



Northeastern University

Report for Assignment_4

K3G MUSIC

Team Member: Jixiao Yang

Cuicui Feng

Miner Yang

Yuning Song

Introduction

K3G Music enterprises is planning a new music station which would enable listeners to listen to music based on how they feel. They plan to use Machine learning to classify songs. Acknowledging there are new songs added to their catalog on a daily basis, they intend to build a machine learning based classifier service which would classify songs as (Happy/Sad).

K3G has three goals in mind. They want to prototype an app which would dynamically generate whether a song is happy/sad from a top-k list in the US. They intend to use Musicmatch to prototype this. They also intend to use Musicmatch's API to get the top-K list of other countries (UK, Canada and Australia) and generate moods for the top lists there. Since the US market is crowded, K3G also is interested in releasing their service in India and China where they could potentially address a 3 Billion target market! In order to test feasibility, they would want to be able to build a classifier that would classify "any" Hindi/Chinese song as Happy/Sad. In addition, K3G researched and found 3 approaches to building models. A google tutorial leveraging Tensorflow and Keras, A research project for mood analysis, Using tone apis.

Link of report:

https://docs.google.com/presentation/d/1fO_fYPO8cIgvywyKeo_X3p1z-JuNghaS7wOpVrhmiog/edit#slide=id.g559a5ae760_1_71

Link of presentation video:

<https://youtu.be/V90gL36fzwY>

PART1 PREPARATION

Because 有时快乐有时会伤心 music Co. will be launched in China that will take any Chinese songs in Simplified Chinese as input and will return whether the song was happy/sad. In order to realize the according functions belows:

1. Have a text box that can take Chinese songs as an input.
2. On the back end, use Google Translate API to translate this to English.
3. Use the modeling approach Tone Analysis APIs to computer the scores.

We need to make concrete plan to prepare and finish it.

- Our team are tasked with using tone analysis APIs for models review.
- Review APIs for Google Translate and Music Match. We need to use these when building out REST APIs and calling services.
- Learn how to build a FLASK app.

PART2 Using Tone Analysis APIs

- **What is Tone Analyzer?**

IBM Watson is an outstanding representative of cognitive computing systems and a technology platform. There are many practical APIs have been released through IBM Watson Developer Cloud, and Tone Analyzer is one of these APIs.

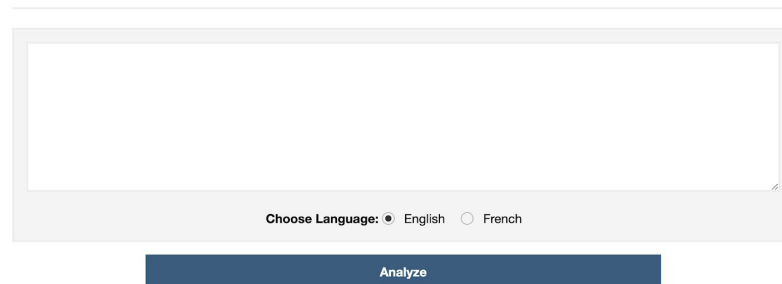
Tone Analyzer service uses linguistic analysis to detect joy, fear, sadness, anger, analytical, confident and tentative tones found in text. You can use the service to understand how your written communications are perceived and then to improve the tone of your communications.

Sample use cases

Choose an example to learn how you can adjust the tone of your content to change people's perceptions, or improve its effectiveness.
[Learn more.](#)

☐ Tweets ☐ Online Review ☐ Email message ☐ Product Review in French ☒ Your own text

Analyzing Customer Engagement Data? Try out the [Tone Analyzer Customer Engagement Endpoint](#).



The screenshot displays the IBM Tone Analyzer web interface. At the top, there's a section titled 'Sample use cases' with a link to 'Learn more'. Below this, there are radio buttons for selecting the type of content: 'Tweets', 'Online Review', 'Email message', 'Product Review in French', and 'Your own text' (which is selected). A link for 'Analyzing Customer Engagement Data' is also present. The main part of the interface is a large, empty text input box. Below the input box, there's a 'Choose Language' section with 'English' selected and 'French' as an option. At the bottom, there is a prominent blue 'Analyze' button.

FIGURE1: Interface of Tone Analysis

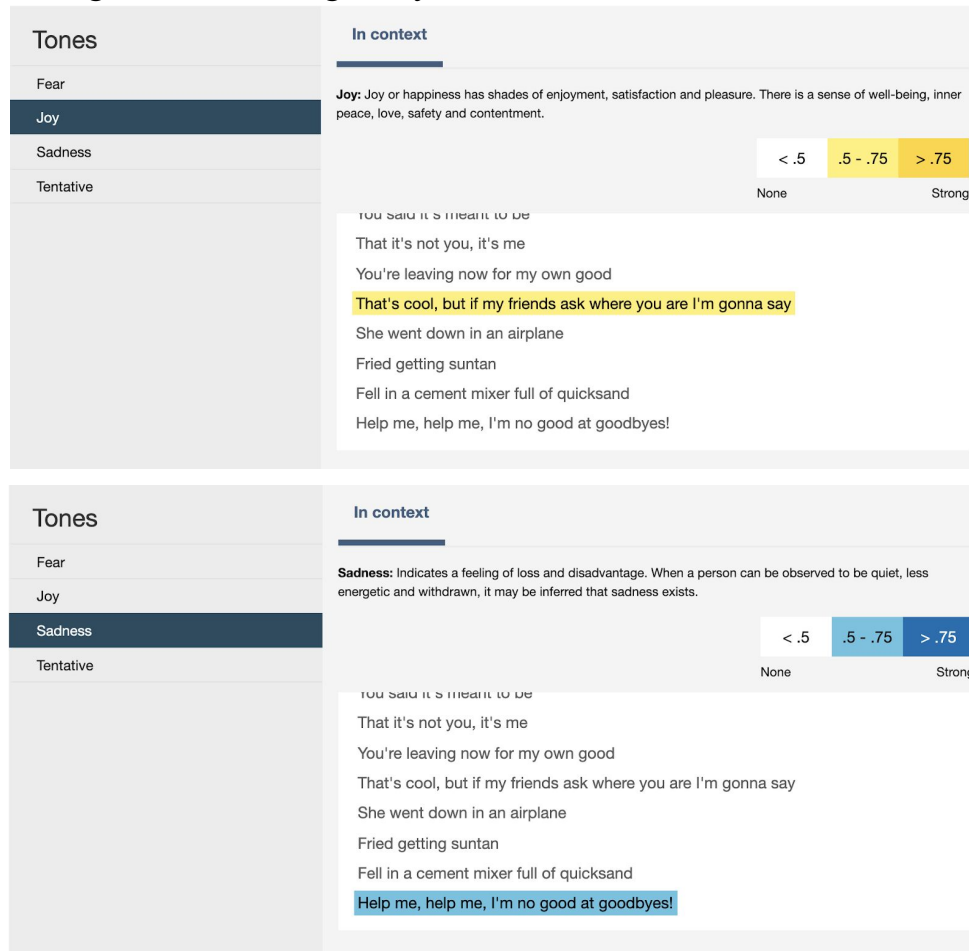
- **Running Instance of Tone Analyzer.**

We can input email, review, tweets or anything you interested to test the function of it. We take the lyrics below as an example:

My heart is paralyzed
My head was over sized
I'll take the high road like I should
You said it's meant to be
That it's not you, it's me
You're leaving now for my own good

That's cool, but if my friends ask where you are I'm gonna say
 She went down in an airplane
 Fried getting suntan
 Fell in a cement mixer full of quicksand
 Help me, help me, I'm no good at goodbyes!

We can get the following analysis results.



● Tone Analyzer's Server Endpoint.

The service endpoint is based on the location of the service instance. For example, when Tone Analyzer is hosted in Frankfurt, the base URL is `https://gateway-fra.watsonplatform.net/tone-analyzer/api`. The URL might also be different when you use IBM Cloud Dedicated.

The Python installation source as follows:

```
pip install --upgrade "watson-developer-cloud>=2.8.0"
```

The Python basic authentication source as follows:

```
from watson_developer_cloud import ToneAnalyzerV3
```

```
tone_analyzer = ToneAnalyzerV3(
    version='{version}',
    username='{username}',
    password='{password}',
    url='{url}'
)
```

The Python examples for the Washington DC location in the constructor and after instantiation source as follows:

```
tone_analyzer = ToneAnalyzerV3(
    version='{version}',
    iam_apikey='{apikey}',
    url='https://gateway-wdc.watsonplatform.net/tone-analyzer/api'
)
```

● Tone Analyzer's Data Handling.

1) Additional headers

Some Watson services accept special parameters in headers that are passed with the request. You can pass request header parameters in all requests or in a single request to the service.

To pass header parameters with every request, use the `setDefaultHeaders` method of the service object. See Data collection for an example use of this method.

To pass header parameters in a single request, use the `addHeader` method as a modifier on the request before you execute the request.

The Python source for example header parameter in a request as follows:

```
response = tone_analyzer.methodName(
    parameters,
    headers = {
        'Custom-Header': '{header_value}'
    })
```

2) Response details

The Tone Analyzer service might return information to the application in response headers.

To access information in the response headers, use one of the request methods that returns details with the response: `executeWithDetails()`, `enqueueWithDetails()`, or `rxWithDetails()`. These methods return a `Response<T>` object, where T is the expected

response model. Use the getResult() method to access the response object for the method, and use the getHeaders() method to access information in response headers.

The Python source for example request to access response headers as follows:

```
tone_analyzer.set_detailed_response(True)
response = tone_analyzer.methodName(parameters)
# Access response from methodName
print(json.dumps(response.get_result(), indent=2))
# Access information in response headers
print(response.get_headers())
# Access HTTP response status
print(response.get_status_code())
```

3) Data collection

Request logging is disabled for the Tone Analyzer service. Regardless of whether you set the X-Watson-Learning-Opt-Out request header, the service does not log or retain data from requests and responses.

PART3 Using Google Translate APIs

- **What is Google Translate?**

The Cloud Translation API dynamically translates between thousands of language pairs. With the Cloud Translation API, we can programmatically integrate websites and programs with translation services.

- **How to use Google Translate Client Library?**

Set up the GCP Console project and enable the Cloud Translation API for this project.

Set the environment variable `GOOGLE_APPLICATION_CREDENTIALS` to the file path of the JSON file that contains your service account key.

The Python source for install client library as follows:

```
pip install --upgrade google-cloud-translate
```

The Python source for run the first example as follows:

```
# Imports the Google Cloud client library
from google.cloud import translate
# Instantiates a client
translate_client = translate.Client()
# The text to translate
text = u'Hello, world!'
# The target language
target = 'ru'
# Translates some text into Russian
translation = translate_client.translate(
    text,
    target_language=target)
print(u'Text: {}'.format(text))
print(u'Translation:{}'.format(translation['translatedText']
))
```

- **The Python source of translate Chinese lyrics to English?**

```
def deal_with_lyrics(text):
    credentials =
service_account.Credentials.from_service_account_file('gcp_
certification.json')
    translate_client =
translate.Client(credentials=credentials)
    translation = translate_client.translate(text, 'en')
    print(translation['translatedText'])
    return translation['translatedText']
```


PART4 Creating Web APIs with Flask

- **Install Python and Flask**

Install Flask using the pip package manager for Python.

```
pip install flask
```

- **What is an API?**

A web API allows for information or functionality to be manipulated by other programs via the internet. For example, with Twitter's web API, you can write a program in a language like Python or Javascript that can perform tasks such as favoriting tweets or collecting tweet metadata.

- **When to Create an API?**

In general, consider an API if:

1. Your data set is large, making download via FTP unwieldy or resource-intensive.
2. Your users will need to access your data in real time, such as for display on another website or as part of an application.
3. Your data changes or is updated frequently.
4. Your users only need access to a part of the data at any one time.
5. Your users will need to perform actions other than retrieve data, such as contributing, updating, or deleting data.

- **What is JSON?**

JavaScript Object Notation is a text-based data storage format that is designed to be easy to read for both humans and machines. JSON is generally the most common format for returning data through an API, XML being the second most common.

- **What is REST?**

REpresentational State Transfer is a philosophy that describes some best practices for implementing APIs. APIs designed with some or all of these principles in mind are called REST APIs. While the API outlined in this lesson uses some REST principles, there is a great deal of disagreement around this term. For this reason, I do not describe the example APIs here as REST APIs, but instead as web or HTTP APIs.

- **What is FLASK?**

Flask is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates. In this section, we will create a basic Flask application. In later sections, we'll add to this application to create our API.

PART5 Result

- The input interface of our app.

Chinese Song Tone Analysis

Analyze Lyrics

Help

Input your lyrics:

Submit

Reset

- The output interface of our app.

Lyric Tones

Tone Name	Score
Sadness	0.702883
Tentative	0.544371
Analytical	0.624611

Sentence Tones

1. It was a sigh that was gently sighed at sunset.

Tone Name	Score
Sadness	0.857056

PART 6 Analysis of Confusion Matrix & ROC

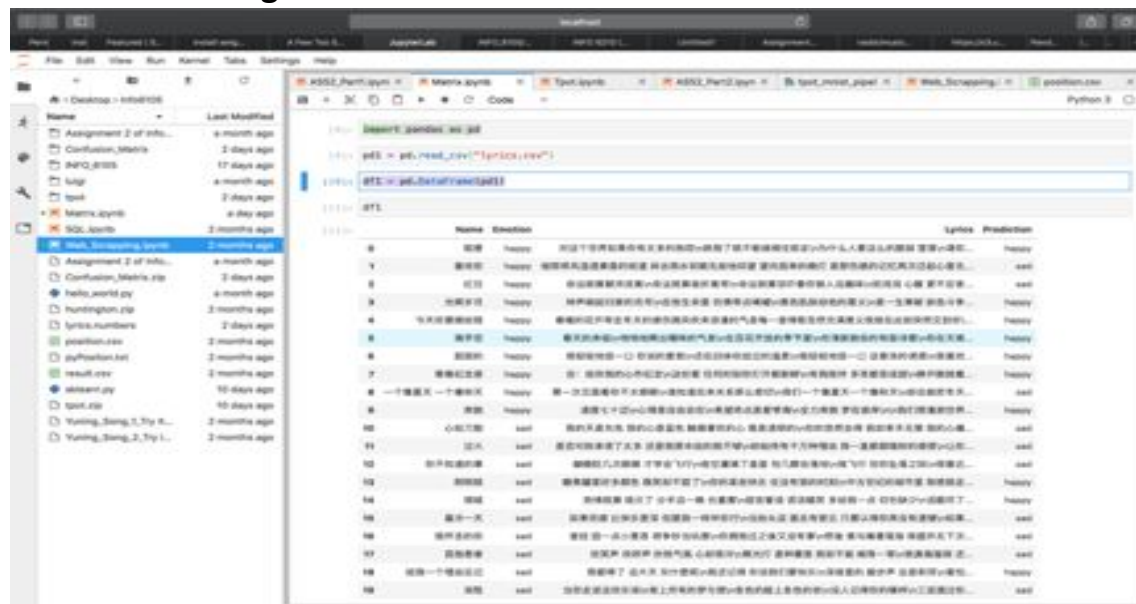
Overview

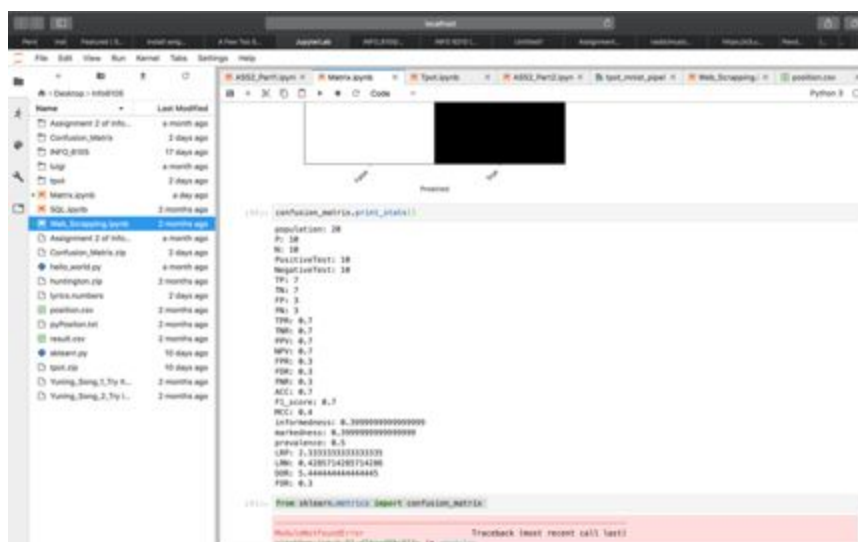
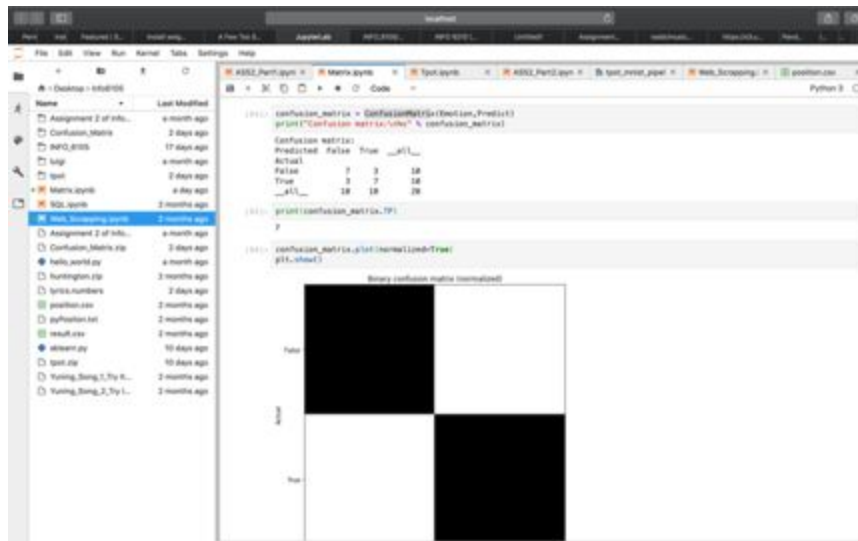
this report to draw the confusion matrices for our analyzer API. Because we do not need to train our model, we can directly use a part of the raw data as the training dataset. We do not need to split that set to training set or validation set. Our training data is based on the Chinese lyrics scraping from the website [10]. We cannot make the raw emotional tag for the dataset. So, we choose a small scaled dataset to add the tag manually. As for the testing data set, I just choose the dataset posted on github:

https://github.com/rasbt/musicmood/blob/master/dataset/validation/valid_lyrics_200.csv

$$F1 = \frac{2TP}{2TP + FP + FN}$$

Matrix of training dataset





According to the above code, the F-1 score of the training data is about 0.7.

Matrix of testing data

Below is the raw data of testing data.

```

In [20]: from sklearn.metrics import confusion_matrix

In [21]: from webson_developer_cloud import TimeAnalyzerV3

In [30]: import pandas as pd

In [31]: pdf = pd.read_csv('data_lyrics_2020.csv')

In [32]: pdf = pd.DataFrame(pdf)

In [77]: pdf

```

#	title	artist	genre	lyrics	result
0	THAT'S WHAT I LOVE ABOUT YOU	Phish	Where is My Mind (2010 Live Version)	Phish	0.9999999999999999
1	THAT'S WHAT I LOVE ABOUT YOU	Queens Of The Stone Age	The Lullaby	Rock	0.9999999999999999
2	THAT'S WHAT I LOVE ABOUT YOU	WCS	Looking At You (Only High School)	Rock	0.9999999999999999
3	THAT'S WHAT I LOVE ABOUT YOU	Dimmu Borgir	The Fundamental Alteration	Rock	0.9999999999999999
4	THAT'S WHAT I LOVE ABOUT YOU	Dark Funeral	The Silver Gate	Rock	0.9999999999999999
5	THAT'S WHAT I LOVE ABOUT YOU	Whisper	Special	Christian	0.9999999999999999
6	THAT'S WHAT I LOVE ABOUT YOU	Headbuck	Follow You Home (Album Version)	Rock	0.9999999999999999
7	THAT'S WHAT I LOVE ABOUT YOU	The Ocean	Cathartes Of A Heron	Rock	0.9999999999999999
8	THAT'S WHAT I LOVE ABOUT YOU	Aut-Pop	Spain's House Destruction Party	Rock	0.9999999999999999
9	THAT'S WHAT I LOVE ABOUT YOU	Wren	Wings	Christian	0.9999999999999999
10	THAT'S WHAT I LOVE ABOUT YOU	Remission	Remission	Rock	0.9999999999999999

```

In [100]: lyrics = pdf['lyrics']

In [140]: type(lyrics)
Out[140]: numpy.ndarray

```

Next is the process of getting emotion array

```

In [585]: mood = []
for text in lyr:
    tone_analysis = tone_analyzer.tone(text, lyrics.get_result())
    for tone in tone_analysis['document_tone']['tones']:
        mood.append(tone.values())

In [386]: mood

```

dict_values([0.829346, 'Joy', 'Joy'])
dict_values([0.547389, 'tentative', 'tentative'])
dict_values([0.539547, 'tentative', 'tentative'])
dict_values([0.629493, 'impr', 'Joy'])
dict_values([0.580348, 'sadness', 'Sadness'])
dict_values([0.81026, 'tentative', 'tentative'])
dict_values([0.61457, 'contentment', 'Contentment'])
dict_values([0.687515, 'analytical', 'Analytical'])
dict_values([0.588276, 'anger', 'Anger'])
dict_values([0.517189, 'sadness', 'Sadness'])
dict_values([0.513893, 'confident', 'Confident'])
dict_values([0.547899, 'sadness', 'Sadness'])
dict_values([0.587277, 'tentative', 'tentative'])
dict_values([0.765874, 'analytical', 'Analytical'])
dict_values([0.636496, 'tentative', 'tentative'])
dict_values([0.630687, 'tentative', 'tentative'])
dict_values([0.766319, 'analytical', 'Analytical'])
dict_values([0.667182, 'tentative', 'tentative'])
dict_values([0.750578, 'tentative', 'tentative'])
dict_values([0.627567, 'sadness', 'Sadness'])

```

In [397]: len(mood)
Out[397]: 153

```

Confusion Matrix

Confusion Matrix of our data sets:

52	26
24	51

Accuracy Score: 0.673202614379085

Other metric:

	precision	recall	F1-score	support
happy	0.68	0.68	0.67	78
sad	0.66	0.68	0.67	75
avg/total	0.67	0.67	0.67	153

According to the above matrix, our accuracy is 0.67. Average F1 score is 0.67. The total valid data is 153 rows but not 200 rows. That's because our API will come out with a multi-emotional result that contains not only happy and sad, but also "fear", "confidence", "analytical" and "tentative". It is hard to just classify the result into happy and sad. I just choose useful results for the processes.

Computing both of above data sets

The results of the both datasets show that the training process is better than the testing process. Because the accuracy of training process is 0.7. But the testing accuracy is around 0.67. But I think this difference is within a reasonable range.

Computing the F-1 score

The F-1 Score of training process is a little bit higher than the testing process. Firstly, I expect the testing score will be higher than the training score. Because the API was built based on English. In other words, we will obviously get a higher score if we use English song to do the prediction. But the truth is that our lyrics in training set has been put through out a translate API. The emotion and the tone of the lyrics may change because of that process. But our training data is so small. I think if the dataset is big enough, the above influence is more visible. Secondly, it is really confused that to classify the 6-parameter output to a single happy-sad output. There might be some error classification and some misleading of tags. Finally, I think not only the F1 score but also the performance of the both processes are pretty similar.

PairPlot Visualization

In this model we can get 7 feature- score of lyrics or a sentence. Definition of these features are as below:

Anger: Likelihood of writer being perceived as angry. Low value indicates unlikely to be perceived as angry. High value indicates very likely to be perceived as angry.

Fear: Likelihood of writer being perceived as scared. Low value indicates unlikely to be perceived as fearful. High value, very likely to be perceived as scared.

Joy: Joy or happiness has shades of enjoyment, satisfaction and pleasure. There is a sense of well-being, inner peace, love, safety and contentment.

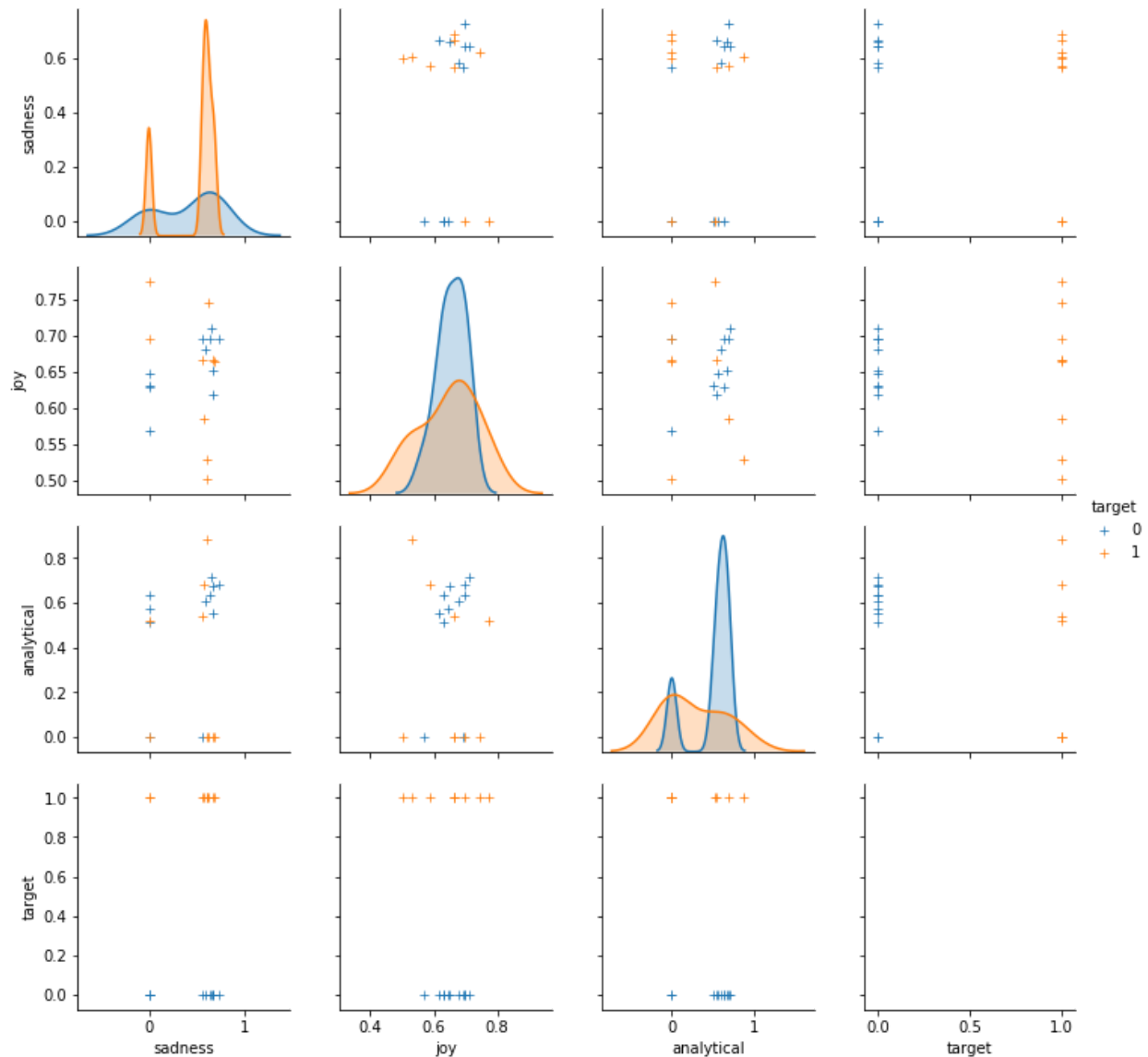
Sadness: Likelihood of writer being perceived as sad. Low value, unlikely to be perceived as sad. High value very likely to be perceived as sad.

Analytic: A writer's reasoning and analytical attitude about things. Higher value, more likely to be perceived as intellectual, rational, systematic, emotionless, or impersonal.

Confident: A writer's degree of certainty. Higher value, more likely to be perceived as assured, collected, hopeful, or egotistical.

Tentative: A writer's degree of inhibition. Higher value, more likely to be perceived as questionable, doubtful, limited, or debatable.

When we do analysis job ,we manually pick three typical ones which are joy, sadness, and analytic since our main job of this assignment is to classify whether the tone of a song is happy or not. We simply obtain 20 songs score and do a pairplot to compare the relationship between each features as below graph:



- After graphing the features in a pair plot, it is clear that the relationship between pairs of features of sadness(represented by 0) is distinctly different from happy(represented by 1) .
- There is some overlap in the pairwise relationships , some sad songs also has happy tones and vice versa.
- Analytic feature is a useable features since when the higher value, more likely to be perceived as intellectual, rational, systematic, emotionless, or impersonal. In other words when the value is higher the less disparity of sadness value and joy value is.

ROC Visualization

As above method ,we can get a matrix of scores of several features and a column result of target score. According to this, we can calculate false-positive-rate(fpr) and true-positive-rate(tp) of each instance and get an auc value and a roc curve of this model.

the procedure is as below:

```
: from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc ###计算roc和auc
# from sklearn import cross_validation
from sklearn import model_selection

random_state = np.random.RandomState(0)
n_samples, n_features = X.shape
X = np.c_[X, random_state.randn(n_samples, 200 * n_features)]

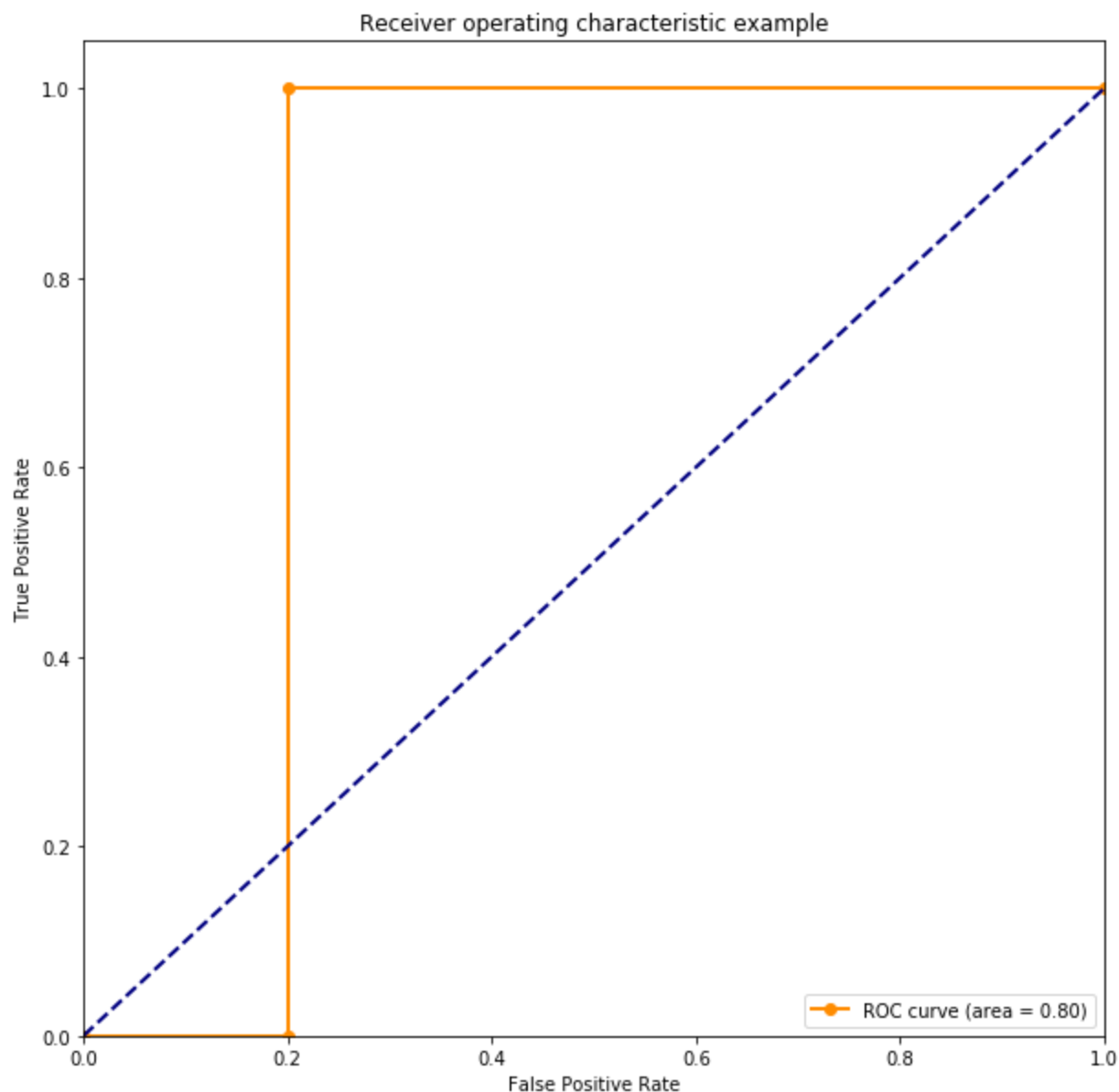
# shuffle and split training and test sets
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=.3, random_state=0)

# Learn to predict each class against the other
svm = svm.SVC(kernel='linear', probability=True, random_state=random_state)

###通过decision_function()计算得到的y_score的值, 用在roc_curve()函数中
y_score = svm.fit(X_train, y_train).decision_function(X_test)
print(y_score)

# Compute ROC curve and ROC area for each class
fpr, tpr, threshold = roc_curve(y_test, y_score) ###计算真正率和假正率
roc_auc = auc(fpr, tpr) ###计算auc的值

plt.figure()
lw = 2
plt.figure(figsize=(10,10))
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc, marker = 'o') ###假正率为横坐标, 真正率为纵坐标做曲线
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



Graph above is the result curve of our ROC analysis

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - SPC$$

As you can see in the graph our AUC (area under curve) value is 0.80. As we know the higher the value is ,the better the model is. Thus generally speaking this tone analysis model performs well.

Conclusion

The tone analyzer API can classify the input text to different emotion like joy, fear, sadness, anger, analytical, confident and tentative. The performance of the API is obviously higher than the result of our experiment. We just classify the emotion to sad and happy, although the emotion like analytical is a really tricky part and hard to define. In other words, the API is a really precise and effective process while testing. If we see each score of their prediction inside the results. We can see if there is a prediction, the score must be higher than 85%. If we set the method into a status that is more suitable for the method, I think it will output a high accuracy and F1 score about the predictions.