

# Bag-of-words

SENTIMENT ANALYSIS IN PYTHON



**Violeta Misheva**  
Data Scientist

# What is a bag-of-words (BOW) ?

- Describes the occurrence of words within a document or a collection of documents (corpus)
- Builds a vocabulary of the words and a measure of their presence

# Amazon product reviews

	score	review
0	1	Stuning even for the non-gamer: This sound tr...
1	1	The best soundtrack ever to anything.: I'm re...
2	1	Amazing!: This soundtrack is my favorite musi...
3	1	Excellent Soundtrack: I truly like this sound...
4	1	Remember, Pull Your Jaw Off The Floor After H...
5	1	an absolute masterpiece: I am quite sure any ...
6	0	Buyer beware: This is a self-published book, ...
7	1	Glorious story: I loved Whisper of the wicked...
8	1	A FIVE STAR BOOK: I just finished reading Whi...
9	1	Whispers of the Wicked Saints: This was a eas...

# Sentiment analysis with BOW: Example

This is the best book ever. I loved the book and highly recommend it!!!

```
{ 'This': 1, 'is': 1, 'the': 2, 'best': 3, 'book': 2,  
  'ever': 1, 'I': 1, 'loved': 1, 'and': 1, 'highly': 1,  
  'recommend': 1, 'it': 1 }
```

- Lose word order and grammar rules!

# BOW end result

- The output will look something like this:

	10	100	12	15	1984	20	30	40	451	50	...	wrong	wrote	year	years	yes	yet	you	young	your	yourself
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	1	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	2	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	3	0	1	0

# CountVectorizer function

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
```

```
vect = CountVectorizer(max_features=1000)
vect.fit(data.review)
X = vect.transform(data.review)
```

# CountVectorizer output

X

```
<10000x1000 sparse matrix of type '<class 'numpy.int64'>'
  with 406668 stored elements in Compressed Sparse Row format>
```

# Transforming the vectorizer

```
# Transform to an array  
my_array = X.toarray()
```

```
# Transform back to a dataframe, assign column names  
X_df = pd.DataFrame(my_array, columns=vect.get_feature_names())
```



# Let's practice!

SENTIMENT ANALYSIS IN PYTHON

# Getting granular with n-grams

SENTIMENT ANALYSIS IN PYTHON



**Violeta Misheva**  
Data Scientist

# Context matters

I am *happy*, not sad.

I am *sad*, not happy.

- Putting 'not' in front of a word (negation) is one example of how context matters.

# Capturing context with a BOW

- **Unigrams** : single tokens
- **Bigrams**: pairs of tokens
- **Trigrams**: triples of tokens
- **n-grams**: sequence of n-tokens

# Capturing context with BOW

The weather today is wonderful.

- **Unigrams** : { The, weather, today, is wonderful }
- **Bigrams**: {The weather, weather today, today is, is wonderful}
- **Trigrams**: {The weather today, weather today is, today is wonderful}

# n-grams with the CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
vect = CountVectorizer(ngram_range=(min_n, max_n))
```

```
# Only unigrams  
ngram_range=(1, 1)
```

```
# Uni- and bigrams  
ngram_range=(1, 2)
```

# What is the best n?

## Longer sequence of tokens

- Results in more features
- Higher precision of machine learning models
- Risk of overfitting

# Specifying vocabulary size

```
CountVectorizer(max_feature, max_df, min_df)
```

- **max\_features**: if specified, it will include only the top most frequent words in the vocabulary
  - If `max_features = None`, all words will be included
- **max\_df**: ignore terms with higher than specified frequency
  - If it is set to integer, then absolute count; if a float, then it is a proportion
  - Default is 1.0, which means it does not ignore any terms
- **min\_df**: ignore terms with lower than specified frequency
  - If it is set to integer, then absolute count; if a float, then it is a proportion
  - Default is 1.0, which means it does not ignore any terms



# Let's practice!

SENTIMENT ANALYSIS IN PYTHON

# Build new features from text

SENTIMENT ANALYSIS IN PYTHON



**Violeta Misheva**  
Data Scientist

# Goal of the video

Goal : Enrich the existing dataset with features related to the text column (capturing the sentiment)

# Product reviews data

```
reviews.head()
```

	score	review
0	1	Stuning even for the non-gamer. This sound tr...
1	1	The best soundtrack ever to anything.: I'm re...
2	1	Amazing!: This soundtrack is my favorite musi...
3	1	Excellent Soundtrack: I truly like this sound...
4	1	Remember, Pull Your Jaw Off The Floor After H...

# Features from the review column

- How long is each review?
- How many sentences does it contain?
- What parts of speech are involved?
- How many punctuation marks?

# Tokenizing a string

```
from nltk import word_tokenize
```

```
anna_k = 'Happy families are all alike, every unhappy family is unhappy in its own way.'
```

```
word_tokenize(anna_k)
```

```
['Happy', 'families', 'are', 'all', 'alike', ',',  
 'every', 'unhappy', 'family', 'is', 'unhappy', 'in',  
 'its', 'own', 'way', '.']
```

# Tokens from a column

```
# General form of list comprehension  
[expression for item in iterable]
```

```
word_tokens = [word_tokenize(review) for review in reviews.review]  
type(word_tokens)
```

```
list
```

```
type(word_tokens[0])
```

```
list
```

# Tokens from a column

```
len_tokens = []  
  
# Iterate over the word_tokens list  
for i in range(len(word_tokens)):  
    len_tokens.append(len(word_tokens[i]))
```

```
# Create a new feature for the length of each review  
reviews['n_tokens'] = len_tokens
```



# Dealing with punctuation

- We did not address it but you can exclude it
- A feature that measures the number of punctuation signs
  - A review with many punctuation signs could signal a very emotionally charged opinion

# Reviews with a feature for the length

```
reviews.head()
```

	score	review	n_tokens
0	1	Stuning even for the non-gamer: This sound tr...	87
1	1	The best soundtrack ever to anything.: I'm re...	109
2	1	Amazing!: This soundtrack is my favorite musi...	165
3	1	Excellent Soundtrack: I truly like this sound...	145
4	1	Remember, Pull Your Jaw Off The Floor After H...	109

# Let's practice!

SENTIMENT ANALYSIS IN PYTHON

# Can you guess the language?

SENTIMENT ANALYSIS IN PYTHON



**Violeta Misheva**  
Data Scientist

# Language of a string in Python

```
from langdetect import detect_langs  
foreign = 'Este libro ha sido uno de los mejores libros que he leído.'
```

```
detect_langs(foreign)
```

```
[es:0.9999945352697024]
```

# Language of a column

- Problem: Detect the language of each of the strings and capture the most likely language in a new column

```
from langdetect import detect_langs
reviews = pd.read_csv('product_reviews.csv')
```

```
reviews.head()
```

	score	review
0	1	Stuning even for the non-gamer. This sound tr...
1	1	The best soundtrack ever to anything.: I'm re...
2	1	Amazing!: This soundtrack is my favorite musi...
3	1	Excellent Soundtrack: I truly like this sound...
4	1	Remember, Pull Your Jaw Off The Floor After H...

# Building a feature for the language

```
languages = []
```

```
for row in range(len(reviews)):
    languages.append(detect_langs(reviews.iloc[row, 1]))
```

```
languages
[it:0.9999982541301151],
[es:0.9999954153640488],
[es:0.7142833997345875, en:0.2857160465706441],
[es:0.9999942365605781],
[es:0.999997956049055] ...
```

# Building a feature for the language

```
# Transform the first list to a string and split on a colon  
str(languages[0]).split(':')  
['[es', '0.9999954153640488]']
```

```
str(languages[0]).split(':')[0]  
'[es'
```

```
str(languages[0]).split(':')[0][1:]  
'es'
```



# Building a feature for the language

```
languages = [str(lang).split(':')[0][1:] for lang in languages]
```

```
reviews['language'] = languages
```

# Let's practice!

SENTIMENT ANALYSIS IN PYTHON