Facet-Based Sentimental Analysis of Images: Bidirectional Tunneling and Analysis of Emoji Impressions

Yunlong Liu, Chen Wang, Dayu Wang[1]

Abstract

Separation and proper demonstration of the features of images are prevalent research topics in machine learning and deep learning sphere. In this paper, a categorization and classification of features of emoticons are presented based on the distorted scaling octagonal model from the Plutchik's wheel model[1]. In this paper, a concise step-by-step derivation of how the model is generated is explicitly articulated. The model relies on the Spark NLP models, generating feature vectors and evaluating cosine similarity to determine how close the data and the feature are related. The next downstream component is the pure NLP processor, which takes the output of the image processing as the input, in order to generate text-based description of the image.

The theoretical facet-based sentimental image analyzing model is then applied to implement a real system that uses emoticons (emoji impressions) as the image input. The system is implemented bidirectional to check if the both-side tunneling is successful. First, the system takes a paragraph as the text input, then selects an emoticon from the library of emoji impressions. On the other hand, when a new emoticon is given as the image input, the system translates that emoji into text-based language, and finds related Tweets using the Twitter API.

In the implemented system, text-data is trained via Apache Spark NLP models, in order to relate text with emotions, then with emoticons; image-data is trained via Tensorflow, an implemented neural network system for deep learning, in order for a better accuracy. Based on the observation of the fact that the "first iteration" output is not good for image and text recognition, the output text and image are trained the second time in order to enhance the precision of the system. The evaluation of the results is based on the categorization of emoticons in Twitter, Google, Android, and so on, to check if the placement of emoji impression are really connected with real emotions amongst creatures.

Key Words

Sentimental Analysis, Machine Learning, Deep Learning, Emotion, Emoji Impression, Emoticon, Twitter

[1] Department of Computer Science and Electrical Engineering, University of Missouri-Kansas City. The three authors contribute to the paper equally.

Introduction

Emoji impression is not new things to the smartphone era. On September 19, 1982, in United States at Carnegie Mellon University, professor Scott got a whim, using a string of ASCII characters: "-)" to express the smiling face, and this emotion which must be watched from sideward opened a piece of history of internet[2].

There already exists a very popular visual description of people's moods, which is the emoji impressions, normally used in social media sphere and chatting applications. Since sometimes pure text message could be misunderstood, letting the receiver incorrectly speculate the actual meaning of the text, emoticons (emoji) greatly prevents such a misunderstanding in most cases by presenting not only the text but also the mood of the sender when he/she was sending the text. Emoji has been proved to be a splendid invention in modern social media sphere. You can see emoticons everywhere in our lives: forums, blogs, all the social applications and some input keyboards. Those applications or webpages always give lots of different kinds of emojis. For some popular emotion, 10 applications can give you 10 kinds of emojis with different styles. And for one emoji, it can have more than one meaning. "Face with tears of joy emoji" are the most popular emoji on the internet from 2015 until present[3]. Both joy and sad, even some more emotions, can be expressed with this emoji, lots of people like it to express their kinds of emotions.

But with emoji has been used anywhere, a question has appeared: some people are so "lazy" that they just publish a single emoji as their status or emotion so that others will confuse with the related things. On the other hand, some others are confused to find a good emoji that matching with their emotion. Our system is very useful under these two conditions: it can find emojis for whom just gives a sentence, and it can also give some sentences that matching with their emotion for someone just gave an emoji.

The system applies the technique named image auto-caption, which serves as a bridge between images and text. It can "translate" between emojis and words. This system can catch user's input sentences and analyze the emotion in it then translate it to emojis. It also can classify emojis to 8 different emotions about creatures based on the Robert Plutchik's Wheel of Emotions Theory[4]. Then the system will support some sentences about this emoji people always express on Twitter. This system can be used on forum, blogs, Twitter, Facebook and all kinds of social application. People can use this system input their emotions to publish then the system will attach proper emojis with their moods they have showed in their status or blogs automatically. And if user just input an emoji to express their emotion, this system will help them to find the matching status with this emoji. Those sentences are extracted through the most popular twitter with this emotion. And it always trains and improves itself to make sure they can give the most useful status for users. It is very convenient for users to express their emotions when they are surfing on internet, publishing status on social application and writing down their own blogs.

The project has two main different features: impression-to-mood and text-to-impression, which completes the bidirectional attribute of the project. Table 1 (a) lists the features' description of our system in the viewpoint of the user, and Table 1 (b) lists the features' description of our system in the viewpoint of the developer.

This project contains mainly two parts, the research part and the practical part. In research part, it focused on the recognition/classification of images, as well as natural language processing. Noticed that most image recognition machine learning/deep learning system has a very low accuracy (all kinds of models behave worse than human beings), our model is not expected to be competitive with human beings, yet reasonably generates trending results. Also, we are looking for a way to use the natural language processing to help

image processing, since sometimes people may manual provide additional information, such as image caption, image description, question in natural language about the image, and so on.

Table 1. Description of the features of the *Emoji Interpreter* system from the viewpoint of the user (a) and the developer (b).

(a)

Feature	Description
Text-To-Impression	Given an input of text (single word, single sentence, or a paragraph), the system returns the top 3 related moods (emoji impressions) to the user, 1 default emoji and other 2 substitutions for the user to choose. The user will have an option to tweet it directly. The user will have an option to tweet it or justify the emoji with the one they like.
Impression-To-Text	Given an input of an emoji image file, the system returns the top 10 popular social media posts related with the input emoji. The user can choose the most appropriate one for himself/herself and tweet it.

(b)

Feature	Description
Text-To-Impression	Given an input of text (single word, single sentence, or a paragraph), the system applies the decision tree method, which was used to build the machine learning model to study people's moods, to correctly find out the possibly related moods. Based on the confidence values, the system sets the most possible mood as the default and returns the second and third most related moods to the user as well. We'll reconstruct words contexts with our own models and uses the word K dimensionality dense vector to express words contexts and user's mood.
Impression-To-Text	Given an input of an emoji image file, the system applied the Clarifai API to find out the main topic of the image, which is considered the key words of the emoji. Based on the key words, we try to find the most related text description of the mood the emoji tried to show us. Finally, we generate 10 most popular tweets which used the mood we just figured out and show them to the user. We will search 100 most recently tweets which contains the mood that the user may be and show the tweets to the user. Since Twitter is not the main part in this project, we will just apply its REST API for our convenience.

Related Work

OpenCV (Open Source Computer Vision Library) is an open-source project that contains hundreds of computer vision algorithms. It provides a huge amount of image processing operations, ranging from basic operations, like *image filtering* and *geometric transformation*, to smart machine learning models, like *histograms*, *feature detection*, and *object detection*. OpenCV 1.*x* API is C-based and Open CV 2.*x* is C++-based. It Java-based API is currently under development.

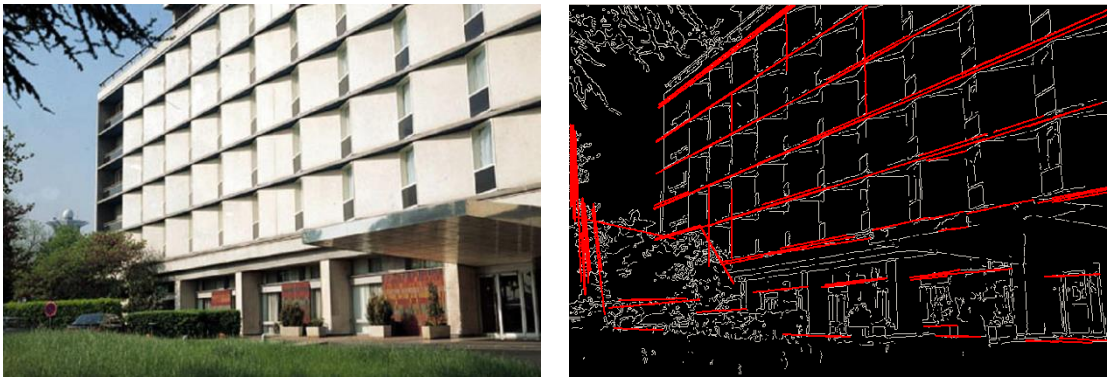
OpenCV is built in a modular structure, which means that the package includes several shared or static libraries. Although it is a C++-based collection of libraries, it handles the memory automatically, which is very similar to those more advanced programming languages and platforms (e.g. Java, Scala, etc.). A rough description of the core modules in OpenCV is listed in Table 2, which can help us understand the basic structure of the entire OpenCV system.

Table 2. Basic Modules in OpenCV System.

Module	Description
`core`	A compact module which defines the basic data structures.
`imgproc`	An image processing module which includes linear/non-linear image filtering, geometrical image transformations, color space conversion, histograms, and more.
`video`	A video analysis module which contains motion estimation, background subtraction, and object tracking algorithms.
`calib3d`	Several basic multiple-view geometric algorithms, including single/stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
`features2d`	Several salient feature detectors, descriptors, and descriptor matchers.
`objdetect`	Detection of objects and instances of the predefined classes.
`highgui`	A convenient interface for video capturing, image/video codecs.
`gpu`	GPU-accelerated algorithms from different OpenCV modules.

In its feature detection functionality, several algorithms are built in, e.g. Canny algorithm[5], Hough transformation[6], and so on. It detects the features by first looking for the *corners* of the shapes in the image (with background template shapes removed), then followed by probabilistic Hough transformation to find the "frame" of the shapes detected from the image file. Figure 3 demonstrates the result after the feature detection of the input image, which shows a hyperfine granularity and the nice performance of the algorithms built in the library. The input image and result are taken from the OpenCV API official documentation website.

The OpenCV projects are very good at object detection and analyze the frame of the images, but it is not good at images classification and "translate" words to images. For our system, the most important features are classifying emoji to different emotions then support a matching express sentence for users. And it is also good at using NLP and Word2Vec to find the meaning in sentence then give a related emoji to users. Our system are much more convenient for user that who surfing on internet with emojis.



(a) (b)

Figure 3. Demonstration of OpenCV feature detection algorithms. (a) Input image; (b) Result image.

Google Vision API allows users to simply integrate vision detection features, like table detection, explicit content detection, face detection, landmark detection, image analysis, logo detection, and optical character recognition. We have learned some of the functions that Google Vision can provide, which is listed in Table 4.

Google Vision is not completely new to us. Last semester, in COMP-SCI 5551 (FS16) (Advance Software Engineering) class, our project contained a receipt recognition system, which had a very good accuracy. It can scan receipt, outputting the location of purchase, automatically classifying the purchase, and extracting the precise amount in USD.

Admittedly, for text recognition and number recognition, the techniques are quite mature at this time. Rather than just for text and numbers, we really would like to see if Google Vision behaves still very well for general images.

From Table 4, we partitioned Google Vision into several principal modules and give a nice description for each module. Acknowledgeably, those modules are not independent with each other. We separate modules to study based on the "loose coupling and high cohesion" policy. In Table 4, we list the modules mainly based on the dependency graph, which means that the module being studied is dependent on the previously studied module. For example, the "Insight from image" module is further than the "Image analysis" module; the "Sentiment analysis" module is deeper than the "Insight from image module", and so on. In other words, we listed all the modules based on the trend of natural study of ourselves, in order to let it help in our project, or set it as a substitute/replacement of Clarifai API or Tensorflow API in image classification/recognition part.

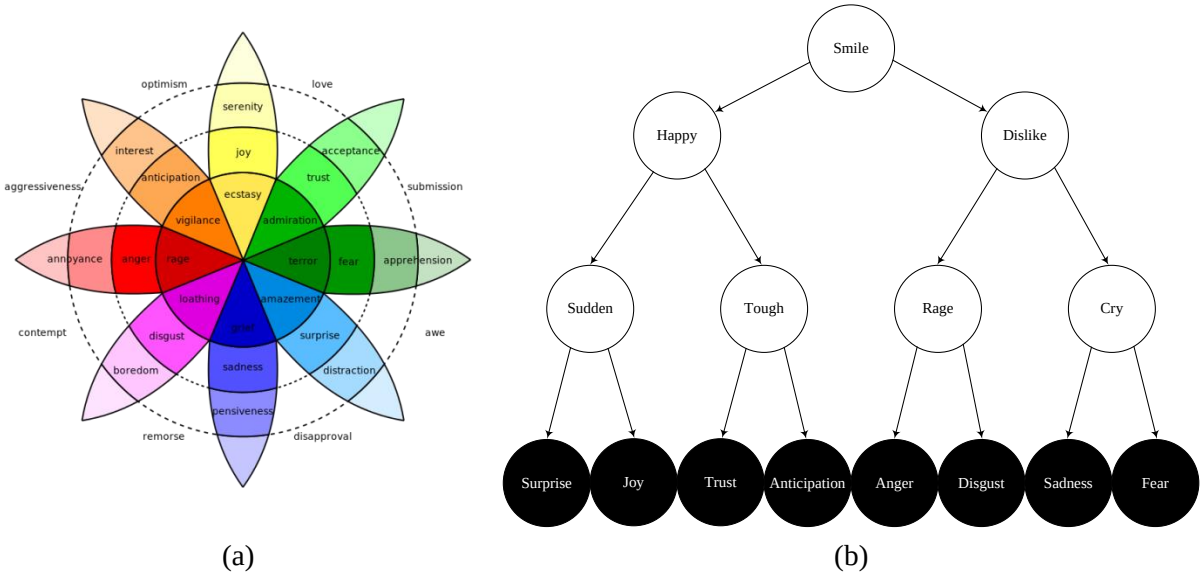
Table 4. Study of Google Vision API.

Function	Explanation
Image analysis	Google Vision API is a REST services API which enables user to have an idea of the image content. It contains machine learning models to classify images into thousands of categories. It includes many different techniques to realize image analysis, such as 2D/3D object recognition, image segmentation, motion detection (e.g. single particle tracking, video tracking, optical flow), medical scan analysis, 3D pose estimation, and automatic number plate recognition. These techniques are being used currently in different fields. Comparing to human abilities of image analysis, these techniques classify images faster and more accurate. Google Vision API classifies the images based on every single detected object in the images, and analyzes the images that uploaded by user or developers in the request.
Insight from image	As we know, images analysis technology is a new, developing field in data science and computational area. These techniques are still being improved. Google Vision API, considered as the "pioneer" in this area, can detect sets or categories of the objects in the images broadly. The categories includes (but not limited to) animals, foods, human, tools, cars, machines, natural landscape. Also, it clearly marks the objects in the images with the most possible labels. Situation seems to be the same with every new concept as that Google Vision API improves over time. In my personal point of view, these kinds of techniques are making big changes to the world.
Inappropriate content detection	This function is powered by Google Search, which can automatically filter pornography and potentially offensive content. This function is significant, since more than half of the youth are accidentally exposed to pornography nowadays. Dark corners always exist somewhere in the internet world, which is really dangerous to the kids. Those offensive contents must be absolutely stamped out.
Sentiment analysis on images	As mentioned above, Google Vision API can analyze images and classify the objects into many categories, this feature also work with people's faces, and it can recognize the emotion on their faces, like joy, anger, sadness, excitement, or trust. Not only it can detect face, but also detect other specific objects, just like product logos, or emoji impressions.
Text extraction	This is another very useful function which enables the developers and users to detect the text from the images, and extract them with automatic language identification.

Google Vision are expert in inappropriate content detection and text extraction which is our project is not skilled. Because our system is just need classify and analyze emotion in emojis. But on the other aspect, our system is good at convert the emotion in emoji to sentence. And it also can finish the "translate" from words to images (emojis) based on NLP and Word2Vec. Those features are the weakness of Google Vision.

Proposed Work

In 1980, Robert Plutchik constructed a wheel-like diagram of emotions, visualizing 8 basic emotions: *joy*, *trust*, *fear*, *surprise*, *sadness*, *disgust*, *anger* and *anticipation*[4]. In order to complete our project of emoji interpretation, the relations amongst all the emotions are absolutely crucial. Figure 5 (a) shows the exact Plutchik's wheel, which has been delved and studied further and further nowadays, and Figure 5 (b) is our speculation of the decision-tree-structure of the relationships amongst different emotions.



(a) (b)

Figure 5. (a) Plutchik's wheel model of the emotions of creatures; (b) Simplified decision tree model of human's emotions for part of implementation.

In this project, our most difficult job is to find an efficient way to model the Plutchik's wheel of emotions. Our algorithm is currently under development. Nevertheless, we still have some basic idea of the classifying algorithm, which will be mentioned later in this report. In this project, we do want to attempt two different algorithms for building up the machine learning models, the decision tree model (which has been taught during the lecture and the tutorials) and the wheel model, in order to accurately classify the eight basic emotions. This requires us to perform "facet-based" sentiment analysis, whereas currently almost all the sentiment analyzing tools can only complete overall analysis. This is why we do need the decision tree, a level-based graphical structure, to help us divide our work into different levels, the decision at one layer can be made if and only if the decisions at all higher levels had been made. We will first complete the decision tree model, and then using the resulting model to the wheel model, which marks the innovation and distinguished significance of our entire project.

Fortunately, in this iteration, we already found a nice and reasonable way to implement the Plutchick's wheel of emotions of creatures. Definitely, our real model is not a regular octagon, since it is impossible that all the emotions that reside at the corners of the octagon have the same distance of any measurement. The details of the implementation will be articulated in the implementation section.

Algorithms

In this project, we are using some of developed algorithms in machine learning and image classification/annotation area. Most of the algorithms are built in Apache Spark or Google Conversion APIs. Also, we will have our own algorithm to build our own machine learning model (see Figure 5). The specific algorithm is currently being developed. In the Implementation section of this report, specific algorithmic specifications will be elaborated. Similar to our previous projects, this time we attempted to implement a "facet-dependent sentiment analytical system for image recognition".

Our analytical tasks include mainly two parts: the **machine learning model** and the **relation extraction** for the emoji impressions with people's moods. Figure 6 demonstrates a decision tree example (not final version) of the relationship of those emoji impressions. For our relation extraction part, differ from classic method found in most textbooks (e.g. IBM Watson), we used a "triangular overlapping" method which is novel and completed designed by ourselves. Nevertheless, we never discarded classic approaches. For instance, we used classic "nearest-neighbor" method which includes Spark Word2Vec module. The algorithm will be described later in this project iteration report.

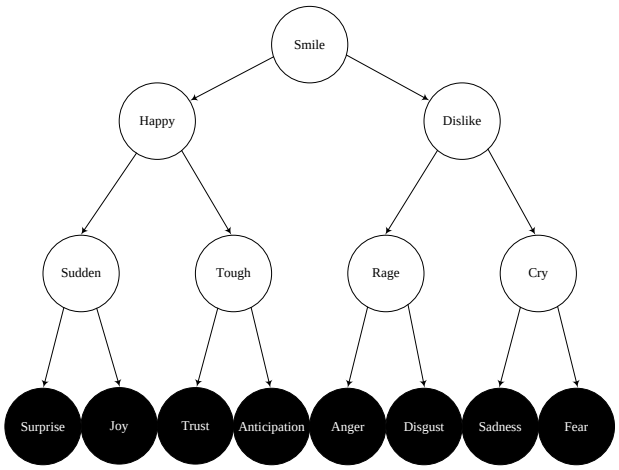


Figure 6. Decision tree model of emoji impressions based on the Plutchik's Wheel of human's moods (left → "Y"; right → "N").

Based on the decision tree model of those impressions, we can further build an ontology model that connects a specific mood with several tweet texts. Based on the time limit and the granularity requirement of the project, this level will be our stop point for the project.

System Architecture (see Figure 7)

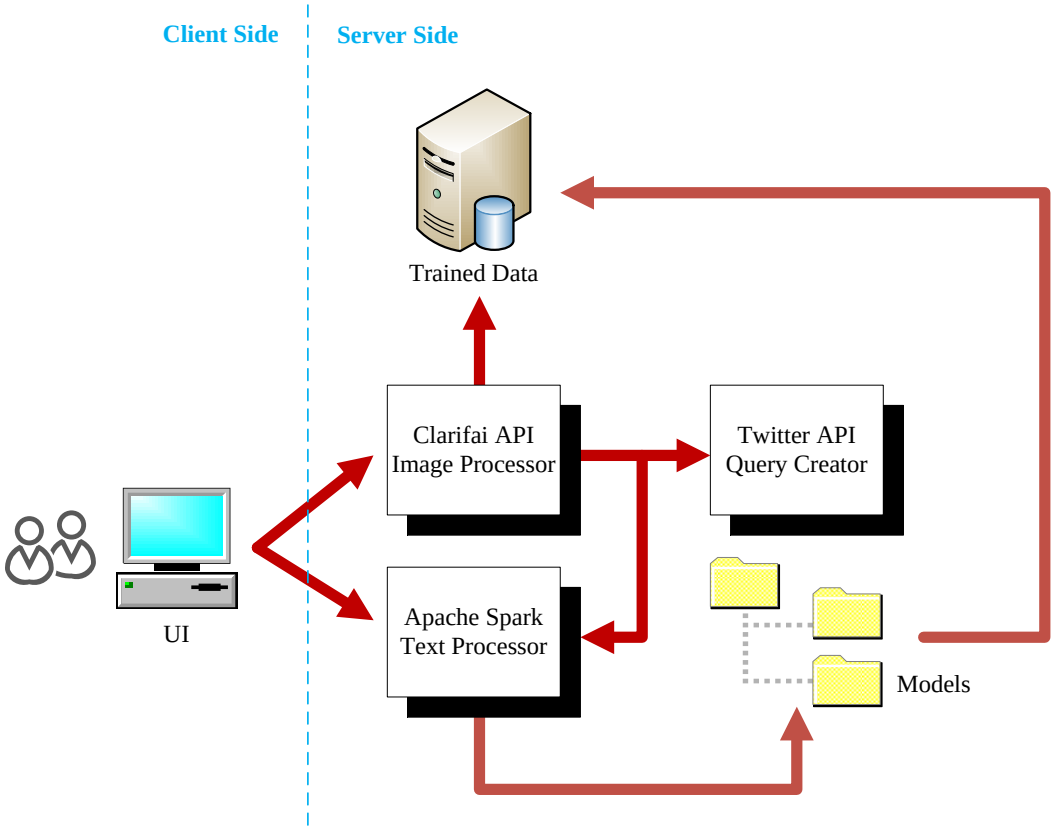


Figure 7. System Architecture of the *Emoji Interpreter* system.

From Figure 7 we can see that our system architecture holds a client-server architectural style. The trained data and generated models are both on the server side. When an input is given by the UI (user), based on the type of the input (text or image), the system passes the input data into its related processors (image or text processor). The two core processors may call each other since sometimes the result of image processor, which is some text, needs further processing via text processor to finally find out the closest classification of the image (emoji impression). After the processing of the input image, the Twitter API is thus called to collect tweet which contains the interpreted meanings of the input emoji. Then, the top 10 tweets are returned to the user for selection. On the other hand, when the text processing is completed, then the result is passed into the created models of people's emotions, in order to find out the emotion class that best fits the input text. In most cases, the resulting text will be determined between two emotions or amongst several emotions. Therefore, top 3 related emotions with the emoji representations will be returned to the user.

Activity Diagram (see Figure 8)

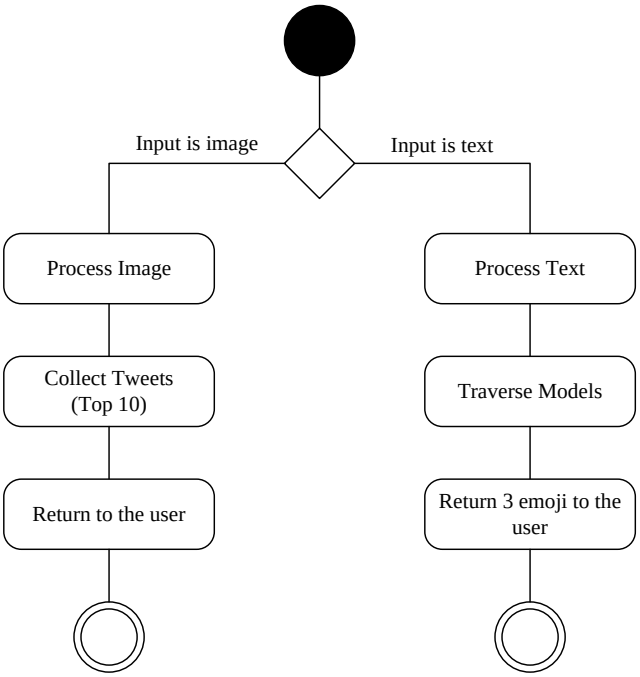


Figure 8. High-level activity diagram of the *Emoji Interpreter* system.

Sequence Diagram (see Figure 9)

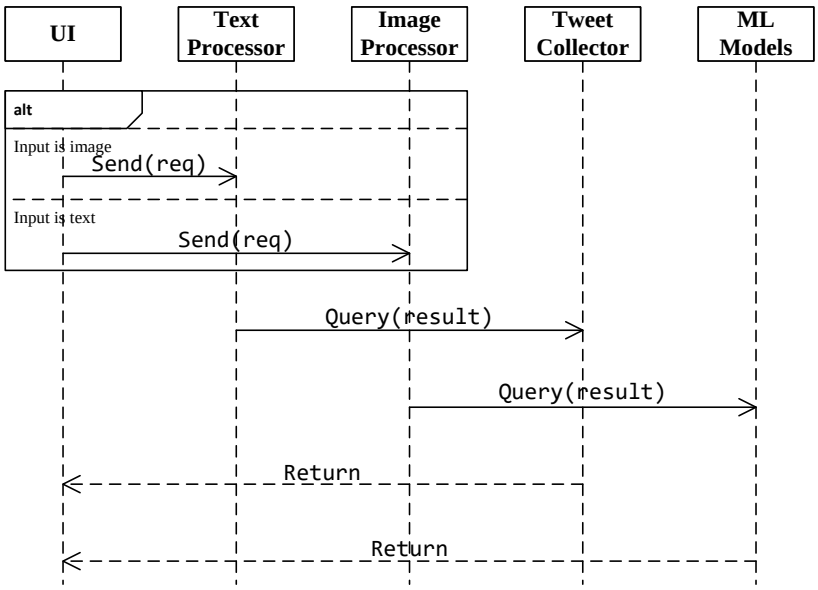


Figure 9. High-level sequence diagram of the *Emoji Interpreter* system.

Implementation Details

The algorithm we designed to implement our system of emotions of human beings is called the *facet-based sentiment analysis algorithm*. In our system, an octagon is placed at the very beginning of time. Then, our model defines each corner of the octagon as an extreme emotion (as shown in Figure 10). Besides, each triangular area between two adjacent emotions is labeled as well. The origin is placed at the center of the octagon.

To convincingly represent our machine learning result, we need a united standard of measurements. Therefore, we chose the Spark Word2Vec as the measurements of text. First of all, we created a vocabulary of emotions and pass into the system. Based on our "instructional input", the natural language processor of Spark will return a set of results, which are the vectors of words. Based on those vectors, we can build our own distorted octagonal model. Referring to the original Plutchik's model, we can connect those sparse vectors into a distorted octagonal shape. The shape is what we will use in future. Nevertheless, this did not solve the problem yet, because for a real vector, it may have multiple possibilities. For example, the vector may reside outside the shape; or, the vector may reside inside the "overlapping area" (will be discussed more later). In this case, the model itself cannot provide any additional help to our system, which means we have to design more algorithms on top of the distorted octagonal model. In this iteration, we mainly focused on this algorithm and finally found out a controllable way to express it (see Figure 11).

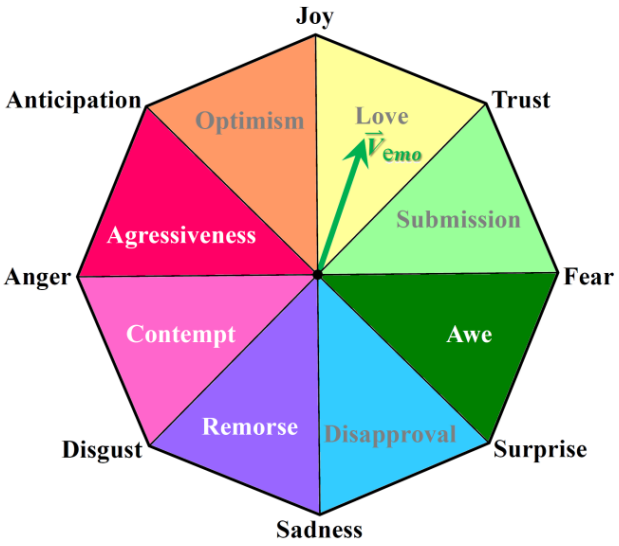


Figure 10. The **octagonal sentiment model** for emoji classification in the *Emoji Interpreter* system. The origin is located at the center of the octagon. Each corner represents an extreme emotion. Each triangular labeled area represents the "combined emotion" which is determined by the two neighboring emotions.

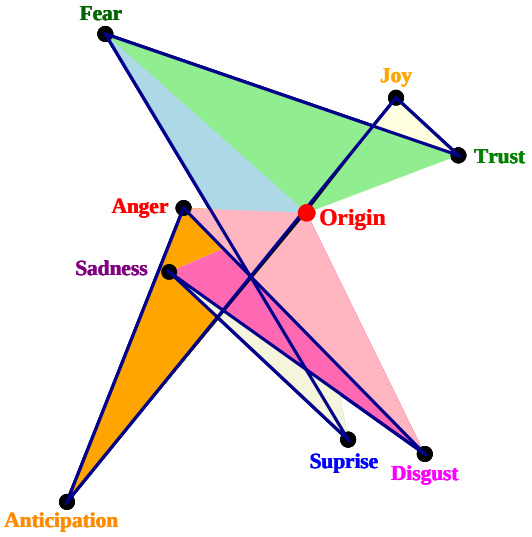


Figure 11. Distorted octagonal model of creature emotions (refer to the Plutchick's wheel of emotions). Black dots represent the vertices in the original regular octagon. Black solid lines represent the edges in the original regular octagon. Colored triangular areas represent the sectors defined in the original regular octagon.

For this model, we should following the steps below to make sure the octagonal model is reasonable (see Table 12). In this iteration, we finished the entire part of this algorithm, which means the text processing part is now complete. Based on this, we can next focus on the image classification/recognition part.

Table 12. General steps of processing the *octagonal machine learning facet-based sentiment model*.

Step	Explanation
Step 1: Word2Vec Data Training	In order to make our machine learning model does make sense, we need to first unite the coordinate system, which means we need to choose the same training algorithm. Therefore, we need to know the coordinates of every corner. Thus, we put all the corner extreme emotions into the Word2Vec model, and record their coordination values.
Step 2: Lining the range for each triangular area	After receiving all the coordinate values for every corner in the octagon, we can define each range by lining those coordinating data. This marks the completeness of model generation.
Step 3: User input data coordination	For user's text input, or the image processing result (still text), we find out the related vector in the Word2Vec system. Definitely, the Word2Vec system requires the input data to be vectors. Therefore, we still need some more modules to transform user's input data into the data that our core system can recognize and process.
Step 4: Generate emotion results	Based on which triangular the vector belongs to, we can simply generate the three closest emotions as the classification. For example, in Figure 10, vector $\vec{V}$ is in the "Love" area, the three closest emotions are: "Joy", "Trust", and "Anticipation", which will be considered as the results used for the downstream components.

Datasets

Table 13 lists the data source for each part of our *Emoji Interpreter* system, which includes the training data, test data, and presentation sample data. Our data was collected manually from the internet.

Table 13. Data source and collection method for each part of the *Emoji Interpreter* system.

Data Description	Size of the Data	Data Source	Collection Method
Image Training Data	800 Emoji Images	Multiple Devices (Apple, Android, Samsung, Twitter, Facebook, etc.)	Manual Collection
Text Training Data	1000 Tweets	Twitter	Twitter API (4j)
Searching Text	Entire Twitter	Twitter	Twitter API (4j)
Representing Data	8 Emoji Images	Website	Manual Setup

Feature Specification (see Table 14)

Table 14. Feature specifications for the *Emoji Interpreter* system.

Feature	Specification

Text-To-Impression	input plain_text *tx*
	---------- 	
	Spark processor (*tx*) → sparkResult[]
	if sparkResult[] contains class
	  return top 3 classes
	else
	  Word2Vec result (top 3 similar)
	---------- 	
	output emoji images to the user

Impression-To-Text	input jpg image file *img*
	---------- 	
	Clarifai processor (*img*) → textResult[]
	if textResult[] contains class
	  extract top class *cls*
	else
	  Word2Vec result (top similar) *cls*
	Twitter query (*cls*) → twitterResult[]
	twitterResult[].get(10)
	---------- 	
	output top 10 tweets to the user

Operation Specification (see Table 15)

Table 15. Operation specifications for the *Emoji Interpreter* system.

Input	Output	Exception
Format: String/File	Format: jpg/String	Empty input
File: jpg only	Output: Arrays	null image file
File content: emoji	Number of results: 10	Size exceeded
File size: less than 1 M	UI: Web-based	Not emoji image

Emoji Classification Workflow (see Figure 16)

Emoji in the training part will process through object detection, feature vector and classification algorithm. Then the Classification Model has been built. The testing data has been used for evaluating the model after process of feature vector. In Figure 16, the classification model locates at both the training part and the test part. We set up emoji impressions as the input, as generate key frames using Clarifai API. After that, the feature vectors are extracted from the emoji impressions, which will be passed into the main classification engine. The algorithm of classifying images will generate output models, which serves as the classification model. On the test part, testing data is going to the feature vector extractor. The using the trained models, we can generate our desired results.

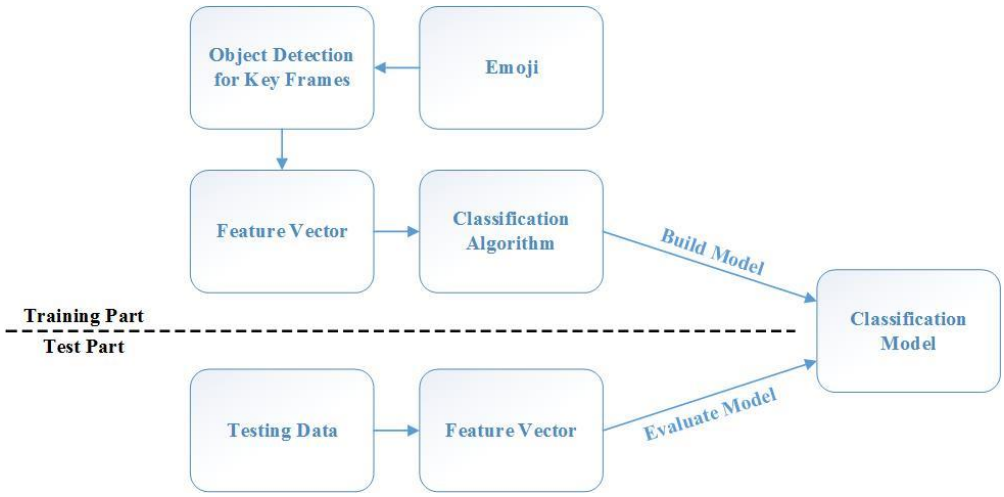


Figure 16. Workflow about the emoji classification model built.

Measurement

For our project, there are two ways to measure its result: the first way to measure it is put the emoji back to its source and get its tag. When the system finished a classification and gave it an label (emotion), we can find the test emoji's source such as twitter, Facebook and so on. Then we can see the tag of the emoji those applications give them. Then we can use those tags to measure if the label is correct and improve the system. The second way to test the output of the system is recognize the emoji and sentence manually, and because the sentence

user input has no reference and no others can give an exact answer for the emotion in it. So we must choose some volunteer to input the words and judge if the system give a matching emoji for it. Both of those measurements can help our system do machine learning and self-improving. The results of this system will more and more accuracy after training and testing.

Evaluation and Results

After initial training and testing, the accuracy of our system is about 41%, the runtime is displayed on Table 17.

Table 17. Runtime of our system in 8 times.

Index	Runtime (ms)
1	3215
2	2816
3	2745
4	3126
5	2920
6	3462
7	2934
8	3103

As comparison, the accuracy of Clarifai has displayed on Figure 18.

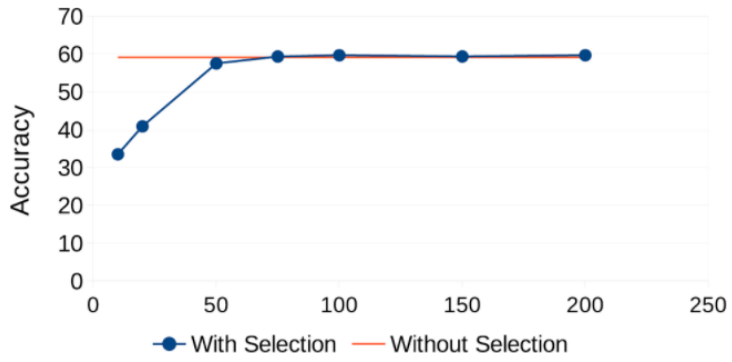


Figure 18. Accuracy of Clarifai Descriptors using a Subset Variable Selection Process[7].

And the accuracy of Spark is nearby 90% for image classification after training. Using the RandomForestModel model, the accuracy of Clarifai even can achieve 96.3%[8].

Discussion and Limitation

The result of our system has different performance between emoji to text and text to emoji. The performance and accuracy about text to emoji is much better than emoji to text. Through test and analyze, we found two main reasons: the first one is images is much difficult to recognize and classify, there are too much parameters about frame, scenery, brightness and outline. All these factors will influence the performance about the image classification. The other reason about this phenomenon is the limited training data sets, because the emoji collection is not very easy and we must classify the training data by ourselves, so even if we choose lots of emojis to train the system, comparing with the huge training data of Word2Vec and NLP, 800 emojis are still not enough to build an ideal model for analyze emojis. So for the further improve the system, we need collect more dataset and train/test more times to build a better model.

Conclusion

From the result specified above, we can conclude that the image recognition power is worse

than the text recognition power. Even though our input data was changed from generic dataset to the results from the previous iteration, the result is still not quite satisfactory. However, we did prove that the tendency of recognition of images in the sphere of emotions of creatures can keep positively correct. The separation of different facets of the features of an image is successful.

Reference

[1] Plutchik, R. and H. Kellerman, *Emotion: Theory, Research and Experience. Vol. 1, Theories of Emotion*. 1980: Academic Press.

[2] Fahlman, S., *Original Bboard Thread in Which ":-)" was Proposed.* 1982.

[3] Lu, X., et al. *Learning from the Ubiquitous Language: An Empirical Analysis of Emoji Usage of Smartphone Users*. in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 2016. ACM.

[4] Plutchik, R., *Emotion: A psychoevolutionary synthesis*. 1980: Harpercollins College Division.

[5] Canny, J., *A computational approach to edge detection.* IEEE Transactions on pattern analysis and machine intelligence, 1986(6): p. 679-698.

[6] Matas, J., C. Galambos, and J. Kittler, *Robust detection of lines using the progressive probabilistic hough transform.* Computer Vision and Image Understanding, 2000. **78**(1): p. 119-137.

[7] Vieira, M., D. Faria, and U. Nunes, *Robot 2015: Second Iberian Robotics Conference*, 2016, Springer International Publishing Lisbon, Portugal:.

[8] Ryza, S., et al., *Advanced Analytics with Spark: Patterns for Learning from Data at Scale*. 2015: " O'Reilly Media, Inc.".