REPORT

Name ： Yiqun Peng

1. Background

Text sentiment analysis (Sentiment Analysis) refers to the process of analyzing, processing and extracting sentimental subjective text using natural language processing and text mining technology. At present, the research of text sentiment analysis covers many fields including natural language processing, text mining, information retrieval, information extraction, machine learning, and ontology. It has attracted the attention of many scholars and research institutions, and has continued to become natural language processing in recent years. And one of the hot issues in the field of text mining. Sentiment analysis tasks can be divided into chapter-level, sentence-level, word or phrase-level according to the granularity of their analysis; according to the type of text processed, they can be divided into sentiment analysis based on product reviews and sentiment analysis based on news reviews; according to their research tasks Types can be divided into sub-problems such as emotion classification, emotion retrieval and emotion extraction.

2. load data

We use negative_tweets.json (5000 tweets with negative emotions) and

positive_tweets.json under twitter_samples: (5000 tweets with positive emotions are used to train the model)

```
po_file_path = 'positive_tweets.json'
ne_file_path = 'negative_tweets.json'

positive_tweets = twitter_samples.strings(po_file_path)
negative_tweets = twitter_samples.strings(ne_file_path)
for i in range(6):
    print(positive_tweets[i])
    print(negative_tweets[i])
```

```
#FollowFriday @France_Inte @PKuchly57 @Milipol_Paris for being top engaged members in my community this week :)
hopeless for tmr :(
@Lamb2ja Hey James! How odd :/ Please call our Contact Centre on 02392441234 and we will be able to assist you :) Many tha
Everything in the kids section of IKEA is so cute. Shame I'm nearly 19 in 2 months :(
@DespiteOfficial we had a listen last night :) As You Bleed is an amazing track. When are you in Scotland?!
@Hegelbon That heart sliding into the waste basket. :(
@97sides CONGRATS :)
"@ketchBurning: I hate Japanese call him "bani" :( :("

Me too
yeaaaah yippppy!!!  my accnt verified rqst has succeed got a blue tick mark on my fb profile :) in 15 days
Dang starting next week I have "work" :(
@BhaktisBanter @PallaviRuhail This one is irresistible :)
#FlipkartFashionFriday http://t.co/EbZ0L2VENM
oh god, my babies' faces :( https://t.co/9fcwGvaki0
```

3. Word segmentation

Word segmentation is to decompose long texts such as sentences, paragraphs, and articles into data structures in units of words to

facilitate subsequent processing and analysis.

```
po_fenci_res = fenci(po_file_path)
be_fenci_res = fenci(ne_file_path)

print('Positive participle result: {}'.format(po_fenci_res))
print('Negative participle result: {}'.format(be_fenci_res))
```

```
Positive participle result: [['#FollowFriday', '@France_Inte', '@PKuchly57', '@Milipol_Paris', 'for', 'being', 'top', 'engaged', 'members', 'in', 'my', 'community', 'this', 'week', ':)'],
Negative participle result: [['hopeless', 'for', 'tmr', ':('], ['Everything', 'in', 'the', 'kids', 'section', 'of', 'IKEA', 'is', 'so', 'cute', '.', 'Shame', "I'm", 'nearly', '19', 'in',
```

4. Data normalization

Data normalization includes the following steps:

Part-of-speech tagging

Junk data processing

Part of speech

```
def cleaned_list_func(evert_tweet):
    new_text = []
    cixing_list = pos_tag(evert_tweet)
    for word, cixing in cixing_list:
        word = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|(?:[0-9a-fA-F][0-9a-fA-F]))+', '', word)
        word = re.sub('(@[A-Za-z0-9_]+)', '', word)
        if cixing.startswith('NN'):
            pos = 'n'
        elif cixing.startswith('VB'):
            pos = 'v'
        else:
            pos = 'a'
        lemmatizer = WordNetLemmatizer()
        new_word = lemmatizer.lemmatize(word, pos)
        if len(new_word) > 0 and new_word not in string.punctuation and new_word.lower() not in stopwords.words(
                'english'):
            new_text.append(new_word.lower())
    return new_text
```

```
Positive tweet results after processing: [['#followfriday', 'top', 'engage', 'member', 'community', 'week', ':)'], ['hey', 'james', 'odd', ':/', 'please', 'call', 'contact', 'centre', '023
original data: ['#FollowFriday @France_Inte @PKuchly57 @Milipol_Paris for being top engaged members in my community this week :)', '@Lamb2ja Hey James! How odd :/ Please call our Contact C
```

5. construct model data

```
def get_tweets_for_model(clean_tokens_list, tag):
    li = []
    for every_tweet in clean_tokens_list:
        data_dict = dict([token, True] for token in every_tweet)
        li.append((data_dict, tag))
    return li
```

positive data: [({'#followfriday': True, 'top': True, 'engage': True, 'member': True, 'community': True, 'week': True, ':)': True}, 'Positive'), ({'hey': True, 'james': True, 'odd': True,

negative data: [({'hopeless': True, 'tmr': True, ':(': True}, 'Negative'), ({'everything': True, 'kid': True, 'section': True, 'ikea': True, 'cute': True, 'shame': True, "i'm": True, 'near

6. Train and test the model

```
def train_model(train_data):
    from nltk import NaiveBayesClassifier
    model = NaiveBayesClassifier.train(train_data)
    return model


def test(test_text):
    from nltk.tokenize import word_tokenize
    custom_tokens = cleaned_list_func(word_tokenize(test_text))
    result = dict([token, True] for token in custom_tokens)
    return result
```

7. Result

```
positive data: [({'#followfriday': True, 'top': True, 'engage': True, 'member': Tru
negative data: [({'hopeless': True, 'tmr': True, ':(': True}, 'Negative'), ({'every
[1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1
                                                    0    1  2  3
0    greenlights is a remarkable first book from an...  4.0  1  1
1    worst book ever written.  the only redeeming v...  1.0  0  0
2    amazing book, clever and funny. not your typic...  5.0  1  1
3    already received and enjoying it immensely. be...  5.0  1  1
4    excellent book  matthew is far more complex th...  5.0  0  1
..                                                   ...  ... .. ..
770  arrived with marks on the cover. was expecting...  1.0  1  0
771  we received our book today and it was damaged....  1.0  0  0
772                                  i love this book   5.0  1  1
773          un de mes livres préféré. juste sublime.   5.0  0  1
774        not what i was expecting. but its kinda ok.  2.0  1  0

[775 rows x 4 columns]
Accuray:  0.7858064516129032

Process finished with exit code 0
```