Open in Colab

Unsupervised Sentiment Analysis with Lexicon Models

We talked about unsupervised learning methods in the past, which refer to specific modeling methods that can be applied directly to data features without the presence of labeled data. One of the major challenges in any organization is getting labeled datasets due the lack of time as well as resources to do this tedious task. Unsupervised methods are very useful in this scenario and we look at some of these methods in this section. Even though we have labeled data, this section should give you a good idea of how lexicon based models work and you can apply them to your own datasets when you do not have labeled data.

Unsupervised sentiment analysis models use well curated knowledgebases, ontologies, lexicons, and databases, which have detailed information pertaining to subjective words, phrases including sentiment, mood, polarity, objectivity, subjectivity, and so on.

A lexicon model typically uses a lexicon, also known as a dictionary or vocabulary of words specifically aligned to sentiment analysis. These lexicons contain a list of words associated with positive and negative sentiment, polarity (magnitude of negative or positive score), parts of speech (POS) tags, subjectivity classifiers (strong, weak, neutral), mood, modality, and so on.

You can use these lexicons and compute the sentiment of a text document by matching the presence of specific words from the lexicon and then looking at other factors like presence of negation parameters, surrounding words, overall context, phrases, and aggregate overall sentiment polarity scores to decide the final sentiment score.

There are several popular lexicon models used for sentiment analysis. Some of them are as follows:

- Bing Liu's lexicon
- MPQA subjectivity lexicon
- Pattern lexicon
- TextBlob lexicon
- AFINN lexicon
- SentiWordNet lexicon
- VADER lexicon

This is not an exhaustive list of lexicon models but these are definitely among the most popular ones available today.

Install Dependencies

```
!pip install textblob
!pip install textsearch
!pip install contractions
```

```
!pip install afinn
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('vader_lexicon')
```

```
Requirement already satisfied: textblob in /usr/local/lib/python3.7/dist-packages (0.15.3)
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.7/dist-packages (from textblob) (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from nltk>=3.1->textblob) (1.15.0)
Requirement already satisfied: textsearch in /usr/local/lib/python3.7/dist-packages (0.0.21)
Requirement already satisfied: pyahocorasick in /usr/local/lib/python3.7/dist-packages (from textsearch) (1.4.4)
Requirement already satisfied: anyascii in /usr/local/lib/python3.7/dist-packages (from textsearch) (0.3.0)
Requirement already satisfied: contractions in /usr/local/lib/python3.7/dist-packages (0.1.66)
Requirement already satisfied: textsearch>=0.0.21 in /usr/local/lib/python3.7/dist-packages (from contractions) (0.0.21)
Requirement already satisfied: pyahocorasick in /usr/local/lib/python3.7/dist-packages (from textsearch>=0.0.21->contractions) (1.4.4)
Requirement already satisfied: anyascii in /usr/local/lib/python3.7/dist-packages (from textsearch>=0.0.21->contractions) (0.3.0)
Requirement already satisfied: afinn in /usr/local/lib/python3.7/dist-packages (0.1)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
True
```

Load Dependencies

```
import pandas as pd
import numpy as np
import nltk
import textblob
from sklearn.metrics import confusion_matrix, classification_report
np.set_printoptions(precision=2, linewidth=80)
```

Load Dataset

```
# adapted code to download files from storage
from google.colab import files

uploaded = files.upload()
```

Choose Files noneutral_o…an2022.xlsx

- **noneutral_orange_solana_15jan2022.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 44806 bytes, last modified: 2/18/2022 - 100% done

```
Saving noneutral_orange_solana_15jan2022.xlsx to noneutral_orange_solana_15jan2022.xlsx
```

```
# adapted code to read xlsx files
import pandas as pd
import io
dataset = pd.read_excel(io.BytesIO(uploaded['noneutral_orange_solana_15jan2022.xlsx']))
```

```
#reviews = np.array(dataset['review'])
reviews = np.array(dataset['text']) # the dataset field is called 'text' on orange xlsx results
sentiments = np.array(dataset['sentiment'])

# extract data for model evaluation
test_reviews = reviews[1:]
test_sentiments = sentiments[1:]
sample_review_ids = [2,5,10,50,100]
```

Sentiment Analysis with TextBlob

The lexicon that TextBlob uses is the same one as pattern and is available in their source code on GitHub (https://github.com/sloria/TextBlob/blob/dev/textblob/en/en-sentiment.xml). Some sample examples are shown from the lexicon as follows.

```
<word form="abhorrent" wordnet_id="a-1625063" pos="JJ" sense="offensive
to the mind" polarity="-0.7" subjectivity="0.8" intensity="1.0"
reliability="0.9" />
<word form="able" cornetto_synset_id="n_a-534450" wordnet_id="a-01017439"
pos="JJ" sense="having a strong healthy body" polarity="0.5"
subjectivity="1.0" intensity="1.0" confidence="0.9" />
```

Typically, specific adjectives have a polarity score (negative/positive, -1.0 to +1.0) and a subjectivity score (objective/subjective, +0.0 to +1.0).

The reliability score specifies if an adjective was hand-tagged (1.0) or inferred (0.7). Words are tagged per sense, e.g., ridiculous (pitiful) = negative, ridiculous (humorous) = positive.

The Cornetto id (lexical unit id) and Cornetto synset id refer to the Cornetto lexical database for Dutch. The WordNet id refers to the WordNet3 lexical database for English. The part-of-speech tags (POS) use the Penn Treebank convention. Let’s look at how we can use TextBlob for sentiment analysis.

Predict sentiment for sample reviews

```
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[sample_review_ids]):
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    print('Predicted Sentiment polarity:', textblob.TextBlob(review).sentiment.polarity)
    print('-'*60)
```

```
REVIEW: @solanatiger_slt @CoinMarketCap @coingecko @solana @RaydiumProtocol This is a project that will be very useful in the future because it is supp
@Shahin61918436
@Hawrekurdd
@hooman_k99
@Shilankurd1
Actual Sentiment: positive
Predicted Sentiment polarity: 0.2914285714285714
------------------------------------------------------------
REVIEW: @solana Sol the best there is!
Actual Sentiment: positive
Predicted Sentiment polarity: 1.0
------------------------------------------------------------
REVIEW: @pushpendrakum With that we are going to see more downtime on these networks as well.

Last year @solana was down for 12 hrs.

Long way to go ✌️
Actual Sentiment: positive
Predicted Sentiment polarity: 0.07361111111111111
------------------------------------------------------------
REVIEW: @TheBigHeartsNFT @ethereum @solana love your project, could you dm me please?
Actual Sentiment: positive
Predicted Sentiment polarity: 0.5
------------------------------------------------------------
REVIEW: @OrionDepp @solana @Orion__Research @Orion__Trading @KhadijaTahsin @sadman788 @Sabbir1478
Good project and strong project
bsc:0x7a4D7667B54c2f99db9aEf78a892840Ef042178A
Actual Sentiment: positive
Predicted Sentiment polarity: 0.5666666666666667
------------------------------------------------------------
```

Predict sentiment for test dataset

```
sentiment_polarity = [textblob.TextBlob(review).sentiment.polarity for review in test_reviews]
```

```
# adapted codo to get positive score above 0, instead of score >= 0.1
predicted_sentiments = ['positive' if score > 0 else 'negative' for score in sentiment_polarity]
```

Evaluate model performance

```
labels = ['negative', 'positive']
print(classification_report(test_sentiments, predicted_sentiments))
pd.DataFrame(confusion_matrix(test_sentiments, predicted_sentiments), index=labels, columns=labels)
```

```
              precision    recall  f1-score   support

    negative       0.45      0.50      0.47        26
    positive       0.92      0.90      0.91       166

    accuracy                           0.85       192
   macro avg       0.68      0.70      0.69       192
weighted avg       0.86      0.85      0.85       192
```

	negative	positive
negative	13	13
positive	16	150

Sentiment Analysis with AFINN

The AFINN lexicon is perhaps one of the simplest and most popular lexicons and can be used extensively for sentiment analysis. Developed and curated by Finn Årup Nielsen, you can find more details on this lexicon in the paper by Finn Årup Nielsen, entitled “A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs,” from the proceedings of the ESWC2011 workshop.

The current version of the lexicon is AFINN-en-165. txt and it contains over 3,300 words with a polarity score associated with each word. You can find this lexicon at the author’s official GitHub repository along with previous versions of this lexicon including AFINN-111 at https://github.com/ fnielsen/afinn/blob/master/afinn/data/.

The author has also created a nice wrapper library on top of this in Python called afinn, which we will be using for our analysis needs

```
from afinn import Afinn

afn = Afinn(emoticons=True)
```

Predict sentiment for sample reviews

```
for review, sentiment in zip(test_reviews[sample_review_ids], test_sentiments[sample_review_ids]):
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    print('Predicted Sentiment polarity:', afn.score(review))
    print('-'*60)
```

```
REVIEW: @solanatiger_slt @CoinMarketCap @coingecko @solana @RaydiumProtocol This is a project that will be very useful in the future because it is supp
@Shahin61918436
@Hawrekurdd
@hooman_k99
@Shilankurd1
Actual Sentiment: positive
Predicted Sentiment polarity: 11.0
------------------------------------------------------------
REVIEW: @solana Sol the best there is!
Actual Sentiment: positive
Predicted Sentiment polarity: 3.0
------------------------------------------------------------
REVIEW: @pushpendrakum With that we are going to see more downtime on these networks as well.

Last year @solana was down for 12 hrs.

Long way to go ✌️
Actual Sentiment: positive
Predicted Sentiment polarity: 0.0
------------------------------------------------------------
REVIEW: @TheBigHeartsNFT @ethereum @solana love your project, could you dm me please?
Actual Sentiment: positive
Predicted Sentiment polarity: 4.0
------------------------------------------------------------
REVIEW: @OrionDepp @solana @Orion__Research @Orion__Trading @KhadijaTahsin @sadman788 @Sabbir1478
Good project and strong project
bsc:0x7a4D7667B54c2f99db9aEf78a892840Ef042178A
Actual Sentiment: positive
Predicted Sentiment polarity: 5.0
------------------------------------------------------------
```

Predict sentiment for test dataset

```
sentiment_polarity = [afn.score(review) for review in test_reviews]
# adapted codo to get positive score above 0, instead of score >= 1.0
predicted_sentiments = ['positive' if score > 0 else 'negative' for score in sentiment_polarity]
```

Evaluate model performance

```
labels = ['negative', 'positive']
print(classification_report(test_sentiments, predicted_sentiments))
pd.DataFrame(confusion_matrix(test_sentiments, predicted_sentiments), index=labels, columns=labels)
```

```
              precision    recall  f1-score   support

    negative       0.56      0.85      0.68        26
    positive       0.97      0.90      0.93       166

    accuracy                           0.89       192
   macro avg       0.77      0.87      0.81       192
weighted avg       0.92      0.89      0.90       192
```

	negative	positive
negative	22	4
positive	17	149

```
import nltk
import re
import numpy as np
import contractions

stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z0-9\s]', '', doc, re.I|re.A)
    doc = doc.strip()
    doc = contractions.fix(doc)
    return doc

normalize_corpus = np.vectorize(normalize_document)

norm_corpus = normalize_corpus(test_reviews)
len(norm_corpus)
```

```
192
```

Sentiment Analysis with VADER

The VADER lexicon, developed by C.J. Hutto, is based on a rule-based sentiment analysis framework, specifically tuned to analyze sentiments in social media. VADER stands for Valence Aware Dictionary and sEntiment Reasoner. Details about this framework can be read in the original paper by Hutto, C.J., and Gilbert, E.E. (2014), entitled “VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text,” from the proceedings of the Eighth International Conference on Weblogs and Social Media (ICWSM-14). You can use the library based on NLTK's interface under the `nltk. sentiment.vader` module.

You can also download the actual lexicon or install the framework from https:// github.com/cjhutto/vaderSentiment, which also contains detailed information about VADER. This lexicon, present in the file titled vader_lexicon.txt, contains necessary sentiment scores associated with words, emoticons, and slangs (like wtf, lol, nah, and so on).

There were a total of over 9,000 lexical features from which over 7,500 curated lexical features were finally selected in the lexicon with proper validated valence scores.

Each feature was rated on a scale from `"[-4] Extremely Negative" to "[4] Extremely Positive", with allowance for "[0] Neutral (or Neither, N/A)"`.

The process of selecting lexical features was done by keeping all features that had a non-zero mean rating and whose standard deviation was less than 2.5, which was determined by the aggregate of ten independent raters. We depict a sample from the VADER lexicon as follows:

```
:( -1.9 1.13578 [-2, -3, -2, 0, -1, -1, -2, -3, -1, -4]
:) 2.0 1.18322 [2, 2, 1, 1, 1, 1, 4, 3, 4, 1]
...
terrorizing -3.0 1.0 [-3, -1, -4, -4, -4, -3, -2, -3, -2, -4]
thankful 2.7 0.78102 [4, 2, 2, 3, 2, 4, 3, 3, 2, 2]
```

Each line in the preceding lexicon sample depicts a unique term, which can either be an emoticon or a word. The first token indicates the word/emoticon, the second token indicates the mean sentiment polarity score, the third token indicates the standard deviation, and the final token indicates a list of scores given by 10 independent scorers.

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

def analyze_sentiment_vader_lexicon(review,
                                    threshold=0.1,
                                    verbose=False):
    # analyze the sentiment for review
    analyzer = SentimentIntensityAnalyzer()
    scores = analyzer.polarity_scores(review)
    # get aggregate scores and final sentiment
    agg_score = scores['compound']
    final_sentiment = 'positive' if agg_score >= threshold\
                                   else 'negative'
    if verbose:
        # display detailed sentiment statistics
        positive = str(round(scores['pos'], 2)*100)+'%'
        final = round(agg_score, 2)
        negative = str(round(scores['neg'], 2)*100)+'%'
        neutral = str(round(scores['neu'], 2)*100)+'%'
```

```
        sentiment_frame = pd.DataFrame([[final_sentiment, final, positive,
                                         negative, neutral]],
                                         columns=pd.MultiIndex(levels=[['SENTIMENT STATS:'],
                                                                       ['Predicted Sentiment', 'Polarity Score',
                                                                        'Positive', 'Negative', 'Neutral']],
                                                               codes=[[0,0,0,0,0],[0,1,2,3,4]]))
        print(sentiment_frame)

    return final_sentiment
```

Predict sentiment for sample reviews

```
for review, sentiment in zip(norm_corpus[sample_review_ids], test_sentiments[sample_review_ids]):
    print('REVIEW:', review)
    print('Actual Sentiment:', sentiment)
    pred = analyze_sentiment_vader_lexicon(review, threshold=0.4, verbose=True)
    print('-'*60)
```

```
REVIEW: solanatigerslt CoinMarketCap coingecko solana RaydiumProtocol This is a project that will be very useful in the future because it is supported
Shahin61918436
Hawrekurdd
hoomank99
Shilankurd1
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Polarity Score Positive Negative Neutral
0            positive           0.92    23.0%     0.0%   77.0%
------------------------------------------------------------
REVIEW: solana Sol the best there is
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Polarity Score Positive Negative Neutral
0            positive           0.64    46.0%     0.0%   54.0%
------------------------------------------------------------
REVIEW: pushpendrakum With that we are going to see more downtime on these networks as well

Last year solana was down for 12 hrs

Long way to go
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Polarity Score            Positive Negative Neutral
0            negative           0.27  7.000000000000001%     0.0%   93.0%
------------------------------------------------------------
REVIEW: TheBigHeartsNFT ethereum solana love your project could you dm me please
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Polarity Score Positive Negative             Neutral
```

```
0           positive          0.76    42.0%     0.0%  57.99999999999999%
------------------------------------------------------------
REVIEW: OrionDepp solana OrionResearch OrionTrading KhadijaTahsin sadman788 Sabbir1478
Good project and strong project
bsc0x7a4D7667B54c2f99db9aEf78a892840Ef042178A
Actual Sentiment: positive
     SENTIMENT STATS:
  Predicted Sentiment Polarity Score Positive Negative Neutral
0           positive          0.74    36.0%     0.0%   64.0%
------------------------------------------------------------
```

Predict sentiment for test dataset

```
predicted_sentiments = [analyze_sentiment_vader_lexicon(review, threshold=0.4, verbose=False) for review in test_reviews]
```

Evaluate model performance

```
labels = ['negative', 'positive']
print(classification_report(test_sentiments, predicted_sentiments))
pd.DataFrame(confusion_matrix(test_sentiments, predicted_sentiments), index=labels, columns=labels)
```

```
              precision    recall  f1-score   support

    negative       0.43      0.81      0.56        26
    positive       0.97      0.83      0.89       166

    accuracy                           0.83       192
   macro avg       0.70      0.82      0.73       192
weighted avg       0.89      0.83      0.85       192
```

	negative	positive
negative	21	5
positive	28	138

✓ 0s completed at 12:34 PM