studi

study

**Word Cloud**

Wordclouds.com

A picture containing text, businesscard, screenshot

Description automatically generated

A tag cloud is a visual representation of text data, which is often used to depict keyword metadata on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color.

For this exercise, we are going to consider a food review from a restaurant dataset. The dataset provides the text reviews and the rating score from 1 to 5. We are now trying to understand what the reviews were saying. If the reviews are in numbers, we can use descriptive statistics to see the data distribution. But the reviews are in text form. How to quickly get a summary of hundreds of reviews text? Reading them one by one is not an efficient solution.

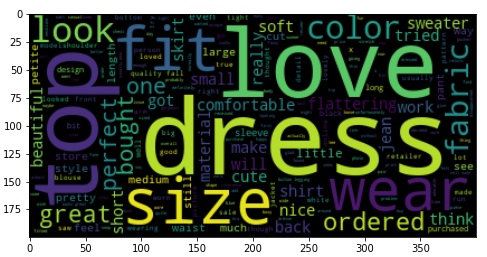
A simple way is to plot the word cloud. Word cloud displays the commonly found words from the whole dataset. A larger font size means more frequently found.

visual\_rev = WordCloud().generate(' '.join(data['Review Text']))

plt.figure(figsize=(8,8))

plt.imshow(visual\_rev, interpolation='bilinear')

plt.show()

Fig. 1 word cloud (source: [image by author](https://www.kaggle.com/rendyk/e-commerce-nlp-and-sentiment-analysis-part-1))

From the word cloud, we can notice that the reviews are talking about dress, love, size, top, wear, and so on as they are the most commonly found words. To display the exact frequency number of each word, we can use Counter(). Let’s demonstrate it using the variable “text2”.

from collections import Counter

Counter(word\_tokenize(text2))

**Output**:

Counter({'The': 2,

'monkeys': 1,

'are': 1,

'eating': 1,

'7': 1,

'bananas': 2,

'on': 1,

'the': 1,

'tree': 3,

'!': 1,

'will': 1,

'only': 1,

'have': 1,

'5': 1,

'left': 1,

'later': 1,

'.': 2,

'One': 1,

'monkey': 1,

'is': 1,

'jumping': 1,

'to': 1,

'another': 1})

The following code does the same thing, but it calls only the top 3 most common words.

Counter(word\_tokenize(text2)).most\_common(3)

Output:

[('tree', 3), ('The', 2), ('bananas', 2)]

**Sentiment Analysis**

Sentiment analysis can be run by using TextBlob or training a Machine Learning model. TextBlob does not require training. It can tell the polarity and subjectivity of the reviews. The polarity ranges from 1 to -1 expressing positive sentiment to negative sentiment. Here is the code to apply sentiment analysis to the text2 = ‘The monkeys are eating 7 bananas on the tree! The tree will only have 5 bananas left later. One monkey is jumping to another tree.’

from textblob import TextBlob

TextBlob(text2).sentiment

:

Sentiment(polarity=-0 Output.0125, subjectivity=0.25)

Now, let’s see how it works on the women’s clothing dataset.

# Applying text blob sentiment

def polarity(t):

a = TextBlob(t).sentiment

return a[0]

def subjectivity(t):

a = TextBlob(t).sentiment

return a[1]

data['polarity'] = data.apply(lambda t: polarity(t['Review Text']), axis=1)

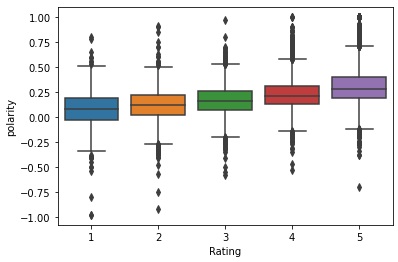
data['subjectivity'] = data.apply(lambda t: subjectivity(t['Review Text']), axis=1)

data.head()

Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Review Text | Rating | polarity | subjectivity |
| 0 | Absolutely wonderful – silky and sexy and comf… | 4 | 0.633333 | 0.933333 |
| 1 | Love this dress! it’s sooo pretty. i happene… | 5 | 0.339583 | 0.725000 |
| 2 | I had such high hopes for this dress and reall… | 3 | 0.073675 | 0.356294 |
| 3 | I love, love, love this jumpsuit. it’s fun, fl… | 5 | 0.550000 | 0.625000 |
| 4 | This shirt is very flattering to all due to th… | 5 | 0.512891 | 0.568750 |

The table above displays the first 5 rows of the review text polarity. Examine how the words in “Review Text” return the “polarity”. In the same dataset, the satisfaction rating is also given by each of the reviewers in the feature “Rating”. The “Rating” ranges from 1 to 5. The figure below visualizes the polarity distribution of each rating class. Examine that a higher rating tends to have more positive polarity.

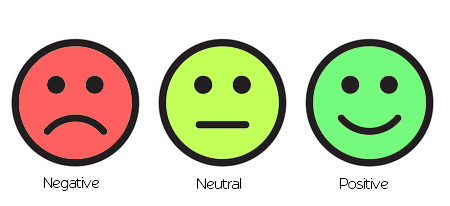


*This article was published as a part of the*[*Data Science Blogathon*](https://datahack.analyticsvidhya.com/contest/data-science-blogathon-13/)

It was 6:30 AM on a Thursday, I got a call from my senior regarding a project on sentiment analysis. Honestly, I was dreaming and I was very much sleepy, Thanks to the call recorder that helped me listen to the requirements and deadline once again. So the conclusion was I have to analyze review data and classify them into Negative, Positive, and Neutral and I had to deliver it the next Monday. I have a good interest in NLP and Sentiment analysis is a major part of it. I have heard of Lexicon-based sentiment analyzers, particularly TextBlob, and I chose to go with it because there was no time to experiment with some new algorithm. TextBlob is developed by Steven Loria, it’s a python library that  
uses Natural Language Toolkit to perform the tasks. I have seen many projects that use TextBlob as a sentiment analyzer, mostly with Twitter data or Movie review data.

Why sentiment analysis is important?

A review on anything(product, movie, person, etc.) is very important these days to get a clear picture of what the end-users think. An enthusiast gives all kinds of reviews on a particular domain or a consumer who has a bad experience provides negative reviews on the product they purchased/used. It’s us who gave “The Shawshank Redemption” a rating of 9.3 and “Student of the Year 2” has a rating of 2.2 on IMDB. Talking about any product’s feedback, We have plenty of user review data if we want to take a look at this particular topic. The sources can be shopping portals like Amazon, Flipkart, Alibaba, Myntra, etc. as well as social media platforms like Twitter, Facebook,  
etc.



The domain in which I was about to work was a little different from movie reviews, It was mobile devices review from various shopping portals and social media. The data extraction part was traditional i.e. to scrap it from the portals. I used web scraping to get all the data. Now it was time to perform the data cleaning and sentiment analysis. Cleaning the data was a time-consuming task as the review contained sentences with irrelevant things that were not useful for my objective. For data cleaning, I used regular expressions and a few ready-made  
python libraries. The corpus(data) was now ready to be analyzed.

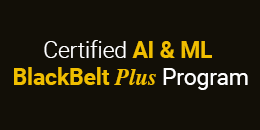
**Begin with TextBlob**

Let’s see some basics of TextBlob, It uses NLTK (Natural Language ToolKit) and the input contains a single sentence, The output of TextBlob is **polarity** and **subjectivity**. Polarity score lies between (-1 to 1) where -1 identifies the most negative words such as***‘disgusting’, ‘awful’, ‘pathetic’,***and 1 identifies the most positive words like***‘excellent’, ‘best’.***Subjectivity score lies between (0 and 1), It shows the amount of personal opinion, If a sentence has high subjectivity i.e. close to 1, It resembles that the text contains more personal opinion than factual information. I was more concerned about the Polarity score as my objective was not to identify factual information, so I skipped the subjectivity score in my project.

To start working with TextBlob it requires preinstalled python, and configured pip. The pip installation command for TextBlob is

**pip install textblob**

To import TextBlob we need to write as



**Become a Full-Stack Data Scientist**

Power Ahead in your AI ML Career | No Pre-requisites Required[Download Brochure](https://blackbelt.analyticsvidhya.com/plus?utm_source=blog&utm_medium=Bannerbetweenarticles&utm_campaign=BlackBelt)

**from textblob import TextBlob**

TextBlob syntax to get polarity score:

**res = TextBlob(sentence)**

**print(res.sentiment.polarity)**

As TextBlob is a Lexicon-based sentiment analyzer It has some predefined rules or we can say word and weight dictionary, where it has some scores that help to calculate a sentence’s polarity. That’s why the Lexicon-based sentiment analyzers are also called “Rule-based sentiment analyzers”.

Let’s check some random sentences’ polarity with TextBlob, The beauty of TextBlob is it has a very easy syntax.

* ***It’s a beautiful day.***
* ***This movie is badly directed.***
* ***The weather today is pleasant.***

We get the polarity values as 0.85, -0.69, 0.73 respectively. In the above data, we have a negative sentence***“This movie is badly directed”***which has a polarity score of -0.69 which resembles one of the most negative sentences,

Let’s change the word***“badly”*** to***“amazingly”*.**

**res = TextBlob("This movie is amazingly directed")**

**print(res.sentiment.polarity)**

The output comes as 0.6000000000000001. Here, TextBlob works *amazingly* as a sentiment analyzer. And I was successful in delivering my project next Monday and got appreciation as well from my colleagues.

The next day I was just looking at the result files and some particular sentence caught my attention.  
It was*“no slow-motion camera”*



As I told that my domain was mobile phone review analysis so if anyone writes this sentence it’s a negative one, but TextBlob classified it as positive with a polarity score of 0.15. That made me curious and forced me to do some more exploration on how TextBlob works and the finding was *when any negation is added with any sentence it simply multiplies -0.5 to the polarity score of the word*. In my case, it was the word “slow” which was a negative word and have a polarity score of -0.3 so when it multiplies -0.5 then the resulting polarity of the sentence becomes positive 0.15.

Another issue I faced with TextBlob was when the negation word is added somewhere in between i.e. not adjacent to the word which has some polarity other than 0.

* ***This is the best Face Recognition at this price. (Polarity: 1.0)***
* ***This is not the best Face Recognition at this price. (Polarity: 1.0)***

In the above example if we see the word***“best”*,**it has a polarity score of 1.0 however in the second sentence it should multiply -0.5 to 1.0 and the value should appear as -0.5 but this is not the case. The answer here is TextBlob considers***“not best”***differently from***“not the best”***and that creates the issue. These things need to be changed as it was impacting the overall sentiment on the product.

The Game Changer

I started exploring again on Sentiment analyzers and found a research paper written by Eric Gilbert and C. Hutto. It was on VADER (Valence Aware Dictionary and Sentiment Reasoner). VADER is another Lexicon-based sentiment analyzer that has pre-defined rules for words or lexicons. VADER not only tells the lexicon is positive, negative, or neutral, it also tells how positive, negative, or neutral a sentence is. The output from VADER comes in a Python dictionary in which we have four keys and their corresponding values. ‘neg’, ‘neu’, ‘pos’, and ‘compound’ which stands for Negative, Neutral, and Positive respectively. The Compound score is an indispensable score that is calculated by normalizing the other 3 scores (neg, neu, pos) between -1 and +1. The decision criteria are similar to TextBlob -1 is for most negative and +1 is for most positive.

It works differently than TextBlob. I took some of the problematic sentences and executed them with VADER and the output was correct.

To start working on VADER we need to install it with pip.

**pip install vaderSentiment**

We need to import and initialize it as

**from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer**

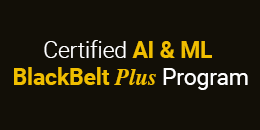
**sid\_obj= SentimentIntensityAnalyzer()**

I checked with my problematic sentence:

**print(sid\_obj.polarity\_scores("no slow motion camera"))**



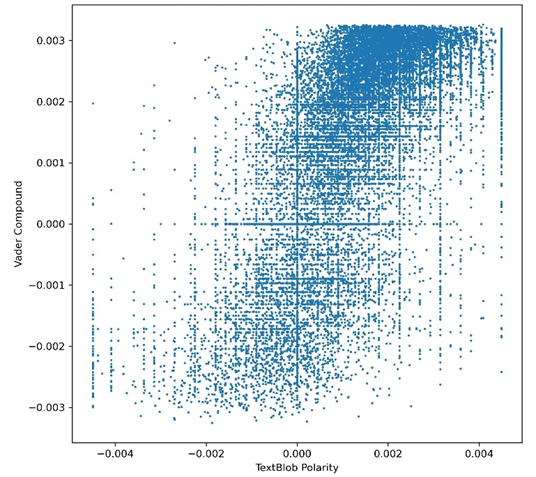
The output of the above sentence is a  compound score of -0.296.



**Become a Full-Stack Data Scientist**

Power Ahead in your AI ML Career | No Pre-requisites Required[Download Brochure](https://blackbelt.analyticsvidhya.com/plus?utm_source=blog&utm_medium=Bannerbetweenarticles&utm_campaign=BlackBelt)

I analyzed the whole corpus with Vader and TextBlob. The output brought me to the conclusion that TextBlob was struggling with negative sentences, particularly negations.

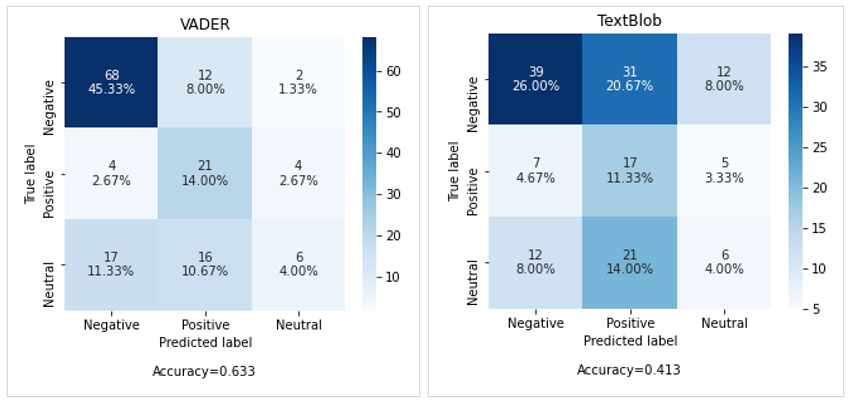


The above graph is a scatter plot of the Pearson correlation coefficient between the mentioned two algorithms VADER and TextBlob. In the above graph, we can see that the sentences considered Negative by VADER were mostly identified as positive by TextBlob. In the case of the 1st, & 3rd quadrant both the algorithms have reached an agreement. But in the case of the 2nd & 4th quadrant, there is a mismatch especially for the 4th quadrant, which has more contradictory data, it belongs to Positive as per TextBlob and Negative as per VADER.  
To get rid of the bias I had for TextBlob I needed more proof to get convinced that VADER is doing the job better than TextBlob in my project. To get the proof I needed more experiments.

**As said by Richard Feynman *“It doesn’t matter how beautiful  
your theory is, it doesn’t matter how smart you are. If it doesn’t agree with  
experiment, it’s wrong.”***

The best way was to compare the two algorithms I had, but the major problem was to compare with what? I wanted some real analysis, but who will decide a correct sentiment? It’s us, Humans. Initially, I thought I will mark all the correct sentiments, but I researched a little and came to know about “Wisdom of Crowd”. In the book “The Wisdom of Crowds”, James Surowiecki has written,***“collective  
knowledge of a group of people as expressed through their aggregated opinions  
can be trusted as an alternative to an expert’s knowledge”***. I decided to go  
with Wisdom of Crowd to get the correct sentiment.

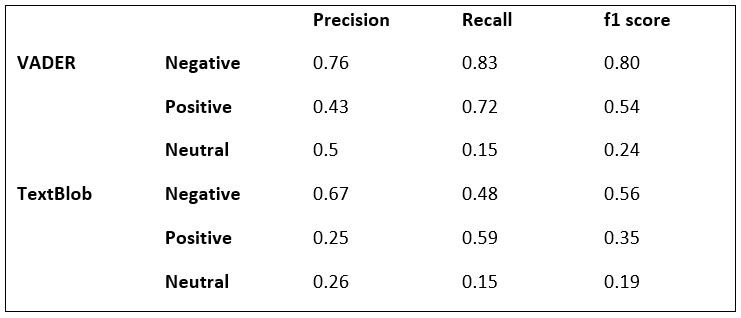
I selected 20 people for this task, of which 10 had expertise in the mobile domain while the rest did not. I gave them 150 random sentences to mark as Positive, Negative, and Neutral. Then from the output provided by each individual, I took the average of 20 people and gave a final correct sentiment rating. That was the gold standard. Now we can compare TextBlob and Vader. To get the accuracy of an algorithm as compared to human analyzed sentences, I created confusion matrices with both the algorithm versus crowdsourcing data.



The result is very convincing that VADER outperforms TextBlob when it comes to negative polarity  
detection. In the above-mentioned confusion matrices VADER gets an overall accuracy of 63.3% however TextBlob gets an accuracy of 41.3%.

**Can we say VADER is better than TextBlob for Sentiment Analysis?**

It depends on the requirement of the user. My answer is No, VADER is not better than TextBlob in all areas. However, I can say that  
VADER works better when it comes to negative sentiment classification.



In the above-mentioned table the f1 score of VADER is 0.80 when it comes to negative polarity detection and for TextBlob it comes as 0.56. From this, we can conclude that VADER does better sentiment analysis when it comes to negative polarity detection.

**Textblob can be used for complex analysis and working with textual data**. When a sentence is passed into Textblob it gives two outputs, which are polarity and subjectivity. Polarity is the output that lies between [-1,1], where -1 refers to negative sentiment and +1 refers to positive sentiment

# Text sentiment analysis with textblob

Sentiment Analysis can help us decipher the mood and emotions of general public and gather insightful information regarding the context. Sentiment Analysis is a process of analyzing data and classifying it based on the need of the research.

!pip install textblob

from textblob import TextBlob

from IPython.display import display, HTML

blob = TextBlob(text\_blob)

### Sentiment analysis

Write your text here:

input\_1



* **Polarity** simply means emotions expressed in a sentence => negative vs. positive (-1.0 => +1.0)
* **Subjectivity** simply expresses some personal feelings, views, or beliefs => objective vs. subjective (+0.0 => +1.0)

def polarity\_to\_text(blob):

if (blob.sentiment.polarity > 0.1):

return 'Polarity is positive 😊'

elif(blob.sentiment.polarity <= 0.1 and blob.sentiment.polarity >= -0.1):

return 'Polarity is neutral 😐'

else:

return 'Polarity is negative 😡'

def subjecitivity\_to\_text(blob):

if (blob.sentiment.subjectivity > 0.1):

return 'Sentence is objective 👨‍🏫'

elif(blob.sentiment.subjectivity <= 0.1 and blob.sentiment.subjectivity >= -0.1):

return 'Sentence is is neutral 😐'

else:

return 'Sentence is subjective 👀'

HTML(f'''<h3>{polarity\_to\_text(blob)}</h3>

<h3>{subjecitivity\_to\_text(blob)}</h3>''')

print(f'Raw output: {blob.sentiment}')

Raw output: Sentiment(polarity=0.0, subjectivity=0.0)

|  |
| --- |
| **from** textblob **import** TextBlob      **def** sentiment(polarity):  **if** blob.sentiment.polarity < 0:          print("Negative")  **elif** blob.sentiment.polarity > 0:          print("Positive")  **else**:          print("Neutral")      blob **=** TextBlob("The movie was excellent!")  print(blob.sentiment)  sentiment(blob.sentiment.polarity)    blob **=** TextBlob("The movie was not bad.")  print(blob.sentiment)  sentiment(blob.sentiment.polarity)    blob **=** TextBlob("The movie was ridiculous.")  print(blob.sentiment)  sentiment(blob.sentiment.polarity) |

Sentiment(polarity=1.0, subjectivity=1.0)

Positive

Sentiment(polarity=0.3499999999999999, subjectivity=0.6666666666666666)

Positive

Sentiment(polarity=-0.3333333333333333, subjectivity=1.0)

Negative

## Using Tweepy to Read Tweets

Tweepy is a Python library that simplifies the interaction between Python code and the Twitter API. More information about Tweepy can be found at [docs.tweepy.org/en/v3.5.0](https://docs.tweepy.org/en/v3.5.0). At this time, return to the Jupyter notebook and enter the following code to install the Tweepy API. The exclamation mark instructs Jupyter to execute a command in the shell:

XMLCopy

!pip install tweepy

Once the code executes successfully, the response text in the cell will read: “Successfully installed tweepy-3.6.0,” although the specific version number may change. In the following cell, enter the code in **Figure 4** into the newly created empty cell and execute it.

Figure 4 Use Tweepy to Access the Twitter API

XMLCopy

import tweepy

consumer\_key = "[Insert Consumer Key value]"

consumer\_secret = "[Insert Consumer Secret value]"

access\_token = "[Insert Access Token value]"

access\_token\_secret = "[Insert Access Token Secret value]"

authentication\_info = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

authentication\_info.set\_access\_token(access\_token, access\_token\_secret)

twitter\_api = tweepy.API(authentication\_info)

spacex\_tweets = twitter\_api.search("#spacex")

for tweet in spacex\_tweets:

  print(tweet.text)

  analysis = TextBlob(tweet.text)

  print(analysis.sentiment)

The results that come back should look similar to the following:

XMLCopy

#ElonMusk deletes own, #SpaceX and #Tesla Facebook pages after #deletefacebook https://t.co/zKGg4ZM2pi https://t.co/d9YFboUAUj

Sentiment(polarity=0.0, subjectivity=0.0)

RT @loislane28: Wow. did @elonmusk just delete #SpaceX and #Tesla from Facebook? https://t.co/iN4N4zknca

Sentiment(polarity=0.0, subjectivity=0.0)

Keep in mind that as the code executes a search on live Twitter data, your results will certainly vary. The formatting is a little confusing to read. Modify the for loop in the cell to the following and then re-execute the code.

XMLCopy

for tweet in spacex\_tweets:

  analysis = TextBlob(tweet.text)

  print('{0} | {1} | {2}'.format(tweet.text, analysis.sentiment.polarity,

    analysis.sentiment.subjectivity))

Adding the pipe characters to the output should make it easier to read. Also note that the sentiment property’s two fields, polarity and subjectivity, can be displayed individually.

[[](https://towardsdatascience.com/?source=post_page-----9b1771e7cdd6--------------------------------)](https://towardsdatascience.com/?source=post_page-----9b1771e7cdd6--------------------------------)

Published in

[Towards Data Science](https://towardsdatascience.com/?source=post_page-----9b1771e7cdd6--------------------------------)

You have **2** free member-only stories left this month.

[Sign up for Medium and get an extra one](https://medium.com/m/signin?operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fcnn-sentiment-analysis-9b1771e7cdd6&source=-----9b1771e7cdd6---------------------smart_meter-----------)

[[](https://ritakurban.medium.com/?source=post_page-----9b1771e7cdd6--------------------------------)](https://ritakurban.medium.com/?source=post_page-----9b1771e7cdd6--------------------------------)

[Rita Kurban](https://ritakurban.medium.com/?source=post_page-----9b1771e7cdd6--------------------------------)

Dec 30, 2019

·

8 min read

·

Listen

# CNN Sentiment Analysis

## Use Convolutional Neural Networks to Analyze Sentiments in the IMDb Dataset

Convolutional neural networks, or CNNs, form the backbone of multiple modern computer vision systems. Image classification, object detection, semantic segmentation — all these tasks can be tackled by CNNs successfully. At first glance, it seems to be counterintuitive to use the same technique for a task as different as Natural Language Processing. This post is my attempt to explain the intuition behind this approach using the famous IMDb dataset.

A picture containing background pattern

Description automatically generated

Source: <https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/>

After reading this post, you will:

1. Learn how to preprocess text using [torchtext](https://torchtext.readthedocs.io/en/latest/index.html" \t "_blank)
2. Understand the idea behind convolutions
3. Learn how to represent text as images
4. Build a basic CNN Sentiment Analysis model in PyTorch

Let’s get started!

# Data

The IMDb dataset for binary sentiment classification contains a set of 25,000 highly polar movie reviews for training and 25,000 for testing. Luckily, it is a part of torchtext, so it is straightforward to load and pre-process it in PyTorch:

# Create an instance that turns text into tensors  
TEXT = data.Field(tokenize = 'spacy', batch\_first = True)  
LABEL = data.LabelField(dtype = torch.float)# Load data from torchtext  
train\_data, test\_data = datasets.IMDB.splits(TEXT, LABEL)  
train\_data, valid\_data = train\_data.split()# Select only the most important 30000 words  
MAX\_VOCAB\_SIZE = 30\_000# Build vocabulary  
TEXT.build\_vocab(train\_data,   
 max\_size = MAX\_VOCAB\_SIZE,   
 # Load pretrained embeddings  
 vectors = "glove.6B.100d",   
 unk\_init = torch.Tensor.normal\_)LABEL.build\_vocab(train\_data)

The data.Fieldclass defines a datatype together with instructions for converting it to Tensor. In this case, we are using [SpaCy](https://spacy.io/api/tokenizer" \t "_blank) tokenizer to segment text into individual tokens (words). After that, we build a vocabulary so that we can convert our tokens into integer numbers later. The vocabulary is constructed with all words present in our train dataset. Additionally, we load pre-trained [GloVe](https://nlp.stanford.edu/projects/glove/" \t "_blank) embeddings so that we don’t need to train our own word vectors from scratch. If you’re wondering what word embeddings are, they are a form of word representation that bridges the human understanding of language to that of a machine. To learn more, read [this article](https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa). Since we will be training our model in batches, we will also create data iterators that output a specific number of samples at a time:

# Create PyTorch iterators to use in training  
train\_iterator, valid\_iterator, test\_iterator = data.BucketIterator.splits(  
 (train\_data, valid\_data, test\_data),   
 batch\_size = BATCH\_SIZE,   
 device = device)

BucketIterator is a module in torchtext that is specifically optimized to minimize the amount of padding needed while producing freshly shuffled batches for each new epoch. Now we are done with text preprocessing, so it’s time to learn more about CNNs.

# Convolutions

Convolutions are sliding window functions applied to a matrix that achieve specific results (e. g., image blur, edge detection.) The sliding window is called a kernel, filter, or feature detector. The visualization shows six 3×3 kernels that multiply their values element-wise with the original matrix, then sum them up. To get the full convolution, we do this for each element by sliding the filter over the entire matrix:

CNNs are just several layers of convolutions with activation functions like [ReLU](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)" \t "_blank) that make it possible to model non-linear relationships. By applying this set of dot products, we can extract relevant information from images, starting from edges on shallower levels to identifying the entire objects on deeper levels of neural networks. Unlike traditional neural networks that simply flatten the input, CNNs can extract spatial relationships that are especially useful for image data. But how about the text?

# CNNs for NLP

Remember the word embeddings we discussed above? That’s where they come into play. Images are just some points in space, just like the word vectors are. By representing each word with a vector of numbers of a specific length and stacking a bunch of words on top of each other, we get an “image.” Computer vision filters usually have the same width and height and slide over local parts of an image. In NLP, we typically use filters that slide over word embeddings — matrix rows. Therefore, filters usually have the same width as the length of the word embeddings. The height varies but is generally from 1 to 5, which corresponds to different n-grams. N-grams are just a bunch of subsequent words. By analyzing sequences, we can better understand the meaning of a sentence. For example, the word “like” alone has an opposite meaning compared to the bi-gram “don’t like”; the latter gives us a better understanding of the real meaning. In a way, by analyzing n-grams, we are capturing the spatial relationships in texts, which makes it easier for the model to understand the sentiment. The visualization below summarizes the concepts we just covered:

Diagram

Description automatically generated

Source: Lopez et al. (2017) Link: <https://arxiv.org/pdf/1703.03091.pdf>

# PyTorch Model

Let’s now build a binary CNN classifier. We will base our model on the built-in PyTorch nn.Module:

class CNN\_Text(nn.Module):  
 ''' Define network architecture and forward path. '''  
 def \_\_init\_\_(self, vocab\_size,   
 vector\_size, n\_filters,   
 filter\_sizes, output\_dim,   
 dropout, pad\_idx):  
   
 super().\_\_init\_\_()  
   
 # Create word embeddings from the input words   
 self.embedding = nn.Embedding(vocab\_size, vector\_size,   
 padding\_idx = pad\_idx)  
   
 # Specify convolutions with filters of different sizes (fs)  
 self.convs = nn.ModuleList([nn.Conv2d(in\_channels = 1,   
 out\_channels = n\_filters,   
 kernel\_size = (fs, vector\_size))   
 for fs in filter\_sizes])  
   
 # Add a fully connected layer for final predicitons  
 self.linear = nn.Linear(len(filter\_sizes) \  
 \* n\_filters, output\_dim)  
   
 # Drop some of the nodes to increase robustness in training  
 self.dropout = nn.Dropout(dropout)  
   
   
   
 def forward(self, text):  
 '''Forward path of the network.'''   
 # Get word embeddings and formt them for convolutions  
 embedded = self.embedding(text).unsqueeze(1)  
   
 # Perform convolutions and apply activation functions  
 conved = [F.relu(conv(embedded)).squeeze(3)   
 for conv in self.convs]  
   
 # Pooling layer to reduce dimensionality   
 pooled = [F.max\_pool1d(conv, conv.shape[2]).squeeze(2)   
 for conv in conved]  
   
 # Dropout layer  
 cat = self.dropout(torch.cat(pooled, dim = 1))  
 return self.linear(cat)

In the initfunction, we specify different layer types: embedding, convolution, dropout, and linear. All these layers are integrated into PyTorch and are very easy to use. The only tricky part is calculating the correct number of dimensions. In the case of the linear layer, it will be equal to the number of filters you use (I use 100, but you can pick any other number) multiplied by the number of different filter sizes (5 in my case.) We can think of the weights of this linear layer as “weighting up the evidence” from each of the 500 n-grams. The forward function specifies the order in which these layers should be applied. Notice that we also use max-pooling layers. The idea behind max-pooling is that the maximum value is the “most important” feature for determining the sentiment of the review, which corresponds to the “most important” n-gram is identified through backpropagation. Max-pooling is also useful for reducing the number of parameters and computations in the network.

Once we specified our network architecture, let’s load the pre-trained GloVe embeddings we imported before:

# Initialize weights with pre-trained embeddings  
model.embedding.weight.data.copy\_(TEXT.vocab.vectors)# Zero the initial weights of the UNKnown and padding tokens.  
UNK\_IDX = TEXT.vocab.stoi[TEXT.unk\_token]# The string token used as padding. Default: “<pad>”.  
PAD\_IDX = TEXT.vocab.stoi[TEXT.pad\_token]model.embedding.weight.data[UNK\_IDX] = torch.zeros(EMBEDDING\_DIM)  
model.embedding.weight.data[PAD\_IDX] = torch.zeros(EMBEDDING\_DIM)  
model = model.to(device)

The second part of this code chunk sets the unknown vectors (the ones that are not present in the vocabulary) and the padding vectors (used in case the input size is smaller than the height of the largest filter) to zeros. We’re now ready to train and evaluate our model.

You can find the full training and evaluation code in this notebook:

Link: <https://gist.github.com/ritakurban/c9ebcbfa0be45952c99ccd199b57af3d>

Before training the model, we need to specify the network optimizer and the loss function. Adam and binary cross-entropy are popular choices for classification problems. To train our model, we get the model predictions, calculate how accurate they are using the loss function, and backpropagate through the network to optimize weights before the next run. We perform all these actions in the model.train() mode. To evaluate the model, don’t forget to turn the model.eval() mode on to make sure we’re not dropping half of the nodes with the dropout (while improving the robustness in the training phase, it will hurt during evaluation). We also don’t need to calculate the gradient in the evaluation phase so that we can turn it off with the help of the torch.no\_grad() mode.

After training the model for several epochs (use GPU to speed it up), I got the following losses and accuracies:

Chart, line chart

Description automatically generated

Losses and Accuracies

The graph indicates signs of overfitting since both training loss and accuracy keep improving while the validation loss and accuracy get worse. To avoid using the overfitted model, we only save the model in case the validation loss increased. In this case, the validation loss was the highest after the third epoch. In the training loop, this part looks as follows:

if valid\_loss < best\_valid\_loss:  
 best\_valid\_loss = valid\_loss  
 torch.save(model.state\_dict(), 'CNN-model.pt')

The performance of this model on the previously unseen test set is quite good: 85.43%. Finally, let’s predict the sentiment of some polar reviews using the CNN-model. To do so, we need to write a function that tokenizes user input and turns it into a tensor. After that, we get predictions using the model we just trained:

def sentiment(model, sentence, min\_len = 5):  
 '''Predict user-defined review sentiment.'''  
 model.eval()  
 tokenized = [tok.text for tok in nlp.tokenizer(sentence)]  
 if len(tokenized) < min\_len:  
 tokenized += ['<pad>'] \* (min\_len - len(tokenized))  
 # Map words to word embeddings  
 indexed = [TEXT.vocab.stoi[t] for t in tokenized]  
 tensor = torch.LongTensor(indexed).to(device)  
 tensor = tensor.unsqueeze(0)  
 # Get predicitons  
 prediction = torch.sigmoid(model(tensor))  
 return prediction.item()

In the original dataset, we have labels “pos” and “negs” that got mapped to 0 and 1, respectively. Let’s see how well our model performs on positive, negative, and neutral reviews:

reviews = ['This is the best movie I have ever watched!',   
 'This is an okay movie',   
 'This was a waste of time! I hated this movie.']  
scores = [sentiment(model, review) for review in reviews]

The model predictions are 0.007, 0.493, and 0.971 respectively, which is pretty good! Let’s try some tricker examples:

tricky\_reviews = ['This is not the best movie I have ever watched!',   
 'Some would say it is an okay movie, but I found it terrific.',   
 'This was a waste of time! I did not like this movie.']  
scores = [sentiment(model, review) for review in tricky\_reviews]  
scores

Unfortunately, since the model has been trained on polar reviews, it finds it quite hard to classify tricky statements. For example, the first tricky review got a score of 0.05, which is quite confident ‘yes’ even though negation is present in the sentence. Try playing around with different n-grams to see whether some of them are more important then others, maybe a model with bi-grams and 3-grams would perform better than a combination of different n-grams we used.

Table

Description automatically generated

Table of reviews and their sentiment scores

# Conclusion

In this post, we went through the concept of convolutions and discussed how they can be used to work with text. We also learned how to preprocess datasets from PyTorch and built a binary classification model for sentiment analysis. Despite being fooled by tricky examples, the model performs quite well. I hope you enjoyed reading this post and feel free to reach out to me if you have any questions!

# References

Britz, D. (2015). Understanding Convolutional Neural Networks for NLP. Retrieved from: <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

Lopez, M. M., & Kalita, J. (2017). Deep Learning applied to NLP. arXiv preprint arXiv:1703.03091. Retrieved from: <https://arxiv.org/pdf/1703.03091.pdf>

Trevett, B. (2019). Convolutional Sentiment Analysis. Retrieved from: <https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/4%20-%20Convolutional%20Sentiment%20Analysis.ipynb>

136

136

## Sign up for The Variable

### By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](https://medium.com/towards-data-science/newsletters/the-variable?source=newsletter_v3_promo--------------------------newsletter_v3_promo-----------)

[Get this newsletter](https://medium.com/m/signin?actionUrl=%2F_%2Fapi%2Fsubscriptions%2Fnewsletters%2Fd6fe9076899&operation=register&redirect=https%3A%2F%2Fmedium.com%2Ftowards-data-science%2Fnewsletters%2Fthe-variable&collection=Towards+Data+Science&collectionId=7f60cf5620c9&newsletterV3=The+Variable&newsletterV3Id=d6fe9076899&user=Ludovic+Benistant&userId=895063a310f4&source=--------------------------newsletter_v3_promo----------d6fe9076899-)

## [More from Towards Data Science](https://towardsdatascience.com/?source=post_page-----9b1771e7cdd6--------------------------------)

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fcollection%2Ftowards-data-science%2F9b1771e7cdd6&operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fcnn-sentiment-analysis-9b1771e7cdd6&collection=Towards+Data+Science&collectionId=7f60cf5620c9&source=post_page-----9b1771e7cdd6---------------------follow_footer-----------)

Your home for data science. A Medium publication sharing concepts, ideas and codes.

[[Rita Kurban](https://ritakurban.medium.com/?source=post_page-----9b1771e7cdd6----0----------------------------)](https://ritakurban.medium.com/?source=post_page-----9b1771e7cdd6----0----------------------------)

[Rita Kurban](https://ritakurban.medium.com/?source=post_page-----9b1771e7cdd6----0----------------------------)

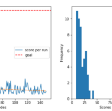
[·Dec 30, 2019](https://towardsdatascience.com/deep-q-learning-for-the-cartpole-44d761085c2f?source=post_page-----9b1771e7cdd6----0----------------------------)

## [Deep Q Learning for the CartPole](https://towardsdatascience.com/deep-q-learning-for-the-cartpole-44d761085c2f?source=post_page-----9b1771e7cdd6----0----------------------------)

[The purpose of this post is to introduce the concept of Deel Q Learning and use it to solve the CartPole environment from the OpenAI Gym. The post will consist of the following components: Open AI Gym Environment Intro Random Baseline Strategy Deep Q Learning Deep Q Learning with Replay…](https://towardsdatascience.com/deep-q-learning-for-the-cartpole-44d761085c2f?source=post_page-----9b1771e7cdd6----0----------------------------)

[Machine Learning](https://medium.com/tag/machine-learning?source=post_page-----9b1771e7cdd6---------------machine_learning-----------------)

[11 min read](https://towardsdatascience.com/deep-q-learning-for-the-cartpole-44d761085c2f?source=post_page-----9b1771e7cdd6----0----------------------------)

[[](https://towardsdatascience.com/deep-q-learning-for-the-cartpole-44d761085c2f?source=post_page-----9b1771e7cdd6----0----------------------------)](https://towardsdatascience.com/deep-q-learning-for-the-cartpole-44d761085c2f?source=post_page-----9b1771e7cdd6----0----------------------------)

Share your ideas with millions of readers.

[Write on Medium](https://medium.com/new-story?source=post_page_footer_cta_write-------------------------------------)

[[Nishakanthi Angulgamuwa](https://medium.com/@nishakanthi?source=post_page-----9b1771e7cdd6----1----------------------------)](https://medium.com/@nishakanthi?source=post_page-----9b1771e7cdd6----1----------------------------)

[Nishakanthi Angulgamuwa](https://medium.com/@nishakanthi?source=post_page-----9b1771e7cdd6----1----------------------------)

[·Dec 30, 2019](https://towardsdatascience.com/working-with-hive-using-aws-s3-and-python-4c7471533f98?source=post_page-----9b1771e7cdd6----1----------------------------)

## [Working with Hive using AWS S3 and Python](https://towardsdatascience.com/working-with-hive-using-aws-s3-and-python-4c7471533f98?source=post_page-----9b1771e7cdd6----1----------------------------)

[Beginner’s guide to maintaining a Hive schema with external data storage and executing Hive queries using Python — In this article, I’m going to share my experience of maintaining a Hive schema. This will be useful to the freshers who are willing to step into Big Data technologies. Mainly this will describe how to connect to Hive using python and How to use AWS S3 as the data…](https://towardsdatascience.com/working-with-hive-using-aws-s3-and-python-4c7471533f98?source=post_page-----9b1771e7cdd6----1----------------------------)

[Nosql Database](https://medium.com/tag/nosql-database?source=post_page-----9b1771e7cdd6---------------nosql_database-----------------)

[6 min read](https://towardsdatascience.com/working-with-hive-using-aws-s3-and-python-4c7471533f98?source=post_page-----9b1771e7cdd6----1----------------------------)

[[](https://towardsdatascience.com/working-with-hive-using-aws-s3-and-python-4c7471533f98?source=post_page-----9b1771e7cdd6----1----------------------------)](https://towardsdatascience.com/working-with-hive-using-aws-s3-and-python-4c7471533f98?source=post_page-----9b1771e7cdd6----1----------------------------)

[[Egemen Zeytinci](https://medium.com/@egemenzeytinci?source=post_page-----9b1771e7cdd6----2----------------------------)](https://medium.com/@egemenzeytinci?source=post_page-----9b1771e7cdd6----2----------------------------)

[Egemen Zeytinci](https://medium.com/@egemenzeytinci?source=post_page-----9b1771e7cdd6----2----------------------------)

[·Dec 30, 2019](https://towardsdatascience.com/predicting-hotel-cancellations-with-machine-learning-fa669f93e794?source=post_page-----9b1771e7cdd6----2----------------------------)

## [Predicting Hotel Reservation Cancellations with Machine Learning](https://towardsdatascience.com/predicting-hotel-cancellations-with-machine-learning-fa669f93e794?source=post_page-----9b1771e7cdd6----2----------------------------)

[As you can imagine, the cancellation rate for bookings in the online booking industry is quite high. Once the reservation has been cancelled, there is almost nothing to be done. This creates discomfort for many institutions and creates a desire to take precautions. …](https://towardsdatascience.com/predicting-hotel-cancellations-with-machine-learning-fa669f93e794?source=post_page-----9b1771e7cdd6----2----------------------------)

[Machine Learning](https://medium.com/tag/machine-learning?source=post_page-----9b1771e7cdd6---------------machine_learning-----------------)

[9 min read](https://towardsdatascience.com/predicting-hotel-cancellations-with-machine-learning-fa669f93e794?source=post_page-----9b1771e7cdd6----2----------------------------)

[[](https://towardsdatascience.com/predicting-hotel-cancellations-with-machine-learning-fa669f93e794?source=post_page-----9b1771e7cdd6----2----------------------------)](https://towardsdatascience.com/predicting-hotel-cancellations-with-machine-learning-fa669f93e794?source=post_page-----9b1771e7cdd6----2----------------------------)

[[Jun M.](https://medium.com/@junma?source=post_page-----9b1771e7cdd6----3----------------------------)](https://medium.com/@junma?source=post_page-----9b1771e7cdd6----3----------------------------)

[Jun M.](https://medium.com/@junma?source=post_page-----9b1771e7cdd6----3----------------------------)

[·Dec 30, 2019](https://towardsdatascience.com/logistic-regression-from-scratch-in-r-b5b122fd8e83?source=post_page-----9b1771e7cdd6----3----------------------------)

## [Logistic Regression from Scratch in R](https://towardsdatascience.com/logistic-regression-from-scratch-in-r-b5b122fd8e83?source=post_page-----9b1771e7cdd6----3----------------------------)

[please click “show embed” to reveal the equations if they didn’t show up Introduction In statistics and data science, logistic regression is used to predict the probability of a certain class or event. Usually, the model is binomial, but can also extend to multinomial. …](https://towardsdatascience.com/logistic-regression-from-scratch-in-r-b5b122fd8e83?source=post_page-----9b1771e7cdd6----3----------------------------)

[Machine Learning](https://medium.com/tag/machine-learning?source=post_page-----9b1771e7cdd6---------------machine_learning-----------------)

[5 min read](https://towardsdatascience.com/logistic-regression-from-scratch-in-r-b5b122fd8e83?source=post_page-----9b1771e7cdd6----3----------------------------)

[[](https://towardsdatascience.com/logistic-regression-from-scratch-in-r-b5b122fd8e83?source=post_page-----9b1771e7cdd6----3----------------------------)](https://towardsdatascience.com/logistic-regression-from-scratch-in-r-b5b122fd8e83?source=post_page-----9b1771e7cdd6----3----------------------------)

[[Vandit Jain](https://medium.com/@vandit_15?source=post_page-----9b1771e7cdd6----4----------------------------)](https://medium.com/@vandit_15?source=post_page-----9b1771e7cdd6----4----------------------------)

[Vandit Jain](https://medium.com/@vandit_15?source=post_page-----9b1771e7cdd6----4----------------------------)

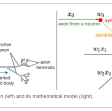
[·Dec 30, 2019](https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253?source=post_page-----9b1771e7cdd6----4----------------------------)

## [Everything you need to know about “Activation Functions” in Deep learning models](https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253?source=post_page-----9b1771e7cdd6----4----------------------------)

[This article is your one-stop solution to every possible question related to activation functions that can come into your mind that are used in deep learning models. These are basically my notes on activation functions and all the knowledge that I have about this topic summed together in one place…](https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253?source=post_page-----9b1771e7cdd6----4----------------------------)

[Machine Learning](https://medium.com/tag/machine-learning?source=post_page-----9b1771e7cdd6---------------machine_learning-----------------)

[8 min read](https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253?source=post_page-----9b1771e7cdd6----4----------------------------)

[[](https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253?source=post_page-----9b1771e7cdd6----4----------------------------)](https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253?source=post_page-----9b1771e7cdd6----4----------------------------)

[Read more from Towards Data Science](https://towardsdatascience.com/?source=post_page-----9b1771e7cdd6--------------------------------)

## Recommended from Medium

[[Gideon Kowadlo](https://medium.com/@gideon.kowadlo?source=post_internal_links---------0----------------------------)](https://medium.com/@gideon.kowadlo?source=post_internal_links---------0----------------------------)

[Gideon Kowadlo](https://medium.com/@gideon.kowadlo?source=post_internal_links---------0----------------------------)

in

[Project AGI](https://medium.com/project-agi?source=post_internal_links---------0----------------------------)

## Continuous Learning

[[Kountay Dwivedi](https://medium.com/@kountaydwivedi?source=post_internal_links---------1----------------------------)](https://medium.com/@kountaydwivedi?source=post_internal_links---------1----------------------------)

[Kountay Dwivedi](https://medium.com/@kountaydwivedi?source=post_internal_links---------1----------------------------)

in

[Analytics Vidhya](https://medium.com/analytics-vidhya?source=post_internal_links---------1----------------------------)

## Model Selection Techniques — Part 3

Graphical user interface

Description automatically generated

[[MJ](https://medium.com/@mjauhari?source=post_internal_links---------2----------------------------)](https://medium.com/@mjauhari?source=post_internal_links---------2----------------------------)

[MJ](https://medium.com/@mjauhari?source=post_internal_links---------2----------------------------)

## Udacity Self Driving Cars Nanodegree- Kidnapped Vehicle project

Chart

Description automatically generated

[[Marvin Martin](https://medium.com/@martin.marvin?source=post_internal_links---------3----------------------------)](https://medium.com/@martin.marvin?source=post_internal_links---------3----------------------------)

[Marvin Martin](https://medium.com/@martin.marvin?source=post_internal_links---------3----------------------------)

in

[The Startup](https://medium.com/swlh?source=post_internal_links---------3----------------------------)

## Mathematics behind Basic Feed Forward Neural Network (3 Layers) + Python from Scratch

A picture containing invertebrate, ctenophore

Description automatically generated

[[Raman Singh](https://medium.com/@raman-nbg?source=post_internal_links---------4----------------------------)](https://medium.com/@raman-nbg?source=post_internal_links---------4----------------------------)

[Raman Singh](https://medium.com/@raman-nbg?source=post_internal_links---------4----------------------------)

in

[ML Aide](https://medium.com/ml-aide?source=post_internal_links---------4----------------------------)

## Manage Machine Learning Lifecycle with ML Aide

ML Aide Logo

[[Nada Alay](https://medium.com/@alay.nada?source=post_internal_links---------5----------------------------)](https://medium.com/@alay.nada?source=post_internal_links---------5----------------------------)

[Nada Alay](https://medium.com/@alay.nada?source=post_internal_links---------5----------------------------)

## Great Resources to Start Learning ‘Deep Learning for Image Recognition’

A person with blue eyes

Description automatically generated with medium confidence

[[Rohit Pandey](https://medium.com/@rohitpandey576?source=post_internal_links---------6----------------------------)](https://medium.com/@rohitpandey576?source=post_internal_links---------6----------------------------)

[Rohit Pandey](https://medium.com/@rohitpandey576?source=post_internal_links---------6----------------------------)

in

[Towards Data Science](https://towardsdatascience.com/?source=post_internal_links---------6----------------------------)

## Support Vector Machines



[[Richard Switzer](https://medium.com/@rswitzer?source=post_internal_links---------7----------------------------)](https://medium.com/@rswitzer?source=post_internal_links---------7----------------------------)

[Richard Switzer](https://medium.com/@rswitzer?source=post_internal_links---------7----------------------------)

## Learning Machine Learning

A picture containing text

Description automatically generated

[Get started](https://medium.com/m/signin?operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fcnn-sentiment-analysis-9b1771e7cdd6&source=post_page--------------------------nav_reg-----------)

[Sign In](https://medium.com/m/signin?operation=login&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fcnn-sentiment-analysis-9b1771e7cdd6&source=post_page--------------------------nav_reg-----------)



[[](https://ritakurban.medium.com/)](https://ritakurban.medium.com/)

## [Rita Kurban](https://ritakurban.medium.com/)

45 Followers

Data Science enthusiast with a strong interest in Technology and Healthcare

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F3855ad42baa6&operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fcnn-sentiment-analysis-9b1771e7cdd6&user=Rita+Kurban&userId=3855ad42baa6&source=post_page-3855ad42baa6-------------------------follow_profile-----------)

## More from Medium

[[Jano Soler](https://medium.com/@janosolerdiaz?source=read_next_recirc---------0---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)](https://medium.com/@janosolerdiaz?source=read_next_recirc---------0---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

[Jano Soler](https://medium.com/@janosolerdiaz?source=read_next_recirc---------0---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

## Disease detection in oranges with Machine learning

Schema of a convolutional network for image classification

[[Amay Gada](https://amaygada.medium.com/?source=read_next_recirc---------1---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)](https://amaygada.medium.com/?source=read_next_recirc---------1---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

[Amay Gada](https://amaygada.medium.com/?source=read_next_recirc---------1---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

in

[Analytics Vidhya](https://medium.com/analytics-vidhya?source=read_next_recirc---------1---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

## Sequence to Sequence models (with implementation) — Part 1

Diagram

Description automatically generated

[[vivek](https://medium.com/@Vivek06?source=read_next_recirc---------2---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)](https://medium.com/@Vivek06?source=read_next_recirc---------2---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

[vivek](https://medium.com/@Vivek06?source=read_next_recirc---------2---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

in

[Towards AI](https://pub.towardsai.net/?source=read_next_recirc---------2---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

## Why convert text to a vector?

Plane PI with normal w is separating positive reviews from negative reviews

[[Captain Nemo](https://medium.com/@captain_nemo?source=read_next_recirc---------3---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)](https://medium.com/@captain_nemo?source=read_next_recirc---------3---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

[Captain Nemo](https://medium.com/@captain_nemo?source=read_next_recirc---------3---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

in

[MLearning.ai](https://medium.com/mlearning-ai?source=read_next_recirc---------3---------------------4ab5df72_97a0_4ae7_bfe4_7c4caa3858bc-------)

## An Artificially Intelligent Ghost-Writer

Ghost Rider Writer

[Help](https://help.medium.com/hc/en-us)

[Status](https://medium.statuspage.io/)

[Writers](https://about.medium.com/creators/)

[Blog](https://blog.medium.com/)

[Careers](https://medium.com/jobs-at-medium/work-at-medium-959d1a85284e)

[Privacy](https://policy.medium.com/medium-privacy-policy-f03bf92035c9)

[Terms](https://policy.medium.com/medium-terms-of-service-9db0094a1e0f)

[About](https://medium.com/about?autoplay=1)

[Knowable](https://knowable.fyi/)