





Open in app

Get started

For this example, we're examining a dataset of Amazon products/reviews which can be found and downloaded for free on [data.world](#). I'll be using Python in Jupyter notebook.

Here are the imports used:

```
1 #Import pandas
2 import pandas as pd
3 #Import Natural Language Toolkit
4 import nltk
5 #Import BeautifulSoup
6 from bs4 import BeautifulSoup
7 #Import string for list of punctuation
8 import string
9 # Import the stop word list
10 from nltk.corpus import stopwords
11 # Import Tokenizer
12 from nltk.tokenize import RegexpTokenizer
13 #Import Lemmatizer
14 from nltk.stem import WordNetLemmatizer
15 # Import stemmer.
16 from nltk.stem.porter import PorterStemmer
```

(You may need to run `nltk.download()` in a cell if you've never previously used it.)

Read in csv file, create DataFrame & check shape. We are starting out with 10,000 rows and 17 columns. Each row is a different product on Amazon.

```
1 df = pd.read_csv('amazon_co-ecommerce_sample.csv')
```

```
1 df.shape
```

```
(10000, 17)
```

I conducted some basic data cleaning that I won't go into detail about now, but you can read my post about EDA [here](#) if you want some tips.

In order to make the dataset more manageable for this example, I first dropped columns with too many nulls and then dropped any remaining rows with null values. I changed the `number_of_reviews` column type from object to integer and then created a new DataFrame using only the rows with no more than 1 review. My new shape is 3,705





Open in app

Get started

The following workflow is what I was taught to use and like using, but the steps are just general suggestions to get you started. Usually I have to modify and/or expand depending on the text format.

1. Remove HTML
2. Tokenization + Remove punctuation
3. Remove stop words
4. Lemmatization or Stemming

While cleaning this data I ran into a problem I had not encountered before, and learned a cool new trick from [geeksforgeeks.org](https://www.geeksforgeeks.org) to split a string from one column into multiple columns either on spaces or specified characters.

The column I am most interested in is `customer_reviews`, however, upon taking a closer look, it currently has the review title, rating, review date, customer name, and review all in one cell separated by `//`.

```
1 review_df['customer_reviews'][3]
```

```
"I love it // 5.0 // 22 July 2013 // By\n\n Lilla Lukacs\n\n on 22 July 2013 // I love it. Perfect with the earlier ordered locomotive. Again: I would recommend it to the masters of the topic. It's not just a toy."
```

Pandas `.str.split` method can be applied to a Series. First parameter is the repeated part of the string you want to split on, `n=` maximum number of separations and `expand=True` will split up the sections into new columns. I set the 4 new columns equal to a new variable called `reviews`.

```
1 #separate review column by // breaks
2 reviews = review_df['customer_reviews'].str.split("//", n=4, expand=True)
3 reviews.head()
```

	0	1	2	3	4
3	I love it	5.0	22 July 2013	By\n\n Lilla Lukacs\n\n on 22 July 2...	I love it. Perfect with the earlier ordered I...
8	Five Stars	5.0	23 Dec. 2015	By\n\n Mr. K. Waller\n\n on 23 Dec. ...	Xmas box for grandson,am sure it will be great
10	steaming good engine!	5.0	11 Jan. 2016	By\n\n Jeremy Joslin\n\n on 11 Jan. ...	Grandson loved it. Hauls good load
11	Five Stars	5.0	12 Nov. 2014	By\n\n alain HENAU\n\n on 12 Nov. 2...	Very good product. Thank you.
12	Five Stars	5.0	2 Oct. 2015	By\n\n Mark Ritzmann\n\n on 2 Oct. 2...	very good

Then you can rename the new 0, 1, 2, 3, 4 columns in the original `reviews_df` and drop the original messy column.

```
1 review_df["review_title"] = reviews[0]
2 review_df['rating'] = reviews[1]
3 review_df['review_date'] = reviews[2]
```





Open in app

Get started

review_title	rating	review_date	customer_name	review
I love it	5.0	22 July 2013	Lilla Lukacs	I love it. Perfect with the earlier ordered I...
Five Stars	5.0	23 Dec. 2015	Mr. K. Waller	Xmas box for grandson,am sure it will be great
steaming good engine!	5.0	11 Jan. 2016	Jeremy Joslin	Grandson loved it. Hauls good load
Five Stars	5.0	12 Nov. 2014	alain HENAU	Very good product. Thank you.
Five Stars	5.0	2 Oct. 2015	Mark Ritzmann	very good

I ran the same method over the new `customer_name` column to split on the `\n \n` and then dropped the first and last columns to leave just the actual customer name. There is a lot more we could do here if this were a longer article! Right off the bat, I can see the names and dates could still use some cleaning to put them in a uniform format.

**Removing HTML** is a step I did not do this time, however, if data is coming from a web scrape, it is a good idea to start with that. This is the function I would have used.

```

1 def remove_html(text):
2     soup = BeautifulSoup(text, 'lxml')
3     html_free = soup.get_text()
4     return html_free

```





Open in app

Get started

go, but I wanted to show the break down and make them easier to customize.

### Remove punctuation:

One way of doing this is by looping through the Series with list comprehension and keeping everything that is not in `string.punctuation`, a list of all punctuation we imported at the beginning with `import string`.

```
1 def remove_punctuation(text):
2     no_punct = "".join([c for c in text if c not in string.punctuation])
3     return no_punct
```

“join will join the list of letters back together as words where there are no spaces.

```
1 review_df['review'] = review_df['review'].apply(lambda x: remove_punctuation(x))
2 review_df['review'].head()

3      I love it Perfect with the earlier ordered lo...
8      Xmas box for grandsonam sure it will be great
10     Grandson loved it Hauls good load
11     Very good product Thank you
12     very good
```

If you scroll up you can see where this text previously had commas, periods, etc.

However, as you can see in the second line of output above, this method does not account for user typos. Customer had typed “grandson,am” which then became one word “grandsonam” once the comma was removed. I still think this is handy to know in case you ever need it though.

### Tokenize:

This breaks up the strings into a list of words or pieces based on a specified pattern using Regular Expressions aka RegEx. The pattern I chose to use this time (`r'\w+'`) also removes punctuation and is a better option for this data in particular. We can also add `.lower()` in the lambda function to make everything lowercase.

```
1 #Instantiate Tokenizer
2 tokenizer = RegexpTokenizer(r'\w+')

1 review_df['review'] = review_df['review'].apply(lambda x: tokenizer.tokenize(x.lower()))
2 review_df['review'].head(20)

3      [i, love, it, perfect, with, the, earlier, ord...
8      [xmas, box, for, grandson, am, sure, it, will,...
10     [grandson, loved, it, hauls, good, load]
11     [very, good, product, thank, you]
12     [very, good]
14     [great, quality, lots, of, detail, run, nic, a...
```





Open in app

Get started

Some other examples of RegEx are:

'\w+|\\$\d\.|+\S+' = splits up by spaces or by periods that are not attached to a digit

'\s+', gaps=True = grabs everything except spaces as a token

'[A-Z]\w+' = only words that begin with a capital letter.

### Remove stop words:

We imported a list of the most frequently used words from the NL Toolkit at the beginning with `from nltk.corpus import stopwords`. You can run `stopwords.words(insert language)` to get a full list for every language. There are 179 English words, including 'i', 'me', 'my', 'myself', 'we', 'you', 'he', 'his', for example. We usually want to remove these because they have low predictive power. There are occasions when you may want to keep them though. Such as, if your corpus is very small and removing stop words would decrease the total number of words by a large percent.

```
1 def remove_stopwords(text):
2     words = [w for w in text if w not in stopwords.words('english')]
3     return words

1 review_df['review'] = review_df['review'].apply(lambda x : remove_stopwords(x))
2 review_df['review'].head(10)

3     [love, perfect, earlier, ordered, locomotive, ...
8         [xmas, box, grandson, sure, great]
10        [grandson, loved, hauls, good, load]
11        [good, product, thank]
12        [good]
14     [great, quality, lots, detail, run, nic, smoot...
15     [fantastic, little, set, beautifully, made, de...
18        [lovely, quality, kato]
19     [second, bachmann, powerhauls, coming, take, r...
20     [excellent, ho, scale, diorama, piece, compris...
```

### Stemming & Lemmatizing:

Both tools shorten words back to their root form. Stemming is a little more aggressive. It cuts off prefixes and/or endings of words based on common ones. It can sometimes be helpful, but not always because often times the new word is so much a root that it loses its actual meaning. Lemmatizing, on the other hand, maps common words into one base. Unlike stemming though, it always still returns a proper word that can be found in the dictionary. I like to compare the two to see which one works better for what I need. I usually prefer Lemmatizer, but surprisingly, this time, Stemming seemed to have more of an affect.







Open in app

Get started

```

1 # Instantiate lemmatizer
2 lemmatizer = WordNetLemmatizer()
3
4 def word_lemmatizer(text):
5     lem_text = [lemmatizer.lemmatize(i) for i in text]
6     return lem_text

```

```

1 review_df['review'].apply(lambda x: word_lemmatizer(x))

```

```

3     [love, perfect, earlier, ordered, locomotive, ...
8         [xmas, box, grandson, sure, great]
10        [grandson, loved, haul, good, load]
11        [good, product, thank]
12        [good]
14     [great, quality, lot, detail, run, nic, smooth...
15     [fantastic, little, set, beautifully, made, de...
18        [lovely, quality, kato]
19     [second, bachmann, powerhauls, coming, take, r...
20     [excellent, ho, scale, diorama, piece, compris...

```

Lemmatizer: can barely even see a difference

You see more of a difference with Stemmer so I will keep that one in place. Since this is the final step, I added "`.join()`" to the function to join the lists of words back together.

```

1 #Instantiate Stemmer
2 stemmer = PorterStemmer()

```

```

1 def word_stemmer(text):
2     stem_text = " ".join([stemmer.stem(i) for i in text])
3     return stem_text

```

```

1 review_df['review'] = review_df['review'].apply(lambda x: word_stemmer(x))

```

```

1 review_df['review']

```

```

3     love perfect earlier order locomot would recom...
8         xma box grandson sure great
10        grandson love haul good load
11        good product thank
12        good
14     great qualiti lot detail run nic smooth great ...
15     fantast littl set beauti made decent price
18        love qualiti kato
19     second bachmann powerhaul come take real train...
20     excel ho scale diorama piec compris white encl...

```





Open in app

Get started

<https://pixabay.com/photos/thank-you-neon-lights-neon-362164/>

## 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.](#)

Get this newsletter

