Avinash Navlani
December 14th, 2019

PYTHON

Text Analytics for Beginners using NLTK

Learn How to analyze text using NLTK. Analyze people's sentiments and classify movie reviews.

In today's area of internet and online services, data is generating at incredible speed and amount. Generally, Data analyst, engineer, and scientists are handling relational or tabular data. These tabular data columns have either numerical or categorical data. Generated data has a variety of structures such as text, image, audio, and video. Online activities such as articles, website text, blog posts, social media posts are generating unstructured textual data. Corporate and business need to analyze textual data to understand customer activities, opinion, and feedback to successfully derive their business. To compete with big textual data, text analytics is evolving at a faster rate than ever before.

Text Analytics has lots of applications in today's online world. By analyzing tweets on Twitter, we can find trending news and peoples reaction on a particular event. Amazon can understand user feedback or review on the specific product. BookMyShow can discover people's opinion about the movie. Youtube can also analyze and understand peoples viewpoints on a video.

In this tutorial, you are going to cover the following topics:

- Text Analytics and NLP
- Compare Text Analytics, NLP and Text Mining
  - Text Analysis Operations using NLTK

    - Stopwords
    - Lexicon Normalization such as Stemming and Lemmatization
    - POS Tagging
- Sentiment Analysis
- Text Classification
- Performing Sentiment Analysis using Text Classification

Text Analytics and NLP

Text communication is one of the most popular forms of day to day conversion. We chat, message, tweet, share status, email, write blogs, share opinion and feedback in our daily routine. All of these activities are generating text in a significant amount, which is unstructured in nature. I this area of the online marketplace and social media, It is essential to analyze vast quantities of data, to understand peoples opinion.

NLP enables the computer to interact with humans in a natural manner. It helps the computer to understand the human language and derive meaning from it. NLP is applicable in several problematic from speech recognition, language translation, classifying documents to information extraction. Analyzing movie review is one of the classic examples to demonstrate a simple NLP Bag-of-words model, on movie reviews.

Compare Text Analytics, NLP and Text Mining

Text mining also referred to as text analytics. Text mining is a process of exploring sizeable textual data and find patterns. Text Mining process the text itself, while NLP process with the underlying metadata. Finding frequency counts of words, length of the sentence, presence/absence of specific words is known as text mining. Natural language processing is one of the components of text mining. NLP helps identified sentiment, finding entities in the sentence, and category of blog/article. Text mining is preprocessed data for text analytics. In Text Analytics, statistical and machine learning algorithm used to classify information.

NLTK is a powerful Python package that provides a set of diverse natural languages algorithms. It is free, opensource, easy to use, large community, and well documented. NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition. NLTK helps the computer to analysis, preprocess, and understand the written text.

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /home/northout/anaconda2/lib/python2.7/site-packages
Requirement already satisfied: six in /home/northout/anaconda2/lib/python2.7/site-packages (fr
←[33mYou are using pip version 9.0.1, however version 10.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.←[0m
```

```
#Loading NLTK
import nltk
```

Tokenization

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

Sentence Tokenization

Sentence tokenizer breaks text paragraph into sentences.

```
from nltk.tokenize import sent_tokenize
text="""Hello Mr. Smith, how are you doing today? The weather is great, and city is awesome.
The sky is pinkish-blue. You shouldn't eat cardboard"""
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['Hello Mr. Smith, how are you doing today?', 'The weather is great, and city is awesome.', 'T
```

Here, the given text is tokenized into sentences.

Word Tokenization

Word tokenizer breaks text paragraph into words.

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', '
```

Frequency Distribution

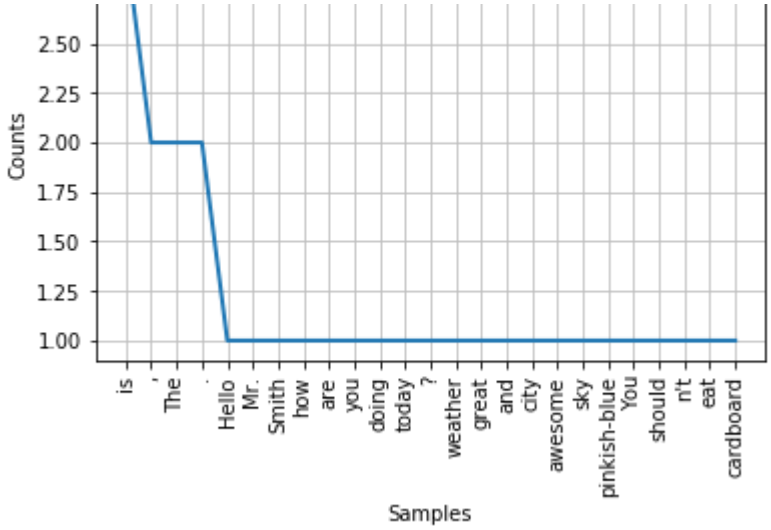
```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
```

```
<FreqDist with 25 samples and 30 outcomes>
```

```
fdist.most_common(2)
```

```
[('is', 3), (',', 2)]
```

```
# Frequency Distribution Plot
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



Stopwords

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{'their', 'then', 'not', 'ma', 'here', 'other', 'won', 'up', 'weren', 'being', 'we', 'those',
```

Removing Stopwords

```
filtered_sent=[]
for w in tokenized_sent:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentence:",tokenized_sent)
print("Filterd Sentence:",filtered_sent)
```

```
Filtered Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?']
```

Lexicon Normalization

Lexicon normalization considers another type of noise in the text. For example, connection, connected, connecting word reduce to a common word "connect". It reduces derivationally related forms of a word to a common root word.

Stemming

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".

```
# Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))

print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)
```

```
Filtered Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?']
Stemmed Sentence: ['hello', 'mr.', 'smith', ',', 'today', '?']
```

Lemmatization

Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without

better --> good

```
#Lexicon Normalization
#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()

word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))


Lemmatized Word: fly
Stemmed Word: fli
```

POS Tagging

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

```
sent = "Albert Einstein was born in Ulm, Germany in 1879."


tokens=nltk.word_tokenize(sent)
print(tokens)


['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879', '.']


nltk.pos_tag(tokens)
```

```
('Einstein', 'NNP'),
('was', 'VBD'),
('born', 'VBN'),
('in', 'IN'),
('Ulm', 'NNP'),
(',', ','),
('Germany', 'NNP'),
('in', 'IN'),
('1879', 'CD'),
('.', '.')]
```

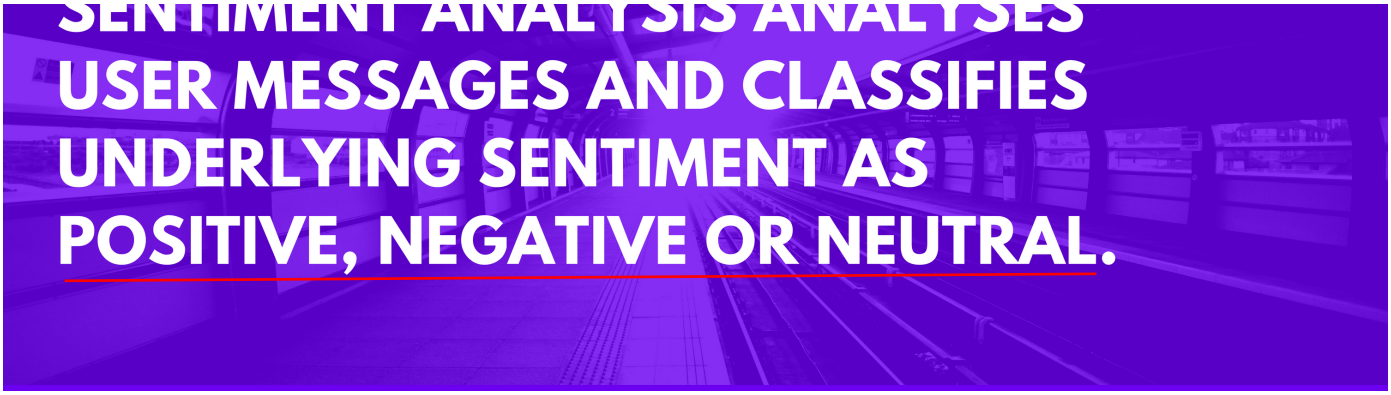
POS tagged: Albert/NNP Einstein/NNP was/VBD born/VBN in/IN Ulm/NNP ,/, Germany/NNP in/IN 1879/CD ./.

Sentiment Analysis

Nowadays companies want to understand, what went wrong with their latest products? What users and the general public think about the latest feature? You can quantify such information with reasonable accuracy using sentiment analysis.

Quantifying users content, idea, belief, and opinion is known as sentiment analysis. User's online post, blogs, tweets, feedback of product helps business people to the target audience and innovate in products and services. Sentiment analysis helps in understanding people in a better and more accurate way. It is not only limited to marketing, but it can also be utilized in politics, research, and security.

Human communication just not limited to words, it is more than words. Sentiments are combination words, tone, and writing style. As a data analyst, It is more important to understand our sentiments, what it really means?



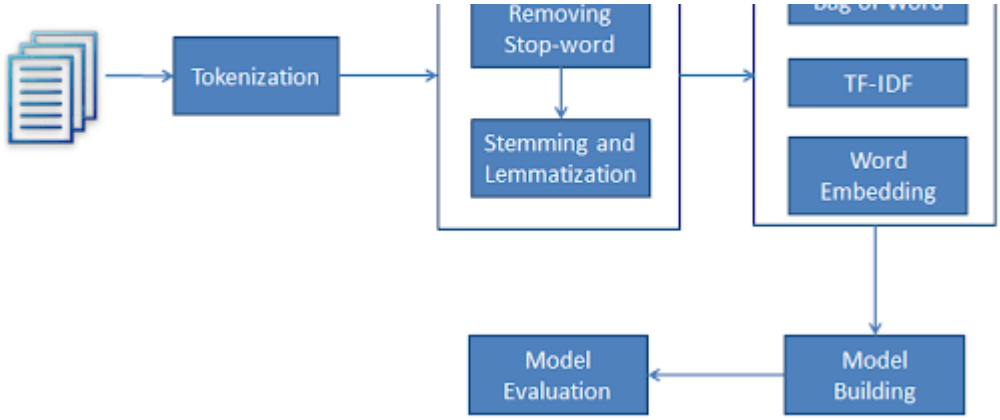
There are mainly two approaches for performing sentiment analysis.

- Lexicon-based: count number of positive and negative words in given text and the larger count will be the sentiment of text.
- Machine learning based approach: Develop a classification model, which is trained using the pre-labeled dataset of positive, negative, and neutral.

In this Tutorial, you will use the second approach(Machine learning based approach). This is how you learn sentiment and text classification with a single example.

Text Classification

Text classification is one of the important tasks of text mining. It is a supervised approach. Identifying category or class of given text such as a blog, book, web page, news articles, and tweets. It has various application in today's computer world such as spam detection, task categorization in CRM services, categorizing products on E-retailer websites, classifying the content of websites for a search engine, sentiments of customer feedback, etc. In the next section, you will learn how you can do text classification in python.



Performing Sentiment Analysis using Text Classification

```
# Import pandas
import pandas as pd
```

Loading Data

Till now, you have learned data pre-processing using NLTK. Now, you will learn Text Classification. You will perform Multi-Nomial Naive Bayes Classification using scikit-learn.

In the model the building part, you can use the "Sentiment Analysis of Movie, Reviews" dataset available on Kaggle. The dataset is a tab-separated file. Dataset has four columns PhraseId, SentenceId, Phrase, and Sentiment.

This data has 5 sentiment labels:

0 - negative 1 - somewhat negative 2 - neutral 3 - somewhat positive 4 - positive

Here, you can build a model to classify the type of cultivar. The dataset is available on Kaggle. You can download it from the following link: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data

```
data=pd.read_csv('train.tsv', sep='\t')
```

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

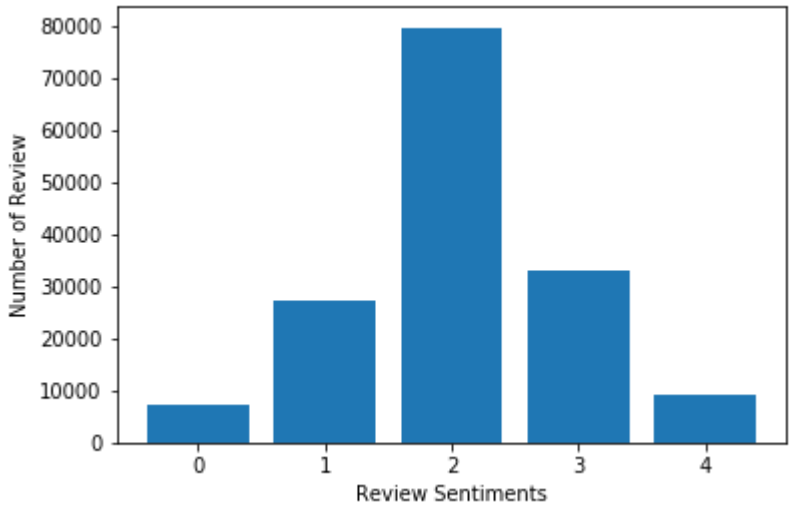
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156060 entries, 0 to 156059
Data columns (total 4 columns):
PhraseId      156060 non-null int64
SentenceId    156060 non-null int64
Phrase        156060 non-null object
Sentiment     156060 non-null int64
dtypes: int64(3), object(1)
memory usage: 4.8+ MB
```

```
data.Sentiment.value_counts()
```

```
2    79582
3    32927
1    27273
4     9206
0     7072
Name: Sentiment, dtype: int64
```

```
plt.bar(Sentiment_count.index.values, Sentiment_count['Phrase'])
plt.xlabel('Review Sentiments')
plt.ylabel('Number of Review')
plt.show()
```



Feature Generation using Bag of Words

In the Text Classification Problem, we have a set of texts and their respective labels. But we directly can't use text for our model. You need to convert these text into some numbers or vectors of numbers.

Bag-of-words model(BoW) is the simplest way of extracting features from the text. BoW converts text into the matrix of occurrence of words within a document. This model concerns about whether given words occurred or not in the document.

Example: There are three documents:

Doc 1: I love dogs. Doc 2: I hate dogs and knitting. Doc 3: Knitting is my hobby and passion.

Now, you can create a matrix of document and words by counting the occurrence of words in the given document. This matrix is known as Document-Term Matrix(DTM).

Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

This matrix is using a single word. It can be a combination of two or more words, which is called a bigram or trigram model and the general approach is called the n-gram model.

You can generate document term matrix by using scikit-learn's CountVectorizer.

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
#tokenizer to remove unwanted elements from out data like symbols and numbers
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range = (1,1),tokenizer = token
text_counts= cv.fit_transform(data['Phrase'])
```

Split train and test set

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split dataset by using function train_test_split(). You need to pass basically 3 parameters features, target, and test_set size. Additionally, you can use random_state to select records randomly.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    text_counts, data['Sentiment'], test_size=0.3, random_state=1)
```

Model Building and Evaluation

Let's build the Text Classification Model using TF-IDF.

Then, fit your model on a train set using fit() and perform prediction on the test set using predict().

```
from sklearn.naive_bayes import MultinomialNB
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))
```

```
MultinomialNB Accuracy: 0.604916912299
```

Well, you got a classification rate of 60.49% using CountVector(or BoW), which is not considered as good accuracy. We need to improve this.

Feature Generation using TF-IDF

In Term Frequency(TF), you just count the number of words occurred in each document. The main issue with this Term Frequency is that it will give more weight to longer documents. Term frequency is basically the output of the BoW model.

IDF(Inverse Document Frequency) measures the amount of information a given word provides across the document. IDF is the logarithmically scaled inverse ratio of the number of documents that contain the word and the total number of documents.

$$\text{idf(W)} = \log \frac{\#(\text{documents})}{\#(\text{documents containing word W})}$$

TF-IDF(Term Frequency-Inverse Document Frequency) normalizes the document term matrix. It is the product of TF and IDF. Word with high tf-idf in a document, it is most of the times occurred in given documents and must be absent in the other documents. So the words must be a signature word.

Doc 1	0.18	**0.48**	0.18							
Doc 2	0.18		0.18	**0.48**	0.18	0.18				
Doc 3					0.18	0.18	**0.48**	**0.95**	**0.48**	**0.48**

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
text_tf= tf.fit_transform(data['Phrase'])
```

Split train and test set (TF-IDF)

Let's split dataset by using function train_test_split(). You need to pass basically 3 parameters features, target, and test_set size. Additionally, you can use random_state to select records randomly.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    text_tf, data['Sentiment'], test_size=0.3, random_state=123)
```

Model Building and Evaluation (TF-IDF)

Let's build the Text Classification Model using TF-IDF.

First, import the MultinomialNB module and create the Multinomial Naive Bayes classifier object using MultinomialNB() function.

Then, fit your model on a train set using fit() and perform prediction on the test set using predict().

```
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))
```

```
MultinomialNB Accuracy: 0.5865209549018
```

Well, you got a classification rate of 58.65% using TF-IDF features, which is not considered as good accuracy. We need to improve the accuracy by using some other preprocessing or feature engineering. Let's suggest in comment box some approach for accuracy improvement.

Conclusion

Congratulations, you have made it to the end of this tutorial!

In this tutorial, you have learned what Text Analytics is, NLP and text mining, basics of text analytics operations using NLTK such as Tokenization, Normalization, Stemming, Lemmatization and POS tagging. What are sentiment analysis and text classification using scikit-learn?

I look forward to hearing any feedback or questions. You can ask the question by leaving a comment, and I will try my best to answer it.

If you are interested in learning more about Python, please take DataCamp's Natural Language Processing Fundamentals in Python course.

COMMENTS

Hugo Nicolau Barbosa de Gusmão
05/09/2018 09:38 PM

Amazing tutorial!

I think you forgot to say that you need to use these commands:

nltk.download (punkt)

▲ 10

Avinash Navlani
06/09/2018 02:34 PM

Thanks for the feedback and spotting that I'd missed.
Yes, we need to download stopwords and punkt.

▲ 2

MILIND PARIKH
18/10/2018 12:04 AM

Great tutorial--- just a minor correction you are missing assignment of tokenized sent...see bwlow

tokenized_sent = tokenized_word

▲ 9

Benjamin III Buensuceso
07/10/2018 05:45 PM

This was an excellent tutorial! :)

▲ 3

Saaket Varma
23/10/2018 01:19 AM

Thanks, Nice overview of the Text analysis. Helped me in the capstone project .

▲ 2

Aida Haliti
25/10/2018 12:55 AM

One recommendation is implementing ULMFiT from fast.ai for classification tasks, certainly, accuracy will improve.

▲ 3

---

olubukola smile

THANKS A BUNCH

▲ 1

Lhicel Joyce Aytona
15/11/2018 12:11 PM

Hi! If I want to use Sentiment Analysis on News Articles, how can I train my model if I will use data from a news website? Of course my data will not have any target value or the y_train because it will come from raw news articles from the web unlike the built in data of movie reviews used in this tutorial which has x_train and y_train.

Can I use the same movie reviews data used in this tutorial to train my model and then use the news articles as a testing set for prediction? Or that will not work? Can I have your suggestion please? Thank you!

▲ 6

Avinash Navlani
17/11/2018 09:02 PM

Hi Lhicel Joyce Aytona,

you can't use movie data model on news articles because both have different context.

Sentiment problem can be solved in two ways: classification and using postive-negative keyword dictionary. For classification, you need labels. In your news articale data you don't have labels. you can add labels to some 200-300 and articles manually and than create model and predict for others and test results manually. in second approach, you can use dictionary of positive negative words and count positive and negative keywords in each article. Find difference of both positive and negative keywords. If difference is postive than sentiment is postive. if difference is negative than sentiment is negative and if difference is zero than neutral sentiment.

Thanks

▲ 5

Guru Charan Pathalla
02/12/2018 12:21 AM

Hi Avinash, this is a great tutorial. I appreciate you work on this to explain everything in a very well manner. I have a question: After I predict and get the accuracy lets say 60% I want to target the negative responses and take some action like further reach out to them and probe some questions. I should only target this 60% (which it accurately predicted and which contains the positive, negative and neutral sentiments I guess). can I identify who all

▲ 2

Heshani Rupasinghe
12/01/2020 09:46 PM

Hi Avinash,

I am also having the same problem. in my data set I had only x values and then i manually put some y values i expect. then i had 400 data. Using them I tained my model and got 63% accuracy. I am planning to improve the code by training more. But after that how do i use this code to predict rest of the values in my file?

▲ 0

Guru Charan Pathalla
02/12/2018 02:57 AM

Hi Avinash, this is a great tutorial. I appreciate you work on this to explain everything in a very well manner. I have a question: After I predict and get the accuracy lets say 60% I want to target the negative responses and take some action like further reach out to them and probe some questions. I should only target this 60% (which it accurately predicted and which contains the positive, negative and neutral sentiments I guess). can I identify who all fall under 60% and who all are under negative sentiment? your response is much appreciated Sir!! :)

▲ 2

Avinash Navlani
08/12/2018 10:50 PM

Yes, you can. you need to separte data in two parts training and test and append the prediction results to test records and evaluate one by one to understand the prediction results. If you still have any question you can mail me your code i will fix it and add the required solution to it. My mail address is: avinashnvln8@gmail.com

▲ 2

Vijay Das
12/12/2018 02:58 AM

Very informative. Thanks for posting .

▲ 1

Great tutorial Avinash with nice step by step approach for Text Analytics. Have you considered using a different classifier instead of multinomial naive bayes? Perhaps, SGDClassifier may be? This is because of the obvious limitations of Naive Bayes i.e. the assumption of independent predictors.

▲ 2

Avinash Navlani
16/12/2018 09:59 PM
Yes, you can use SGDClassifier.

▲ 1

Srujana Reddy
08/01/2019 03:51 AM
Hi, Thanks for this great tutorial. Which version of Python do you recommend for the NLTK?

▲ 2

Avinash Navlani
14/01/2019 06:09 PM
Any version of Python 3.

▲ 1

Amitrajit Bose
16/01/2019 08:08 PM
Why not go for a linear support vector machine ?

▲ 2

Avinash Navlani
18/01/2019 11:03 AM
You can use linear SVM. Linear SVM is better compare to other non-linear SVM models because text classification has lots of features. Mapping the data in linear SVM with higher dimensional (or large number of features) will improve the performance.

▲ 2

Ab Sa
23/01/2019 10:19 PM
Hi, I'm asking about DTM, why we have 2 for (my) in Doc3, it is only mentioned once !

30/01/2019 09:35 AM

Good Spot, It should be 1.

▲ 1

Bipul kumar singh
25/01/2019 05:04 PM

Thank you for such a nice tutorial , sir i want to develop python program for Hindi sentiment analysis. how to stem hindi words and there is no any option for pos tagging. can you provide us tutorial on Hindi sentiment analysis.

▲ 2

Avinash Navlani
30/01/2019 09:38 AM

Thank you for your feedback. I will try to do it for hindi tweets, or blogs and publish it.

▲ 2

Mohamad Raychan
30/01/2019 03:44 PM

Awesome tutorial, many thanks sir!

▲ 2

Achyut Namdeo
31/01/2019 11:08 AM

- Very informative article on NLP

▲ 4

Achyut Namdeo
31/01/2019 11:11 AM

Thanks for suggesting sir

▲ 2

Venkateswarlu Aduru

na_filter=False)

▲ 2

Yogesh Bhardwaj
04/02/2019 01:21 PM

Thanks for such a great Tutorial.

▲ 1

Christopher Ifeanyi Eke
16/03/2019 05:40 PM

Nice tutorial. Thank you. Please I need a guide on the steps to download and install NLTK data and packages in python 3.5/ Jupyter.

▲ 2

Awais Alvi
18/03/2019 01:31 PM

Dear Mr. Avinash,

I am doing work on a Urdu grammar checker which is N Gram base .Kindly guide how can train model on N Gram. Kindly guide from where can help to create model.

▲ 2

Avinash Navlani
18/03/2019 07:43 PM

Sorry, I have no idea about Urdu language NLP library.

▲ 1

Prameela Janardanan
26/03/2019 02:23 AM

When I try to execute the above code, I am getting "**OSError**: Initializing from file failed". Not sure if I am going wrong anywhere. I am a beginner. Trying to understand how the classification works in AI. Can someone help?

▲ 2

Hi, I am able to fix it. This was Nice Tutorial.

▲ 2

saurabh ahuja
05/05/2019 11:17 AM

Such an error :(, it's confusing hence, especially for a beginner

a) tokenized_sent hasn't been defined & it is being used (Removing Stopwords)

▲ 6

akshaya kumar
15/05/2019 09:47 PM

Such a beautiful explanation.

Could you please suggest what i should use for anomaly detection in application logs. There is no labeled data, so thinking it to be un-supervised text classification probably.

Sample 2 lines of log:

2018-08-08 20:04:53.904 | INFO | [GeneratingProcessName] | APP_ID | USER_ID | GUID | PROCESS_DESC

2018-08-08 20:04:54.887 | ERROR | [GeneratingProcessName] | ERR_CODE | USER_ID | GUID | ERROR_DESC

ERROR_DESC and PROCESS_DESC are the fields with paragraph texts and that's the information we want to make use of. Of course nltk, tf-idf will be applied first on these columns. And rest all fields can be taken care of using LabelEncoding or OneHotEncoding.

▲ 1

Dana Nourie
05/06/2019 11:41 PM

Great article! Can you explain the stop words you chose? I thought stop words were long lists of words not to include like and is the, etc. What's the best practice for creating a stop word list?

▲ 1

great

▲ 1

---

m s
09/06/2019 01:53 PM
very useful

▲ 1

Mohammad Heydari
14/06/2019 03:23 AM
nice tutorial for beginners

▲ 1

Guru Charan Pathalla
03/07/2019 10:23 PM
Hi Avinash,

I am working on a machine learning project in which I need your suggestion and inputs. I do different types of requests for reports. Some examples of reports are like New Hires report (where the list of professionals joined the firm in a particular date range are reported), Separations report(where the list of professionals who separated from the firm for a specific span of time/ period are reported), Actives report(where the list of active professionals as on a particular date is reported).

I worked on automating the extraction of this reports. All we need to do is understand the type of request and then run the query for Hires or Separations or Actives based on the request. I now want to automate this using machine learning and link it to my already automated code so that the machine learning code understands the requirement and runs the query.

I was able to do half of this and am in the middle of it. I am not sure how to complete this.

I have done the Text processing (cleaning, removing stop words, stemming). I am ready with the word where I think I need to create 3 bags for Hires, Separation and Actives. Currently I was able to make these bags with very few words and used the Logistic regression model for classification.

What do you suggest me to do for this classification. I am happy to discuss it live.

your help in this is greatly appreciated!! :)

Jack marco
17/07/2019 04:44 PM

We have been knowing , reading & learning text analytics through various resources online and I find this to be an interesting one to know about text analytics using nltk.

▲ 2

Ankit Nayan
21/07/2019 06:52 AM

thanks alot

▲ 2

Madhumita Roy
23/07/2019 06:08 AM

Loved how the tutorial was simple and well explained

▲ 2

José Ângelo de Assis Júnior
26/07/2019 03:08 AM

I have tried to reproduce this article using messages from a chatbot to do the training and then using facebook messages for validation.

▲ 1

José Ângelo de Assis Júnior
26/07/2019 03:08 AM

Both datasets went through the TFIDF process. While running the model on the validation data I received the error message: Number of features of the model must match the input. Model n_features.

▲ 1

22/08/2019 09:10 AM

Thank you for this. Its very helpful . This was an excellent tutorial!

▲ 2

Avinash Navlani
23/08/2019 09:22 AM

Thanks Ibrahim

▲ 1

---

syeda tooba
05/09/2019 04:30 AM

i want to know that can we find tf-idf from nltk?

▲ 1

Akash Raj
29/10/2019 06:03 AM

tf-idf is very easy to find by using coding and as per my knowledge maximum people write their own function for finding tf-idf....but i provide some resources for you,once go through it

https://markhneedham.com/blog/2015/02/15/pythonscikit-learn-calculating-tfidf-on-how-i-met-your-mother-transcripts/

https://iyzico.engineering/how-to-calculate-tf-idf-term-frequency-inverse-document-frequency-from-the-beatles-biography-in-c4c3cd968296

▲ 1

Akash Raj
29/10/2019 05:00 AM

at the time of removing stop words,we are checking stopwords in tokenized_words not in tokenized_sent(there is no such variable declared above).....it's a silly writing mistake but create big confusion for some of us.

▲ 1

For some reason my Tokenized Sentence and my Filtered Sentence are the same.

▲ 1

Fatima-ezzahra Badaoui
10/11/2019 10:11 AM

Because Stopword filter works on words, not on sentences.

▲ 1

Fatima-ezzahra Badaoui
10/11/2019 10:45 AM

Thank you so much for this amazing tutorial.

▲ 1

Mohinder Pal Goyal
16/11/2019 02:59 PM

Maybe matrix factorization can improve the accuracy.

▲ 1

Anil Mavati
04/12/2019 01:05 PM

Sir this is excellent tutorial.

Can you tell how to do the sentiment analysis for the patient discharge summary. Which contains the fields like Diagnosis , Echo_before_admission, History , Procedure, Examination,Investigation,etc..

If you have research papers regarding this that will be helpful. Thank you

▲ 1

leela v
23/12/2019 04:51 PM

Thank you very much for the tutorial, well explained, appreciate your effort.

▲ 1

Hi Avinash

Thanks for the tutorial. Great learning. I need one clarification. I have a dataset where only 2 columns are text with special characters. Others are integers.

Can i use this to get the TF-IDF for each row of text and then use that value for my prediction? Is that a good approach?

▲ 1

Ikram EL Mbarki
30/12/2019 01:52 AM

How can we do a proposal of content that can solve the problems detected from email exchanges between employees or between groups in a workspace using NLP and machine learning (Clustering)

▲ 1

Sugantha Raja
31/12/2019 03:58 PM

Great post! I am actually getting ready to across this information, is very helpful. Keep up the good work you are doing here. Python Training

▲ 1

Pham Huyen
10/02/2020 10:06 AM

Hello!

With the current movie theme, everyone can also enjoy the showbox, mobdro, titanium tv, ... applications provided by: https://showboxvpn.com

▲ 1

Máy Phun Sương
12/02/2020 03:42 PM

máy phun sương làm mát này sẽ giúp đảm bảo sức khỏe của công nhân làm việc lâu dài, tạo không gian thoáng mát, dễ chịu và duy trì độ ẩm trên mức 75%, nhưng không ảnh hưởng đến đồ đạc trong nhà xưởng.

▲ 1

Máy Phun Sương
12/02/2020 03:44 PM

Hệ thống máy phun sương cung cấp một lượng hơi nước phù hợp và duy trì và điều chỉnh nhiệt độ và độ ẩm cho rau và hoa màu phát triển tươi tốt. Sử dụng máy phun sương nén nức với áp lực cao, nước được đưa qua đường ống và phun ra thông qua các đầu béc phun sương của hệ thống với áp suất lên đến 200 PSI hoặc lớn nhỏ tùy vào các loại máy.

▲ 1

Ben Mohamed kater nada
12/02/2020 11:05 PM

hey , Great tutorial , I was wondering can i use this in my project , my project aim to conduct a sentiments analyse on comments and tweets , positive , negative or neutral . I'm newbie to this so any useful information or suggestions are welcome and appreciated.

▲ 1

Hoàng Oanh
18/02/2020 04:29 PM

Các hệ thống phun sương ngày càng được áp dụng nhiều hơn trong ngành trồng trọt. Đối với cây phong lan, việc sử dụng phương pháp tưới theo hệ thống ***máy phun sương tưới lan*** sẽ nâng cao hiệu quả khi thu hoạch và tiết kiệm nguồn nhân lực cùng chi phí chăm sóc.

▲ 1

liwibee a
25/02/2020 05:10 PM

Very informative article he wrote. Thankyou Also do check omggamer

▲ 1

Anne Beck
26/02/2020 08:50 PM

This is the book that changed my approach when I started using 501 for ProwritingAid. The authors have done great work, and have made it more enjoyable to use it.

muraliadya
14/03/2020 01:11 AM

Great Example and useful explanation helps lot for beginners,

▲ 1