Rule-Based Sentiment Analysis in Python

Home

Harika Bonthu — June 18, 2021

Advanced Classification NLP Project Python Structured Data

This article was published as a part of the Data Science Blogathon

Image by Author made online on befunky.com

Intro:

According to experts, 80% of the world's existing data is in the form of unstructured data(images, videos, text, etc). This data could be generated by Social media tweets/posts, call transcripts, survey or interview reviews, text across blogs, forums, news, etc.

It is humanly impossible to read all the text across the web and find patterns. Yet, there is definitely a need for the business to analyze this data for better actions.

One such process of drawing insights from textual data is Sentiment Analysis. To obtain the data for sentiment analysis, one can directly scrape the content from the web pages using different web scraping techniques.

If you are new to web scraping, feel free to check out my article "Web scraping with Python: BeautifulSoup".

What is Sentiment Analysis?

Sentiment Analysis (also known as opinion mining or emotion AI) is a sub-field of NLP that measures the inclination of people's opinions (Positive/Negative/Neutral) within the unstructured text.

Sentiment Analysis can be performed using two approaches: Rule-based, Machine Learning based.

Few applications of Sentiment Analysis

Rule-Based Sentiment Analysis in Python

- Market research

What is Natural Language Processing(NLP)?

Natural Language is the way we, humans, communicate with each other. It could be Speech or Text. NLP is the automatic manipulation of the natural language by software. NLP is a higher-level term and is the combination of Natural Language Understanding (NLU) and Natural Language Generation (NLG).

```
NLP = NLU + NLG
```

Some of the Python Natural Language Processing (NLP) libraries are:

- Natural Language Toolkit (NLTK)
- TextBlob
- SpaCy
- Gensim
- CoreNLP

I hope we have got a basic understanding of the terms Sentiment Analysis, NLP.

This article focusses on the Rule-based approach of Sentiment Analysis

Rule-based approach

This is a practical approach to analyzing text without training or using machine learning models. The result of this approach is a set of rules based on which the text is labeled as positive/negative/neutral. These rules are also known as lexicons. Hence, the Rule-based approach is called Lexicon based approach.

Widely used lexicon-based approaches are TextBlob, VADER, SentiWordNet.

Data preprocessing steps:

1. Cleaning the text
2. Tokenization
3. Enrichment – POS tagging
4. Stopwords removal
5. Obtaining the stem words

Before deep-diving into the above steps, lemme import the text data from a txt file.

Importing a text file using Pandas read CSV function

```
# install and import pandas library
import pandas as pd
# Creating a pandas dataframe from reviews.txt file
data = pd.read_csv('reviews.txt', sep='t')
data.head()
```

Rule-Based Sentiment Analysis in Python

0	0	I called because my food was cold and not done...
1	1	OMG, hands down the best pizza I've had from D...
2	2	This Domino's has the best pizza delivery and ...
3	3	My Sweetheart & I are very pleased with the qu...
4	4	I called to place an order, The lady answered ...

This doesn't look cool. So, we will now drop the "Unnamed: 0" column using the df.drop function.

```
mydata = data.drop('Unnamed: 0', axis=1)
mydata.head()
```

	review
0	I called because my food was cold and not done...
1	OMG, hands down the best pizza I've had from D...
2	This Domino's has the best pizza delivery and ...
3	My Sweetheart & I are very pleased with the qu...
4	I called to place an order, The lady answered ...

Our dataset has a total of 240 observations(reviews).

Step 1: Cleaning the text

In this step, we need to remove the special characters, numbers from the text. We can use the regular expression operations library of Python.

```
# Define a function to clean the text
def clean(text):
# Removes all special characters and numericals leaving the alphabets
    text = re.sub('[^A-Za-z]+', ' ', text)
    return text

# Cleaning the text in the review column
mydata['Cleaned Reviews'] = mydata['review'].apply(clean)
mydata.head()
```

Explanation: "clean" is the function that takes text as input and returns the text without any punctuation marks or numbers in it. We applied it to the 'review' column and created a new column 'Cleaned Reviews' with the cleaned text.

	review	Cleaned Reviews
0	I called because my food was cold and not done...	I called because my food was cold and not done...
1	OMG, hands down the best pizza I've had from D...	OMG hands down the best pizza I ve had from Do...
2	This Domino's has the best pizza delivery and ...	This Domino s has the best pizza delivery and ...
3	My Sweetheart & I are very pleased with the qu...	My Sweetheart I are very pleased with the qual...
4	I called to place an order, The lady answered ...	I called to place an order The lady answered a...

Great, look at the above image, all the special characters and the numbers are removed.

Rule-Based Sentiment Analysis in Python

tokenization) or word level(word tokenization).

I will be performing word-level tokenization using nltk tokenize function word_tokenize().

Note: As our text data is a little large, first I will illustrate steps 2-5 with small example sentences.

Let's say we have a sentence "**This is an article on Sentiment Analysis**". It can be broken down into small pieces(tokens) as shown below.

```
text = "This is an article on Sentiment Analysis"
tokens = word_tokenize(text)
tokens
```

```
['This', 'is', 'an', 'article', 'on', 'Sentiment', 'Analysis']
```

Step 3: Enrichment – POS tagging

Parts of Speech (POS) tagging is a process of converting each token into a tuple having the form (word, tag). POS tagging essential to preserve the context of the word and is essential for Lemmatization.

This can be achieved by using the nltk pos_tag function.

Below shown are the POS tags of the example sentence "This is an article on Sentiment Analysis".

```
pos = nltk.pos_tag(tokens)
pos
```

```
[('This', 'DT'),
 ('is', 'VBZ'),
 ('an', 'DT'),
 ('article', 'NN'),
 ('on', 'IN'),
 ('Sentiment', 'NN'),
 ('Analysis', 'NN')]
```

Check out the list of possible pos tags from here.

Step 4: Stopwords removal

Stopwords in English are words that carry very little useful information. We need to remove them as part of text preprocessing. nltk has a list of stopwords of every language.

See the stopwords in the English language.

```
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'its
elf', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these',
'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doin
g', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'o
ut', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'al
l', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o',
're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'has
n', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'sha
n', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Rule-Based Sentiment Analysis in Python

```
tokens = word_tokenize(text)

new_text = (" ").join(ele for ele in tokens if ele.lower() not in stopwords.words('english'))
new_text
```

```
'article Sentiment Analysis'
```

The stopwords This, is, an, on are removed and the output sentence is 'article Sentiment Analysis'.

Step 5: Obtaining the stem words

A stem is a part of a word responsible for its lexical meaning. The two popular techniques of obtaining the root/stem words are Stemming and Lemmatization.

The key difference is Stemming often gives some meaningless root words as it simply chops off some characters in the end. Lemmatization gives meaningful root words, however, it requires POS tags of the words.

Example to illustrate the difference between Stemming and Lemmatization: Click here for code

```
Text: He glanced up from his computer when she came into his office
Stem: ['glanc', 'comput', 'came', 'offic']
Lemma: ['glance', 'computer', 'come', 'office']
```

If we look at the above example, the output from Stemming is Stem, and the output from Lemmatizatin is Lemma.

For the word glanced, the stem **glanc** is meaningless. Whereas, the Lemma **glance** is perfect.

We now understood steps 2-5 by taking simple examples. Without any further delay, let us bounce back to our actual problem.

Code for Steps 2-4: Tokenization, POS tagging, Stopwords removal

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk import pos_tag
nltk.download('stopwords')
from nltk.corpus import stopwords
nltk.download('wordnet')
from nltk.corpus import wordnet

# POS tagger dictionary
pos_dict = {'J':wordnet.ADJ, 'V':wordnet.VERB, 'N':wordnet.NOUN, 'R':wordnet.ADV}
def token_stop_pos(text):
    tags = pos_tag(word_tokenize(text))
    newlist = []
    for word, tag in tags:
        if word.lower() not in set(stopwords.words('english')):
        newlist.append(tuple([word, pos_dict.get(tag[0])]))
    return newlist

mydata['POS tagged'] = mydata['Cleaned Reviews'].apply(token_stop_pos)
mydata.head()
```

Explanation: token_stop_pos is the function that takes the text and performs tokenization, removes stopwords, and tags the words to their POS. We applied it to the 'Cleaned Reviews' column and created a new column for 'POS tagged' data.

Rule-Based Sentiment Analysis in Python

the POS tags obtained from pos_tag are in the form of 'NN', 'ADJ', etc.

To map pos_tag to wordnet tags, we created a dictionary pos_dict. Any pos_tag that starts with J is mapped to wordnet.ADJ, any pos_tag that starts with R is mapped to wordnet.ADV, and so on.

Our tags of interest are Noun, Adjective, Adverb, Verb. Anything out of these four is mapped to None.

	review	Cleaned Reviews	POS tagged
0	I called because my food was cold and not done...	I called because my food was cold and not done...	[(called, v), (food, n), (cold, a), (done, v),...
1	OMG, hands down the best pizza I've had from D...	OMG hands down the best pizza I ve had from Do...	[(OMG, n), (hands, v), (best, a), (pizza, n), ...
2	This Domino's has the best pizza delivery and ...	This Domino s has the best pizza delivery and ...	[(Domino, n), (best, a), (pizza, n), (delivery...
3	My Sweetheart & I are very pleased with the qu...	My Sweetheart I are very pleased with the qual...	[(Sweetheart, n), (pleased, a), (quality, n), ...
4	I called to place an order, The lady answered ...	I called to place an order The lady answered a...	[(called, v), (place, v), (order, n), (lady, n...

In the above fig, we can observe that each word of column 'POS tagged' is mapped to its POS from pos_dict.

Code for Step 5: Obtaining the stem words – Lemmatization

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
def lemmatize(pos_data):
    lemma_rew = " "
    for word, pos in pos_data:
    if not pos:
        lemma = word
        lemma_rew = lemma_rew + " " + lemma
    else:
        lemma = wordnet_lemmatizer.lemmatize(word, pos=pos)
        lemma_rew = lemma_rew + " " + lemma
    return lemma_rew

mydata['Lemma'] = mydata['POS tagged'].apply(lemmatize)
mydata.head()
```

Explanation: lemmatize is a function that takes pos_tag tuples, and gives the Lemma for each **word** in pos_tag based on the **pos** of that word. We applied it to the 'POS tagged' column and created a column 'Lemma' to store the output.

	review	Cleaned Reviews	POS tagged	Lemma
0	I called because my food was cold and not done...	I called because my food was cold and not done...	[(called, v), (food, n), (cold, a), (done, v),...	call food cold do right miss item call answe...
1	OMG, hands down the best pizza I've had from D...	OMG hands down the best pizza I ve had from Do...	[(OMG, n), (hands, v), (best, a), (pizza, n), ...	OMG hand best pizza Domino pizza Southaven M...
2	This Domino's has the best pizza delivery and ...	This Domino s has the best pizza delivery and ...	[(Domino, n), (best, a), (pizza, n), (delivery...	Domino best pizza delivery customer service ...
3	My Sweetheart & I are very pleased with the qu...	My Sweetheart I are very pleased with the qual...	[(Sweetheart, n), (pleased, a), (quality, n), ...	Sweetheart pleased quality service excellent...
4	I called to place an order, The lady answered ...	I called to place an order The lady answered a...	[(called, v), (place, v), (order, n), (lady, n...	call place order lady answer already know na...

Yay, after a long journey, we are done with preprocessing of the text.

Now, take a minute to look at the 'review', 'Lemma' columns and observe how the text is processed.

Rule-Based Sentiment Analysis in Python

	review	Lemma
0	I called because my food was cold and not done...	call food cold do right miss item call answe...
1	OMG, hands down the best pizza I've had from D...	OMG hand best pizza Domino pizza Southaven M...
2	This Domino's has the best pizza delivery and ...	Domino best pizza delivery customer service ...
3	My Sweetheart & I are very pleased with the qu...	Sweetheart pleased quality service excellent...
4	I called to place an order, The lady answered ...	call place order lady answer already know na...
...	...	...
235	Good quality and cheap pizzas. They offer wide...	Good quality cheap pizza offer wide range pi...
236	Every time I order online they forget to put m...	Every time order online forget put olive ord...
237	Store: SHOP NO. 17 & 18, A WING, HALLMARK, VAS...	Store SHOP WING HALLMARK VASANT OSCAR LBS MA...
238	Chicken topping was so good, I could not sit s...	Chicken topping good could sit still excelle...
239	People reading this don't place an order in Do...	People read place order Domino still want pl...

240 rows × 2 columns

As we are done with the data preprocessing, our final data looks clean. Take a short break, and come back to continue with the real task.

Sentiment Analysis using TextBlob:

TextBlob is a Python library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

The two measures that are used to analyze the sentiment are:

- Polarity – talks about how positive or negative the opinion is
- Subjectivity – talks about how subjective the opinion is

TextBlob(text).sentiment gives us the Polarity, Subjectivity values.

Polarity ranges from -1 to 1 (1 is more positive, 0 is neutral, -1 is more negative)

Subjectivity ranges from 0 to 1(0 being very objective and 1 being very subjective)

```
res = TextBlob("I love horror films").sentiment
res
```

```
Sentiment(polarity=0.5, subjectivity=0.6)
```

Example of TextBlob sentiment

Python Code:

Rule-Based Sentiment Analysis in Python

```
# function to calculate subjectivity
def getSubjectivity(review):
    return TextBlob(review).sentiment.subjectivity
    # function to calculate polarity
    def getPolarity(review):
        return TextBlob(review).sentiment.polarity

# function to analyze the reviews
def analysis(score):
    if score < 0:
        return 'Negative'
    elif score == 0:
        return 'Neutral'
    else:
        return 'Positive'
```

Explanation: created functions to obtain Polarity, Subjectivity values and to Label the review based on the Polarity score.

Creating a new data frame with the review, Lemma columns and apply the above functions

```
fin_data = pd.DataFrame(mydata[['review', 'Lemma']])
```

```
# fin_data['Subjectivity'] = fin_data['Lemma'].apply(getSubjectivity)
fin_data['Polarity'] = fin_data['Lemma'].apply(getPolarity)
fin_data['Analysis'] = fin_data['Polarity'].apply(analysis)
fin_data.head()
```

	review	Lemma	Polarity	Analysis
0	I called because my food was cold and not done...	call food cold do right miss item call answe...	0.217143	Positive
1	OMG, hands down the best pizza I've had from D...	OMG hand best pizza Domino pizza Southaven M...	0.538889	Positive
2	This Domino's has the best pizza delivery and ...	Domino best pizza delivery customer service ...	0.355000	Positive
3	My Sweetheart & I are very pleased with the qu...	Sweetheart pleased quality service excellent...	0.238542	Positive
4	I called to place an order, The lady answered ...	call place order lady answer already know na...	0.650000	Positive

Count the number of positive, negative, neutral reviews.

```
tb_counts = fin_data.Analysis.value_counts()
```

```
tb_counts
```

```
Negative    119
Positive    114
Neutral       7
Name: Analysis, dtype: int64
```

Count of positive, negative, neutral reviews

Sentiment Analysis using VADER

VADER stands for Valence Aware Dictionary and Sentiment Reasoner. Vader sentiment not only tells if the statement is positive or negative along with the intensity of emotion.

```
vs = analyzer.polarity_scores("I love horror films")
vs
```

Rule-Based Sentiment Analysis in Python

positive if compound >= 0.5

neutral if -0.5 < compound < 0.5

negative if -0.5 >= compound

Python Code:

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
# function to calculate vader sentiment
def vadersentimentanalysis(review):
    vs = analyzer.polarity_scores(review)
    return vs['compound']
    fin_data['Vader Sentiment'] = fin_data['Lemma'].apply(vadersentimentanalysis)
# function to analyse
def vader_analysis(compound):
    if compound >= 0.5:
        return 'Positive'
    elif compound <= -0.5 :
        return 'Negative'
    else:
        return 'Neutral'
fin_data['Vader Analysis'] = fin_data['Vader Sentiment'].apply(vader_analysis)
fin_data.head()
```

Explanation: Created functions to obtain the Vader scores and to label the reviews based on compound scores

Count the number of positive, negative, neutral reviews.

```
vader_counts = fin_data['Vader Analysis'].value_counts()
vader_counts
```

Sentiment Analysis using SentiWordNet

SentiWordNet uses the WordNet database. It is important to obtain the POS, lemma of each word. We will then use the lemma, POS to obtain the synonym sets(synsets). We then obtain the positive, negative, objective scores for all the possible synsets or the very first synset and label the text.

if positive score > negative score, the sentiment is positive

if positive score < negative score, the sentiment is negative

if positive score = negative score, the sentiment is neutral

Python Code:

Rule-Based Sentiment Analysis in Python

```
from nltk.corpus import sentiwordnet as swn
def sentiwordnetanalysis(pos_data):
    sentiment = 0
    tokens_count = 0
    for word, pos in pos_data:
        if not pos:
            continue
            lemma = wordnet_lemmatizer.lemmatize(word, pos=pos)
        if not lemma:
            continue
            synsets = wordnet.synsets(lemma, pos=pos)
        if not synsets:
            continue
            # Take the first sense, the most common
            synset = synsets[0]
            swn_synset = swn.senti_synset(synset.name())
            sentiment += swn_synset.pos_score() - swn_synset.neg_score()
            tokens_count += 1
            # print(swn_synset.pos_score(),swn_synset.neg_score(),swn_synset.obj_score())
        if not tokens_count:
            return 0
        if sentiment>0:
            return "Positive"
        if sentiment==0:
            return "Neutral"
        else:
            return "Negative"

fin_data['SWN analysis'] = mydata['POS tagged'].apply(sentiwordnetanalysis)
fin_data.head()
```

Explanation: We created a function to obtain the positive and negative scores for the first word of the synset then label the text by calculating the sentiment as the difference of positive and negative scores.

Count the number of positive, negative, neutral reviews.

```
swn_counts= fin_data['SWN analysis'].value_counts()
swn_counts
```

Till here, we have seen the implementation of sentiment analysis using some of the popular lexicon-based techniques. Now quickly do some visualization and compare the results.

Visual representation of TextBlob, VADER, SentiWordNet results

We will plot the count of positive, negative, and neutral reviews for all three techniques.

Rule-Based Sentiment Analysis in Python

```
%matplotlib inline
plt.figure(figsize=(15,7))
plt.subplot(1,3,1)
plt.title("TextBlob results")
plt.pie(tb_counts.values, labels = tb_counts.index, explode = (0, 0, 0.25), autopct='%1.1f%%', shadow=False)
plt.subplot(1,3,2)
plt.title("VADER results")
plt.pie(vader_counts.values, labels = vader_counts.index, explode = (0, 0, 0.25), autopct='%1.1f%%',
shadow=False)
plt.subplot(1,3,3)
plt.title("SentiWordNet results")
plt.pie(swn_counts.values, labels = swn_counts.index, explode = (0, 0, 0.25), autopct='%1.1f%%', shadow=False)
```

If we observe the above image, TextBlob and SentiWordNet results look a little close while the VADER results show a large variation.

End Notes:

Congratulations 🎉 to us. By the end of this article, we have learned the various steps of data preprocessing and different lexicon-based approaches for Sentiment Analysis. We compared the results of TextBlob, VADER, SentiWordNet results using Pie plots.

References:

TextBlob documentation

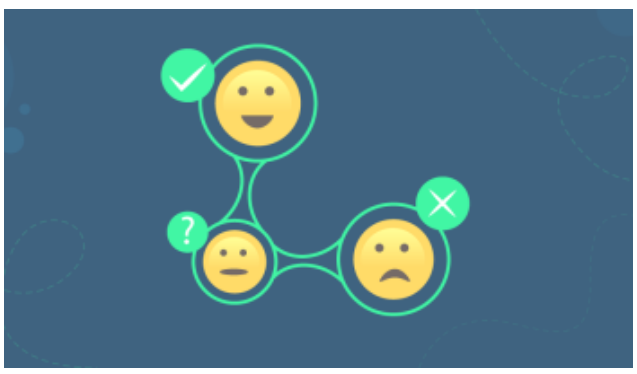
VADER sentiment analysis

SentiWordNet

Check out the complete Jupyter Notebook here hosted on GitHub.

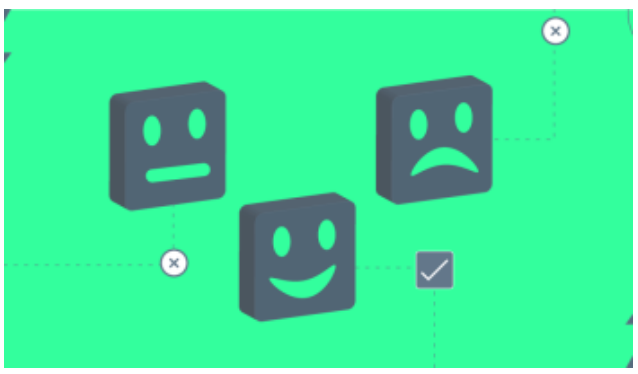
The media shown in this article are not owned by Analytics Vidhya and are used at the Author's discretion.

Related



Sentiment Analysis with TextBlob and Vader

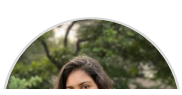
How to Perform Basic Text Analysis without Training Dataset



Web Scraping a News Article and performing Sentiment Analysis using NLP

blogathon Lexicon Lexicon based sentiment analysis Rule-Based sentiment analysis sentiment analysis

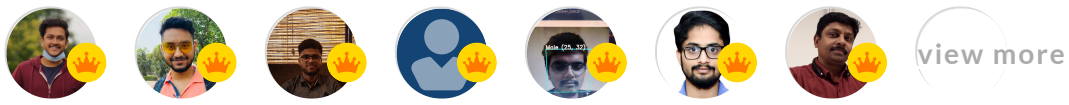
About the Author



Harika Bonthu

Rule-Based Sentiment Analysis in Python

Our Top Authors



Download

Analytics Vidhya App for the Latest blog/Article

Previous Post

Data Manipulation Using Pandas you need to know!

Next Post

Juicing out the Diabetes Patterns amongst Indians using Machine Learning

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

Submit

Top Resources

Rule-Based Sentiment Analysis in Python

TOP 10 GitHub Repositories for Data Science

Ayushi Gupta - JAN 06, 2022

Python Tutorial: Working with CSV file for Data Science

Harika Bonthu - AUG 21, 2021

Machine Learning Algorithms

Amrutha K - JAN 05, 2022

3 Interesting Python Projects With Code for Beginners!

GAURAV SHARMA - JUL 18, 2021