of the reviews for a given movie, and various qualities of the clusters from the t-SNE point map generated during the first phase. These data qualities were then mapped into visual elements of the final visualization, by using generative algorithm, specifically attractors and Perlin noise. 20 different movies were visualized twice, to compare and assess the effective complexity of the artwork. For evaluating the visual form, two aspects were looked into, its communicativeness of the data and visual qualities, and effectiveness of the algorithm. The evaluation procedure consisted of an online questionnaire with 6 questions regarding two of the randomly given visuals for each participants, and two matching exercises. Total number of 79 participants completed the questionnaire. Although further investigations into calibrating the mapping are needed to evaluate whether higher success rates for communicativeness could be obtained, the results confirmed the opportunity to communicate data characteristics through an affective visualization For both, the valence and for the data qualities, the intended characteristics were somewhat communicated. This answers the second final problem statement, and the aim of mapping and communicating specific data qualities, through an affective visualization was partially achieved.

We believe there is a big potential in merging the functional and aesthetic approach, through the use of NLP processing and generative art, such, as investigated in this paper, for the aim of understanding and communicating textual data qualities through an affect intend.

APPENDIX A1

Final Visualizations

APPENDIX A2

Evaluation Questionnaire

APPENDIX A3

Evaluation Results

APPENDIX A4

Implementation

APPENDIX B

Image scraper code

APPENDIX C

Entropy code

APPENDIX D

Preprocessing code

APPENDIX E

NLP code

References

1. Part of sentence tagging (pos), https://www.nltk.org/book/ch05.html
2. Ahmad, U., Zahid, A., Shoaib, M., AlAmri, A.: Harvis: An integrated social media content analysis framework for youtube platform. Information Systems **69**, 25–39 (2017)
3. Bohnacker, H., Gross, B., Laub, J., Lazzeroni, C.: Generative design: visualize, program, and create with processing. Princeton Architectural Press (2012)
4. Brown, S., Gao, X., Tisdelle, L., Eickhoff, S.B., Liotti, M.: Naturalizing aesthetics: brain areas for aesthetic appraisal across sensory modalities. Neuroimage **58**(1), 250–258 (2011)
5. Cairo, A.: The Functional Art: An introduction to information graphics and visualization. New Riders (2012)
6. Cairo, A.: The truthful art: Data, charts, and maps for communication. New Riders (2016)
7. Cawthon, N., Moere, A.V.: Qualities of perceived aesthetic in data visualization (2007)
8. Chatterjee, A.: The neuropsychology of visual art: Conferring capacity. International review of neurobiology **74**, 39–49 (2006)
9. Chatterjee, A., Vartanian, O.: Neuroaesthetics. Trends in cognitive sciences **18**(7), 370–375 (2014)
10. Chatterjee, A., Widick, P., Sternschein, R., Smith, W.B., Bromberger, B.: The assessment of art attributes. Empirical Studies of the Arts **28**(2), 207–222 (2010)
11. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: a real-world web image database from national university of singapore. In: Proceedings of the ACM international conference on image and video retrieval. p. 48. ACM (2009)
12. Corp., A.: Amazon mechanical turk, https://www.mturk.com/
13. De Gemmis, M., Lops, P., Semeraro, G., Basile, P.: Integrating tags in a semantic content-based recommender. In: Proceedings of the 2008 ACM conference on Recommender systems. pp. 163–170. ACM (2008)
14. Desmet, B., Hoste, V.: Emotion detection in suicide notes. Expert Systems with Applications **40**(16), 6351–6358 (2013)
15. Ekman, P.: An argument for basic emotions. Cognition & emotion **6**(3-4), 169–200 (1992)
16. Feng, C., Bartram, L., Gromala, D.: Beyond data: Abstract motionscapes as affective visualization. Leonardo **50**(2), 205–206 (2017)
17. Feng, C., Bartram, L., Riecke, B.E.: Evaluating affective features of 3d motionscapes. In: Proceedings of the ACM Symposium on Applied Perception. pp. 23–30. ACM (2014)
18. Friendly, M., Denis, D.J.: Milestones in the history of thematic cartography, statistical graphics, and data visualization. URL http://www. datavis. ca/milestones **32**, 13 (2001)
19. Galanter, P.: What is generative art? complexity theory as a context for art theory. In: In GA2003–6th Generative Art Conference. Citeseer (2003)
20. Gedikli, F.: Recommender systems and the social web: Leveraging tagging data for recommender systems. Springer Science & Business Media (2013)
21. Graves, M.E.: The Art of Color and Design. Psychological Corporation (1941)
22. Ha, H., Hwang, W., Bae, S., Choi, H., Han, H., Kim, G.N., Lee, K.: Cosmovis: Semantic network visualization by using sentiment words of movie review data. In: 2015 19th International Conference on Information Visualisation. pp. 436–443. IEEE (2015)

23. Healey, C.G., Booth, K.S., Enns, J.T.: Visualizing real-time multivariate data using preattentive processing. ACM Transactions on Modeling and Computer Simulation (TOMACS) **5**(3), 190–221 (1995)
24. Honnibal, M.: spacy industrial-strength natural language processing in python, https://spacy.io/
25. Hope, C., Ryan, J.C.: Digital Arts: An Introduction to New Media. Bloomsbury Publishing USA (2014)
26. Huang, E.H., Socher, R., Manning, C.D., Ng, A.Y.: Improving word representations via global context and multiple word prototypes. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. pp. 873–882. Association for Computational Linguistics (2012)
27. Hupka, R.B., Zaleski, Z., Otto, J., Reidl, L., Tarabrina, N.V.: The colors of anger, envy, fear, and jealousy: A cross-cultural study. Journal of cross-cultural psychology **28**(2), 156–171 (1997)
28. Hutchins, J.: The history of machine translation in a nutshell. Retrieved December **20**, 2009 (2005)
29. Hutto, C.J., Gilbert, E.: Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Eighth international AAAI conference on weblogs and social media (2014)
30. Kao, E.C.C., Liu, C.C., Yang, T.H., Hsieh, C.T., Soo, V.W.: Towards text-based emotion detection a survey and possible improvements. In: 2009 International Conference on Information Management and Engineering. pp. 70–74. IEEE (2009)
31. Klee, P.: The Notebooks of Paul Klee: The thinking eye, vol. 15. G. Wittenborn (1964)
32. Kret, M.E., Bocanegra, B.R.: Adaptive hot cognition: How emotion drives information processing and cognition steers affective processing. Frontiers in psychology **7**, 1920 (2016)
33. Lang, A.: Aesthetics in information visualization. Trends in information visualization **8** (2009)
34. Li, Q.: Data visualization as creative art practice. Visual Communication **17**(3), 299–312 (2018)
35. Liu, N.: word2color, https://github.com/nelson-liu/word2color
36. Lockyer, M., Bartram, L.: Affective motion textures. Computers & Graphics **36**(6), 776–790 (2012)
37. Loria, S.: Simplified text processing, https://textblob.readthedocs.io/en/dev/
38. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1. pp. 142–150. Association for Computational Linguistics (2011)
39. Manning, C.D., Manning, C.D., Schütze, H.: Foundations of statistical natural language processing. MIT press (1999)
40. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The stanford corenlp natural language processing toolkit (2014)
41. MATSON, P., MUELLER, M., TIPTON, E.: Affective data visualization: A preliminary study. In: International Symposium on Affective Science and Engineering. pp. 1–6. Japan Society of Kansei Engineering (2018)
42. Mikolov, T., Le, Q.V., Sutskever, I.: Exploiting similarities among languages for machine translation. arXiv preprint arXiv:1309.4168 (2013)
43. Mischel, W., Shoda, Y.: A cognitive-affective system theory of personality: reconceptualizing situations, dispositions, dynamics, and invariance in personality structure. Psychological review **102**(2), 246 (1995)

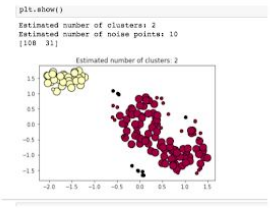
44. Moere, A.V.: Aesthetic data visualization as a resource for educating creative design. In: Computer-Aided Architectural Design Futures (CAADFutures) 2007, pp. 71–84. Springer (2007)
45. Mohammad, S.: Colourful language: Measuring word-colour associations. In: Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics. pp. 97–106. Association for Computational Linguistics (2011)
46. Mohammad, S.M.: Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In: Proceedings of The Annual Conference of the Association for Computational Linguistics (ACL). Melbourne, Australia (2018)
47. Mulcrone, K.: Detecting emotion in text (2012)
48. Murray, S.: Interactive data visualization for the web: an introduction to designing with. " O'Reilly Media, Inc." (2017)
49. Nalbantian, S.: Neuroaesthetics: Neuroscientific theory and illustration from the arts. Interdisciplinary Science Reviews **33**(4), 357–368 (2008)
50. Nasser, A., Hamad, D., Nasr, C.: Visualization methods for exploratory data analysis. In: 2006 2nd International Conference on Information & Communication Technologies. vol. 1, pp. 1379–1384. IEEE (2006)
51. Nell, V.: Lost in a book: The psychology of reading for pleasure. Yale University Press (1988)
52. Neviarouskaya, A., Prendinger, H., Ishizuka, M.: Emoheart: conveying emotions in second life based on affect sensing from text. Advances in Human-Computer Interaction **2010**, 1 (2010)
53. Ngoc, P.T., Yoo, M.: The lexicon-based sentiment analysis for fan page ranking in facebook. In: The International Conference on Information Networking 2014 (ICOIN2014). pp. 444–448. IEEE (2014)
54. Norman, D.A.: Emotional design: Why we love (or hate) everyday things. Basic Civitas Books (2004)
55. Pearson, M.: Generative Art. Manning Publications Co. (2011)
56. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
57. Perikos, I., Hatzilygeroudis, I.: Recognizing emotions in text using ensemble of classifiers. Engineering Applications of Artificial Intelligence **51**, 191–201 (2016)
58. Picard, R.W.: Affective computing mit press. Cambridge, Massachsusetts (1997)
59. Posner, J., Russell, J.A., Peterson, B.S.: The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. Development and psychopathology **17**(3), 715–734 (2005)
60. Ramirez Gaviria, A.: When is information visualization art? determining the critical criteria. Leonardo **41**(5), 479–482 (2008)
61. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender systems handbook, pp. 1–35. Springer (2011)
62. Russell, J.A.: A circumplex model of affect. Journal of personality and social psychology **39**(6), 1161 (1980)
63. Sack, W.: Aesthetics of information visualization. Context providers: Conditions of meaning in media arts pp. 123–50 (2011)
64. Schnall, S.: Affect, mood and emotions. Social and emotional aspect of learning pp. 59–64 (2010)
65. Sebastiani, F.: Machine learning in automated text categorization. ACM computing surveys (CSUR) **34**(1), 1–47 (2002)
66. Seyeditabari, A., Zadrozny, W.: Can word embeddings help find latent emotions in text? preliminary results. In: The Thirtieth International Flairs Conference (2017)

67. Shaheen, S., El-Hajj, W., Hajj, H., Elbassuoni, S.: Emotion recognition from text based on automatically generated rules. In: 2014 IEEE International Conference on Data Mining Workshop. pp. 383–392. IEEE (2014)
68. Shivhare, S.N., Khethawat, S.: Emotion detection from text. CoRR **abs/1205.4944** (2012), http://arxiv.org/abs/1205.4944
69. Skilling, J., Bryan, R.: Maximum entropy image reconstruction-general algorithm. Monthly notices of the royal astronomical society **211**, 111 (1984)
70. Steven Bird, Edward Loper, E.K.: Natural language toolkit, https://www.nltk.org/
71. Strapparava, C., Valitutti, A., et al.: Wordnet affect: an affective extension of wordnet. In: Lrec. vol. 4, p. 40. Citeseer (2004)
72. Tateosian, L.G., Healey, C.G., Enns, J.T.: Engaging viewers through nonphotorealistic visualizations. In: Proceedings of the 5th international symposium on Nonphotorealistic animation and rendering. pp. 93–102. ACM (2007)
73. Tellex, S., Katz, B., Lin, J., Fernandes, A., Marton, G.: Quantitative evaluation of passage retrieval algorithms for question answering. In: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval. pp. 41–47. ACM (2003)
74. Terwogt, M.M., Hoeksma, J.B.: Colors and emotions: Preferences and combinations. The Journal of general psychology **122**(1), 5–17 (1995)
75. Tufte, E.R.: The visual display of quantitative information, vol. 2. Graphics press Cheshire, CT (2001)
76. Tufte, E.R., Robins, D.: Visual explanations. Graphics Cheshire (1997)
77. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics. pp. 384–394. Association for Computational Linguistics (2010)
78. Vartanian, O., Goel, V.: Neuroanatomical correlates of aesthetic preference for paintings. Neuroreport **15**(5), 893–897 (2004)
79. Viégas, F.B., Wattenberg, M.: Artistic data visualization: Beyond visual analytics. In: International Conference on Online Communities and Social Computing. pp. 182–191. Springer (2007)
80. Vrehuuvrek, R., Sojka, P.: Gensimstatistical semantics in python. statistical semantics; gensim; Python; LDA; SVD (2011)
81. Ware, C.: Information visualization: perception for design. Elsevier (2012)
82. Warriner, A.B., Kuperman, V., Brysbaert, M.: Norms of valence, arousal, and dominance for 13,915 english lemmas. Behavior research methods **45**(4), 1191–1207 (2013)
83. You, Y.S., Lee, S., Kim, J.: Design and development of visualization tool for movie review and sentiment analysis. In: Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory. pp. 117–123. ACM (2016)
84. Zhai, C.: Fast statistical parsing of noun phrases for document indexing. In: Proceedings of the fifth conference on Applied natural language processing. pp. 312–319. Association for Computational Linguistics (1997)

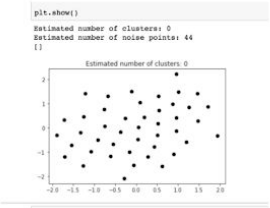
Appendix A1

Results of metadata analysis and final visualizations for each movie

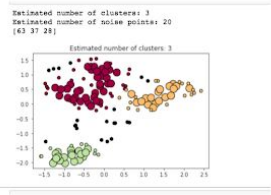
Harry Potter and the Sorcerer's Stone (2001)



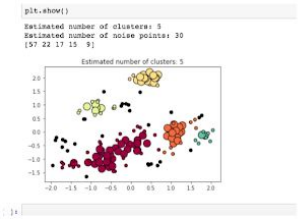
Three Wishes for Cinderella (1973)

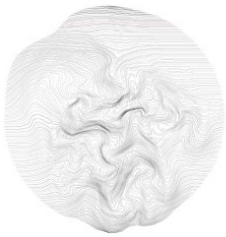


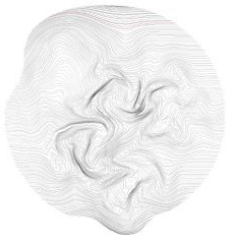
The Matrix (1999)



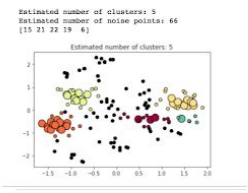
Shazam! (2019)

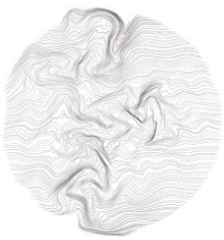




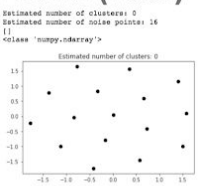


Toy Story (1995)

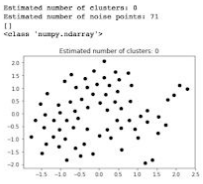




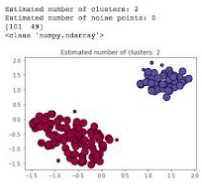
Cosy Dens (1999)



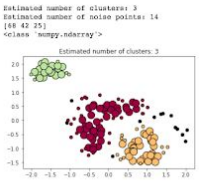
Nim's Island (2008)



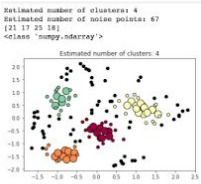
Brüno (2009)

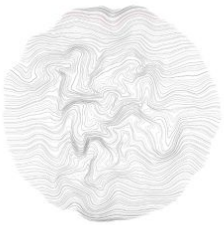


Dances with Wolves (1990)

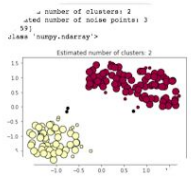


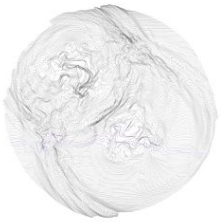
Rio Bravo (1959)

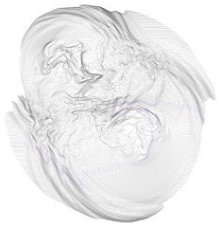




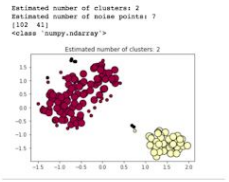
Alone in the Dark (2005)



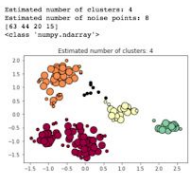


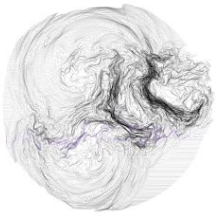


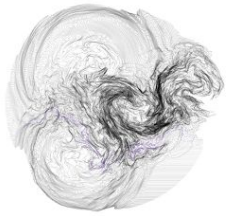
Catwoman (2004)



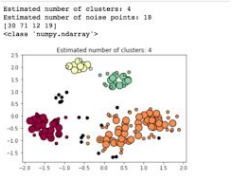
A Wrinkle in Time (2018)

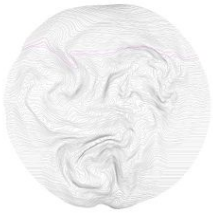


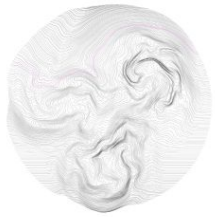




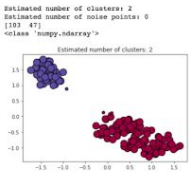
What Men Want (2019)

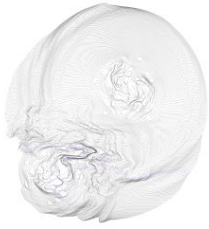


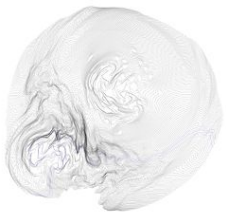




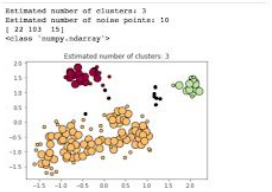
Disaster Movie (2008)



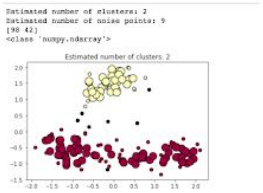


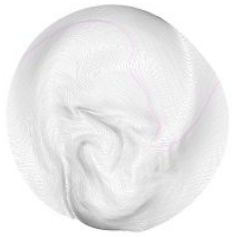


The Emoji Movie (2017)

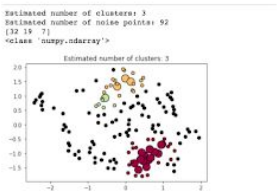


Mary Shelley's Frankenstein (1994)

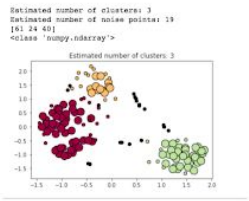


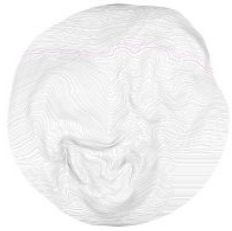


Life of Pi (2012)

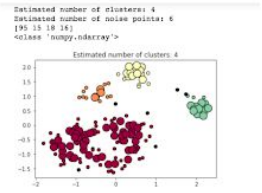


Titanic (1997)

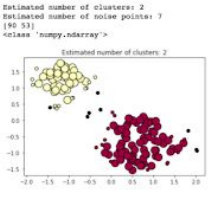




Free Solo (2018)



Sex and the City 2 (2010)



Appendix A2
Evaluation questionnaire

Data Vibes

Dear participant,

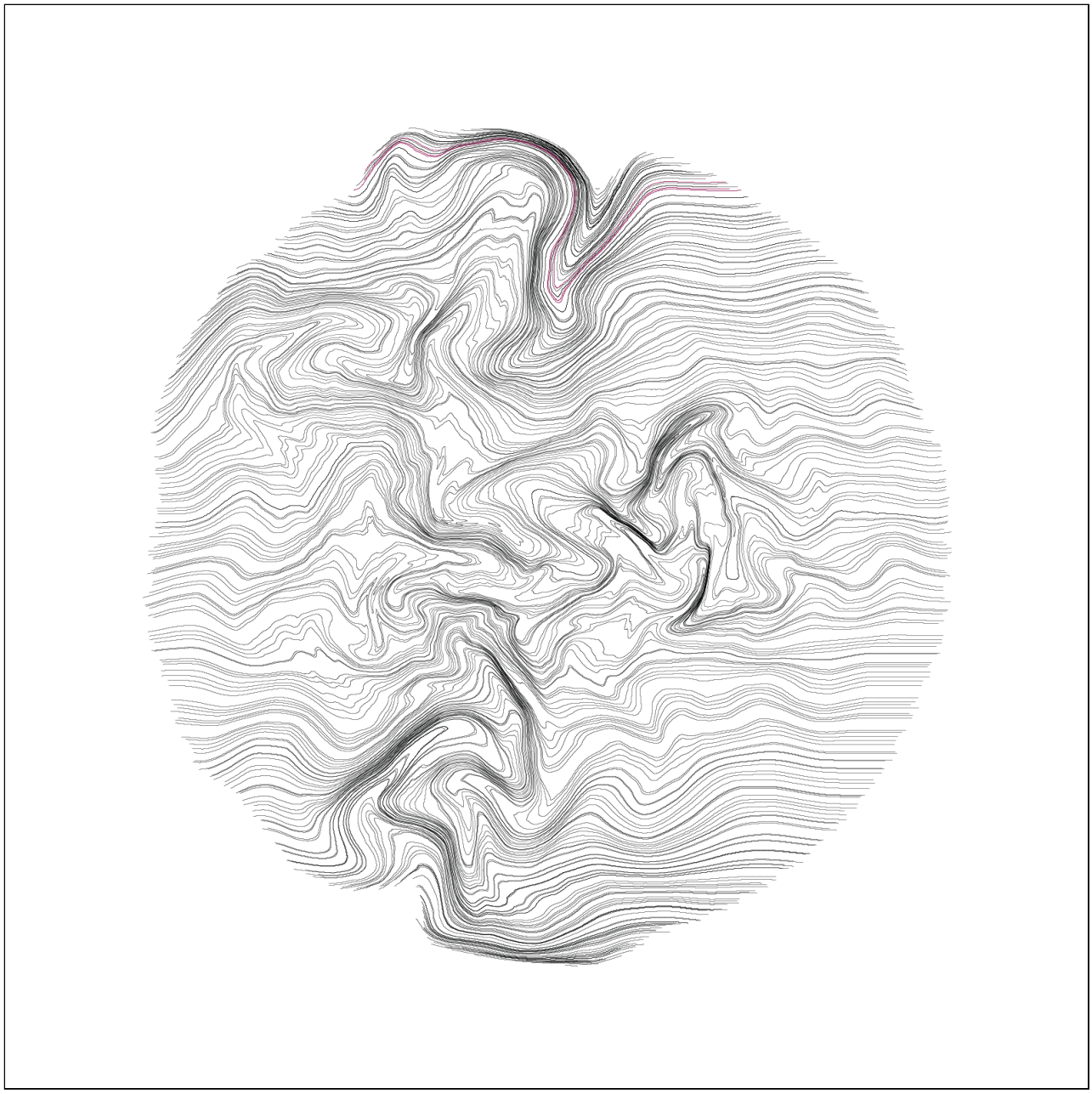
Thanks for being part of this experiment!

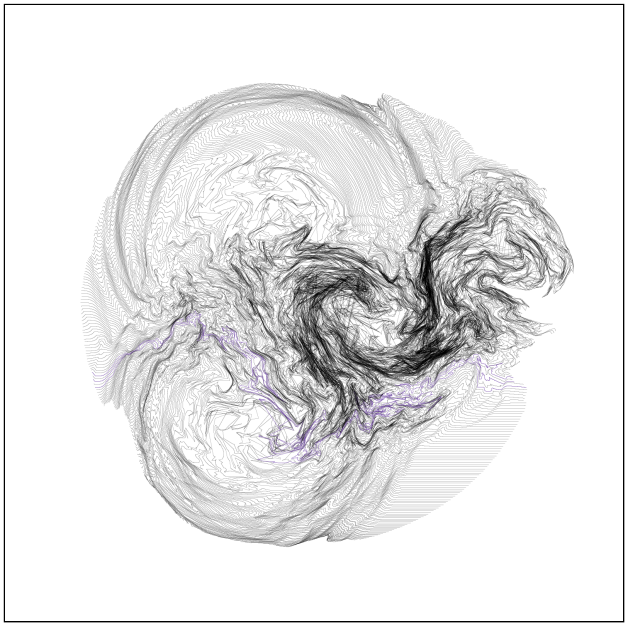
This is a short interactive questionnaire, where you can help us in questioning our hypothesis. During our project we played around with analyzing IMDb reviews, with focus on the text characteristics such as its sentiment, topic, subjectivity and assigned color value. Afterwards, the data of analyzed reviews, led us into idea of creating a “movie signature” visualization. The following pictures are communicating the results, where each picture represents reviews of one movie.

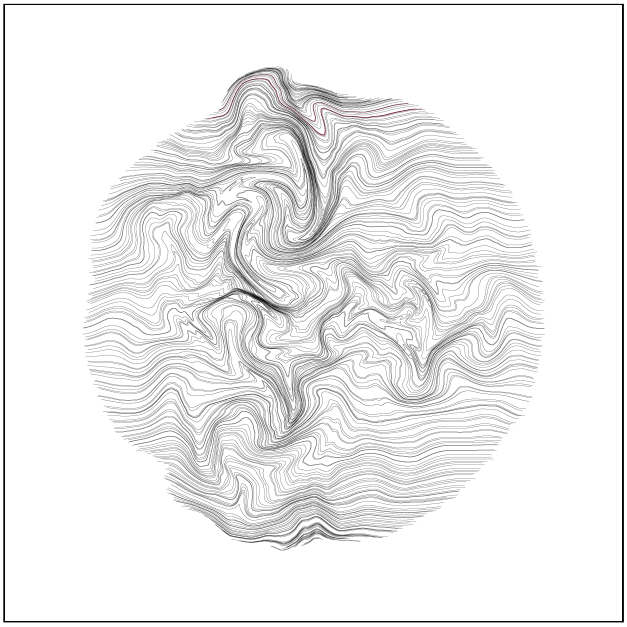
How are they representing them? That’s part of the secret :).
Let’s start.

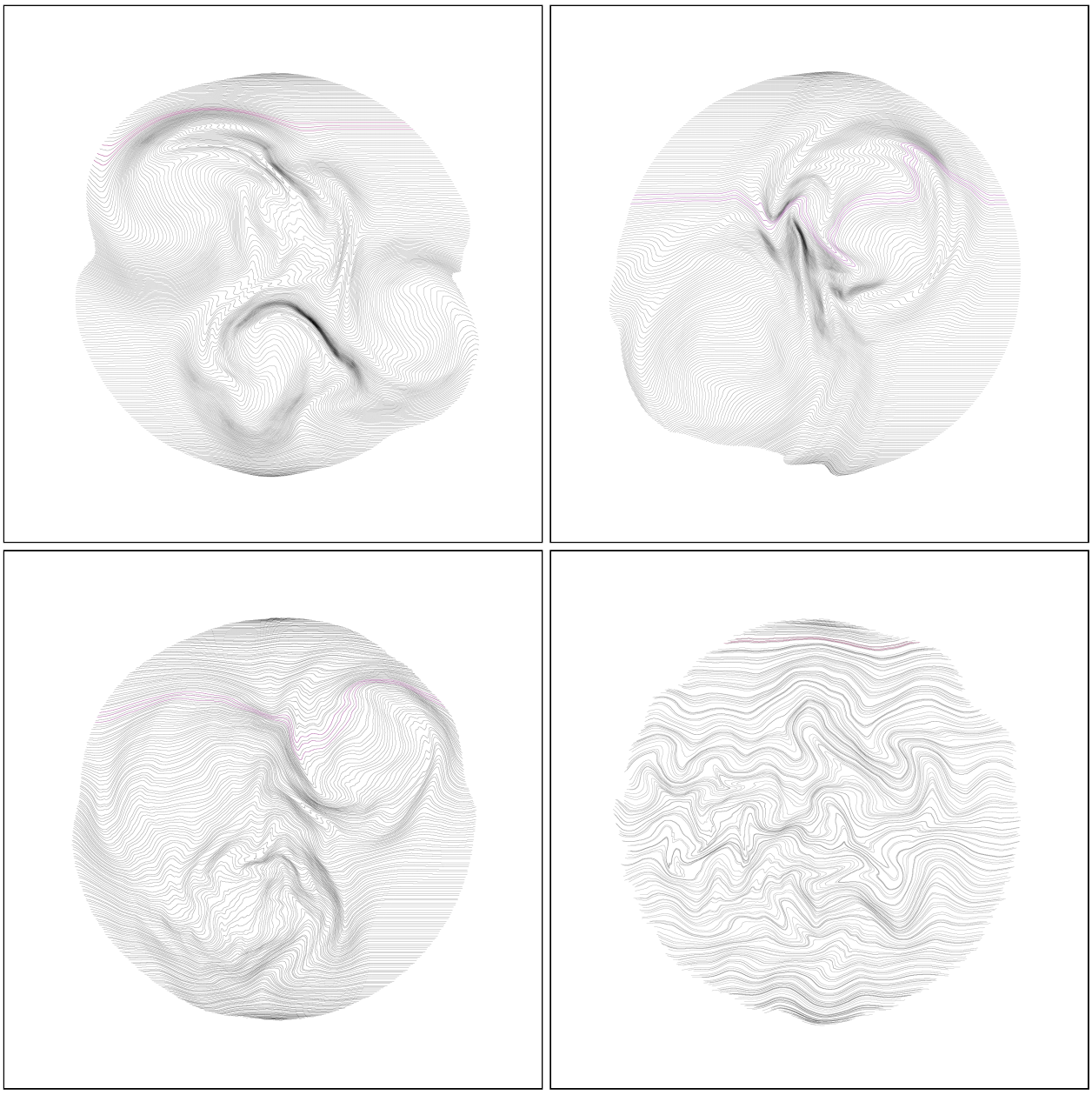
1/4

Looking at the following picture and its features (balance, depth, curvature, complexity, animacy, emotion, etc.) can you find a picture that you feel like is most similar ?

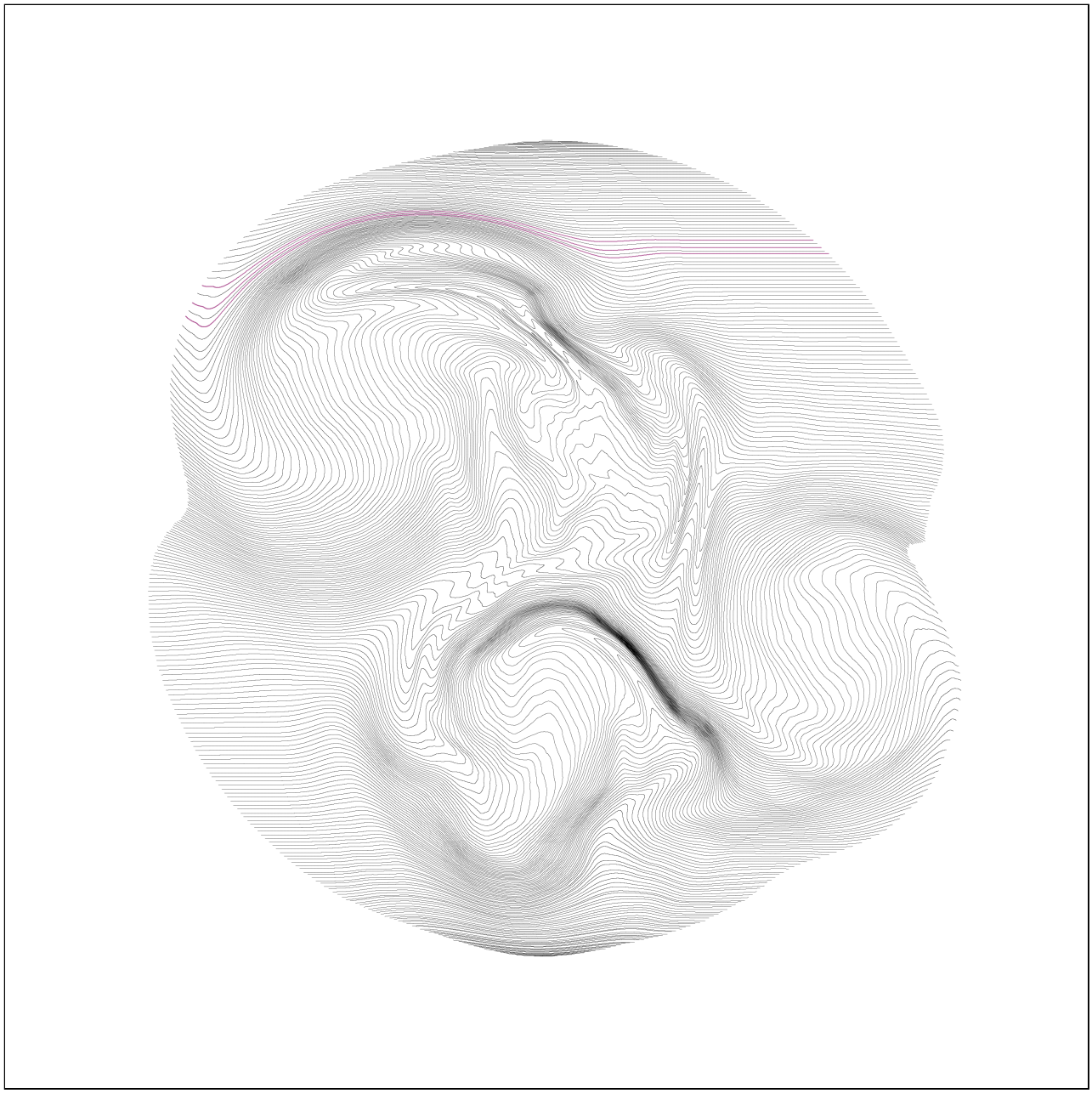


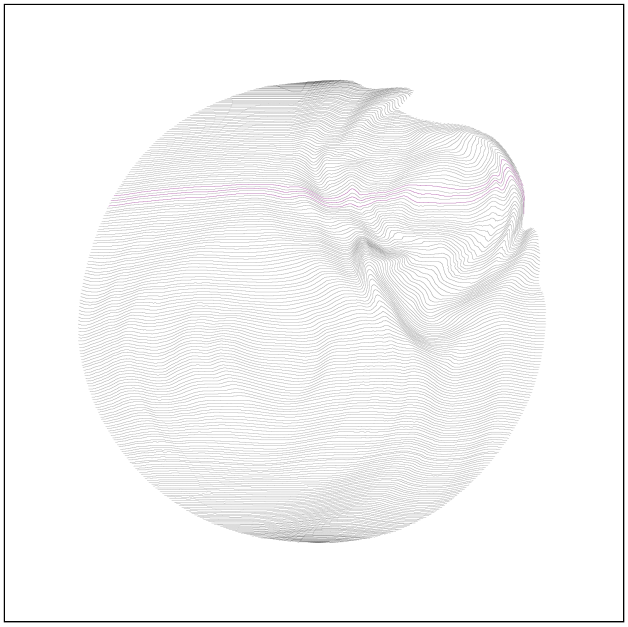


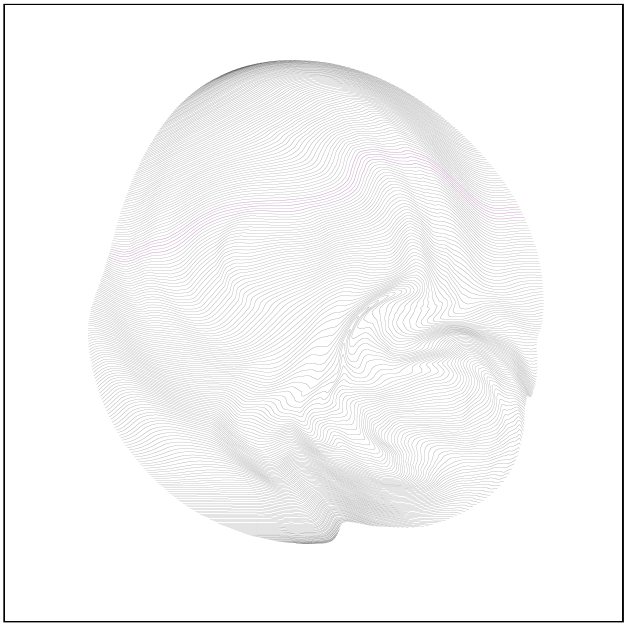


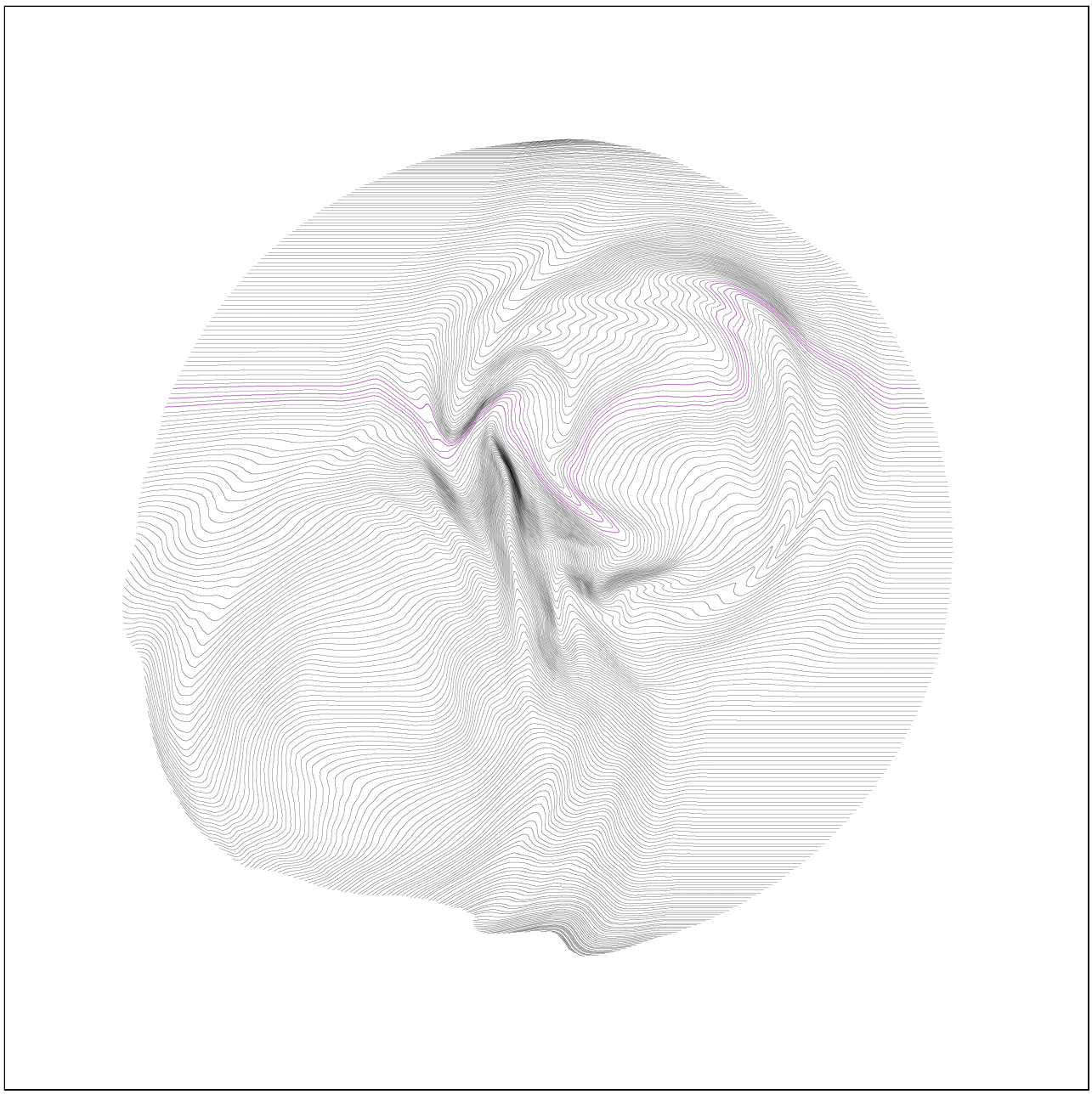


Looking at the following picture and its features (balance, depth, curvature, complexity, animacy, emotion, etc.) can you find a picture that you feel like is most similar ?









On a scale between positive and negative, how do you perceive the mood of the visual?

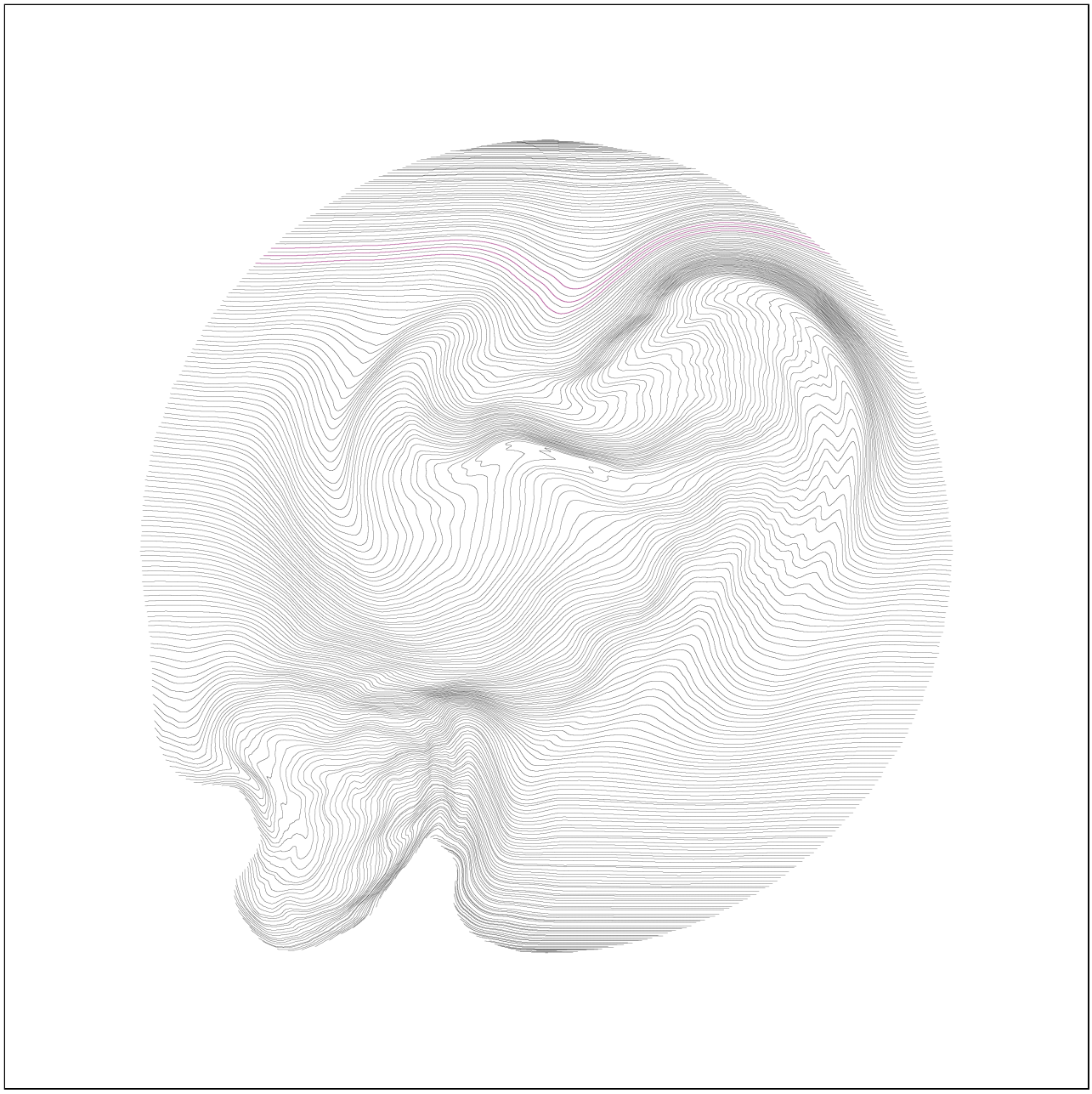
Negative Positive

What first words comes to your mind describing the visual?

Type the words here...

From the following words, what characteristics do you think the visual communicates?

Random	◯ ◯	Structured
Harmonious	◯ ◯	Dissonant
Simple	◯ ◯	Complex
In movement	◯ ◯	Static
Diversity	◯ ◯	Uniformness
Deep	◯ ◯	Flat
Loose	◯ ◯	Tight
Unitedness	◯ ◯	Separation
Calm	◯ ◯	Vibrant
Cold	◯ ◯	Warm
Smooth	◯ ◯	Noisy
Strong	◯ ◯	Weak



On a scale between positive and negative, how do you perceive the mood of the visual?

Negative Positive

What first words comes to your mind describing the visual?

Type the words here...

From the following words, what characteristics do you think the visual communicates?

Random	◯ ◯	Structured
Harmonious	◯ ◯	Dissonant
Simple	◯ ◯	Complex
In movement	◯ ◯	Static
Diversity	◯ ◯	Uniformness
Deep	◯ ◯	Flat
Loose	◯ ◯	Tight
Unitedness	◯ ◯	Separation
Calm	◯ ◯	Vibrant
Cold	◯ ◯	Warm
Smooth	◯ ◯	Noisy
Strong	◯ ◯	Weak

That’s it. Thanks for participating!

PS: If you wish to receive signature of your favourite movie, include your email and name of the movie and you will get a mail from me after the end of this research:

Email Movie name

If you have any comments, please let us know:

Type any comments you have here...

Submit

Appendix A3

Results

Timestamp	Correct1	Main1	Selected1	Correct2	MainPic2
5/20/2019 8:02:55	YES	71	72	YES	01
5/20/2019 8:38:23	YES	121	122	YES	11
5/20/2019 8:41:57	YES	41	42	YES	01
5/20/2019 8:49:39	YES	91	92	YES	121
5/20/2019 9:39:07	YES	71	72	YES	111
5/20/2019 9:39:16	NO	51	62	YES	81
5/20/2019 9:44:17	YES	31	32	NO	151
5/20/2019 10:21:17	YES	201	202	NO	111
5/20/2019 10:21:26	NO	111	61	YES	31
5/20/2019 10:22:14	NO	21	122	YES	61
5/20/2019 10:23:25	NO	31	181	NO	91
5/20/2019 10:26:19	NO	51	62	YES	11
5/20/2019 10:26:32	NO	161	121	NO	191
5/20/2019 10:33:17	YES	171	172	YES	21
5/20/2019 10:38:51	YES	81	82	YES	101
5/20/2019 10:41:26	YES	41	42	YES	11
5/20/2019 10:42:25	YES	161	162	YES	141
5/20/2019 10:42:27	YES	81	82	YES	131
5/20/2019 10:43:14	YES	61	62	YES	201
5/20/2019 10:43:33	YES	201	202	YES	51
5/20/2019 10:55:01	YES	41	42	YES	31
5/20/2019 10:59:06	YES	131	132	NO	11
5/20/2019 11:06:53	NO	61	71	YES	191
5/20/2019 11:10:30	NO	161	21	YES	131
5/20/2019 11:36:14	YES	101	102	NO	131
5/20/2019 11:39:49	YES	171	172	YES	91
5/20/2019 11:57:18	YES	61	62	NO	161
5/20/2019 12:12:42	YES	51	52	YES	81
5/20/2019 12:17:13	YES	01	02	YES	171
5/20/2019 12:44:42	YES	161	162	YES	91
5/20/2019 12:57:38	YES	41	42	YES	71

SelectedPic2	RatePic1	Mood1	Words1	Pairs1 [Row 1]	Pairs1 [Row 2]
02	81	6	calm, smooth	Column 1	Column 1
12	21	6	fluffy cotton	none	none
02	152	5	mysterious, hidden secret	Column 2	Column 2
122	71	5	nothing happens	Column 2	Column 1
112	61	5	Looks like a fingerprint or	Column 2	Column 1
82	172	3	crooked	Column 2	Column 2
162	42	3	Dark, vibrant	Column 1	Column 2
151	12	5	wood, map, military, plane	Column 2	Column 1
32	12	6	Focus	Column 2	Column 2
62	82	5	Baby, calm	Column 2	Column 1
31	112	7	Calming, whole	Column 2	Column 1
12	151	3	paper baby	Column 1	Column 2
121	71	1	Moody, static	none	none
22	72	4	Calm, cold, melancholic, s	Column 2	Column 1
102	192	7	smooth	Column 2	Column 1
12	101	1	Anger	Column 1	Column 2
142	42	2	Disruption, frequency, not	Column 1	Column 2
132	182	4	calm	Column 2	Column 1
202	111	2	Ueven and starting chaos	Column 2	Column 2
52	32	2	erratic	Column 1	Column 2
32	161	7	harmonious calm waves v	Column 2	Column 1
51	182	6	calm but energetic	Column 2	Column 1
192	161	3	none	Column 2	Column 2
132	192	2	Opposites, fight	Column 1	Column 2
121	02	5	none	Column 2	Column 1
92	121	1	Agression	Column 2	Column 2
72	101	2	Messy, outliers	Column 1	Column 2
82	101	1	disturbed, in motion	none	none
172	31	2	Wringed out	Column 2	Column 1
92	31	1	Grumpy	Column 1	Column 2
72	191	1	anger	Column 1	Column 2

Pairs1 [Row 3]	Pairs1 [Row 4]	Pairs1 [Row 5]	Pairs1 [Row 6]	Pairs1 [Row 7]	Pairs1 [Row 8]
Column 1	Column 1	Column 2	Column 1	Column 1	Column 2
none	Column 1	none	none	Column 1	none
Column 1	Column 2	Column 1	Column 2	Column 2	Column 1
Column 1	Column 2	Column 2	Column 2	Column 1	Column 1
Column 1	Column 2	Column 2	Column 1	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	Column 1	Column 1
Column 1	Column 1	Column 2	Column 1	Column 2	Column 1
Column 1	Column 1	Column 2	Column 1	Column 1	Column 1
Column 1	Column 2	Column 2	Column 1	Column 1	Column 1
Column 1	Column 1	Column 2	Column 1	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	Column 1	Column 1
Column 2	Column 1	none	none	none	none
Column 1	Column 2	Column 2	Column 2	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 2
Column 2	Column 1	Column 1	Column 1	Column 1	Column 2
Column 2	Column 1	Column 1	Column 1	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 1
Column 2	Column 1	Column 1	Column 2	Column 1	Column 2
Column 2	Column 1	Column 1	Column 1	Column 1	Column 2
Column 1	Column 1	Column 2	Column 1	Column 1	Column 1
Column 1	Column 1	Column 1	Column 1	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 2
Column 1	Column 2	Column 2	Column 1	Column 2	Column 1
none	none	none	none	none	none
Column 1	Column 2	Column 1	Column 2	Column 1	Column 1
none	Column 1	none	Column 1	Column 2	Column 2
Column 1	Column 1	Column 2	Column 1	Column 2	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 1
Column 2	Column 1	Column 1	Column 1	Column 1	Column 2

Pairs1 [Row 9]	Pairs1 [Row 10]	Pairs1 [Row 11]	Pairs1 [Row 12]	RatePic2	Mood2
Column 1	Column 1	Column 1	Column 1	91	2
Column 1	none	Column 1	none	52	7
Column 1	Column 1	Column 1	Column 2	111	6
Column 1	Column 1	Column 1	Column 2	02	5
Column 1	Column 2	Column 1	Column 1	01	3
Column 2	Column 2	Column 2	Column 1	01	2
Column 2	Column 2	Column 2	Column 1	102	3
Column 1	Column 1	Column 1	Column 1	31	2
Column 1	Column 1	Column 1	Column 1	21	2
Column 1	Column 2	Column 1	Column 2	152	2
Column 1	Column 1	Column 1	Column 1	192	4
Column 2	Column 1	Column 2	Column 1	31	2
Column 2	none	none	none	182	7
Column 1	Column 1	Column 1	Column 1	121	1
Column 1	Column 2	Column 1	Column 1	62	2
Column 2	Column 1	Column 2	Column 1	91	7
Column 2	Column 2	Column 2	Column 2	72	6
Column 1	Column 1	Column 1	Column 1	171	4
Column 2	Column 2	Column 2	Column 1	52	7
Column 2	Column 1	Column 2	Column 1	182	4
Column 1	Column 1	Column 1	Column 1	192	3
Column 1	Column 2	Column 1	Column 1	162	2
Column 2	Column 2	Column 2	Column 1	32	4
Column 2	Column 2	Column 1	Column 1	102	0
Column 1	Column 2	Column 1	Column 1	141	2
none	none	none	Column 1	12	6
Column 1	Column 1	Column 2	Column 2	82	5
none	Column 1	none	Column 1	122	3
Column 2	Column 1	Column 2	Column 1	102	4
Column 2	Column 1	Column 2	Column 1	22	5
Column 2	Column 1	Column 1	Column 1	22	3

Words2	Pairs2 [Row 1]	Pairs2 [Row 2]	Pairs2 [Row 3]	Pairs2 [Row 4]	Pairs2 [Row 5]
jagged, nervous	Column 1	Column 2	Column 1	Column 1	Column 2
vortex	none	Column 1	Column 1	Column 1	none
calm, pleasant, smooth cu	Column 2	Column 1	Column 1	Column 2	Column 2
none	Column 2	Column 1	Column 1	Column 1	Column 1
DissonantInterrupted by s	Column 1	Column 2	Column 2	Column 1	Column 1
broken	Column 1	Column 2	Column 2	Column 1	Column 1
Alien head	Column 2	Column 2	Column 2	Column 2	Column 2
defected ball, damage, illr	Column 1	Column 2	Column 2	Column 1	Column 1
none	Column 1	Column 2	Column 2	Column 1	Column 1
Disorientation, mess	Column 1	Column 2	Column 2	Column 1	Column 1
Light turmoil	Column 2	Column 2	Column 1	Column 1	Column 1
none	Column 1	Column 2	Column 2	Column 1	Column 2
Vibrant, evolving	none	none	Column 2	Column 1	none
Chaotic, violent	Column 1	Column 2	Column 2	Column 1	Column 1
unconnected	Column 1	Column 2	Column 1	Column 1	Column 1
Calm	Column 2	Column 1	Column 1	Column 2	Column 2
Harmony, peaceful, still, c	Column 2	Column 1	Column 1	Column 1	Column 2
energitic, un-calm	Column 1	Column 2	Column 2	Column 1	Column 1
Whirlpool	Column 2	Column 1	Column 2	Column 1	Column 2
twisted	Column 1	Column 1	Column 1	Column 1	Column 2
random twisted human ea	Column 1	Column 2	Column 2	Column 1	Column 1
sad unenergized	Column 2	Column 1	Column 1	Column 2	Column 1
none	none	none	none	none	none
Dissonans	Column 1	Column 2	Column 2	Column 2	Column 1
none	Column 1	Column 2	Column 2	Column 1	Column 1
Frustrated apex	none	none	none	none	none
Clear, normal	Column 2	Column 2	Column 1	Column 2	Column 2
in progress	Column 2	none	Column 2	Column 1	none
mitosis	Column 2	Column 1	Column 2	Column 1	Column 2
Waves.	Column 2	Column 1	Column 1	Column 1	Column 1
cyclone	Column 1	Column 2	Column 1	Column 1	Column 1

Pairs2 [Row 6]	Pairs2 [Row 7]	Pairs2 [Row 8]	Pairs2 [Row 9]	Pairs2 [Row 10]	Pairs2 [Row 11]
Column 2	Column 2	Column 2	Column 2	Column 1	Column 2
none	none	none	none	none	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 1
Column 1	Column 1	Column 2	Column 1	Column 1	Column 1
Column 1	Column 1	Column 2	Column 2	Column 2	Column 2
Column 1	Column 1	Column 2	Column 2	Column 1	Column 2
Column 2	Column 2	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 1	Column 2	Column 2	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 1	Column 2	Column 1	Column 2
Column 1	Column 1	Column 1	Column 2	Column 1	Column 2
Column 1	none	none	Column 2	none	none
Column 2	Column 2	Column 2	Column 2	Column 1	Column 2
Column 2	Column 1	Column 2	Column 1	Column 2	Column 2
Column 1	Column 1	Column 1	Column 1	Column 2	Column 1
Column 2	Column 2	Column 1	Column 1	Column 2	Column 1
Column 1	Column 1	Column 1	Column 2	Column 1	Column 2
Column 1	Column 1	Column 1	Column 2	Column 2	Column 1
Column 2	Column 1	Column 1	Column 1	Column 1	Column 1
Column 1	Column 2	Column 2	Column 2	Column 1	Column 2
Column 2	Column 1	Column 1	Column 1	Column 1	Column 1
none	none	none	none	none	none
Column 2	Column 1	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 2
none	none	none	Column 1	none	Column 2
Column 2	Column 1	Column 1	Column 1	Column 1	Column 2
none	none	none	none	none	none
Column 2	Column 2	Column 2	Column 2	Column 2	Column 1
Column 1	Column 1	Column 2	Column 1	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 1

Pairs2 [Row 12]	Gift		Movie	
Column 1	YES		cast away	
none	NO		none	
Column 1	YES		Monty python and the holy grail.	
Column 1	YES		Into the wild	
Column 2	NO		none	
Column 2	NO		none	
Column 1	NO		none	
Column 1	NO		none	
Column 1	NO		none	
Column 1	NO		none	
Column 1	YES		Howl's moving castle	
Column 1	NO		none	
none	YES		Eternal sunshine of a spotless mind	
Column 1	NO		none	
Column 1	YES		Warrior	
Column 1	NO		none	
Column 1	YES		Astroguru	
Column 1	NO		none	
Column 1	NO		none	
Column 2	NO		none	
Column 1	YES		Lord of the rings: Return of the King	
Column 2	NO		none	
none	NO		none	
Column 1	NO		none	
Column 2	NO		none	
none	YES		*	
Column 2	NO		none	
Column 1	YES		Leon	
Column 1	NO		none	
Column 2	NO		none	
Column 2	YES		Avengers End Game	

Timestamp	Correct1	Main1	Selected1	Correct2	MainPic2
5/20/2019 13:51:00	YES	71	72	YES	121
5/20/2019 14:09:28	YES	151	152	NO	01
5/20/2019 14:12:30	YES	121	122	YES	201
5/20/2019 14:15:57	YES	111	112	NO	201
5/20/2019 14:39:38	NO	21	132	YES	201
5/20/2019 15:25:21	YES	111	112	YES	201
5/20/2019 15:33:02	YES	201	202	NO	21
5/20/2019 15:42:27	NO	111	201	YES	151
5/20/2019 17:08:41	YES	201	202	YES	31
5/20/2019 18:04:41	YES	191	192	YES	31
5/20/2019 18:11:32	YES	01	02	YES	131
5/20/2019 19:00:12	YES	91	92	YES	51
5/20/2019 19:05:21	YES	111	112	YES	201
5/20/2019 19:16:51	YES	81	82	YES	111
5/20/2019 20:12:59	NO	81	151	YES	171
5/20/2019 20:36:05	YES	41	42	YES	71
5/20/2019 20:58:27	NO	191	161	YES	181
5/20/2019 21:45:51	YES	41	42	YES	61
5/20/2019 21:58:55	YES	151	152	YES	101
5/20/2019 22:27:48	YES	151	152	YES	191
5/20/2019 23:11:33	YES	101	102	YES	21
5/20/2019 23:50:41	YES	171	172	YES	51
5/21/2019 4:51:56	YES	171	172	YES	41
5/21/2019 6:26:51	YES	91	92	NO	01
5/21/2019 7:04:15	YES	111	112	YES	201
5/21/2019 7:01:54	NO	101	192	NO	61
5/21/2019 8:25:39	NO	181	31	YES	41
5/21/2019 8:57:32	NO	11	51	YES	131
5/21/2019 10:26:31	YES	191	192	NO	51
5/21/2019 10:52:20	YES	81	82	YES	91

SelectedPic2	RatePic1	Mood1	Words1	Pairs1 [Row 1]	Pairs1 [Row 2]
122	82	6	gentle touches, slowly pla	Column 2	Column 1
182	121	0	complex	none	none
202	192	5	Face	none	Column 1
62	61	5	Fingerprint	Column 2	Column 1
202	161	2	In shock	Column 1	Column 1
202	142	3	brain, chaos	Column 1	Column 2
182	172	2	Complexity	Column 1	Column 2
152	21	1	Scull	Column 2	Column 1
32	22	3	Writhing	Column 2	Column 2
32	162	2	Imperfect, weak, messy	Column 1	Column 2
132	121	2	hectic	none	none
52	01	2	Disturbance	Column 2	Column 2
202	51	8	Romantic, gentle, soft and	Column 2	Column 1
112	71	5	none	Column 1	Column 1
172	181	0	devil covered by silk	Column 1	Column 2
72	122	3	it feels kind of broken and	none	Column 2
182	152	2	Boring rough	Column 2	Column 2
62	91	4	Comet	Column 1	none
102	201	2	Tense	none	Column 2
192	62	2	Valleys	Column 2	Column 1
22	162	2	Scared scream	Column 1	Column 2
52	01	2	Broken but not irreparable	Column 1	Column 2
42	32	5	Wrinkles, bumpy,	Column 2	Column 2
11	32	8	Mouvement	none	Column 1
202	51	8	Romantic, gentle, soft and	Column 2	Column 1
11	151	5	Ears	Column 1	Column 1
42	152	2	Wrinly, irritated.	Column 2	Column 2
132	41	2	none	Column 1	Column 2
62	31	2	Disruption and little bit of	Column 1	Column 2
92	161	6	Ear or baby	Column 1	Column 1

Pairs1 [Row 3]	Pairs1 [Row 4]	Pairs1 [Row 5]	Pairs1 [Row 6]	Pairs1 [Row 7]	Pairs1 [Row 8]
Column 1	Column 1	Column 1	Column 2	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	none	none
none	Column 1	none	none	none	Column 2
Column 1	Column 1	none	none	none	none
Column 1	Column 1	Column 2	Column 1	Column 2	Column 1
Column 2	Column 1	Column 1	Column 1	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	Column 1	Column 2
Column 1	Column 2	Column 2	Column 1	Column 2	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 2
Column 2	Column 1	Column 1	Column 1	Column 1	Column 1
Column 2	none	none	none	Column 2	Column 2
Column 1	Column 1	Column 1	Column 1	Column 2	Column 2
Column 1	Column 1	Column 2	Column 1	Column 1	Column 1
Column 1	none	none	Column 1	Column 1	Column 1
Column 2	Column 1	Column 2	Column 1	Column 2	Column 2
none	Column 2	Column 1	Column 2	none	Column 2
Column 1	Column 2	Column 2	Column 2	Column 2	Column 1
none	none	none	none	none	none
none	none	none	none	none	none
Column 1	Column 2	Column 2	Column 1	Column 2	Column 1
Column 2	Column 2	Column 2	Column 1	Column 1	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 1
Column 1	Column 1	Column 2	Column 2	Column 1	Column 2
none	Column 1	none	Column 1	none	none
Column 1	Column 1	Column 2	Column 1	Column 1	Column 1
Column 1	Column 1	Column 2	Column 1	Column 2	Column 2
Column 2	Column 1	Column 1	Column 1	Column 2	Column 1
Column 1	Column 2	Column 1	Column 1	Column 2	Column 2
Column 2	Column 2	Column 1	Column 1	Column 1	Column 2
Column 1	Column 2	Column 2	Column 1	Column 1	Column 1

Pairs1 [Row 9]	Pairs1 [Row 10]	Pairs1 [Row 11]	Pairs1 [Row 12]	RatePic2	Mood2
Column 1	Column 1	Column 1	Column 1	151	7
Column 2	Column 1	Column 2	Column 1	112	5
none	none	Column 1	none	182	8
none	none	none	none	31	4
Column 1	Column 2	Column 1	Column 1	62	0
Column 2	Column 2	Column 1	Column 1	161	3
Column 2	Column 1	Column 2	Column 2	102	0
Column 1	Column 1	Column 1	Column 1	72	0
Column 2	Column 1	Column 1	Column 1	91	4
Column 1	Column 2	Column 2	Column 2	02	6
none	none	Column 2	Column 1	12	5
Column 1	Column 1	Column 2	Column 2	31	1
Column 1	Column 2	Column 1	Column 1	101	1
Column 1	Column 2	Column 1	Column 2	51	0
Column 2	Column 1	Column 2	Column 1	52	1
Column 2	Column 1	Column 2	Column 2	81	5
Column 1	Column 1	Column 2	Column 2	01	5
none	none	none	none	52	6
none	none	none	none	91	6
Column 2	Column 1	Column 2	Column 2	161	5
Column 2	Column 2	Column 2	Column 1	111	7
Column 2	Column 2	Column 2	Column 1	22	6
Column 1	Column 2	Column 1	Column 2	111	6
Column 1	none	none	none	201	6
Column 1	Column 2	Column 1	Column 1	101	1
Column 1	Column 1	Column 1	Column 2	81	6
Column 2	Column 1	Column 2	Column 1	191	6
Column 2	Column 1	Column 2	Column 1	191	4
Column 2	Column 1	Column 2	Column 1	171	4
Column 1	Column 1	Column 1	Column 1	171	2

Words2	Pairs2 [Row 1]	Pairs2 [Row 2]	Pairs2 [Row 3]	Pairs2 [Row 4]	Pairs2 [Row 5]
cloth twirls, movement im	Column 2	Column 1	Column 1	Column 1	Column 1
smooth	Column 2	Column 1	Column 1	Column 2	none
Degenerated	Column 2	none	Column 2	Column 1	none
none	Column 2	Column 1	Column 1	Column 1	Column 2
hard	Column 2	Column 2	Column 1	Column 2	Column 2
mountains, obsession	Column 1	Column 2	Column 1	Column 2	Column 2
Anxiety	Column 1	Column 2	Column 2	Column 1	Column 1
□ ant	Column 2	Column 1	Column 1	Column 2	Column 2
Symbol	Column 2	Column 1	Column 1	Column 2	Column 2
none	Column 1	Column 1	Column 2	Column 1	Column 1
structure	Column 2	none	Column 1	Column 1	none
Wrinkles	Column 1	Column 2	Column 2	Column 1	Column 2
Crush, mixing up, echo, h	Column 2	Column 2	none	Column 1	Column 1
none	none	none	Column 2	Column 1	Column 2
lava	Column 2	Column 1	Column 1	Column 1	Column 2
smooth	none	Column 1	Column 1	none	none
Smooth interesting	Column 2	Column 1	Column 1	Column 2	Column 2
Fingerprint	none	Column 1	none	none	none
Soft	none	none	none	Column 1	none
Smooth	Column 1	Column 1	Column 1	Column 1	Column 1
Moon chill	Column 2	Column 1	Column 2	Column 1	Column 1
Life in motion	Column 2	Column 1	Column 1	Column 1	Column 2
Calm	Column 2	Column 1	Column 1	Column 2	Column 2
Calm	none	none	Column 1	none	Column 1
Crush, mixing up, echo, h	Column 2	Column 2	none	Column 1	Column 1
River	Column 2	Column 1	Column 2	Column 1	Column 2
calm	Column 1	Column 1	Column 2	Column 1	Column 1
none	Column 2	Column 1	Column 1	Column 1	Column 2
Neither good nor bad	Column 1	Column 2	Column 1	Column 1	Column 1
Waves	Column 1	Column 2	Column 1	Column 1	Column 2

Pairs2 [Row 6]	Pairs2 [Row 7]	Pairs2 [Row 8]	Pairs2 [Row 9]	Pairs2 [Row 10]	Pairs2 [Row 11]
Column 2	Column 1	Column 1	Column 1	none	Column 1
Column 2	none	none	Column 1	none	Column 1
none	Column 1	none	none	Column 1	none
Column 1	Column 1	Column 1	Column 2	Column 2	Column 1
Column 2	Column 2	Column 2	Column 2	Column 1	Column 2
Column 1	Column 1	Column 1	Column 2	Column 1	Column 2
Column 1	Column 1	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 2
Column 2	Column 2	Column 1	Column 1	Column 2	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 1
none	none	none	Column 1	none	Column 1
Column 2	Column 2	Column 1	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 2
Column 2	Column 2	none	Column 2	Column 1	Column 2
Column 1	Column 1	Column 1	Column 2	Column 2	Column 2
Column 1	Column 1	none	none	none	Column 1
Column 2	Column 2	Column 1	Column 2	Column 2	Column 1
none	none	none	none	none	none
none	none	none	none	none	none
Column 1	Column 1	Column 2	Column 1	Column 2	Column 1
Column 1	Column 1	Column 1	Column 1	Column 1	Column 1
none	Column 1	none	Column 1	Column 2	Column 1
Column 2	Column 1	Column 2	Column 1	Column 2	Column 1
none	none	none	Column 1	none	none
Column 1	Column 2	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 2	Column 1
Column 1	Column 1	Column 1	Column 2	Column 2	Column 1
Column 2	Column 1	Column 1	Column 1	Column 2	Column 1
Column 2	Column 1	Column 2	Column 2	Column 1	Column 2
Column 2	Column 1	Column 2	Column 2	Column 2	Column 1

Pairs2 [Row 12]	Gift		Movie	
none	NO		none	
Column 2	YES		The Hunt(danish movie)	
Column 2	YES		Malena	
Column 1	NO		none	
Column 1	NO		none	
Column 2	NO		none	
Column 2	NO		none	
Column 1	YES		Bad Boys 2	
Column 1	YES		Porco Rosso	
Column 1	NO		none	
Column 1	NO		none	
Column 2	YES		scott pilgrim and the seven evil exes	
Column 1	YES		About time	
none	NO		none	
Column 1	NO		none	
none	YES		The last samurai	
Column 1	YES		Butterfly effect	
none	NO		none	
none	YES		Inception	
Column 1	YES		Your Name	
Column 1	YES		Blood in blood out	
Column 1	YES		Amélie	
Column 2	NO		none	
none	NO		none	
Column 1	YES		About time	
Column 2	NO		none	
Column 1	NO		none	
Column 2	NO		none	
Column 2	YES		Deadpool	
Column 2	YES		The Notebook	

Timestamp	Correct1	Main1	Selected1	Correct2	MainPic2
5/21/2019 12:24:54	YES	51	52	YES	31
5/21/2019 13:23:36	YES	161	162	YES	51
5/21/2019 13:30:26	YES	121	122	YES	181
5/21/2019 13:32:48	NO	181	51	NO	161
5/21/2019 14:59:43	YES	131	132	YES	191
5/21/2019 15:30:59	YES	71	72	YES	201
5/21/2019 15:51:25	YES	121	122	YES	81
5/21/2019 17:00:21	YES	21	22	YES	71
5/21/2019 20:32:22	NO	51	62	YES	191
5/22/2019 4:44:48	YES	101	102	NO	141
5/22/2019 6:35:56	YES	11	12	YES	191
5/22/2019 6:43:58	YES	81	82	YES	121
5/22/2019 9:44:43	YES	191	192	YES	141
5/22/2019 14:11:21	YES	181	182	YES	01
5/22/2019 15:03:02	YES	21	22	YES	41
5/22/2019 22:26:12	NO	181	131	YES	191
5/24/2019 6:24:58	NO	01	72	YES	131
5/24/2019 11:11:32	YES	151	152	NO	01

SelectedPic2	RatePic1	Mood1	Words1	Pairs1 [Row 1]	Pairs1 [Row 2]
32	12	7	Calm but edgy	Column 2	Column 1
52	21	4	Waves	Column 1	Column 1
182	111	4	Perturbed	Column 1	Column 2
112	121	2	Messy, chaotic	Column 1	Column 2
192	102	2	unease, shakey, intense,	Column 1	Column 2
202	12	2	Sad personMountain	Column 2	Column 1
82	161	3	Contour, heightmap	Column 2	Column 1
72	112	1	crawling, despair, hiding,	Column 2	Column 1
192	171	2	Chaos, disturbance	Column 1	Column 2
121	152	6	Beautiful, uniform, pure, li	Column 2	Column 1
192	181	2	Face with a judging mouth	Column 1	Column 1
122	12	4	Geographical map withou	Column 2	Column 2
142	92	2	nervous, sharp	Column 2	Column 2
02	131	2	Nerves, earthquake, soun	Column 1	Column 2
42	51	5	simple, calm	Column 2	Column 1
192	21	1	none	Column 1	Column 2
132	11	7	Calm, water	Column 2	Column 1
202	11	5	Flow with small jerks in th	Column 2	Column 1

Pairs1 [Row 3]	Pairs1 [Row 4]	Pairs1 [Row 5]	Pairs1 [Row 6]	Pairs1 [Row 7]	Pairs1 [Row 8]
Column 1	Column 1	Column 2	Column 1	Column 1	Column 1
Column 2	Column 1	Column 2	Column 1	Column 2	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 1
Column 2	Column 2	Column 1	Column 1	Column 1	Column 2
Column 2	Column 1	Column 1	Column 2	Column 2	Column 2
Column 1	Column 2	Column 2	Column 1	Column 2	Column 1
Column 2	Column 1	Column 1	Column 1	Column 1	Column 2
Column 2	Column 1	Column 2	Column 1	Column 2	Column 1
Column 1	Column 1	Column 1	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 1
Column 2	Column 1	Column 1	Column 2	Column 1	Column 1
Column 2	Column 1	none	Column 1	Column 2	Column 2
Column 2	Column 1	none	Column 1	Column 2	Column 1
Column 1	Column 1	Column 1	Column 1	Column 2	Column 2
Column 1	Column 1	Column 2	Column 2	Column 1	Column 1
none	Column 2	Column 1	Column 1	none	none
Column 1	Column 1	Column 1	Column 2	Column 1	Column 2
Column 1	Column 1	Column 2	Column 2	Column 2	Column 1

Pairs1 [Row 9]	Pairs1 [Row 10]	Pairs1 [Row 11]	Pairs1 [Row 12]	RatePic2	Mood2
Column 1	Column 1	Column 1	Column 1	61	3
Column 2	Column 2	Column 1	Column 1	131	2
Column 1	Column 1	Column 1	Column 2	32	3
Column 2	Column 1	Column 2	Column 1	201	6
Column 1	Column 1	Column 2	Column 1	12	7
Column 1	Column 1	Column 1	Column 1	162	2
Column 1	Column 1	Column 1	Column 1	111	6
Column 1	Column 2	Column 1	Column 1	41	4
Column 2	Column 1	Column 2	Column 2	182	2
Column 1	Column 1	Column 1	Column 2	42	3
Column 1	Column 1	Column 1	Column 1	142	3
Column 2	Column 2	Column 2	Column 2	161	4
Column 2	Column 2	Column 2	Column 1	111	6
Column 2	Column 1	Column 2	Column 1	192	7
Column 1	Column 1	Column 1	Column 2	101	1
Column 2	none	Column 2	none	01	1
Column 1	Column 2	Column 2	Column 2	142	2
Column 1	Column 2	Column 1	Column 1	91	5

Words2	Pairs2 [Row 1]	Pairs2 [Row 2]	Pairs2 [Row 3]	Pairs2 [Row 4]	Pairs2 [Row 5]
Sad	Column 2	Column 1	Column 1	Column 2	Column 2
Hurricane	Column 1	Column 2	Column 2	Column 1	Column 1
Uneven	Column 1	Column 2	Column 2	Column 1	Column 1
Subtlety, change	Column 2	Column 1	Column 2	Column 1	Column 1
sea, horizont, landscape,	Column 2	Column 1	Column 1	Column 1	Column 1
WrinkleLaundry	Column 1	Column 2	Column 2	Column 1	Column 2
Moon crater	Column 1	Column 1	Column 1	Column 2	Column 1
dance, excitement, chang	Column 1	Column 2	Column 2	Column 1	Column 1
light pain	Column 2	Column 2	Column 2	Column 1	Column 1
Complicated, twisting, inte	Column 2	Column 2	Column 2	Column 1	Column 1
Doodling	Column 1	Column 2	Column 2	Column 2	Column 1
weather forecast in depth	Column 2	Column 2	Column 2	Column 1	none
slow	Column 2	Column 1	Column 1	Column 2	Column 2
Circles, flow, river,	Column 2	Column 1	Column 1	Column 1	Column 1
aggressive, angry, harsh,	Column 1	Column 2	Column 2	Column 1	Column 1
none	Column 1	Column 2	none	none	Column 1
Tear apart	Column 2	Column 2	Column 2	Column 1	Column 1
Some sort of symmetry m	Column 2	Column 1	Column 2	Column 1	Column 2

Pairs2 [Row 6]	Pairs2 [Row 7]	Pairs2 [Row 8]	Pairs2 [Row 9]	Pairs2 [Row 10]	Pairs2 [Row 11]
Column 1	Column 1	Column 1	Column 1	Column 1	Column 1
Column 1	Column 1	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 1	Column 2
Column 1	Column 1	Column 1	Column 1	Column 2	Column 1
Column 1	Column 1	Column 1	Column 1	Column 1	Column 1
Column 2	Column 1	Column 2	Column 2	Column 1	Column 2
Column 2	Column 1	Column 2	Column 2	Column 2	Column 1
Column 2	Column 2	Column 1	Column 2	Column 1	Column 2
Column 1	Column 1	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 2	Column 2
Column 2	Column 1	Column 2	Column 2	Column 1	Column 2
Column 1	Column 2	Column 2	Column 2	Column 2	Column 2
Column 2	Column 1	Column 2	Column 1	Column 1	Column 1
Column 1	Column 1	Column 1	Column 1	Column 2	Column 1
Column 1	Column 2	Column 2	Column 2	Column 2	Column 2
none	none	Column 2	none	none	Column 2
Column 1	Column 1	Column 1	Column 2	Column 2	Column 2
Column 1	Column 2	Column 1	Column 2	Column 2	Column 1

Pairs2 [Row 12]	Gift		Movie	
Column 1	YES		Den eneste ene	
Column 1	YES		Fight club	
Column 1	YES		Catch me if you can	
Column 1	YES		Your Name	
Column 1	YES		Eyes wide shut	
Column 2	YES		Avatar	
Column 1	NO		none	
Column 1	YES		Gladiator	
Column 1	NO		none	
Column 1	YES		Kung Fu Panda	
Column 1	NO		none	
Column 2	YES		donnie darko	
Column 2	YES		Pesho	
Column 1	YES		Juno	
Column 1	NO		none	
Column 2	NO		none	
Column 2	YES		Limitless	
Column 1	YES		Back to the Future	

Appendix A4

Implementation- Metadata Analysis

In [3]:
```
%matplotlib inline
```

In [1]:
```
import numpy as np

import pylab
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
MovieTsnePoints=pd.read_csv("0F.csv")
MovieTsnePoints.columns
MovieTsnePoints=MovieTsnePoints[['t1','t2']]
MovieTsnePoints.dropna()
print(np.mean(MovieTsnePoints))

MovieSentiment=pd.read_csv("0F.csv")
MovieSentiment.columns
MovieSentiment=MovieSentiment[['valence','arousal']]
MovieSentiment.dropna()
print(np.mean(MovieSentiment))

MovieTsnePoints = StandardScaler().fit_transform(MovieTsnePoints)
```

```
t1    0.201741
t2   -0.245986
dtype: float64
valence    0.511266
arousal   -0.121217
dtype: float64
```

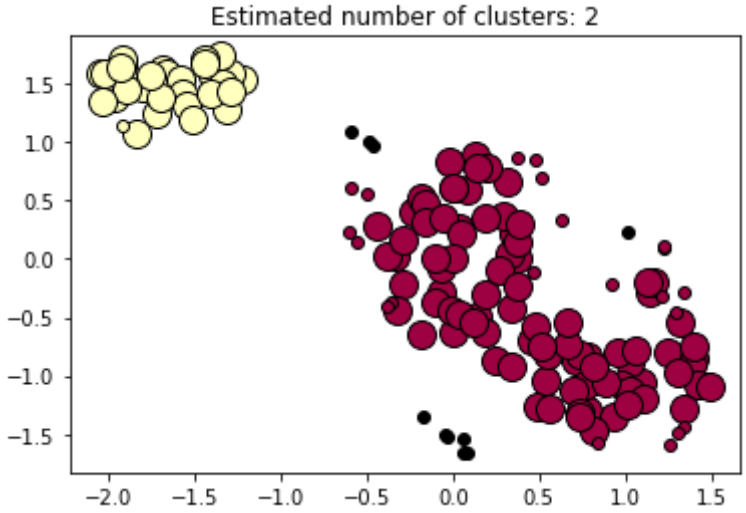
In [3]:
```
# Compute DBSCAN
db = DBSCAN(eps=0.4, min_samples=10).fit(MovieTsnePoints)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

counts = np.bincount(labels[labels>=0])




print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
print(counts)
print(np.std(labels))
```

```
Estimated number of clusters: 2
Estimated number of noise points: 10
[108  31]
0.5052759791870796
```

In [ ]:

In [4]:
```
# Black removed and is used for noise instead.

import matplotlib.pyplot as plt

unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = (labels == k)

    xy = MovieTsnePoints[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=14)

    xy = MovieTsnePoints[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)

plt.title('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
print(counts)

print(type(labels))

plt.show()
```

```
Estimated number of clusters: 2
Estimated number of noise points: 10
[108  31]
<class 'numpy.ndarray'>
```



In [5]:
```
from sklearn.cluster import KMeans
import numpy as np
X = MovieTsnePoints
kmeans = KMeans(n_clusters=n_clusters_, random_state=0).fit(X)
kmeans.labels_

kmeans.cluster_centers_
```

Out[5]:
```
array([[-1.50228259,  1.39148265],
       [ 0.4786033 , -0.44330421]])
```

In [90]:
```
pocitamStd = (labels == 0)
pocitamStd2 = (labels == 1)
MovieTsnePoints[pocitamStd]

print(np.std(MovieTsnePoints[pocitamStd][:,0]), np.std(MovieTsnePoints[p
ocitamStd][:,1]))
print(np.std(MovieTsnePoints[pocitamStd2][:,0]), np.std(MovieTsnePoints[
pocitamStd2][:,1]))
```

```
0.5944361799422452 0.38463891107136006
0.32256469887245603 0.3643187044030861
```

In [ ]:

Implementation- MovieSignature

Raw | Blame | History

403 lines (295 sloc) 11.2 KB

```
1
2  import generativedesign.*;
3  import processing.pdf.*;
4  import java.util.Calendar;
5  import java.util.Date;
6
7  boolean savePDF = false;
8  boolean saveToPrint = false;
9  boolean recording = false;
10
11 int h=1600;
12 int w=1600;
13
14 float cnt01, cnt02;
15 int currentDrawIterationForCluster = 0;
16 float valenceMean ; //calculated inside load data
17 float pointX, pointY; //used for mapping X and Y position of attractors
18 int count = 250; //int(random(40,90)); //number of dots
19 int crclSize=w/2-200;
20 int crclDist=int(sqrt(pow(crclSize, 2)/2));
21
22 //attractors parameters
23 int radius; //radius for attractors
```

```
float ramp; //ramp default before mapping

String fileName="0F.csv";

int clusterCount = 2; //number of clusters  ******
int time = clusterCount*120;

int c1x = int(map(-1.50228259, -2, 2, w/2-crclDist, w/2+crclDist));     //mapping mean of f
int c1y = int(map(1.39148265, 2, -2, w/2+crclDist, w/2-crclDist));
int r1 = int(map(31, 0, 150, 100, 800));        //mapping radius to be each cluster's size

int c2x = int(map(0.4786033, -2, 2, w/2-crclDist, w/2+crclDist));     //mapping mean of sec
int c2y = int(map(-0.44330421, 2, -2, w/2+crclDist, w/2-crclDist));
int r2 = int(map(108, 0, 150, 100, 800));

int c3x = int(map(0, -2, 2, 150, w-150));     //mapping mean of third cluster for x,y posit
int c3y = int(map(0, 2, -2, h-150, 0+150));
int r3 = int(map(0, 0, 150, 100, 800));

int c4x = int(map(0, -2, 2, 150, w-150));     //mapping mean of 4 cluster for x,y positions
int c4y = int(map(0, 2, -2, h-150, 0+150));
int r4 = int(map(0, 0, 150, 100, 800));

int c5x = int(map(0, -2, 2, 150, w-150));     //mapping mean of 5 cluster for x,y positions
int c5y = int(map(0, 2, -2, h-150, 0+150));
int r5 = int(map(0, 0, 150, 100, 800));

int c6x = int(map(0, -2, 2, 150, w-150));     //mapping mean of 6 cluster for x,y positions
int c6y = int(map(0, 2, -2, h-150, 0+150));
int r6 = int(map(0, 0, 150, 100, 800));


float noise= 10*100/150 ; // noise in percentage, based on number of noise points (first nu

int drawingRadius= int(map(noise, 0, 100, 100, 600));

float strength = 45*int(random(-2, 2));//map(0.9, 0, 1, 0, 45); //strength of attractors
float rampS= 0.19;                // ramp on starting= initial point effect

float circleRadius = map(100-noise, 0, 100, 0.98, 1); //how spreaded are the initial points

float finalnodeY, finalnodeX;
color range;
int coloredline;

int left [][]=new int [2][count]; //2 2D arrays with values
int right [][]=new int [2][count];

int xCount=0; //initial value
```

```
73
74  ArrayList<ClusterXY> clusterStartPoints = new ArrayList<ClusterXY>();
75
76  ArrayList<Node> nodeArraylist; //vsetky spolu
77  ArrayList<ArrayList> linesList; //vsetky lines
78
79  Attractor myAttractor;
80
81  PointAnna[]points;
82
83  Table table;
84
85  void setup() {
86    size(1600, 1600);
87    smooth();
88    pixelDensity(2);
89    strokeCap(ROUND);
90    strokeJoin(ROUND);
91    loadData();
92    setupClusterPoints();
93
94    float mappedValence=map(valenceMean, 1, -1, count-20, 20);
95    coloredline=floor(mappedValence);
96    int colorMapped=int(map(valenceMean, 1, -1, 340, 220));
97
98    colorMode(HSB, 360, 100, 100);
99    range=color(colorMapped, 42, 73);
100
101   ramp = map(valenceMean, -1, 1, -0.99, 0.99); //ramp radius to be based on valence
102
103   println("points= " +count + " strength= " + strength+" ramp= " +ramp+ " radius= " +radius
104
105   makepoints(); //calculates the positions of points around a circle, and stores them in th
106   sortMe(left); //sorting all left points by y value, smallest to highest
107   sortMe(right);
108
109   nodeArraylist = new ArrayList<Node>(10000);
110   linesList=new ArrayList();
111
112   initGrid(); // setup node grid. This is where I save the position of each node (from all
113
114   myAttractor = new Attractor(0, 0); // setup attractor
115   myAttractor.strength = strength;
116   myAttractor.ramp = ramp;
117   myAttractor.radius=radius;
118 }
119
120 boolean first=true;
121 int c = 0;
```

```
//float noiseMax = map(noise, 0, 100, 1, 5);

void setupClusterPoints()
{
  for (int i=0; i< clusterCount; i++)
  {
    if (i==0)
      clusterStartPoints.add(new ClusterXY(c1x, c1y, r1));
    else if (i==1)
      clusterStartPoints.add(new ClusterXY(c2x, c2y, r2));
    else if (i==2)
      clusterStartPoints.add(new ClusterXY(c3x, c3y, r3));
    else if (i==3)
      clusterStartPoints.add(new ClusterXY(c4x, c4y, r4));
    else if (i==4)
      clusterStartPoints.add(new ClusterXY(c5x, c5y, r5));
    else if (i==5)
      clusterStartPoints.add(new ClusterXY(c6x, c6y, r6));
  }
}


void draw() {

  if (savePDF) beginRecord(PDF, fileName+ c+".pdf");

  background(0, 0, 100);
  strokeWeight(0.3);
  stroke(0);
  fill(0);
  float lastx=0;
  float lasty=0;

  float directionx=random(-1, 1);
  float directiony=random(-1, 1);
  int randomizedDrawingRadiusX=int(random(0, drawingRadius)*directionx);
  int randomizedDrawingRadiusY=int(random(0, drawingRadius)*directiony);

  ClusterXY currentClusterPoint = clusterStartPoints.get(currentDrawIterationForCluster);

  pointX = currentClusterPoint.x;
  pointY = currentClusterPoint.y;
  myAttractor.radius=currentClusterPoint.radius;

  currentDrawIterationForCluster++;

  c++;
  cnt01+=0.7;
  cnt02+=0.34;
```

```
171
172   if (first) {
173     myAttractor.ramp = rampS;
174     myAttractor.radius = 150;
175     myAttractor.x = pointX;
176     myAttractor.y = pointY;
177
178     //first=false;
179   } else {
180     myAttractor.ramp = ramp;
181     myAttractor.x = pointX+(noise(cnt01)-0.5)*350-randomizedDrawingRadiusX;
182     myAttractor.y = pointY+(noise(cnt02)-0.5)*350-randomizedDrawingRadiusY;
183   }
184
185   if (c%20==0 & directionx<0)myAttractor.strength*=-1;
186
187   //if(c%30==0 & directionx>0)myAttractor.strength= int(random(25,45))*directionx;
188
189   for (int i = 0; i < linesList.size(); i++) {  //loop for going through the nodes drawing
190     ArrayList pointsOnLine=linesList.get(i);
191
192     if (i==coloredline ||i==coloredline-2 || i==coloredline+2) {
193       stroke(range);
194       strokeWeight(0.8);
195     } else {
196       stroke(0);
197       strokeWeight(0.3);
198     }
199
200     for (int j=0; j<pointsOnLine.size(); j++) {
201       Node node = (Node)pointsOnLine.get(j);
202       myAttractor.attract(node);
203       node.update();
204
205       finalnodeX=node.x;
206       finalnodeY=node.y;
207
208       if (lastx>0) {
209         line(finalnodeX, finalnodeY, lastx, lasty);
210       }
211       lastx=finalnodeX;
212       lasty=finalnodeY;
213     }
214
215     lastx=0;
216     lasty=0;
217   }
218
219
```

```
  if (currentDrawIterationForCluster==clusterCount)
  {
    currentDrawIterationForCluster = 0;
    first=false;
  }



  if (recording)saveFrame("output/attractingPixs_####.png");

  if (c%50==0)
  {
    println("saving shot");
    saveFrame(timestamp()+"_####.png");
  }

  if (c==time)
  {
    println("I'm done. Thanks");
    saveFrame(timestamp()+"_####.png");
    noLoop();
  }

  if (savePDF) {
    savePDF = false;
    println("saving to pdf – finishing");
    endRecord();
  }
}

void loadData() {

  Table table = loadTable(fileName); //data file
  float[][] points = new float[2][table.getRowCount()-1];


  for (int i=1; i<table.getRowCount(); i++) { //accesing all table rows and storing them as

    TableRow row=table.getRow(i);

    float xax=(float)row.getDouble(4);
    float yax =(float)row.getDouble(5);


    points[0][i-1] = xax;
    points[1][i-1] = yax;
  }

  float[] meanX = meanAndStd(points[0]);
```

```
  float[] meanY = meanAndStd(points[1]);
  println(" mean for valence and arousal  is " + meanX[0], meanY[0]);
  println("std  is " + meanX[1], meanY[1]);
  valenceMean= meanX[0];
  //arousalMean=meanY[0];
  //valenceStd=meanX[1];
  //arousalStd=meanY[1];
}

float[] meanAndStd(float numArray[])
{
  float[] ret = new float[2];
  float sum = 0.0, standardDeviation = 0.0;
  int length = numArray.length;

  for (float num : numArray) {
    sum += num;
  }

  float mean = sum/length;

  for (float num : numArray) {
    standardDeviation += Math.pow(num - mean, 2);
  }

  ret[0] = mean;
  ret[1] = (float)Math.sqrt(standardDeviation/length);

  return ret;
}

void sortMe(int[][] arr) {
  String [] sortArray= new String[count];


  for (int j = 0; j < count; ++j) {       //Moving the data from Matrix to single array to
    String combineColumns;
    combineColumns = str(arr[0][j]);   //Adding the two columns together, seperated by a "d
    //Converting the datatype to String, to fit a single array
    combineColumns = str(arr[1][j]) + "-" + combineColumns; //saving string with y position
    sortArray[j] = combineColumns;
  }

  sortArray = sort(sortArray);  //Sorting, using processings sorting function

  for (int i = 0; i < count; ++i) {  //Moving the sorted data back in the Matrix
    String[] tmp = split(sortArray[i], "-"); //Using the split function, for each string in
    arr[0][i] = int(tmp[1]);
    arr[1][i] = int(tmp[0]); //Using the temp array, to write the data into the matrix agai
```

```
  }

  // println(""); println("FINAL RESULT"); //checking if sorting worked
  //for(int i = 0; i < 10; ++i) {
  // print(arr[0][i],arr[1][i]);
  // println("");
  //}
}

void makepoints() {

  for (int i=0; i<count; i++) {   //calculating coordinates for each point on the right sid

    float angle = radians(180/float(count));
    float randomX = random(0, width);
    float randomY = random(0, height);
    float circleX = width/2 + sin(angle*i)*(crclSize);
    float circleY = height/2 + cos(angle*i)*(crclSize);

    int x = floor(lerp(randomX, circleX, circleRadius));
    int y = floor(lerp(randomY, circleY, circleRadius));

    right [0][i]=x;
    right [1][i]=y;
  }


  for (int i=0; i<count; i++) {  //calculating coordinates for each point on left side
    float angle = radians(-180/float(count));
    float randomX = random(0, width);
    float randomY = random(0, height);
    float circleX = width/2 + sin(angle*i)*(crclSize);
    float circleY = height/2 + cos(angle*i)*(crclSize);

    int x = floor(lerp(randomX, circleX, circleRadius));
    int y = floor(lerp(randomY, circleY, circleRadius));

    left [0][i]=x;
    left [1][i]=y;
  }
}

void initGrid() {


  for (int y = 0; y < count; y++) {

    int middleToPoint = int(dist(width/2, left[1][y], left[0][y], left[1][y]));

```

```
    ArrayList<Node> nodesInOneLine =new ArrayList(); //vsetky body v jednej line

    for (int x = left[0][y]; x <= left[0][y]+2*middleToPoint; x++) {
      int xPos = x;
      int yPos = left[1][y];

      Node myOneNode = new Node(xPos, yPos); //save position in Node object
      myOneNode.setBoundary(0, 0, width, height);
      myOneNode.setDamping(0.8);  //// 0.0 - 1.0

      nodeArraylist.add(myOneNode);
      nodesInOneLine.add(myOneNode);
    }

    linesList.add(nodesInOneLine);
  }
}


/////

void keyReleased() {  //saving
  if (key == 's' || key == 'S') saveFrame(timestamp()+"_####.png");
  if (key == 'p' || key == 'P') savePDF = true;
  println("I'm safed!");
  if  (key == 'r' || key == 'R') {
    recording = !recording;
    if (recording) println("recording on");
    if (!recording) println("recording stopped");
  }
}

String timestamp() {
  Calendar now = Calendar.getInstance();
  return String.format("%1$ty%1$tm%1$td_%1$tH%1$tM%1$tS", now);
}
```

Appendix B

```
from bs4 import BeautifulSoup
import cv2
import numpy as np
from numpy import genfromtxt
import os
import re
import requests
import urllib3
urllib3.disable_warnings()
from IPython.display import clear_output
from sklearn.cluster import KMeans
import argparse

clusters = 5
colorSpace = 'BGR'
combined = True
resizeW = 250

class YahooScraper():
    def get_soup(self, url):
        return BeautifulSoup(requests.get(url).text, "html.parser")

    def download_images(self, color, num_images):

        # Num of images
        if num_images > 9:
            raise ValueError('Number of images to download must be less than 20.')

        download_directory = 'images/' + color

        # Dir check
        if not os.path.exists(download_directory):
            os.mkdir(download_directory)

        # Images check
        nfiles = len([name for name in os.listdir(download_directory) if '.jpg' in name])
        if nfiles >= num_images:
            return False

        # Query
        query = color
        url = 'https://images.search.yahoo.com/search/images?p=' + query
        +'&imgty=photo&imgsz=medium'

        soup = self.get_soup(url)
        images = [a['src']
                  for a in soup.find_all('img', {'src': re.compile('http')})]
        images = images[nfiles:num_images]

        if len(images) < 1:
            return False


        for img in images:
            http = urllib3.PoolManager()
            raw_img = http.request('GET', img).data
            image = np.asarray(bytearray(raw_img), dtype="uint8")
            image = cv2.imdecode(image, cv2.IMREAD_COLOR)

            x_size, y_size, _ = image.shape
            cropped_image = image[0:y_size, int(x_size*.18):int(x_size*1.18)]

            cntr = len([i for i in os.listdir(download_directory)
                        if color in i]) + 1
            filename = ("images/" + color + '/' + color +
                        "_" + str(cntr) + '.jpg')
            cv2.imwrite(filename, cropped_image)
```

```
        return True

# Combine images into one for processing
def combine_images(folder, img_array):
    if len(img_array) == 0 or img_array is None:
        return (None, None)

    # Make combined image
    h = []
    w = []

    # Sizes
    for i in img_array:
        img = cv2.imread(i)
        h.append(img.shape[0])
        w.append(img.shape[1])

    # Final shape
    image = np.zeros((max(h), sum(w),3), np.uint8)

    # Join to final
    pivot = 0
    for i, j in enumerate(img_array):
        img = cv2.imread(j)
        image[:h[i], pivot:pivot+w[i],:3] = img
        pivot = pivot + w[i]

    # Scale down to 250px width
    r = resizeW / image.shape[1]
    dim = (resizeW, int(image.shape[0] * r))
    image = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)

    # Save combined
    cv2.imwrite(folder+'/combined.jpg', image)

    return folder+'/combined.jpg'

# Get Dominant colors of input images
def get_dominant_colors(img_array):
    if len(img_array) == 0 or img_array is None:
        return (None, None)

    colors_list = []
    cluster_list = []
    for image in img_array:
        dominant_colors, cluster = get_dominant_colors_img(image)
        colors_list.append(dominant_colors)
        cluster_list.append(cluster)

    return (colors_list, cluster_list)

# Dominant colors of an image
def get_dominant_colors_img(image_path):
    global clusters

    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) if colorSpace == 'RGB' else image

    # reshape the image to be a list of pixels
    image = image.reshape((image.shape[0] * image.shape[1], 3))

    # cluster the pixel intensities
    clt = KMeans(n_clusters = clusters)
    clt.fit(image)

    # print(clt.cluster_centers_)
```

```
134
135         # show our color bart
136         return (clt.cluster_centers_, clt)
137
138     def getColors(color, n):
139         color = color.replace('/', ' ')
140
141         download_directory = 'images/' + color
142
143         # Fetch images and store in images/<color>
144         scraper = YahooScraper()
145         new = scraper.download_images(color, n)
146
147         # Read from files
148         # BGR
149         if colorSpace == 'BGR':
150             # Updated BGR file exists
151             if os.path.exists(download_directory+'/color.txt') and not new:
152                 return (genfromtxt(download_directory+'/color.txt', delimiter=',',
                    dtype=int).tolist(), None, False)
153
154             # Updated RGB file exists : Load, transform and save as BGR
155             if os.path.exists(download_directory+'/color_rgb.txt') and not new:
156                 temp = genfromtxt(download_directory+'/color_rgb.txt', delimiter=',',
                    dtype=int).tolist()
157                 temp = swap3(temp)
158                 np.savetxt(download_directory+'/color.txt', temp, delimiter=",")
159                 return (temp, None, False)
160         # RGB
161         else:
162             # Updated RGB file exists
163             if os.path.exists(download_directory+'/color_rgb.txt') and not new:
164                 return (genfromtxt(download_directory+'/color_rgb.txt', delimiter=',',
                    dtype=int).tolist(), None, False)
165
166             # Updated BGR file exists : Load, transform and save as RGB
167             if os.path.exists(download_directory+'/color.txt') and not new:
168                 temp = genfromtxt(download_directory+'/color.txt', delimiter=',',
                    dtype=int).tolist()
169                 temp = swap3(temp)
170                 np.savetxt(download_directory+'/color_rgb.txt', temp, delimiter=",")
171                 return (temp, None, False)
172
173         # Get list of resulting images
174         images = list(map(lambda x: download_directory + '/' + x,
            os.listdir(download_directory)))
175         images = [x for x in images if '.jpg' in x]
176         images = images[:n]
177
178         # No images
179         if len(images)<1:
180             return ([0], None, False)
181
182         # Separate clustering
183         if not combined:
184             dom_colors, clusters = get_dominant_colors(images)
185
186         # Combined clustering
187         if combined:
188             dom_colors, clusters = get_dominant_colors([combine_images(download_directory,
                images)])
189
190         return (dom_colors, clusters, True)
191
192     # Color (phrase, number of images, clusters per image, color treshold, clusters in
        colors only)
193     def scrape(s, n, ct, t, cd, cs, comb):
```

```
194         global clusters, colorSpace, combined
195         combined = comb
196         clusters = ct
197         colorSpace = cs
198         download_directory = 'images/' + s
199 
200         # Image colors
201         color, clts, new = getColors(s, n)
202 
203         if not new:
204             return (color, False)
205 
206         # 3D into 2D
207         final = []
208         for img in color:
209             for c in img:
210                 final.append(list(c))
211 
212         colorTreshold = t
213 
214         # Filter out colors only
215         temp = [a for a in final if np.std(a) > colorTreshold]
216 
217         # Any colors >> continue
218         if not temp:
219             return ([0], True)
220 
221         # More than 1 color >> choose dominant one
222         if len(temp) == 1:
223             result = temp[0]
224         else:
225             result = domColor(temp, cd)
226 
227         if colorSpace == 'BGR':
228             np.savetxt(download_directory+'/color.txt', result, delimiter=",")
229         else:
230             np.savetxt(download_directory+'/color_rgb.txt', result, delimiter=",")
231 
232         return (result, True)
233 
234     def domColor(d, c):
235         # Clusters
236         clt = KMeans(n_clusters = c)
237         clt.fit(d)
238 
239         # Max population of dominant colors
240         numLabels = np.arange(0, len(np.unique(clt.labels_)) + 1)
241         (h, _) = np.histogram(clt.labels_, bins = numLabels)
242         h = h.tolist()
243         m = h.index(max(h))
244 
245         return [int(i) for i in clt.cluster_centers_[m]]
246 
247     def show(c, mode):
248         c = c if mode == 'BGR' else swap3(c)
249         w = 512
250         h = 512
251 
252         img = np.zeros((w,h,3), np.uint8)
253 
254         cv2.rectangle(img, (0,0), (w,h), c, cv2.FILLED)
255 
256         cv2.imshow("Image", img)
257         cv2.waitKey(0)
258         cv2.destroyAllWindows()
259 
260     def showAll(c, mode):
```

```
261         c = c if mode == 'BGR' else [swap3(a) for a in c]
262         w = 512
263         h = 512
264         size = [int(w/len(c)), h]
265 
266         img = np.zeros((w,h,3), np.uint8)
267 
268         for j in range(len(c)):
269             cv2.rectangle(img, (size[0]*j,0), (size[0]*j+size[0],size[1]), c[j], cv2.FILLED)
270 
271         cv2.imshow("Image", img)
272         cv2.waitKey(0)
273         cv2.destroyAllWindows()
274 
275     def swap3(c):
276         c = [c[2], c[1], c[0]]
277         return c
```

Appendix C

```python
import numpy as np
from skimage import io, color, img_as_ubyte
from sklearn.metrics.cluster import entropy
import os
from numpy import genfromtxt

def e(img):
    rgbImg = io.imread(img)
    grayImg = img_as_ubyte(color.rgb2gray(rgbImg))

    return entropy(grayImg)

def ent(phrase, n, new):
    download_directory = 'images/' + phrase

    # Entrophy file exists
    if os.path.exists(download_directory+'/entrophy.txt') and not new:
        x = None
        with open(download_directory+'/entrophy.txt', 'r') as f:
            x = float(f.readline())
            f.close()
        if x:
            return x

    # Get list of resulting images
    images = list(map(lambda x: download_directory + '/' + x,
    os.listdir(download_directory)))
    images = [x for x in images if '.jpg' in x]
    images = images[:n]

    if len(images)<1:
        return 0

    x = 0
    for image in images:
        x = x + e(image)
    x = x/len(images)

    # Save entrophy file
    with open(download_directory+'/entrophy.txt', 'w') as f:
        f.write('%f' % x)
        f.close()

    return x
```

Appendix D

In [8]:

```
import urllib.request, json
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from textblob import TextBlob
from textblob import Word
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from gensim.scripts.glove2word2vec import glove2word2vec
from gensim.models import KeyedVectors
import warnings
import re
from string import punctuation

warnings.filterwarnings("ignore", category=FutureWarning)

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

```
C:\Users\OndrejSpetko\Anaconda3\lib\site-packages\gensim\utils.py:1209: Us
erWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\OndrejSpetko\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\OndrejSpetko\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\OndrejSpetko\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[8]:

```
True
```

In [3]:

```
limit = 1000
#mode = 'duckling'
mode = 'imbd'
```

In [6]:

```
##JSON

if mode == 'duckling':
    # tags = 'tech%2Cfood%2Clove%2Cfamily'
    tags = ''

    u = "https://api.duckling.me/v3/public/api/stories?tags="+tags+"&offset=0&limit="+s
tr(limit)+"&fbclid=IwAR0LTjyryCcI5Y_5AXdGOQeKWxfAL4n-L6JcIWSphnBDB4qgqg2uOk7q_CQ";
    with urllib.request.urlopen(u) as url:
        data = json.loads(url.read())
    #data
    len(data['stories'])
print(mode)
```

imbd

In [15]:

```
##JSON to DataFrame
##User ID?
##Short text provides no info

if mode == 'duckling':
    minTextLen = 4
    minTagLen = 4

    stories = data['stories']
    nodes = []
    count = 0
    #final = ""

    for story in stories :
        slides = story['story']['slides']
        tags = story['story']['tags']
        content = ""
        content2 = ""

        #content
        for slide in slides :

            f = None
            if 'layers' in slide['slide'] :
                f = slide['slide']['layers']['foreground']['layer']['content']
            #let b = slide.layers.background.layer.content.overlayText
            #print(f)
            if f and 'overlayText' in f:
                content += f['overlayText']['text']+" "
        #tags
        for tag in tags :

            f = None
            if 'text' in tag['tag'] :
                f = tag['tag']['text']
            #print(f)
            if f :
                content2 += f+" "

        if len(content) > minTextLen :
            count = count + 1
            nodes.append({"text":content, "tags":content2 if len(content2)>minTagLen el
se " "})

        if count == limit:
            break

    #final = final.join(nodes)
    #type(nodes[0])
    df = pd.DataFrame(nodes)

if mode == 'imbd':
    d = pd.read_csv('IMBD.tsv', '\t')[:limit]
    df = pd.DataFrame(data={'text':d['review'], 'sentibin':d['sentiment']})

##Save raw

df.to_csv(r'dataRaw.csv',index=False)
```

```
# df.index
# df['text'][3]
print(df.index)
df.head()
```

```
RangeIndex(start=0, stop=1000, step=1)
```

Out[15]:

	sentibin	text
0	1	With all this stuff going down at the moment w...
1	1	\The Classic War of the Worlds\" by Timothy Hi...
2	0	The film starts with a manager (Nicholas Bell)...
3	0	It must be assumed that those who praised this...
4	1	Superbly trashy and wondrously unpretentious 8...

In [16]:

```
## Remove "." where it glues sentances/words

df['text'] = df['text'].apply(lambda x: " "+re.sub('(?<=[a-zA-Z])[\.](?=[a-zA-Z])', '',
x, re.I|re.A))

df['text'][4]
```

Out[16]:

```
' Superbly trashy and wondrously unpretentious 80\'s exploitation, hooray!
The pre-credits opening sequences somewhat give the false impression that
we\'re dealing with a serious and harrowing drama, but you need not fear b
ecause barely ten minutes later we\'re up until our necks in nonsensical c
hainsaw battles, rough fist-fights, lurid dialogs and gratuitous nudity! B
o and Ingrid are two orphaned siblings with an unusually close and even sl
ightly perverted relationship. Can you imagine playfully ripping off the t
owel that covers your sister\'s naked body and then stare at her unshaven
genitals for several whole minutes? Well, Bo does that to his sister and,
judging by her dubbed laughter, she doesn\'t mind at all. Sick, dude! Anyw
ay, as kids they fled from Russia with their parents, but nasty soldiers b
rutally slaughtered mommy and daddy. A friendly smuggler took custody over
them, however, and even raised and trained Bo and Ingrid into expert smugg
lers. When the actual plot lifts off, 20 years later, they\'re facing thei
r ultimate quest as the mythical and incredibly valuable White Fire diamon
d is coincidentally found in a mine. Very few things in life ever made as
little sense as the plot and narrative structure of \\White Fire\\", but i
t sure is a lot of fun to watch. Most of the time you have no clue who\'s
beating up who or for what cause (and I bet the actors understood even les
s) but whatever! The violence is magnificently grotesque and every single
plot twist is pleasingly retarded. The script goes totally bonkers beyond
repair when suddenly \x96 and I won\'t reveal for what reason \x96 Bo need
s a replacement for Ingrid and Fred Williamson enters the scene with a big
cigar in his mouth and his sleazy black fingers all over the local prostit
utes. Bo\'s principal opponent is an Italian chick with big breasts but a
hideous accent, the preposterous but catchy theme song plays at least a do
zen times throughout the film, there\'s the obligatory \\"we\'re-falling-i
n-love\\" montage and loads of other attractions! My God, what a brilliant
experience. The original French title translates itself as \\"Life to Surv
ive\\", which is uniquely appropriate because it makes just as much sense
as the rest of the movie: None!"'
```

In [17]:

```
## Punctuation, digits, ?lowercase(if not sentiment)

remove_terms = punctuation + '0123456789'

#df['text'] = df['text'].apply(lambda x: " ".join(("".join(z for z in y if z not in rem
ove_terms)).lower() for y in x.split()))
df['text'] = df['text'].apply(lambda x: " ".join(("".join(z for z in y if z not in remo
ve_terms)) for y in x.split()))

df
df['text'][4]
```

Out[17]:

'Superbly trashy and wondrously unpretentious s exploitation hooray The pr
ecredits opening sequences somewhat give the false impression that were de
aling with a serious and harrowing drama but you need not fear because bar
ely ten minutes later were up until our necks in nonsensical chainsaw batt
les rough fistfights lurid dialogs and gratuitous nudity Bo and Ingrid are
two orphaned siblings with an unusually close and even slightly perverted
relationship Can you imagine playfully ripping off the towel that covers y
our sisters naked body and then stare at her unshaven genitals for several
whole minutes Well Bo does that to his sister and judging by her dubbed la
ughter she doesnt mind at all Sick dude Anyway as kids they fled from Russ
ia with their parents but nasty soldiers brutally slaughtered mommy and da
ddy A friendly smuggler took custody over them however and even raised and
trained Bo and Ingrid into expert smugglers When the actual plot lifts off
years later theyre facing their ultimate quest as the mythical and incredi
bly valuable White Fire diamond is coincidentally found in a mine Very few
things in life ever made as little sense as the plot and narrative structu
re of White Fire but it sure is a lot of fun to watch Most of the time you
have no clue whos beating up who or for what cause and I bet the actors un
derstood even less but whatever The violence is magnificently grotesque an
d every single plot twist is pleasingly retarded The script goes totally b
onkers beyond repair when suddenly \x96 and I wont reveal for what reason
\x96 Bo needs a replacement for Ingrid and Fred Williamson enters the scen
e with a big cigar in his mouth and his sleazy black fingers all over the
local prostitutes Bos principal opponent is an Italian chick with big brea
sts but a hideous accent the preposterous but catchy theme song plays at l
east a dozen times throughout the film theres the obligatory werefallingin
love montage and loads of other attractions My God what a brilliant experi
ence The original French title translates itself as Life to Survive which
is uniquely appropriate because it makes just as much sense as the rest of
the movie None'

In [18]:

```
##Special chars, whitespaces, length>=4

wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('english')

def process_doc(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I|re.A)
    #doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = wpt.tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words and len(tok
en)>=4]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc


df['text'] = df['text'].apply(lambda x: process_doc(x))

df
df['text'][4]
```

Out[18]:

```
'Superbly trashy wondrously unpretentious exploitation hooray precredits o
pening sequences somewhat give false impression dealing serious harrowing
drama need fear barely minutes later necks nonsensical chainsaw battles ro
ugh fistfights lurid dialogs gratuitous nudity Ingrid orphaned siblings un
usually close even slightly perverted relationship imagine playfully rippi
ng towel covers sisters naked body stare unshaven genitals several whole m
inutes Well sister judging dubbed laughter doesnt mind Sick dude Anyway ki
ds fled Russia parents nasty soldiers brutally slaughtered mommy daddy fri
endly smuggler took custody however even raised trained Ingrid expert smug
glers When actual plot lifts years later theyre facing ultimate quest myth
ical incredibly valuable White Fire diamond coincidentally found mine Very
things life ever made little sense plot narrative structure White Fire sur
e watch Most time clue whos beating cause actors understood even less what
ever violence magnificently grotesque every single plot twist pleasingly r
etarded script goes totally bonkers beyond repair suddenly wont reveal rea
son needs replacement Ingrid Fred Williamson enters scene cigar mouth slea
zy black fingers local prostitutes principal opponent Italian chick breast
s hideous accent preposterous catchy theme song plays least dozen times th
roughout film theres obligatory werefallinginlove montage loads attraction
s brilliant experience original French title translates Life Survive uniqu
ely appropriate makes much sense rest movie None'
```

In [20]:

```
## Lemmatization

df['text'] = df['text'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.
split()]))
df['text'][4]
```

Out[20]:

```
'Superbly trashy wondrously unpretentious exploitation hooray precredits o
pening sequence somewhat give false impression dealing serious harrowing d
rama need fear barely minute later neck nonsensical chainsaw battle rough
fistfight lurid dialog gratuitous nudity Ingrid orphaned sibling unusually
close even slightly perverted relationship imagine playfully ripping towel
cover sister naked body stare unshaven genitals several whole minute Well
sister judging dubbed laughter doesnt mind Sick dude Anyway kid fled Russi
a parent nasty soldier brutally slaughtered mommy daddy friendly smuggler
took custody however even raised trained Ingrid expert smuggler When actua
l plot lift year later theyre facing ultimate quest mythical incredibly va
luable White Fire diamond coincidentally found mine Very thing life ever m
ade little sense plot narrative structure White Fire sure watch Most time
clue who beating cause actor understood even le whatever violence magnific
ently grotesque every single plot twist pleasingly retarded script go tota
lly bonkers beyond repair suddenly wont reveal reason need replacement Ing
rid Fred Williamson enters scene cigar mouth sleazy black finger local pro
stitute principal opponent Italian chick breast hideous accent preposterou
s catchy theme song play least dozen time throughout film there obligatory
werefallinginlove montage load attraction brilliant experience original Fr
ench title translates Life Survive uniquely appropriate make much sense re
st movie None'
```

In [21]:

```
## NaN clear

temp = df[df['text']==""].index
l = int(len(temp))

df = df.drop(temp).reset_index(drop=True)
print(len(temp))
```

```
0
```

In [22]:

```
##Save

df.to_csv(r'data.csv',index=False)
```

Appendix E

In [ ]:

```
import urllib.request, json
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from textblob import TextBlob
from textblob import Word
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from gensim.scripts.glove2word2vec import glove2word2vec
from gensim.models import KeyedVectors
import warnings
import re
from string import punctuation
from IPython.display import clear_output
import importlib
import matplotlib.pyplot as plt
import matplotlib as mpl

warnings.filterwarnings("ignore", category=FutureWarning)

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('brown')
```

Data

In [ ]:

```
# mode = 'duckling'
mode = 'imbd'
```

In [ ]:

```
# ## Load IMBD raw 20
# df = pd.read_csv('data.csv')
# df
```

In [ ]:

```
## Load pre-processed data
df = pd.read_csv('data.csv')
df
```

In [ ]:

```
df['text'][0]
```

In [ ]:

```
## Load raw data
dfR = pd.read_csv('dataRaw.csv')
dfR
```

In [ ]:

```
dfR['text'][0]
```

In [ ]:

```
# temp = [a for a in TextBlob(df['text'][0]).noun_phrases if len(a.split())>1 and len
(a.split())<4]
# # temp = TextBlob(df['text'][0]).noun_phrases
# temp
```

In [ ]:

```
# temp = nltk.word_tokenize(dfR['text'][40])
# nltk.pos_tag(temp)
```

In [ ]:

```
##Tokenize
# df['tokens'] = df['text'].apply(lambda x: list(pd.Series(x.split())))
df['tokens'] = df['text'].apply(lambda x: list(nltk.word_tokenize(x)))
dfR['tokens'] = dfR['text'].apply(lambda x: list(nltk.word_tokenize(x)))
df.head()
```

Glove

In [ ]:

```
## word2vec from pre-trained data: glove (100 dimensions)
dim = 300
dimS = str(dim)

glove_input_file = 'glove.6B.'+dimS+'d.txt'
word2vec_output_file = 'glove.6B.'+dimS+'d.txt.word2vec'
glove2word2vec(glove_input_file, word2vec_output_file)
```

In [ ]:

```
filename = 'glove.6B.'+dimS+'d.txt.word2vec'
model = KeyedVectors.load_word2vec_format(filename, binary=False)
#(model['go'] + model['away'])/2
#model.most_similar(positive=['woman', 'king'], negative=['man'])[:1]
```

In [ ]:

```
## Words, count columns
df['words'] = df['text'].apply(lambda x: list(pd.Series(x.split()).value_counts().index
))
df['counts'] = df['text'].apply(lambda x: list(pd.Series(x.split()).value_counts().valu
es))
df['wordsT'] = df['text'].apply(lambda x: len(pd.Series(x.split()).value_counts().value
s))

df.head()
```

Feature: Topic

In [ ]:

```
# model['car']
```

In [ ]:

```
## Average vector per document: topicV

def average_word_vectors(words, counts):

    global model, dim
    feature_vector = np.zeros(dim,dtype="float64")
    nwords = 0
    vocab = model.vocab

    for w in range(len(words)):
        word = words[w]
        if word in vocab:
            c = counts[w]
            nwords = nwords + c
            feature_vector = np.add(feature_vector, model[word]*c)

    if nwords:
        feature_vector = np.divide(feature_vector, nwords)

    return feature_vector

# df['topicV'] = df['tokens'].apply(lambda x: average_word_vectors(x, model, dim))

temp = []
wordsLimit = 5

for line in range(len(df)):
    temp.append(average_word_vectors(df['words'][line][:wordsLimit], df['counts'][line]
[:wordsLimit]))


# type(temp[:5])
topicV = pd.DataFrame({'topicV': temp})
df['topicV'] = topicV
topicV = pd.DataFrame(topicV.topicV.values.tolist(), index= topicV.index)
# topicV.head()

df.head()
```

In [ ]:

```
## Top n similar words to the average of a document
topn=10

df['topicWS'] = df['topicV'].apply(lambda x: np.array(model.most_similar([np.array(x)],
topn=topn))[:, 0])
df
```

In [ ]:

```
## Affinity propagation for unsupervised clustering: Similarities

from sklearn.cluster import AffinityPropagation

ap = AffinityPropagation()
ap.fit(topicV)
cluster_labels = ap.labels_
cluster_labels = pd.DataFrame(cluster_labels, columns=['cluster'])
df = pd.concat([df, cluster_labels], axis=1)
df
```

In [ ]:

```
# inc = int(255/np.array(df['cluster'].tolist()).max())
# inc
```

In [ ]:

```
## PCA from topic vectors

from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import matplotlib as mpl

pca = PCA(n_components=2, random_state=0)
pcs = pca.fit_transform(topicV)
```

In [ ]:

```
# ## K-means
# from sklearn.cluster import KMeans

# kmeans = KMeans(n_clusters=10, random_state=0).fit_transform(topicV)
```

In [ ]:

```
import plotly
import plotly.graph_objs as go

plotly.offline.init_notebook_mode(connected=True)

trace = go.Scatter(
    x = pcs[:,0],
    y = pcs[:,1],
    mode = 'markers',
    hoverinfo = 'text',
    text = list(df.index),
    marker = dict(
#         color = df['cluster'].tolist(),
        colorbar=dict(
                title='Colorbar'
            ),
        colorscale='Viridis'
    )
)

plotly.offline.iplot({
    "data": [trace],
    "layout": go.Layout(title="PCA of topic vectors from IMBD reviews (color=clusters)"
)
})
```

In [ ]:

```
from sklearn.manifold import TSNE

X_embedded = TSNE(n_components=2).fit_transform(topicV)
```

In [ ]:

```
trace = go.Scatter(
    x = X_embedded[:,0],
    y = X_embedded[:,1],
    mode = 'markers',
    hoverinfo = 'text',
    text = list(df.index),
    marker = dict(
        color = df['cluster'].tolist(),
        colorbar=dict(
                title='Colorbar'
            ),
        colorscale='Viridis'
    )
)

plotly.offline.iplot({
    "data": [trace],
    "layout": go.Layout(title="t-SNE of topic vectors of IMBD reviews (color=clusters)"
)
})
```

Feature: Sentiment

In [ ]:

```
# from nltk import tokenize

# def vaderThis(paragraph):
#     sentence_list = tokenize.sent_tokenize(paragraph)
#     paragraphSentiments = 0.0
#     for sentence in sentence_list:
#         vs = analyzer.polarity_scores(sentence)
#         paragraphSentiments += vs["compound"]
#     return round(paragraphSentiments / len(sentence_list), 4)
```

In [ ]:

```
## vaderSentiment on Raw data (emoticons and punctuaction)

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()
# Paragraph level
df['sentiment'] = dfR['text'].apply(lambda x: analyzer.polarity_scores(x)['compound'])
# Sentence level
# df['sentiment'] = dfR['text'].apply(lambda x: vaderThis(x))

df['sentiment'].head()
```

In [ ]:

```
## Vader sentiment precision against labeled IMBD dataset sentiment
# vader/nrc
senti = 'vader'

if mode == 'imbd' :
    if senti == 'nrc':
        temp = pd.DataFrame(df)
        temp['v'] = temp['VA'].apply(lambda x: x[0])
        temp['v'] = pd.DataFrame(data={'v':np.interp(np.array(temp.v.values.tolist()),
(0, 1), (-1, +1))})

        pos = temp[temp['v'] > 0]
        posA = np.array(pos.sentiment.values.tolist()).mean()

        posT = pos[pos['sentibin'] == 1]
        posTA = np.array(posT.sentiment.values.tolist()).mean()

        posN = pos[pos['sentibin'] == 0]
        posNA = np.array(posN.sentiment.values.tolist()).mean()

        posAcc = len(posT)/len(pos)*100

        neg = temp[temp['v'] < 0]
        negA = np.array(neg.sentiment.values.tolist()).mean()

        negT = neg[neg['sentibin'] == 0]
        negTA = np.array(negT.sentiment.values.tolist()).mean()

        negN = neg[neg['sentibin'] == 1]
        negNA = np.array(negN.sentiment.values.tolist()).mean()

        negAcc = len(negT)/len(neg)*100

    if senti == 'vader':

        pos = df[df['sentiment'] > 0]
        posA = np.array(pos.sentiment.values.tolist()).mean()

        posT = pos[pos['sentibin'] == 1]
        posTA = np.array(posT.sentiment.values.tolist()).mean()

        posN = pos[pos['sentibin'] == 0]
        posNA = np.array(posN.sentiment.values.tolist()).mean()

        posAcc = len(posT)/len(pos)*100

        neg = df[df['sentiment'] < 0]
        negA = np.array(neg.sentiment.values.tolist()).mean()

        negT = neg[neg['sentibin'] == 0]
        negTA = np.array(negT.sentiment.values.tolist()).mean()

        negN = neg[neg['sentibin'] == 1]
        negNA = np.array(negN.sentiment.values.tolist()).mean()

        negAcc = len(negT)/len(neg)*100

    print('Sentiment: '+senti)
    print('---')
```

```
    print("Positive: "+str(len(pos))+" ~ "+str("%.2f" % posA))
    print("PositiveT: "+str(len(posT))+" ~ "+str("%.2f" % posTA))
    print("PositiveN: "+str(len(posN))+" ~ "+str("%.2f" % posNA))
    print("Acc: "+str("%.2f" % posAcc)+" %")

    print("---")

    print("Negative: "+str(len(neg))+" ~ "+str("%.2f" % negA))
    print("NegativeT: "+str(len(neg[neg['sentibin'] == 0]))+" ~ "+str("%.2f" % negTA))
    print("NegativeN: "+str(len(neg[neg['sentibin'] == 1]))+" ~ "+str("%.2f" % negNA))
    print("Acc: "+str("%.2f" % negAcc)+" %")

    print("---")

    print(str("%.2f" % ((posAcc+negAcc)/2))+" %")
```

In [ ]:

```
trace = go.Scatter(
    x = X_embedded[:,0],
    y = X_embedded[:,1],
    mode = 'markers',
    hoverinfo = 'text',
    text = list(df.index),
    marker = dict(
        color = df['sentiment'].tolist(),
        colorbar=dict(
                title='Colorbar'
            ),
        colorscale='Viridis'
    )
)

plotly.offline.iplot({
    "data": [trace],
    "layout": go.Layout(title="t-SNE of topic vectors of IMBD reviews (color=sentimen
t)")
})
```

In [ ]:

```
## Heatmap of sentiments

import seaborn as sns;
sns.set()

l = len(df.groupby('cluster').groups.keys())
l

temp = []
clusters = []
for i in range(l):
    temp.append([])
    clusters.append([])

for i in range(l):
    index = df['cluster'].apply(lambda x: x==i)
    temp[i] = np.asarray(index)

temp = np.asarray(temp)
temp.shape

for c in range(l):
    for index in range(len(temp[c])):
        if temp[c][index]:
            #clusters[c].append({'cluster':c, 'sentiment':df['sentiment'][index]})
            clusters[c].append(df['sentiment'][index])
        else :
            clusters[c].append(-1.1)

ax = sns.heatmap(clusters, center=0)
```

Feature: Subjectivity

In [ ]:

```
## Returns average subj 0-1
def subjThis(sentences):
    temp = 0.0
    for sentence in sentences:
        temp = temp + sentence.sentiment.subjectivity
    return temp/len(sentences)

def polarThis(sentences):
    temp = 0.0
    for sentence in sentences:
        temp = temp + sentence.sentiment.polarity
    return temp/len(sentences)
```

In [ ]:

```
## TextBlob seubjectivity on raw

# Paragraph level
# df['subjectivity'] = dfR['text'].apply(lambda x: TextBlob(x).sentiment.subjectivity)
# Sentence level
df['subjectivity'] = dfR['text'].apply(lambda x: subjThis(TextBlob(x).sentences))
# df['sentiment'] = dfR['text'].apply(lambda x: polarThis(TextBlob(x).sentences))
df.head()
```

In [ ]:

```
df['sentiment'].min()
```

In [ ]:

```
trace = go.Scatter(
    x = df['subjectivity'].tolist(),
    y = X_embedded[:,1],
    mode = 'markers',
    hoverinfo = 'text',
    text = list(df.index),
    marker = dict(
        color = df['subjectivity'].tolist(),
        colorbar=dict(
                title='Colorbar'
            ),
        colorscale='Viridis'
    )
)

plotly.offline.iplot({
    "data": [trace],
    "layout": go.Layout(title="t-SNE of topic vectors of IMBD reviews ordered by subjec
tivity [X Axis] (color=subjectivity)")
})
```

Feature: Arousal

In [ ]:

```
## Load NRC-VAD lexicon
file = "NRC\\NRC-VAD-Lexicon\\NRC-VAD-Lexicon.txt"
dfL = pd.read_csv(file,'\t')
dfL.head()
```

In [ ]:

```
## NRC-VA from tokens
# iters = 0
# length = len(df)
# def getVA(tokens):
#     global iters, length
#     vec = np.zeros(2)
#     count = 0
#     for token in tokens:
#         tokenL = token.lower()
#         if (dfL['word'] == tokenL).any():
#             temp = dfL[dfL['word'] == tokenL].as_matrix(columns=dfL.columns[1:3])
#             vad = np.array(temp[0])
#             count = count + 1
# #             print(tokenL, temp, vec, vad)
#             vec =  vec + vad
#     clear_output(wait=True)
#     iters = iters + 1
#     print("("+str(iters)+") "+str("%.0f" % (iters/length*100))+"%")
#     return np.divide(vec, count) if count>0 else np.array([0.5, 0.5])
```

In [ ]:

```
## NRC-VA Arousal from unique words
length = len(df)
def getVA(tokens, counts):
    global iters, length
    vec = np.zeros(2)
    count = 0
    for t in range(len(tokens)):
        tokenL = tokens[t].lower()
        if (dfL['word'] == tokenL).any():
            temp = dfL[dfL['word'] == tokenL].as_matrix(columns=dfL.columns[1:3])
            c = counts[t]
            vad = np.array(temp[0])*c
            count = count + c
#             print(tokenL, temp, vec, vad, c, count)
            vec =  vec + vad
    r = np.divide(vec, count)
    clear_output(wait=True)
    iters = iters + 1
    print("("+str(iters)+") "+str("%.0f" % (iters/length*100))+"%")
    return r[1] if count>0 else 0.5
```

In [ ]:

```
# iters = 0
# df[:5].apply(lambda x: getVA(x['words'], x['counts']) if len(x['words']) < wordsLimit
else getVA(x['words'][:wordsLimit], x['counts']), axis=1)
```

In [ ]:

```
iters = 0
wordsLimit = 5
# df['VA'] = df['tokens'].apply(lambda x: getVA(x))
df['arousal'] = df.apply(lambda x: getVA(x['words'], x['counts']) if len(x['words']) <
wordsLimit else getVA(x['words'][:wordsLimit], x['counts']), axis=1)
df.head()
```

In [ ]:

```
# x = np.array(df.sentiment.values.tolist())
# np.var(x)
```

In [ ]:

```
import plotly
import plotly.graph_objs as go

plotly.offline.init_notebook_mode(connected=True)

temp = df.arousal.values

# x = temp[:,0]
# x = np.interp(x, (0, 1), (-1, +1))
x = np.array(df.sentiment.values.tolist())

y = temp
y = np.interp(y, (0, 1), (-1, +1))

# z = np.array(df.sentiment.values.tolist())

trace = go.Scatter(
    x = x,
    y = y,
    mode = 'markers',
    hoverinfo = 'text',
    text = list(df.index),
    marker = dict(
        color = 'blue',
        colorscale='Viridis'
    )
)

plotly.offline.iplot({
    "data": [trace],
    "layout": go.Layout(title="NRC-VA distribution of stories in space of Circumplex of
Affection as clusters")
})
```

In [ ]:

```
dfR['text'][71]
```

In [ ]:

```
# temp = np.array(df.VA.values.tolist())
emoCut = 0.1

# x = temp[:,0]
# x = np.interp(x, (0, 1), (-1, +1))
x = np.array(df.sentiment.values.tolist())

y = np.array(df.arousal.values.tolist())
y = np.interp(y, (0, 1), (-1, +1))

r = np.clip(y, 0, 1)
r = np.where(r < emoCut, 0, r)

g = np.clip(x, 0, 1)
g = np.where(g < emoCut, 0, g)

bb = np.clip(x, -1, 0)
b = np.interp(bb, (-1, 0), (+1, 0))
b = np.where(b < emoCut, 0, b)

# plt.figure(figsize=(12, 12))
# # plt.margins(.1, .1)
# plt.title('NRC-VA distribution of stories in space of Circumplex of Affection as emot
ions > 0.2')

# for i in range(len(x)):
#     m = (r[i]+b[i]+g[i])/1
#     m = m if m>0.1 else 0
#     m = m if m<=1 else 1
#     color = (r[i], g[i], b[i], m)
#     #annotation_label = str(i)+'-'+str(ap.labels_[i])
#     xx = x[i]
#     yy = y[i]
#     plt.scatter(xx, yy, c=color)
#     #plt.scatter(xx, yy, c=color, edgecolors='k')
#     #plt.annotate(annotation_label, xy=(xx+1e-4, yy+1e-3), xytext=(0, 0), textcoords
='offset points')

# plt.show()
```

In [ ]:

```
X_embedded = TSNE(n_components=2).fit_transform(topicV)
```

In [ ]:

```
# print(math.degrees((np.arctan2([1,1], [1, 0]))[0]))
```

In [ ]:

```
temp = np.interp(X_embedded, (X_embedded.min(), X_embedded.max()), (-1, +1))
```

In [ ]:

```
wMult = 1
tMult = 1
#full vectors
# dfF = pd.DataFrame(topicV.values.tolist(), index= topicV.index)
#tsne n= 2 of vectors
dfF = pd.DataFrame(temp*tMult, index= topicV.index)
# dfF = pd.DataFrame(X_embedded*tMult, index= topicV.index)
dfF['valence'] = pd.DataFrame(x*wMult)
dfF['arousal'] = pd.DataFrame(y*wMult)
dfF['subjectivity'] = pd.DataFrame(df.subjectivity.values.tolist()*wMult, index=df.inde
x)
dfF.head()
```

In [ ]:

```
# ## copy features
# dfF = pd.concat([dfF, dfF['valence'], dfF['arousal'], dfF['valence'], dfF['arousal'],
dfF['valence'], dfF['arousal'], dfF['valence'], dfF['arousal'], dfF['valence'], dfF['ar
ousal']], axis=1)
# dfF.head()
```

In [ ]:

```
# ##PCA from topic vectors, valence, arousal

# from sklearn.decomposition import PCA
# import matplotlib.pyplot as plt
# import matplotlib as mpl

# pca = PCA(n_components=2, random_state=0).fit_transform(dfF)
```

In [ ]:

```
# labels = ap.labels_
# categories = list(df.index)
# plt.figure(figsize=(15, 13))
# plt.title('PCA distribution of IMBD reviews by Arousal, Sentiment and topic (color=se
ntiment)')

# for i in range(len(labels)):
#     m = (r[i]+b[i]+g[i])/1
#     m = m if m>0.1 else 0
#     m = m if m<=1 else 1
#     color = (r[i], g[i], b[i], m)
#     #annotation_label = categories[i]
#     x, y = pca[i]
#     plt.scatter(x, y, c=color)
#     #plt.annotate(annotation_label, xy=(x+1e-4, y+1e-3), xytext=(0, 0), textcoords='o
ffset points')

# plt.show()
```

In [ ]:

```
from sklearn.manifold import TSNE

X_embedded2 = TSNE(n_components=2, perplexity=30, n_iter=1000).fit_transform(dfF)
```

In [ ]:

```
labels = ap.labels_
categories = list(df.cluster)
plt.figure(figsize=(15, 13))
plt.title('t-SNE distribution of IMBD reviews by Arousal(1), Sentiment(1) and Topic(2)
 (color=sentiment)')

for i in range(len(labels)):
    m = (r[i]+b[i]+g[i])/1
    m = m if m>0.1 else 0
    m = m if m<=1 else 1
#     color = (r[i], g[i], b[i], m)
    color = (r[i], g[i], b[i], 1)
    annotation_label = i
    x, y = X_embedded2[i]
    plt.scatter(x, y, c=color)
    plt.annotate(annotation_label, xy=(x+1e-4, y+1e-3), xytext=(0, 0), textcoords='offs
et points')

plt.show()
```

In [ ]:

```
dfR['text'][609]
```

(Image) Feature: Color, Entrophy

In [ ]:

```
# temp = [a for a in TextBlob(df['text'][0]).noun_phrases if len(a.split())>1 and len
(a.split())<4]
# len(temp), temp
```

In [ ]:

```
# Phrase count
# count = 0
# iters = 0
# for text in df['text']:
#     iters = iters +1
#     count = count + len([a for a in TextBlob(text).noun_phrases if len(a.split())>1 a
nd len(a.split())<4])
#     clear_output(wait=True)
#     print(iters/len(df)*100, '%')
# count
```

In [ ]:

```
# Scraper hook function
from Scraper import scraper
from Entrophy import entrophy
import colorsys
importlib.reload(scraper)
importlib.reload(entrophy)

# BGR!
# scrape(phrase, number of images, clusters per image, color treshold, clusters in colo
rs only)
# c = scraper.scrape('tranquility', 5, 5, 35, 2)

def color_entrophy(text, row, total, chunk, chunkSize, chunks):
    combined = True
    imgClusters = 5
    colTreshold = 35
    colOnlyClusters = 2
    numImages = 5
    phraseCountLimit = 10
    temp = [a for a in TextBlob(text).noun_phrases if len(a.split())>1 and len(a.split
())<4]
    temp = temp[:phraseCountLimit]
    mode = 'RGB'
    colors = []
    ent = 0
    faults = []
    iters = 0

#     print('Total: (', row, '/', total, ') ', row/total, ' %')

    for phrase in temp:
        s, new = scraper.scrape(phrase, numImages, imgClusters, colTreshold, colOnlyClu
sters, mode, combined)
        e = entrophy.ent(phrase, numImages, new)

        # Color add
        if len(s)>1:
            colors.append(s)
        else:
            faults.append(phrase)

        # Entrophy add
        ent = ent + e

        iters = iters + 1
        clear_output(wait=True)
        print('Total:  (', row+(chunk*chunkSize), '/', chunkSize*chunks, ') ', ((row+ch
unk*chunkSize)/(chunkSize*chunks))*100, ' %')
        print('Chunk:    (', row, '/', total, ') ', (row/total)*100, ' %')
        print("Phrase: (",iters,"/", len(temp), ") : [",phrase,", ",s,", ",e,"]")
#         print("\t(",iters,"/", len(temp), ": ",phrase,", ",s,", ",e,") ")

    # Final color
    if len(colors)>1:
        c = scraper.domColor(colors, colOnlyClusters)
    else:
        if len(colors)>0:
            c = colors[0]
        else:
```

```
            c = [0,0,0]

    # RGB normalized
    rgbn = (np.array(c)/255).tolist()
    if mode == 'BGR':
        rgbn = scraper.swap3(rgbn)

    # RGB to HSV
    c = colorsys.rgb_to_hsv(rgbn[0], rgbn[1], rgbn[2])

    # Final entrophy
    if iters > 0:
        ent = ent/iters

#     print('Phrases:', len(temp), 'True:', len(colors), ' False:', len(faults), 'Colo
r:', c, ' Entrophy:', e)

    return pd.Series({'h': c[0], 's': c[1], 'v':c[2], 'entrophy': ent})
```

In [ ]:

```
# %%time
# imgClusters = 5
# colTreshold = 35
# colOnlyClusters = 2
# numImages = 5
# scraper.scrape('tranquility', numImages, imgClusters, colTreshold, colOnlyClusters,
 'RGB', True)
```

In [ ]:

```
# source = pd.DataFrame(data=df['text'][:2])
# source
# temp = source.apply(lambda x: color_entrophy(x['text'], x.name+1, len(source)), axis=
1)
# df = pd.concat([df, temp], axis=1, join='inner')
# df.head()
```

In [ ]:

```
# result = pd.concat([temp,temp]).reset_index(drop=True)
# result
```

In [ ]:

```
# test = pd.DataFrame(data={'h':[], 's':[], 'v':[], 'entrophy':[]})
# result = pd.concat([result,test]).reset_index(drop=True)
# result
```

In [ ]:

```
# # 500 done
# # 150 Full
# # 350 restricted(10 phrases per text)
# wall = 500
# chunkSize = 50
# packSize = 500

# # Final
# result = pd.DataFrame(data={'h':[], 's':[], 'v':[], 'entrophy':[]})
# # print(result)

# # Chunks (each chunk starts with double check on 1st row; use chunkSize>5)
# for chunk in range(int(packSize/chunkSize)):
#     pivot = int(chunk * chunkSize)+wall
#     source = pd.DataFrame(data=df['text'][pivot:(chunk * chunkSize + chunkSize + wal
l)])
#     temp = source.apply(lambda x: color_entrophy(x['text'], x.name-pivot+1, len(sourc
e), chunk, chunkSize, int(packSize/chunkSize)), axis=1)
#     result = pd.concat([result,temp]).reset_index(drop=True)

# result
```

In [ ]:

```
##Save

# result.to_csv(r'result5002.csv',index=False)
```

In [ ]:

```
# # Load
# resultT = pd.read_csv('result500.csv')
# # resultT
# # result2 = pd.concat([result, resultT])
# result2 = pd.read_csv('result1000.csv')
# result2
```

In [ ]:

```
# resultF = result2.reset_index(drop=True)
# resultF
```

In [ ]:

```
##Save

# resultF.to_csv(r'result1000.csv',index=False)
```

In [ ]:

```
result = pd.read_csv('result1000.csv')
```

In [ ]:

```
## Interpolate entrophy

# Entrophy
zz = np.array(result.entrophy.values.tolist())
zz = pd.DataFrame(np.interp(zz, (zz.min(), zz.max()), (-1, +1)))

# h
hh = np.array(result.h.values.tolist())
hh = pd.DataFrame(np.interp(hh, (hh.min(), hh.max()), (-1, +1)))

# s
ss = np.array(result.s.values.tolist())
ss = pd.DataFrame(np.interp(ss, (ss.min(), ss.max()), (-1, +1)))

# v
vv = np.array(result.v.values.tolist())
vv = pd.DataFrame(np.interp(vv, (vv.min(), vv.max()), (-1, +1)))
```

In [ ]:

```
# # 500
# # HSV >> TSNE >> 2?
# eMult = 0.25

# # Join
# temp = pd.DataFrame(dfF[:len(result)])
# dfF2 = pd.concat([temp,result], axis=1).reset_index(drop=True)

# # Mult?
# # dfF2['entrophy'] = pd.DataFrame([a*eMult for a in dfF2.entrophy.values.tolist()])
# dfF2['entrophy'] = zz
# dfF2['h'] = hh
# dfF2['s'] = ss
# dfF2['v'] = vv

# dfF2.head()
```

In [ ]:

```
# ##Save

# dfF2.to_csv('dfF2.csv',index=False)
```

In [ ]:

```
from sklearn.decomposition import PCA

resultPCA = PCA(n_components=2, random_state=0).fit_transform(pd.DataFrame(data=dfF2[[
'h','s','v']]))
```

In [ ]:

```
# from sklearn.manifold import TSNE

# resultTSNE = TSNE(n_components=2, perplexity=30, n_iter=1000).fit_transform(result)
```

In [ ]:

```
s1 = pd.DataFrame(resultPCA[:,0])
s2 = pd.DataFrame(resultPCA[:,1])
```

In [ ]:

```
dfF2.head()
```

Feature Selection

In [ ]:

```
dfF2 = pd.read_csv('dfF2.csv')
```

In [ ]:

```
## Feature list

dfF2
```

In [ ]:

```
## Feature selection

dfF22 = pd.DataFrame(data=dfF2[['arousal', 'subjectivity', 'entrophy', 'h', 's', 'v']])
dfF22.head()
```

In [ ]:

```
## Reduced Scraping features

dfF22['s1'] = r1
dfF22['s2'] = r2

dfF22.head()
```

In [ ]:

```
import plotly
import plotly.graph_objs as go

plotly.offline.init_notebook_mode(connected=True)

trace = go.Scatter(
    x = resultPCA[:,0],
    y = resultPCA[:,1],
    mode = 'markers',
    hoverinfo = 'text',
    marker = dict(
        color = hh[0],
        colorscale='Viridis'
    )
)

plotly.offline.iplot({
    "data": [trace],
    "layout": go.Layout(title="PCA of HSV features")
})
```

In [ ]:

```
pca = PCA()
X_pca = pca.fit_transform(pd.DataFrame(data=dfF2[['h','s','v']]))

np.set_printoptions(formatter={'float_kind':'{:f}'.format})
ev = pca.explained_variance_ratio_

plt.plot(pd.DataFrame(data=dfF2[['h','s','v']]).columns, ev)
```

In [ ]:

```
dfF2 = pd.DataFrame(data=dfF22)
dfF2.head()
```

Feature statistics

Correlation

In [ ]:

```
import seaborn as sb
from pylab import rcParams

corrcoef = dfF2.corr()
rcParams['figure.figsize']=10, 9
sb.heatmap(corrcoef, xticklabels=corrcoef.columns.values, yticklabels=corrcoef.columns.
values)
```

Mean, Variance, Std

In [ ]:

```
mx=dfF2.mean()
stdx=dfF2.std()
var=dfF2.var()



# mx, stdx, var
sb.heatmap([mx,var,stdx], xticklabels=dfF2.columns.values, yticklabels=['Mean', 'Varian
ce', 'STD'])
```

Feature clustering

In [ ]:

```
clusters_ = np.arange(3, 12, 1)
repeat = 20
```

DBSCAN

In [ ]:

```
# from sklearn.cluster import DBSCAN
# from sklearn import metrics

# # Compute DBSCAN
# db = DBSCAN(eps=0.3, min_samples=7).fit(dfF2)
# core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
# core_samples_mask[db.core_sample_indices_] = True
# labels = db.labels_

# # Number of clusters in labels, ignoring noise if present.
# n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
# n_noise_ = list(labels).count(-1)

# print('Estimated number of clusters: %d' % n_clusters_)
# print('Estimated number of noise points: %d' % n_noise_)
# print("Silhouette Coefficient: %0.3f"
#       % metrics.silhouette_score(dfF2, labels))
```

Silhouette score (GaussianMixture)

In [ ]:

```
from sklearn.mixture import GaussianMixture
from scipy.stats import multivariate_normal as mvn
from sklearn.metrics import silhouette_samples, silhouette_score

rawScores1 = np.zeros(9)

for i in range(repeat):
    for clusters in range(3,12):
        gmm = GaussianMixture(n_components=clusters, covariance_type='full').fit(dfF2)
        prediction_gmm = gmm.predict(dfF2)
        probs = gmm.predict_proba(dfF2)

        silhouette_avg = silhouette_score(dfF2, prediction_gmm)
        rawScores1[clusters-3] = rawScores1[clusters-3]+silhouette_avg

rawScores1 = rawScores1/repeat

for i in range(len(rawScores1)):
    print(i+3," clusters ",rawScores1[i])

fig, ax = plt.subplots()
ax.grid()
ax.set(xlabel='N. of clusters', ylabel='Score',
       title='Average silhouette score for cluster number (Gaussian Mixture)')
ax.plot(np.arange(3, 12, 1), rawScores1)
```

Silhouette score (K-means)

In [ ]:

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

rawScores2 = []

for n in range(3,12):
    kmeans = KMeans(n_clusters=n, random_state=0).fit(dfF2)
    silhouette_avg = silhouette_score(dfF2, kmeans.labels_)
    rawScores2.append(silhouette_avg)
    print(n," clusters ",silhouette_avg)

fig, ax = plt.subplots()
ax.grid()
ax.set(xlabel='N. of clusters', ylabel='Score',
       title='Average silhouette score for cluster number (K-means)')
ax.plot(np.arange(3, 12, 1), rawScores2)
```

Summary

In [ ]:

```
fig, ax = plt.subplots()
ax.grid()
ax.plot(clusters_, rawScores1, color='blue', linestyle='dashed', linewidth=2, markersiz
e=12, label='Original data (GM)')
ax.plot(clusters_, rawScores2, color='blue', linewidth=2, markersize=12, label='Origina
l data (K-means)')
ax.set(xlabel='N. of clusters', ylabel='Score', title='Average silhouette score for clu
ster number')
ax.legend(loc='upper right', shadow=True, fontsize='x-large')
plt.show()
```

t-SNE

In [ ]:

```
from sklearn.manifold import TSNE

X_embedded2 = TSNE(n_components=2, perplexity=30, n_iter=1000).fit_transform(dfF2)
```

PCA

In [ ]:

```
from sklearn.decomposition import PCA

X_embedded3 = PCA(n_components=2, random_state=0).fit_transform(dfF2)
# X_embedded3R = PCA(n_components=2, random_state=0).fit(dfF2)
```

Saving/Loading

In [ ]:

```
## Join
X_embedded2 = np.interp(X_embedded2, (X_embedded2.min(), X_embedded2.max()), (-1, +1))
dfF3 = pd.DataFrame(data={'t1': X_embedded2[:,0],'t2': X_embedded2[:,1]})
dfF33 = pd.DataFrame(data={'p1': X_embedded3[:,0],'p2': X_embedded3[:,1]})
dfF4 = pd.concat([dfF3, dfF33, dfF2], axis=1)
dfF4.head()
```

In [ ]:

```
# ## Save
# dfF4.to_csv('Final.csv',index=False)
```

In [ ]:

```
# dfF4 = pd.read_csv('Final.csv')
# dfF4.head()
```

In [ ]:

```
X_embedded2 = dfF4.as_matrix(columns=dfF4.columns[0:2])
X_embedded3 = dfF4.as_matrix(columns=dfF4.columns[2:4])
```

t-SNE

In [ ]:

```
plt.figure(figsize=(15, 13))

## HSV
# tt = np.array(dfF2.h.values.tolist())
# r = np.interp(tt, (tt.min(), tt.max()), (0, +1))

# tt = np.array(dfF2.s.values.tolist())
# g = np.interp(tt, (tt.min(), tt.max()), (0, +1))

# tt = np.array(dfF2.v.values.tolist())
# b = np.interp(tt, (tt.min(), tt.max()), (0, +1))

## Reduced HSV
tt = hh[0]
r = np.interp(tt, (tt.min(), tt.max()), (0, +1))

tt = ss[0]
g = np.interp(tt, (tt.min(), tt.max()), (0, +1))

tt = vv[0]
b = np.interp(tt, (tt.min(), tt.max()), (0, +1))

for i in range(len(dfF2)):
    c = colorsys.hsv_to_rgb(r[i], b[i], g[i])
    m = (c[0]+c[1]+c[2])/1
    m = m if m>0.1 else 0
    m = m if m<=1 else 1
    color = (c[0], c[1], c[2], 1)
    x, y = X_embedded2[i]
    plt.scatter(x, y, c=color)

plt.show()
```

PCA

In [ ]:

```
plt.figure(figsize=(15, 13))

## HSV
# tt = np.array(dfF2.h.values.tolist())
# r = np.interp(tt, (tt.min(), tt.max()), (0, +1))

# tt = np.array(dfF2.s.values.tolist())
# g = np.interp(tt, (tt.min(), tt.max()), (0, +1))

# tt = np.array(dfF2.v.values.tolist())
# b = np.interp(tt, (tt.min(), tt.max()), (0, +1))

## Reduced HSV
tt = hh[0]
r = np.interp(tt, (tt.min(), tt.max()), (0, +1))

tt = ss[0]
g = np.interp(tt, (tt.min(), tt.max()), (0, +1))

tt = vv[0]
b = np.interp(tt, (tt.min(), tt.max()), (0, +1))

for i in range(len(dfF2)):
    c = colorsys.hsv_to_rgb(r[i], b[i], g[i])
    m = (c[0]+c[1]+c[2])/1
    m = m if m>0.1 else 0
    m = m if m<=1 else 1
    color = (c[0], c[1], c[2], 1)
    x, y = X_embedded3[i]
    plt.scatter(x, y, c=color)

plt.show()
```

In [ ]:

```
pca = PCA()
X_pca = pca.fit_transform(dfF2)

np.set_printoptions(formatter={'float_kind':'{:f}'.format})
ev = pca.explained_variance_ratio_

plt.plot(dfF2.columns, ev)
```

In [ ]:

```
# comps = pd.DataFrame(pca.components_,columns=dfF2.columns)
# sb.heatmap(comps)
```

In [ ]:

```
# scraper.showAll(colors, mode)
```

In [ ]:

```
# # RGB!

# scraper.show(c, mode)
```

Saving/Loading complete dataset

In [ ]:

```
# final = pd.read_csv('Final.csv')
```

In [ ]:

```
# dfF = pd.concat([final,dfR['text']], axis=1).reset_index(drop=True)
# dfF.head()
```

In [ ]:

```
# ## Save
# dfF.to_json('FinalText.json')
```

Clustering & Validation

Silhouette score (GaussianMixture) : t-SNE

In [ ]:

```
from sklearn.mixture import GaussianMixture
from scipy.stats import multivariate_normal as mvn
from sklearn.metrics import silhouette_samples, silhouette_score

avgs1 = np.zeros(9)

for i in range(repeat):
    for clusters in range(3,12):
        gmm = GaussianMixture(n_components=clusters, covariance_type='full').fit(X_embe
dded2)
        prediction_gmm = gmm.predict(X_embedded2)
        probs = gmm.predict_proba(X_embedded2)

        silhouette_avg = silhouette_score(X_embedded2, prediction_gmm)
        avgs1[clusters-3] = avgs1[clusters-3]+silhouette_avg

avgs1 = avgs1/repeat

for clusters in range(3,12):
    gmm = GaussianMixture(n_components=clusters, covariance_type='full').fit(X_embedded
2)
    prediction_gmm = gmm.predict(X_embedded2)
    probs = gmm.predict_proba(X_embedded2)

    print('Silhouette_avg for ',clusters,' cluster: ',avgs1[clusters-3])

    centers = np.zeros((clusters,2))
    for i in range(clusters):
        density = mvn(cov=gmm.covariances_[i], mean=gmm.means_[i]).logpdf(X_embedded2)
        centers[i, :] = X_embedded2[np.argmax(density)]

    n = clusters - 2
    plt.subplot(int(str(33)+str(n)))
    plt.title(str(clusters))
    plt.scatter(X_embedded2[:, 0], X_embedded2[:, 1],c=prediction_gmm ,s=50, cmap='viri
dis')
    plt.scatter(centers[:, 0], centers[:, 1],c='black', s=300, alpha=0.6);

plt.show()
```

In [ ]:

```
fig, ax = plt.subplots()
ax.grid()
ax.set(xlabel='N. of clusters', ylabel='Score',
       title='Average silhouette score for cluster number (GaussianMixture)')
ax.plot(np.arange(3, 12, 1), avgs1)
```

Silhouette score (GaussianMixture) : PCA

In [ ]:

```
from sklearn.mixture import GaussianMixture
from scipy.stats import multivariate_normal as mvn
from sklearn.metrics import silhouette_samples, silhouette_score

avgs2 = np.zeros(9)

for i in range(repeat):
    for clusters in range(3,12):
        gmm = GaussianMixture(n_components=clusters, covariance_type='full').fit(X_embe
dded3)
        prediction_gmm = gmm.predict(X_embedded3)
        probs = gmm.predict_proba(X_embedded3)

        silhouette_avg = silhouette_score(X_embedded3, prediction_gmm)
        avgs2[clusters-3] = avgs2[clusters-3]+silhouette_avg

avgs2 = avgs2/repeat

for clusters in range(3,12):
    gmm = GaussianMixture(n_components=clusters, covariance_type='full').fit(X_embedded
3)
    prediction_gmm = gmm.predict(X_embedded3)
    probs = gmm.predict_proba(X_embedded3)

    print('Silhouette_avg for ',clusters,' cluster: ',avgs2[clusters-3])

    centers = np.zeros((clusters,2))
    for i in range(clusters):
        density = mvn(cov=gmm.covariances_[i], mean=gmm.means_[i]).logpdf(X_embedded3)
        centers[i, :] = X_embedded3[np.argmax(density)]

    n = clusters - 2
    plt.subplot(int(str(33)+str(n)))
    plt.title(str(clusters))
    plt.scatter(X_embedded3[:, 0], X_embedded3[:, 1],c=prediction_gmm ,s=50, cmap='viri
dis')
    plt.scatter(centers[:, 0], centers[:, 1],c='black', s=300, alpha=0.6);
plt.show()
```

In [ ]:

```
fig, ax = plt.subplots()
ax.grid()
ax.set(xlabel='N. of clusters', ylabel='Score',
       title='Average silhouette score for cluster number (GaussianMixture)')
ax.plot(np.arange(3, 12, 1), avgs2)
```

K-means

t-SNE

In [ ]:

```
## t-SNE

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

scores1 = []

for clusters in range(3,12):
    kmeans = KMeans(n_clusters=clusters, random_state=0).fit(X_embedded2)

    silhouette_avg = silhouette_score(X_embedded2, kmeans.labels_)
    scores1.append(silhouette_avg)
    print(clusters," clusters ",silhouette_avg)

    n = clusters - 2
    plt.subplot(int(str(33)+str(n)))
    plt.title(str(clusters))
    plt.scatter(X_embedded2[:, 0], X_embedded2[:, 1],c=kmeans.labels_ ,s=50, cmap='viri
dis')
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],c='black',
s=300, alpha=0.6);
plt.show()
```

In [ ]:

```
fig, ax = plt.subplots()
ax.grid()
ax.set(xlabel='N. of clusters', ylabel='Score',
       title='Average silhouette score for cluster number (K-means)')
ax.plot(np.arange(3, 12, 1), scores1)
```

PCA

In [ ]:

```
## PCA

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

scores2 = []

for clusters in range(3,12):
    kmeans = KMeans(n_clusters=clusters, random_state=0).fit(X_embedded3)

    silhouette_avg = silhouette_score(X_embedded3, kmeans.labels_)
    scores2.append(silhouette_avg)
    print(clusters," clusters ",silhouette_avg)

    n = clusters - 2
    plt.subplot(int(str(33)+str(n)))
    plt.title(str(clusters))
    plt.scatter(X_embedded3[:, 0], X_embedded3[:, 1],c=kmeans.labels_ ,s=50, cmap='viri
dis')
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],c='black',
s=300, alpha=0.6);
plt.show()
```

In [ ]:

```
fig, ax = plt.subplots()
ax.grid()
ax.set(xlabel='N. of clusters', ylabel='Score',
       title='Average silhouette score for cluster number (K-means)')
ax.plot(np.arange(3, 12, 1), scores2)
```

Summary

In [ ]:

```
fig, ax = plt.subplots()
ax.grid()
ax.plot(clusters_, avgs1, color='green', linestyle='dashed', linewidth=2, markersize=12
, label='t-SNE (GM)')
ax.plot(clusters_, scores1, color='green', linewidth=2, markersize=12, label='t-SNE (K-
means)')
ax.plot(clusters_, avgs2, color='blue', linestyle='dashed', linewidth=2, markersize=12,
label='PCA (GM)')
ax.plot(clusters_, scores2, color='blue', linewidth=2, markersize=12, label='PCA (K-mea
ns)')
ax.plot(clusters_, rawScores1, color='red', linestyle='dashed', linewidth=2, markersize
=12, label='Original data (GM)')
ax.plot(clusters_, rawScores2, color='red', linewidth=2, markersize=12, label='Original
data (K-means)')
ax.set(xlabel='N. of clusters', ylabel='Score', title='Average silhouette score for clu
ster number')
ax.legend(loc='upper right', shadow=True)
plt.show()
```

PDF (t-SNE)

In [ ]:

```
import plotly
import plotly.plotly as py
import plotly.graph_objs as go
from scipy.stats import multivariate_normal

plotly.offline.init_notebook_mode(connected=True)

clusters = 6
kmeans = KMeans(n_clusters=clusters, random_state=0).fit(X_embedded2)

d = []
temp = np.array(kmeans.labels_)

for n in range(clusters):

    dfF4Temp = dfF4[['t1','t2']].loc[np.where(temp==n)]
    mu = dfF4Temp.mean()
    Sigma = dfF4Temp.cov()
    X_embeddedTemp = X_embedded2[np.where(temp==n)]

    X, Y = np.meshgrid(X_embeddedTemp[:,0], X_embeddedTemp[:,1])

    F = multivariate_normal(mu, Sigma)
    Z = F.pdf(X_embeddedTemp)

    d.append(go.Mesh3d(
        x = X_embeddedTemp[:,0],
        y = X_embeddedTemp[:,1],
        z = Z,
        colorscale = [[0, 'rgb(0, 0, 255)'],
                      [0.5, 'rgb(0, 255, 0)'],
                      [1, 'rgb(255, 255, 0)']],
        intensity = Z,
        name = 'y',
        showscale = False
    ))

data = d

layout = go.Layout(
    xaxis=go.layout.XAxis(
        title='x'
    ),
    yaxis=go.layout.YAxis(
        title='y'
    )
)

fig = go.Figure(data=data, layout=layout)
plotly.offline.plot(fig, filename='PDF_all_TSNE.html')
```

PDF (PCA)

In [ ]:

```
import plotly
import plotly.plotly as py
import plotly.graph_objs as go
from scipy.stats import multivariate_normal

plotly.offline.init_notebook_mode(connected=True)

clusters = 5
kmeans = KMeans(n_clusters=clusters, random_state=0).fit(X_embedded3)

d = []
temp = np.array(kmeans.labels_)

for n in range(clusters):

    dfF4Temp = dfF4[['p1','p2']].loc[np.where(temp==n)]
    mu = dfF4Temp.mean()
    Sigma = dfF4Temp.cov()

    X_embeddedTemp = X_embedded3[np.where(temp==n)]

    X, Y = np.meshgrid(X_embeddedTemp[:,0], X_embeddedTemp[:,1])

    F = multivariate_normal(mu, Sigma)
    Z = F.pdf(X_embeddedTemp)

    d.append(go.Mesh3d(
        x = X_embeddedTemp[:,0],
        y = X_embeddedTemp[:,1],
        z = Z,
        colorscale = [[0, 'rgb(0, 0, 255)'],
                      [0.5, 'rgb(0, 255, 0)'],
                      [1, 'rgb(255, 255, 0)']],
        intensity = Z,
        name = 'y',
        showscale = False
    ))

data = d

layout = go.Layout(
    xaxis=go.layout.XAxis(
        title='x'
    ),
    yaxis=go.layout.YAxis(
        title='y'
    )
)

fig = go.Figure(data=data, layout=layout)
plotly.offline.plot(fig, filename='PDF_all_PCA.html')
```