File Last Saved: 02/27/17 06:35 PM

COMP-SCI 5542 (SP17) - Big Data Analytics and Applications

Project Report 1 - Due 02/27/17 by 11:59 PM

Yunlong Liu **(22)**
Chen Wang **(44)**
Dayu Wang **(45)**

1. Project Objectives

1.1. Significance

The *image auto-caption* technique has successfully connected image files with text descriptions of the image. Based on this technique, how to connect images with something else became a quite hot topic in data science. In this project, we would like to attempt to link people's moods to representative images about the moods.

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, 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.

In this project, we would like to do **bidirectional linkage between emoji impressions and people's moods**, which means when an emoji is given, a mood is returned to describe the emoji impression; also, when some text is provided, our system can automatically add emoji labels to the text. This technique might save a lot of time for those bloggers and social media fans, for that they do not need to enter both the text and emoji, based on the fact that in most cases, text and emoji consistent in a post in social media.

1.2. Features: Use Case/Scenario

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.

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.
• 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.
• 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.

2. Approach

2.1. Data Sources

Table 2 lists the data source for each part of our *Emoji Interpreter* system, which includes the training data, test data, and presentation sample data.

Table 2. 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

2.2. Analytic Tools

Table 3 lists the current analytic tools we would like to use in our project. Also, we attempted to explain about how the analytic approaches may help us solving different kinds of problems. Some of the techniques are mature and some of them are still being developed.

Table 3. Analytic tools with explanations that will be applied in the development of the *Emoji Interpreter* system.

Analytic Tool	Explanation
• Clarifai API	• We apply the Clarifai API to detect the topics from image files. The detected topics will be used as the text descriptors of emoji impressions, in order to correctly identify the related objects from our machine learning models.
• Twitter API (4j)	• Twitter is the most popular social media platform which includes a "sea" of emoji impression with text. Text from tweets can explicitly give a prevalent interpretation of the emoji impression. Based on the study of the text and emoji within the same tweet, we can well build our machine learning model for our system.
• Apache Spark	• We use Spark basically for the text processing part of the system, which includes TF-IDF, LDA, Word2Vec, and etc. If we cannot find the class from our pool of data which perfectly match the user's input, we will use the most similar words as our keys instead.

2.3. Analytical Tasks

Our analytical tasks include mainly two parts: the **machine learning model** and the **relation extraction** for the emoji impressions with people's moods. Figure 4 demonstrates a decision tree example (not final version) of the relationship of those emoji impressions.

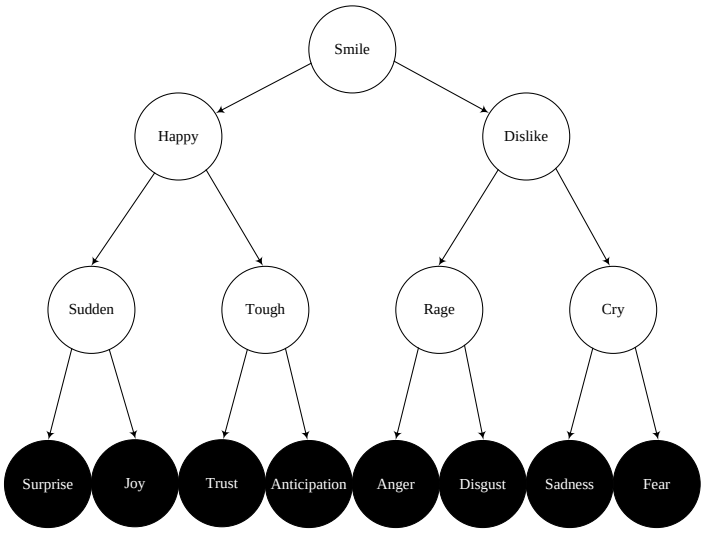


Figure 4. Decision tree model of emoji impressions based on the Plutchik's Wheel of human's moods (left → "Y"; right → "N").

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.

2.4. Expected Inputs/Outputs

Table 5 shows the expected inputs and outputs for the two features of the system.

Table 5. Expected inputs and outputs for the *Emoji Interpreter* system.

Expected Inputs	Expected Outputs
• JPG image	• A list of 10 most popular text posts from Twitter with similar impressions as the input emoji file.
• Text input (Paragraphs, words, sentences, etc.)	• Three emoji impressions, 1 default and 2 substitutes, for the user to select one that best describe his/her current mood.

2.5. 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 4). The specific algorithm is currently being developed.

3. Related Work

3.1. Open Source Projects

3.1.1. OpenCV

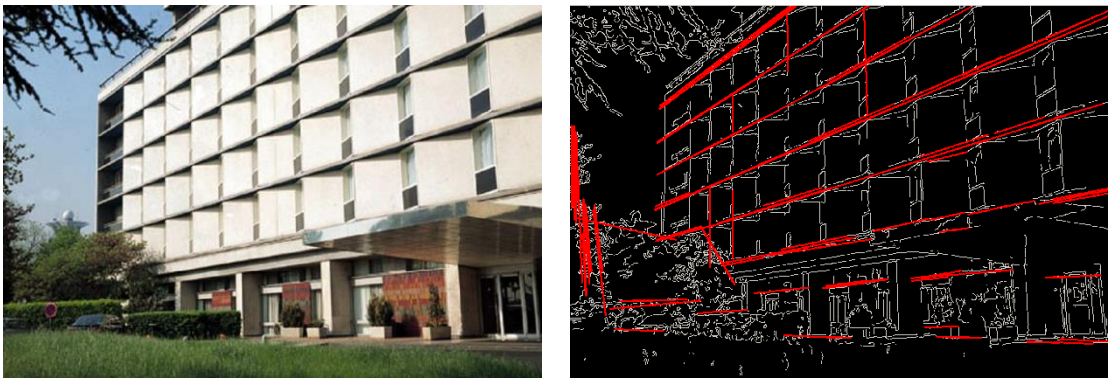
OpenCV (Open Source Computer Vision Library) is an open-source project that contains hundreds of computer vision algorithms. It provides a huge amount of image processing operations, ranging from basic operations, like *image filtering* and *geometric transformation*, to smart machine learning models, 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 6, which can help us understand the basic structure of the entire OpenCV system.

Table 6. 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[1], Hough transformation[2], 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 7 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.



(a) (b)

Figure 7. Demonstration of OpenCV feature detection algorithms. (a) Input image; (b) Result image.

OpenCV Office Website: http://www.opencv.org
OpenCV API Documentation: http://docs.opencv.org/2.4/index.html

3.1.2. Google Vision

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 8.

Table 8. Study of Google Vision API.

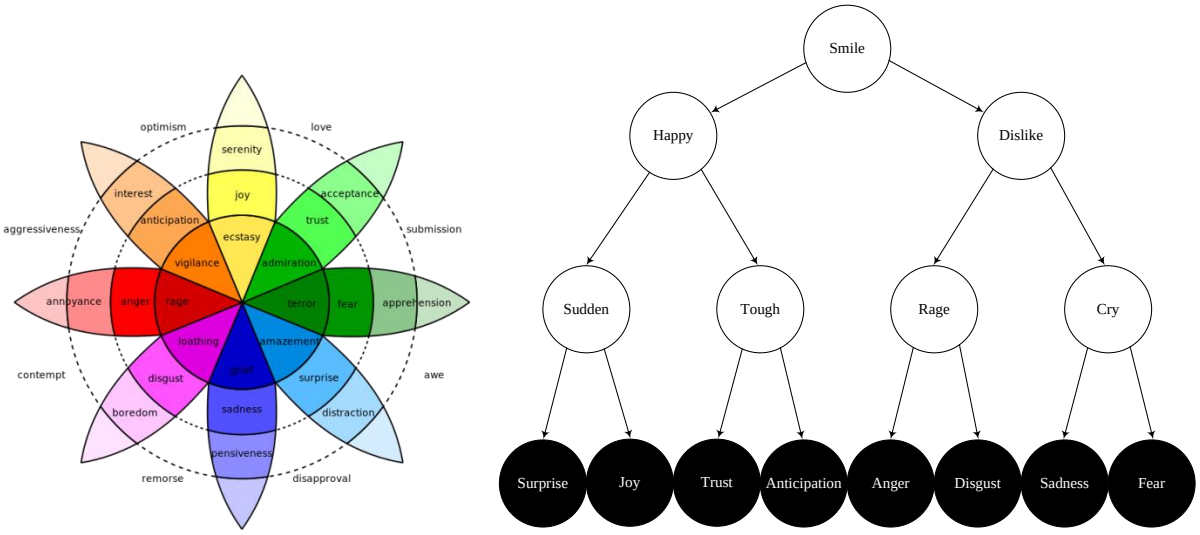
Function	Explanation
• Image analysis	• Google Vision API is a REST services API which enables user to have an idea of the image content. It contains machine learning models to classify images into thousands of categories. It includes many different techniques to realize image analysis, such as 2D/3D 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 Official Website: https://cloud.google.com/vision
Google Vision API Documentation: https://cloud.google.com/vision/docs

3.2. Literature Reviews

Plutchik, 1980, Emotion: A Psychoevolutionary Synthesis[3].

In 1980, Robert Plutchik constructed a wheel-like diagram of emotions, visualizing 8 basic emotions: *joy*, *trust*, *fear*, *surprise*, *sadness*, *disgust*, *anger* and *anticipation*. In order to complete our project of emoji interpretation, the relations amongst all the emotions are absolutely crucial. Figure 9 (a) shows the exact Plutchik's wheel, which has been delved and studied further and further nowadays, and Figure 9 (b) is our speculation of the decision-tree-structure of the relationships amongst different emotions.



(a) (b)

Figure 9. (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.

4. Application Specification

4.1. System Specification

4.1.1. Software Architecture (see Figure 10)

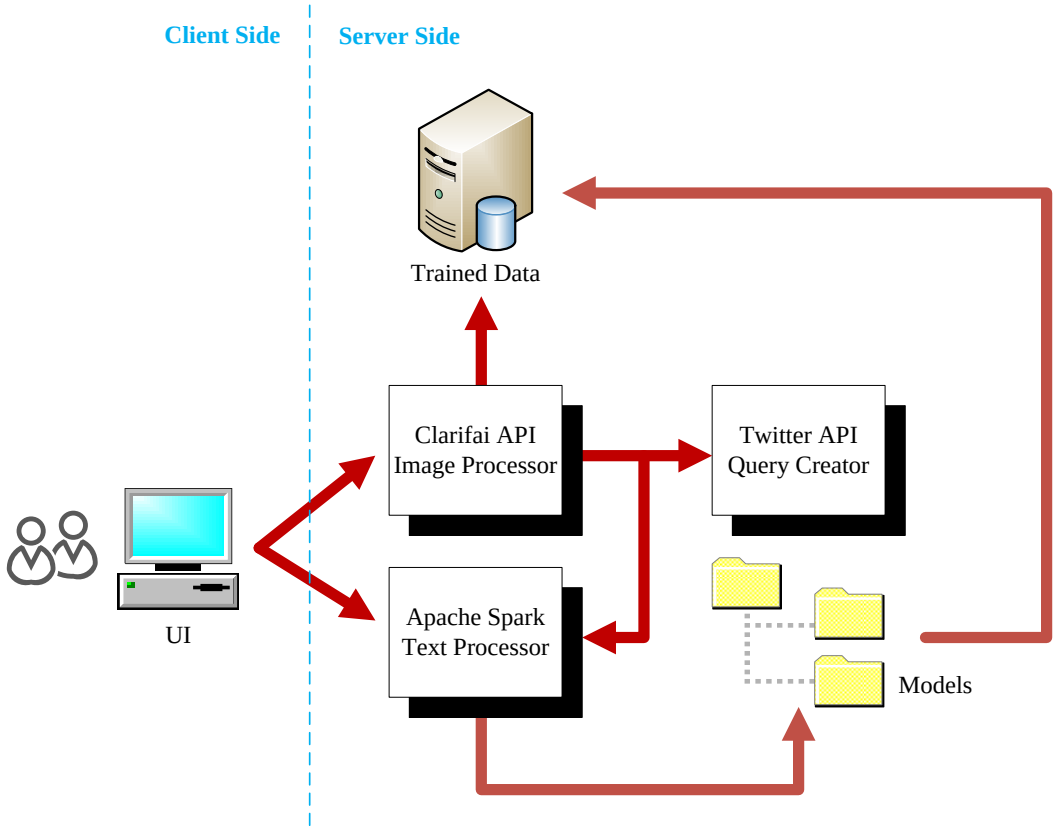


Figure 10. System Architecture of the *Emoji Interpreter* system.

From Figure 10 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.

4.1.2. Features, Workflow, and Technologies

4.1.2.1. Activity Diagram (see Figure 11)

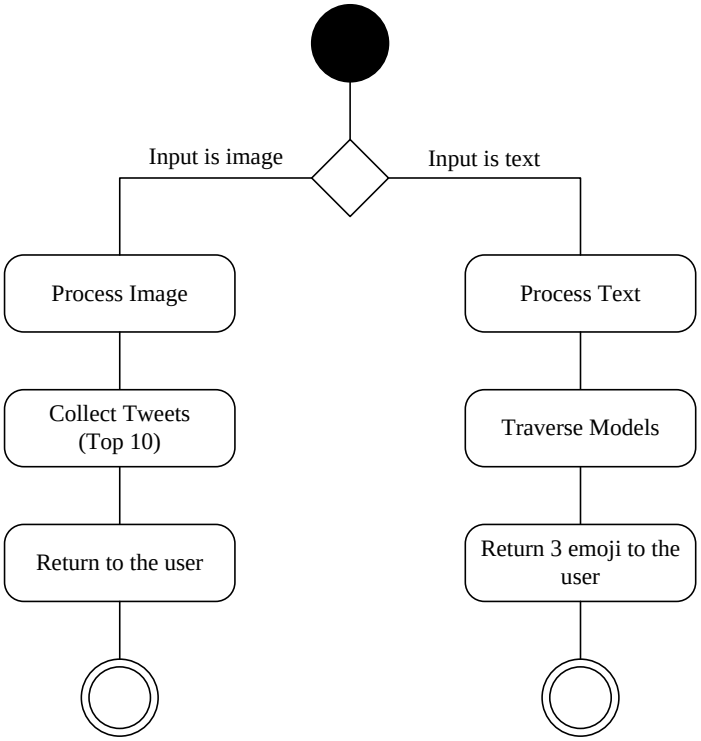


Figure 11. High-level activity diagram of the *Emoji Interpreter* system.

4.1.2.2. Sequence Diagram (see Figure 12)

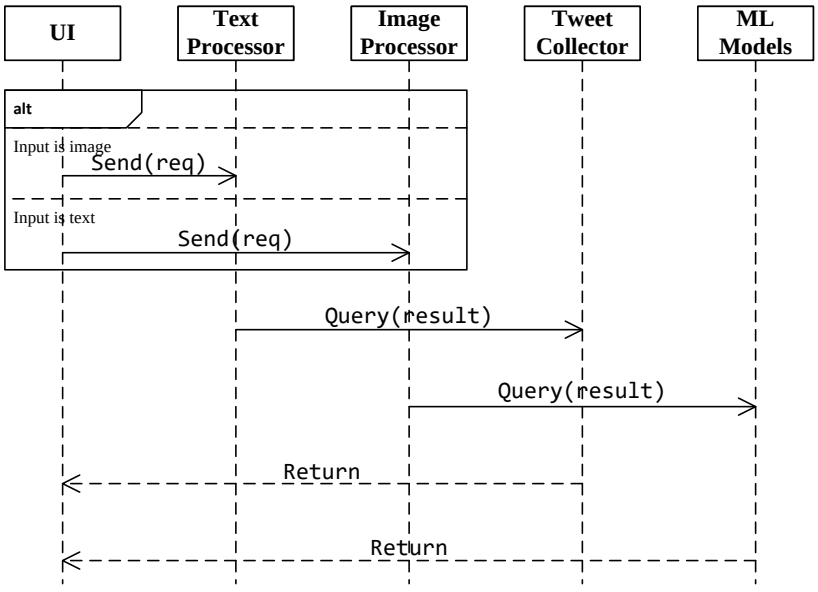


Figure 12. High-level sequence diagram of the *Emoji Interpreter* system.

4.1.2.3. Feature Specification (see Table 13)

Table 13. 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`

4.1.2.4. Operation Specification (see Table 14)

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

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

4.2. Existing Applications/Services Used

4.2.1. Apache Spark

Apache Spark is an open source cluster-computing framework. It uses mathematical operation in the memory units instead of putting the operated datasets into the disk. It can analyze and compute the datasets before they have been written into disk. The speed of Apache Spark finishing the process of computing and analyzing is 10 to 100 times faster than Hadoop MapReduce. It allows users load data to cluster-process and inquiry it more than one time, so Apache Spark are very easy and suitable for machine learning.

Apache Spark has 5 main parts. Spark Core (RDDs) is the foundation of the whole project; it offers distributed assignment dispatch, regular dispatch and basic I/O function.

Spark SQL offers structured and semi-structured data. Spark Streaming analyzes data uses streaming by the rapid dispatch ability of Spark Core. MLlib is the framework of distribute machine learning. GraphX is a distributed graph processing framework.

Apache Spark is easy to user because it can be used on Java, Scala, Python and so on. It can run on lots of platforms, such as Hadoop, Mesos, standalone, or in the cloud.

In our application, Apache Spark will be used for analyzing emoji, words and sentences, training and testing the machine learning model, finding the matching emoji and emotion sentences with users' input. All the analyzing and processing of images and words are based on Apache Spark, and it can build a model of analyzing emoji and improve it continuously.

Url: http://spark.apache.org

4.2.2. Clarifai API

Clarify API is a self-service API that user can process and analyze images, audios and videos. It is a REST API that can extract important information in the images, audios and videos and it has the basic operation about REST API: `GET`, `PUT`, `POST` and `DELETE`.

The core parts of Clarify API are those models: `bundle`, `metadata`, `tracks` and `insight`. They work together to process the images, audios and videos. It can filter the secondary information and extract most important information. It can remove the background noise in the images and videos, the noise in the audios, then get the useful things and analyze them and return one or more information about them. It also gives a parameter to show the definition about the information, such that higher value means more accuracy for this estimation.

For our project, lots of emoji datasets need train and test for the system, the Clarify API can help the system to find the main topic of the emoji, then the topic will be used for training and testing the system. Absolutely, when the model has been built, it can also use to help to finding related emoji based on the input sentences.

Url: http://docs.clarify.io/overview

4.2.3. Twitter API

Twitter API is an API that can connect users, tweets and entities in objects. It is a bridge for users and twitter database. It can be used for various kinds of languages, e.g. Java, Objective C and other programming languages.

Twitter API can be separated to several different smaller APIs: one is REST API and the other one is Stream API. REST API that can help users to get and write Twitter data. It requires developer using OAuth to do it and it will return a JSON file as the output. Without this API, user can use the Twitter Streaming API to develop a real-time application. For the REST API, users can make requests from website or other applications, and then the server will send those requests to the Twitter REST API and Twitter will response the request and return related output through server. For the Twitter Streaming API, user will see the

rendered site directly, and then the data is rendered into view at real-time. For the Twitter Streaming API, a stable web connection is necessary all the time.

For our application, Twitter API will be used when the model has been built, at the function that returns emoji based on user's input. Apache Spark has been used to analyze user's emotion based on their input sentences, the most popular emoji matched this emotion will be returned attached with user's input. Another two reserved choices will also be supported so that user can choose the most accurate emotion they want to express. These emoji are chosen based on the most popular emoji to express related emotion on Twitter. This process will be finished before the system build the model and the chosen emoji will be our standard emoji. On the other hand, when users input emoji, we will give some sentences matched their emoji. Clarify API will be used for analyze the emotion in emoji of user's input, then based on the emotion Clarify API extracted, our system will return some matched sentences with the related emotion. At this step, Twitter API has been used to extract the most popular twitter has been post with this emotion. When user input an emoji to express their emotion, the most probable saying they want to post will be show then they can use it directly. Of course, some reserved saying will be given as the same.

Url: https://dev.twitter.com/overview/api

5. Implementation

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 15). Besides, each triangular area between two adjacent emotions is labeled as well. The origin is placed at the center of the octagon.

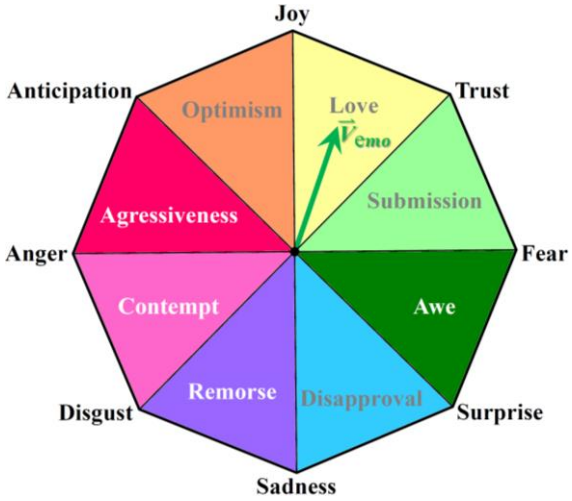


Figure 15. The **octagonal sentiment model** for emoji classification in the *Emoji Interpreter* system. The origin is located at the center of the octagon. Each corner represents an extreme emotion. Each triangular labeled area represents the "combined emotion" which is determined by the two neighboring emotions.

For this model, we should following the steps below to make sure the octagonal model is reasonable (see Table 16).

Table 16. 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.
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 15, 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.

6. Documentation

6.1. User Interface Mock-Up (see Figure 17)

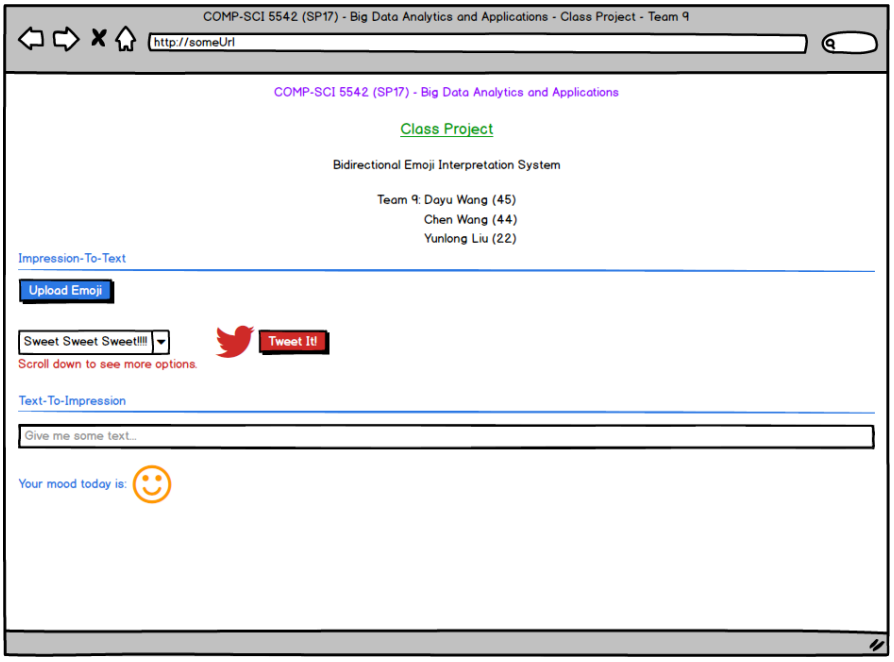


Figure 17. User interface mock-up for the *Emoji Interpreter* system.

6.2. Standard Emoji Impressions for the Eight Corners in Pluchik's Wheel (see Figure 18)

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 18. Standard emoji impressions for the eight corners in Pluchik's wheel in the *Emoji Interpreter* system. (a) Anger; (b) Anticipation; (c) Disgust; (d) Fear; (e) Joy; (f) Sadness; (g) Surprise; (h) Trust.

7. Project Management

7.1. Work Completed (see Table 19)

Table 19. COMP-SCI 5542 (SP17) Project Team 9 - Project Report 1 - Work Completed.

Name (Class ID)	Source Code Completed (Hours)	Report Completed (Hours)	Other Documentation Completed (Hours)
***Dayu Wang* (45)**	• User Interface (1)	• Project Objectives (2) • Approach (5) • Related Work (2) • App Specification (6) • Implementation (3) • Documentation (2) • Management (2)	• Architecture Diagram (2) • Activity Diagram (1) • Sequence Diagram (1) • **Octagon Design** (5) • UI Mock-Up (1)
***Chen Wang* (44)**	• Clarifai API (2)	• App Specification (4) • Documentation (2)	• Emoji Image Handling (3)
***Yunlong Liu* (22)**	• N/A	• Related Work (3)	• N/A

7.2. Work to be Completed (see Table 20)

Table 20. COMP-SCI 5542 (SP17) Project Team 9 - Project Report 1 - Work to be Completed.

Name (Class ID)	Work Description	Estimated Hours	Deadline
***Dayu Wang* (45)**	• Set up the `Word2Vec` model. • Set up the **octagonal ML model**. • Make the IntellIJ project executable.	• 8 • 8 • 8	• 03/06/17 • 03/13/17 • 03/20/17
***Chen Wang* (44)**	• Complete the functionality of Clarifai. • Connect the Clarifai with txt processor. • Make final version Clarifai report.	• 8 • 8 • 8	• 03/06/17 • 03/13/17 • 03/20/17
***Yunlong Liu* (22)**	• UI beautification. • Decision tree model of emotions. • Google conversion API.	• 8 • 8 • 8	• 03/06/17 • 03/13/17 • 03/20/17

7.3. Issues/Concerns: There is **no issue discovered** in this iteration.

8. References

[1] Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6), 679-698.

[2] Matas, J., Galambos, C., & Kittler, J. (2000). Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. *Computer Vision and Image Understanding*, 78(1), 119-137.

[3] Plutchik, R. (1980). Emotion: A Psychoevolutionary Synthesis. Harpercollins College Division.