How to build: VADER sentiment analysis

For any business, great services and products are always the prime focus. Understanding whether you are delivering on that is why companies put so much emphasis on understanding their customers. They want to know what they truly thought of the product or service, and will use surveys, social media, and customer reviews to glean any insight they can.

But, until recently, parsing, coding, and analyzing this feedback was a laborious task. Given the vast amount of feedback a company might see, analyzing reams of text wasn't easy. But, with the advancement of Natural Language Processing (NLP), it has become easier and much faster to analyze text and quickly extract any insight it offers.

For any brand, customers can have three tones: positive, negative, and neutral. **Sentiment analysis** is an NLP technique used to determine and

extract the sentiments and opinions expressed in the text data. The primary goal of sentiment analysis is to classify text into negative, neutral, and positive categories. These categories can vary based on the use case and the nature of the training data. Sentiment analysis is not limited to understanding the customer's reactions towards a product or service, but it can be used for other scenarios as follows:

- **Social media monitoring:** Track sentiments of people across various social platforms such as Facebook, Twitter, LinkedIn, etc., to understand their opinions on various events, products, or social issues.
- **Financial Market Analysis:** You can analyze news articles, social media content, and financial reports to understand investors' sentiment and predict the market trend.
- **Political Analysis:** Monitor the sentiment of public statements toward political figures, parties, and policies to understand voter opinions and anticipate political outcomes.
- **Travel and Hospitality:** This one is quite popular these days; you can analyze customer sentiment towards hotel stays, flights, and other travel services to make an informed decision.

These are just a few use cases where sentiment analysis makes its way. Various methods exist to perform sentiment analysis on a given text (we will see in the upcoming section). Still, one method that is quite popular for social media text sentiment analysis is VADER (Valence Aware Dictionary and sEntiment Reasoner).

In this article, you will learn about sentiment analysis and how to implement your sentiment classifier with the help of Python, VADER, and Hex.

Understanding Sentiment Analysis

The type of sentiment analysis can vary based on the use cases, as some business problems may need more fine-grained sentiment categories or sentiment scores. Typically, there are three types of sentiment analysis:

- **Binary Class Sentiment Analysis:** Binary class sentiment analysis, also known as two-class sentiment analysis, involves categorizing a piece of text's sentiments into negative and positive. This type of sentiment analysis aims to identify if the expressed sentiment is favorable or unfavorable. The best example of binary sentiment analysis is product reviews, where the distinction in whether customers liked the product is sufficient.
- **Multiclass Sentiment Analysis:** Multiclass sentiment classification involves categorizing the sentiment of a piece of text into more than two classes. These classes can include negative, neutral, positive, and other use case-based courses, such as very negative or positive. Apart from this, some use cases may also classify sentiment on a scale, for example, 1-5, with one being very bad and five being very good. One example of multiclass sentiment analysis is classifying text sentiment from social media.
- **Aspect-Based Sentiment Analysis:** This type of sentiment analysis focuses on identifying and analyzing the sentiment of a given piece of text concerning a specific aspect or feature. It usually has two components: identifying the aspects or entities within the text that users are expressing opinions about. Second, where the actual sentiment analysis takes place. Product and service reviews are an example of aspect-based sentiment analysis, where users may express different sentiments about various features or aspects.

Approaches of Sentiment Analysis

Now that you know about the categories of sentiment analysis, one question that might come to your mind is which type of algorithm is suitable for sentiment analysis. To answer this, let's have a look at broad categories of algorithms for sentiment analysis:

- **Rule-Based Approach:** Rule-based approaches rely on a predefined set of rules, patterns, and dictionaries to identify the sentiment of the text. These rules typically include the list of positive, neutral, and negative keywords, grammatical structures, and linguistic patterns. Rule-based methods are intuitive and interpretable as they rely only on predefined rules and not on any statistical or machine learning-based technique to analyze the text. One important thing to remember about rule-based approaches is that these methods require good domain knowledge for the use case you are working on. This is because sometimes a few keywords might not be part of a predefined rule set (which happens mostly in legal use cases), so domain experts are given the flexibility to add those keywords to the rule set, increasing the overall transparency of the system. Also, these methods can work with labeled and unlabeled data, making them super flexible. Some of the popular rule-based techniques are VADER and Textblob.
- **Machine learning Approach:**
  Another approach for sentiment analysis is to train the machine learning models on the labeled data to learn the patterns and relationships in textual features. These patterns are then used to classify the sentiments of a given piece of text. Some popular traditional machine learning methods for sentiment analysis are Support Vector Machines (SVM), Naive Bayes, or Logistic Regression.

Although these models can classify the text into one of the given categories, they fail to capture a text's contextual understanding. This is why deep learning models like RNN, LSTM, and GRU became famous for text processing. These models have the short memory to capture the short-term context and dependencies. However, these models also struggle to capture the long-term dependencies and context, resulting in the rise of one new category of deep learning models called Transformers. These models, including sentiment analysis, are currently state-of-the-art for any NLP use case.

- **Hybrid Approach:** Sometimes, the use cases may require you to combine the rule-based and machine-learning approaches for sentiment analysis; this results in a new kind of model called Hybrid models for sentiment analysis. These models require domain expertise and the knowledge of machine learning and statistical approaches. This can improve the overall adaptability and accuracy of the sentiment analysis system.

Common NLP Concepts for Sentiment Analysis

Now that you know about sentiment analysis and various methods, it is time to discuss some of the most popular concepts used in almost all NLP use cases, including sentiment analysis.

Tokenization

Tokenization is the process of breaking down a larger piece of text into smaller components such as words, phrases, and sub-words. These smaller units of text are called tokens and serve as a building block for further analysis. This is the most crucial stage for rule-based

approaches, as these approaches work on tokens (words) for sentiment analysis.

Stop Words

Stop words are common vocabulary words such as "a," "an," "and," "the," etc. These words do not contribute to any NLP tasks. This is why we try to remove them from the entire text corpus while working on tasks like sentiment analysis. Removing these stop words can reduce noise and improve the efficiency of sentiment analysis by focusing on more meaningful terms.

TF–IDF

TF-IDF stands for term frequency-inverse document frequency, and it evaluates the importance of any word in a document relative to a collection of documents. It first calculates the frequency of a term in a document (TF) and then calculates its rarity across the documents (IDF). The terms that appear regularly are given less weight, while the terms appearing rarely are given more weight.

Stemming and Lemmatization

In English vocabulary, using different forms for a word is quite common. For example, "do," "did," and "done" represent other information, but for a machine learning model such as sentiment analysis, this does not matter. This is why we aim to normalize words and reduce inflections; stemming and lemmatization help us in it. Stemming involves reducing words to their root or base form, while lemmatization involves transforming words to their base or dictionary form.

Word Embeddings

Machine learning models can not directly work on the text data as they only understand the digits. This is why you need to convert the text into numbers, and you can do so with the help of vector representation. Dense vector representations of words in a continuous vector space are called word embeddings. Methods like Word2Vec, GloVe, and FastText are used to generate word embeddings.

Now, you might be wondering since these are a lot of concepts, and implementing them before working on the sentiment classifier might take a lot of time. Don't worry; languages like Python and R already have a lot of modules (packages) that have an implementation of all these concepts. One such library that is widely used in Python and has an implementation of all these concepts is called the Natural Language Toolkit (NLTK). You can use this library to perform any NLP task.

Understanding VADER

As you now know, there are a lot of approaches to implementing sentiment analysis; let me bring your focus to one of the simplest but highly used rule-based algorithms called VADER (of course, not to confuse it with the Sith Lord from Star Wars). VADER is a rule-based sentiment analysis tool highly optimized for social media text and uses a dictionary called **Lexicon**, which is designed explicitly for sentiment analysis. This lexicon comprises various words and phrases with their corresponding sentiment ratings.

In identifying the sentiment of a given piece of text, VADER first breaks down the text into individual words. Then, it assigns a score to each word to identify if it is positive or negative. Based on these set scores,

VADER finally calculates the overall sentiment score of the text. One crucial thing about VADER is that it also considers the intensity of the sentiment, which is usually altered by capitalization and punctuation. It also searches for modifiers that could change the meaning of neighbor words. This way, it portrays the emotion more effectively.

The scores returned by VADER range from -1 to 1, -1 being very negative and one being very positive. The main advantage of the VADER tool is that it does not require you to train any ML model, making it a lot more intuitive and transparent for both technical and non-technical audiences. Also, as it uses a dictionary of terms to compute the sentiment, it is much faster than any other rule-based or ML-based sentiment analysis algorithm. As each tool has its limitations, VADER also has some. It usually has a hard time understanding Sarcasm and Irony. Also, negations are not easily identified by VADER. For example, “not bad" is a positive word but can be classified as unfavorable based on the lexicon. Finally, it has limited support for multilingual analysis.

Note: *The NLTK library in Python provides the implementation of VADER we’ll use.*

Sentiment Analysis with Python, VADER and Hex

In this section, you will see a practical implementation of sentiment analysis on Twitter data with the help of Python, VADER, and Hex. Hex offers an adaptable and robust workspace that enables users to use SQL and Python to analyze sentiment quickly and parse unstructured text input efficiently. Being a polyglot platform, Hex allows you to query data from various data sources with the help of multiple programming languages within the same environment. By leveraging VADER's

capabilities within Hex, you can uncover and understand social media users' emotional tones and opinions, make data-driven decisions, and enhance your understanding of customer sentiment. Hex also allows you to create different customer dashboards with no code visualizations and helps you to deploy any of your applications with just a simple click.

We will use Python 3.11, SQL languages, and Hex as a development environment for development. The data we will use is stored in the Snowflake warehouse.

Install and Load Dependencies

To begin with, you need to install the necessary dependencies, which you can do with the help of Python Package Manager (PIP) as follows:

```
$ pip install nltk
$ pip install numpy
$ pip install scikit-learn
```

Once the libraries are installed, you can import them in the hex environment with the following lines of code:

```
import nltk
nltk.download("vader_lexicon")
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.metrics import accuracy_score
import numpy as np
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/hexuser/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In the above code, we have first imported the `nltk` library that has an implementation for some widely used NLP algorithms. Then using the `d`

`ownload()` method, we installed the necessary files and definitions for the VADER model and later imported the `SentimentIntensityAnalyzer` model from VADER. Then we loaded the `sklearn` module that provides implementation to various performance metrics like accuracy, precision, and recall. Finally, we have imported the numpy library which is helpful for faster numeric computation of our data.

Note: *Similar to VADER other NLP models and methods can be downloaded using the* `downLoad()` *method.*

Load and Use Model

After loading the necessary dependencies, we need to create an object of the `SentimentIntensityAnalyzer` model that will be used for predicting the sentiments of text.

```
# Create an instance of SentimentIntensityAnalyzer
model = SentimentIntensityAnalyzer()
```

The result from the VADER model is a set of four different numbers that belong to classes such as negative, neutral, positive, and compound. The break up of the score for these classes is as follows:

- **Negative:** It depicts the negative sentiment and its value ranges from 0 to 1.
- **Neutral:** It depicts the neutral sentiment and its value ranges from 0 to 1.
- **Positive:** It depicts the positive sentiment and its value ranges from 0 to 1.

- **Compound:** It provides us the overall sentiment score and its value ranges from -1 to 1, -1 being the strong negative sentiment and 1 being the strong positive sentiment.

Now, let us implement a Python method where we will use the `model` object to predict the sentiment of a given piece of text using the `polarity_scores()` method. Then we will get the compound score returned by the model. We will treat a compound score less than or equal to `-0.05` as negative sentiment, a score greater than or equal to `0.05` as positive sentiment, and the score in between as neutral sentiment.

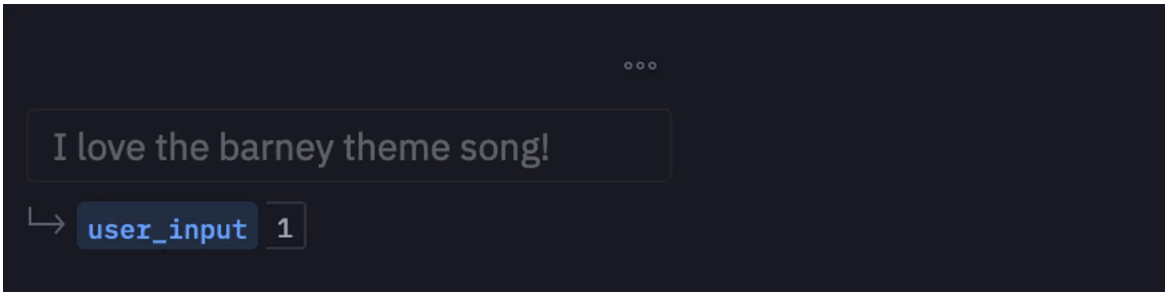
```
def extract_score(text):
        score = model.polarity_scores(text)
        compound = score['compound']

        sentiment = 'neutral'
        if(compound >= 0.05):
        sentiment = "positive"

        elif(compound <= -0.05):
        sentiment = "negative"

        return sentiment
```

After this, you can use the feature of Hex which allows you to take variable values with the help of interactive components. We will read the value of the `user_input` variable as follows:



Finally, to test the above method, you can call it on the example input test as follows:

```
# What about this sentence with repeated exclamations and capi
talization?
# print(model.polarity_scores(user_input))
extract_score(user_input)
```

```
'positive'
```

As you can see in the above image, the predicted sentiment is positive for the given text which seems accurate.

Load and Preprocess Data

Now that we have our model ready, it is time to load the actual tweet data from the data warehouse with the help of SQL cells in the Hex environment as follows:

```
select * from "tweets.csv";
```

	123 column0	Datetime	123 Tweet Id	Text
0	0	2022-09-30T23:29:15	1575991191170342912	@Logitech @apple @Google @Microsoft @Dell
1	1	2022-09-30T21:46:35	1575965354425131008	@MK_habit_addict @official_stier @MortalKor
2	2	2022-09-30T21:18:02	1575958171423752203	As @CRN celebrates its 40th anniversary, Bob
3	3	2022-09-30T20:05:24	1575939891485032450	@dell your customer service is horrible especi
4	4	2022-09-30T20:03:17	1575939359160750080	@zacokalo @Dell @DellCares @Dell give the m
5	5	2022-09-30T19:49:40	1575935931969785856	The screenshot is acting up from the website c

Note: *In the Hex environment, the data is always loaded in the Pandas DataFrame format and it can easily be read with SQL and Python within the same environment. Also, the data is processed in cascade format i.e. the output from previous cells is easily available for the next cells.*

Since our dataset is big enough, as part of data processing we can drop the null values from it using the `dropna()` method from pandas.

```
data.dropna(inplace = True)
data.isnull().sum()
```

```
column0           0
Datetime          0
Tweet Id          0
Text              0
Username          0
sentiment         0
sentiment_score   0
emotion           0
emotion_score     0
dtype: int64
```

The `isnull().sum()` method provides the sum of null values in each column in the dataset.

Since there are a lot of columns in the dataset and we don't need all of them for sentiment classification, we will filter out the dataframe as follows:

```
data = data[["Datetime", "Tweet Id", "Text", "sentiment", "Use
rname"]].head(15000)
data
```

	Datetime	Tweet Id	Text	sentiment	Username
0	2022-09-30 23:29:15	1575991191170342912	@Logitech @apple @Google @Microsoft @Dell @Len...	neutral	ManjuSreedaran
1	2022-09-30 21:46:35	1575965354425131008	@MK_habit_addict @official_stier @MortalKombat...	neutral	MiKeMcDnet
2	2022-09-30 21:18:02	1575958171423752203	As @CRN celebrates its 40th anniversary, Bob F...	positive	jfollett
3	2022-09-30 20:05:24	1575939891485032450	@dell your customer service is horrible especi...	negative	daveccarr
4	2022-09-30 20:03:17	1575939359160750080	@zacokalo @Dell @DellCares @Dell give the man ...	neutral	heycamella
...	...	...	...	...	...
14995	2022-04-28 17:58:44	1519737895891918850	Okay @dell your précision are great laptops bu...	negative	H_Miser
14996	2022-04-28 17:37:32	1519732560842182657	Five Key Benefits of an Intelligent Storage Ap...	positive	MelilloConsult
14997	2022-04-28 17:33:10	1519731463641513984	Interested in @Dell's Self-Dispatch training? ...	neutral	DellClients

As you can see in the above code, only the `Datetime`, `Tweet Id`, `Text`, `sentiment`, and `Username` columns are filtered out from the original data.

Implement VADER Sentiment Analysis

To apply the VADER model in the tweet data, you simply need to use the `apply()` method from pandas and need to pass the `extract_score()` method to it for the `Text` column in our dataframe.

```
data["predicted_sentiment"] = data["Text"].apply(extract_score
) # extract score was defined in a cell above (hidden in the p
ublished app)
```

As you can see in the above code, we will be storing the sentiment results for each text in the `predicted_sentiment` column in our dataframe.

To take a look at a few random rows of the data, you can use the `sample()` method from pandas.

```
data.sample(5)
```

	Datetime	Tweet Id	Text	sentiment	Username	predicted_sentiment
1016	2022-09-18 13:21:30	1571489592183758848	@tkcaddie @warriors @NFL @robmillertime @Dell ...	neutral	NickGoblirsch	neutral
5214	2022-08-01 14:25:14	1554111014957977601	Delphix and @Dell launch new appliances for ra...	positive	MickTorok	positive
3064	2022-08-25 23:30:45	1562945605672267776	@jasminecrowe @Dell We're so glad you're here!	positive	ATLTechVillage	positive
348	2022-09-27 11:47:30	1574727427859050496	@itweetnothing__ @ASUS @Lenovo @HP @Dell @msig...	neutral	quiverloaded	neutral
500	2022-09-24 15:40:16	1573698843757068288	Dear \n@Dell I'm using your Dell latitude 5...	negative	SPratapyS	positive

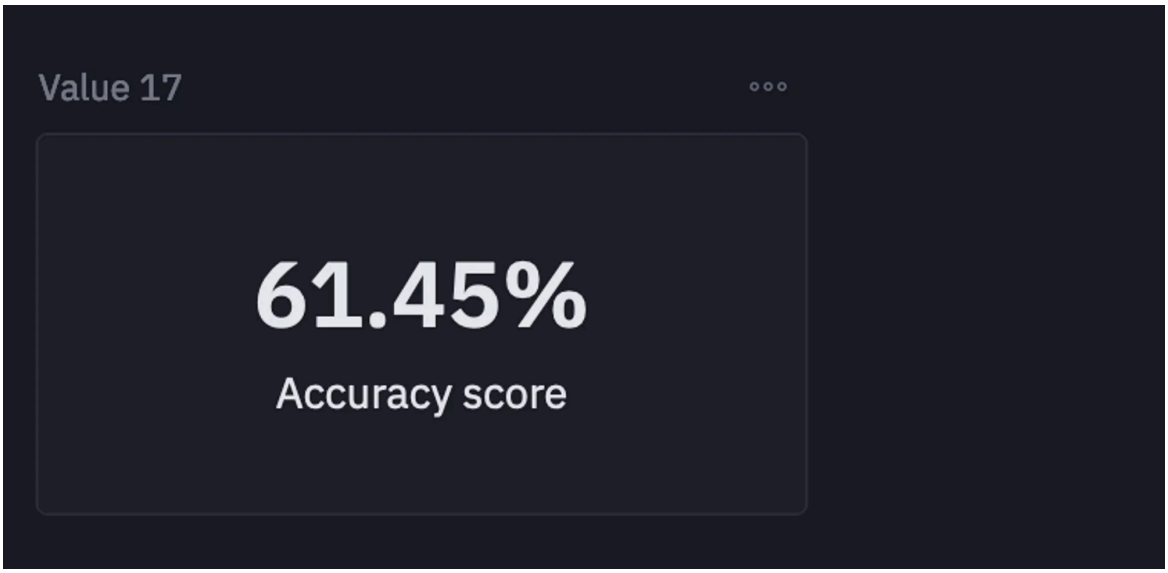
The above image shows the dataframe that contains the `predicted_sentiment` column.

Model Evaluation

After applying the VADER model, we have obtained the compound score for each tweet that represents the sentiment as well as its intensity. It is time to evaluate the performance of the model. For sentiment analysis, various performance metrics can be used, which include accuracy, precision, recall, F1-score, confusion matrix, ROC curve and AUC, and human evaluation.

While working on rule-based approaches, accuracy metrics might work well enough. But if you are using ML models for sentiment analysis, accuracy might not be a good choice for an imbalanced dataset (where the number of positive classes is more as compared to negative classes or vice versa). In that case, metrics like precision, recall, and F1-Score make much more sense. For this article, we will use the accuracy metrics and will calculate the accuracy by comparing the ground truth sentiments and the predicted sentiments, this will give us the percentage of correctly predicted sentiments.

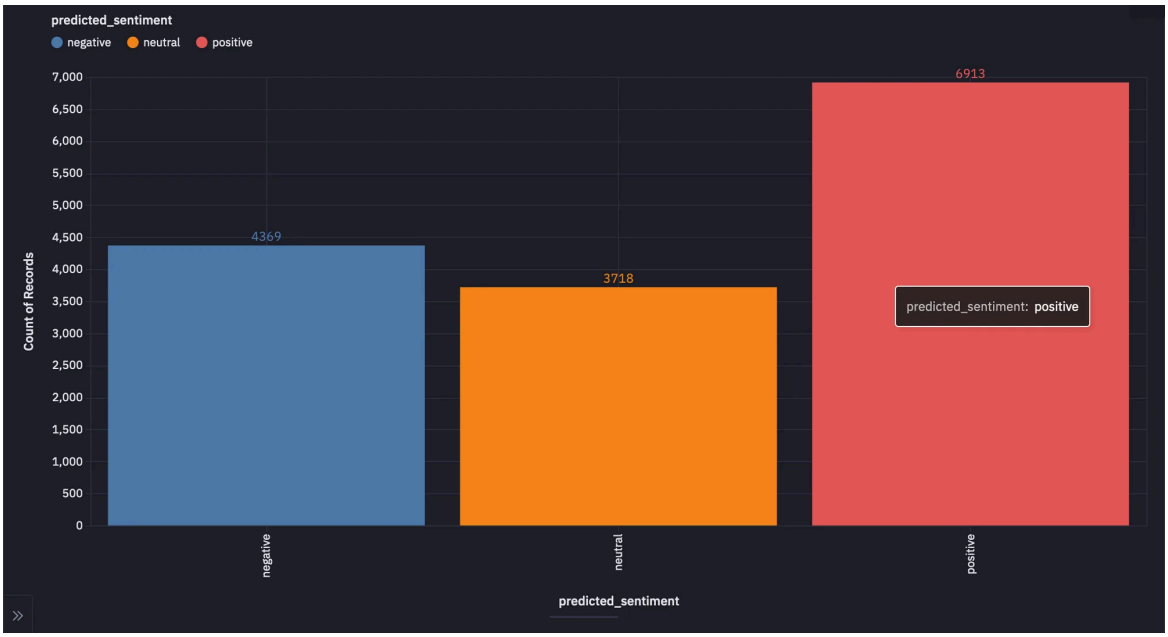
```
score = accuracy_score(data['sentiment'], data['predicted_sentiment'])
```



The accuracy that we got from the VADER model is `60%` which is neither too bad nor too good. This typically means that our rule-based model was able to identify 60% of the sentiments accurately. This score can surely be improved with better text processing (using the methods mentioned above) or using the model-based approach for sentiment classification.

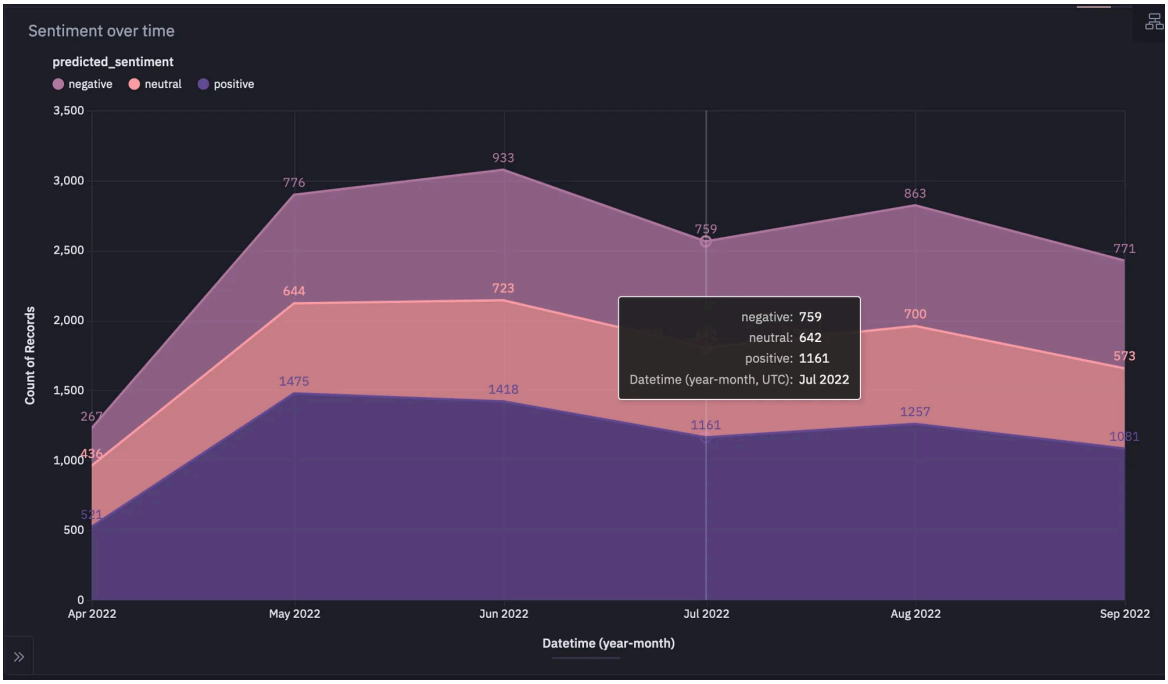
Visualizing Results

Now that you have your predictions, you can enhance the visibility of results with the help of a few visualizations. You can use the no-code functionality of Hex and can create multiple types of graphs for your data. For example, a bar graph for the `predicted_sentiment` column may look something like this:



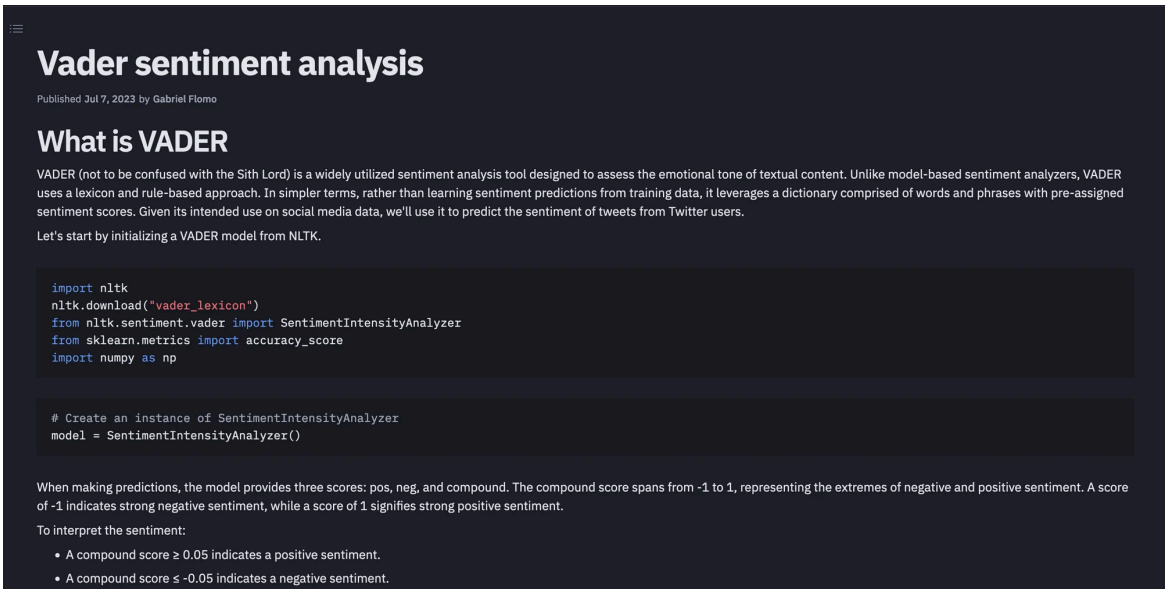
On the right, we see that the positive sentiment is the most present category with negative sentiment being in second.

You can also create a line plot explaining different types of sentiments over time which will look something like this:



This visual shows how sentiment changes over time. Based on the graph, it looks like as one sentiment category increases or decreases, all other categories also change at the same time.

Once you are done with writing all the code, you can head over to the `App` section in the Hex environment to get yourself an appealing dashboard. You can rearrange the components of the dashboard according to your needs and once done, you can click on the publish button to deploy your app with the help of Hex.



This is it, you have now created a sentiment classifier with a few simple steps. VADER is a pretty handy tool for analyzing the sentiment of social media text as these platforms allow the use of casual and informal language. We were able to achieve around 61% accuracy which is not too good but it does the job. Since you have already seen that VADER has its limitations such as pre-defined rules and word lists, it might miss out on some specific or ever-changing language nuances. The use of VADER totally depends on the use case for example in the case of the finance and legal department, it is not a wise choice to use VADER given the new terms that can occur at any time in the text. Finally, while VADER shows promise, there's always room to explore advanced techniques for better sentiment analysis accuracy.

The Evolution of Customer Understanding Through VADER

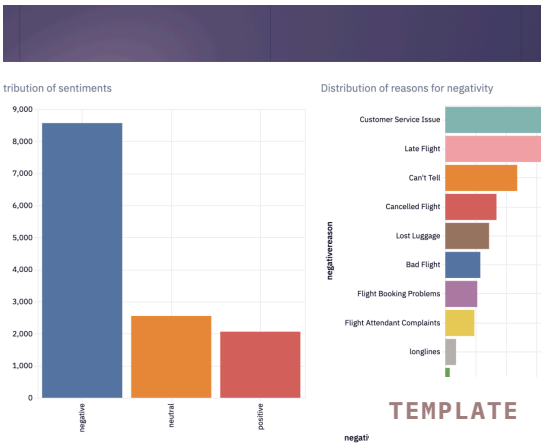
VADER allows analysts to efficiently parse and interpret large volumes of customer feedback in real-time. By quantifying the emotional tone behind words, this tool transforms subjective opinions into actionable data. This capability enables businesses to quickly identify trends in customer sentiment, gauge overall satisfaction, and tailor their strategies to meet the evolving needs and preferences of their target audience. With VADER, companies can navigate the vast ocean of customer opinions with precision, making informed decisions that resonate with their clientele and foster stronger, more meaningful connections.

Related tags

TABLE PYTHON CHART

See what else Hex can do

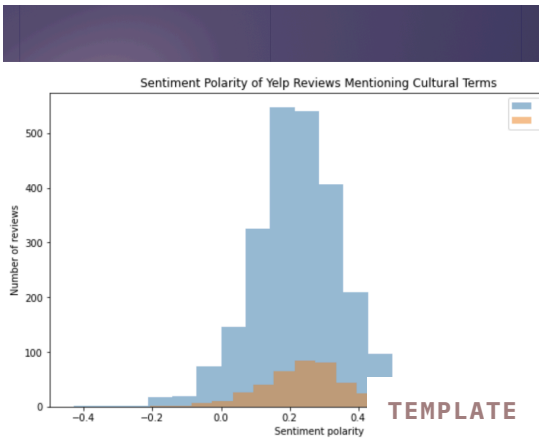
Discover how other data scientists and analysts use Hex for everything from dashboards to deep dives.



Social Media Sentiment Analysis

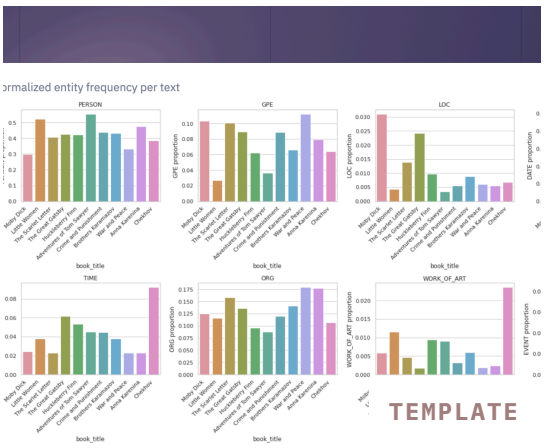
Izzy Miller

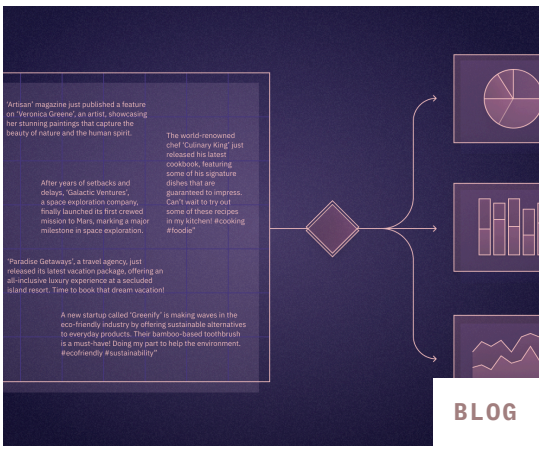
Parse out insights and trends from the sea of tweets, posts, and threads



Sentiment Analysis

Understand the meaning behind your unstructured text data, from reviews to support requests





Named Entity Recognition

Uncover the hidden entities in your text data, from classic literature to customer feedback.

A Comprehensive Guide to Natural Language Processing Algorithms

Andrew Tate · May 25, 2023

Learn about the simpler text processing cousins of LLMs like GPT-4

Natural Language Processing

Multiple contributors

Empower your AI and ML applications to interpret, recognize, and generate human language with powerful Natural Language Processing (NLP) algorithms.