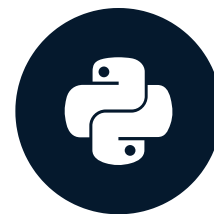# Classifying fake news using supervised learning with NLP

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

What is supervised learning?

- Form of machine learning
  - Problem has predefined training data

  - This data has a label (or outcome) you want the model to learn

  - Classification problem

  - Goal: Make good hypotheses about the species based on geometric features

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	I. setosa
7.0	3.2	4.77	1.4	I.versicolor
6.3	3.3	6.0	2.5	I.virginica

Supervised learning with NLP

- Need to use language instead of geometric features

- `scikit-learn` : Powerful open-source library

- How to create supervised learning data from text?
  - Use bag-of-words models or tf-idf as features

IMDB Movie Dataset

Plot	Sci-Fi	Action
In a post-apocalyptic world in human decay, a ...	1	0
Mohei is a wandering swordsman. He arrives in ...	0	1
#137 is a SCI/FI thriller about a girl, Marla,...	1	0

- Goal: Predict movie genre based on plot summary

- Categorical features generated using preprocessing

Supervised learning steps

- Collect and preprocess our data

- Determine a label (Example: Movie genre)

- Split data into training and test sets

- Extract features from the text to help predict the label
  - Bag-of-words vector built into `scikit-learn`

- Evaluate trained model using the test set

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Building word count vectors with scikit-learn

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON
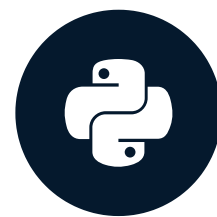
Katharine Jarmul
Founder, kjamistan

Predicting movie genre

- Dataset consisting of movie plots and corresponding genre

- Goal: Create bag-of-word vectors for the movie plots
  - Can we predict genre based on the words used in the plot summary?

Count Vectorizer with Python

```python
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer
df = ... # Load data into DataFrame
y = df['Sci-Fi']
X_train, X_test, y_train, y_test = train_test_split(
                                    df['plot'], y,
                                    test_size=0.33,
                                    random_state=53)

count_vectorizer = CountVectorizer(stop_words='english')
count_train = count_vectorizer.fit_transform(X_train.values)
count_test = count_vectorizer.transform(X_test.values)
```
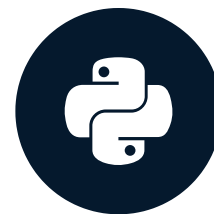
Let's practice!

datacamp

Training and testing a classification model with scikit-learn

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

datacamp

Naive Bayes classifier

- Naive Bayes Model
  - Commonly used for testing NLP classification problems
  - Basis in probability

- Given a particular piece of data, how likely is a particular outcome?

- Examples:
  - If the plot has a spaceship, how likely is it to be sci-fi?
  - Given a spaceship **and** an alien, how likely **now** is it sci-fi?

- Each word from `CountVectorizer` acts as a feature

- Naive Bayes: Simple and effective

Naive Bayes with scikit-learn

```python
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
nb_classifier = MultinomialNB()


nb_classifier.fit(count_train, y_train)
pred = nb_classifier.predict(count_test)
metrics.accuracy_score(y_test, pred)
```

```
0.85841849389820424
```

Confusion matrix

```
metrics.confusion_matrix(y_test, pred, labels=[0,1])
```
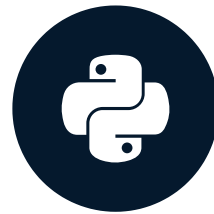
```
array([[6410,  563],
       [ 864, 2242]])
```

	Action	Sci-Fi
Action	6410	563
Sci-Fi	864	2242

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Simple NLP, complex problems

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

datacamp

Translation

source:

Sentiment analysis

Ex. contexts in r/sports		Ex. contexts in r/TwoX
"big men are very <u>soft</u>"	soft	"some <u>soft</u> pajamas"
"freakin raging <u>animal</u>"	animal	"stuffed <u>animal</u>"
"went from the <u>ladies</u> tees"	ladies	"lovely <u>ladies</u>"
"two <u>dogs</u> fighting"	dogs	"hiking with the <u>dogs</u>"
"being able to <u>hit</u>"	hit	"it didn't really <u>hit</u> me"
"insanely <u>difficult</u> saves"	difficult	"a <u>difficult</u> time"
"amazing <u>shot</u>"	shot	"totally <u>shot</u> me down"
"he is still <u>crazy</u> good"	crazy	"overreacting <u>crazy</u> woman"
"his stats are <u>insane</u>"	insane	"people are just <u>insane</u>"

more positive in r/sports, more negative in r/TwoX

more positive in r/TwoX, more negative in r/sports

(source: https://nlp.stanford.edu/projects/socialsent/)

Language biases

Englisch | Rumänisch | Türkisch | Sprache erkennen | ▼   Türkisch | Englisch | Deutsch | ▼   **Übersetzen**

She's a professor. He's a babysitter.   O bir profesör. O bir bebek bakıcısı.

37/5000

Google Übersetzer

Englisch | Rumänisch | **Türkisch** | Sprache erkennen | ▼   Türkisch | **Englisch** | Deutsch | ▼   **Übersetzen**

O bir profesör. O bir bebek bakıcısı.   He's a professor. She's a babysitter.

37/5000

(related talk: **https://www.youtube.com/watch?
v=j7FwpZB1hWc**)

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Named Entity Recognition

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON
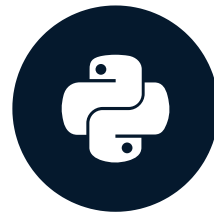
Katharine Jarmul
Founder, kjamistan

What is Named Entity Recognition?

- NLP task to identify important named entities in the text
  - People, places, organizations

  - Dates, states, works of art

  - ... and other categories!

- Can be used alongside topic identification
  - ... or on its own!

- Who? What? When? Where?

Example of NER

In **1917**, **Einstein** applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the **United States** when **Adolf Hitler** came to power in **1933** and did not go back to **Germany**, where he had been a professor at the **Berlin Academy of Sciences**. He settled in the **U.S.**, becoming an American citizen in **1940**. On the eve of World War II, he endorsed a letter to President **Franklin D. Roosevelt** alerting him to the potential development of "extremely powerful bombs of a new type" and recommending that the **U.S.** begin similar research. This eventually led to what would become the **Manhattan** Project. Einstein supported defending the Allied forces, but largely denounced using the new discovery of nuclear fission as a weapon. Later, with the British philosopher **Bertrand Russell**, **Einstein** signed the **Russell–Einstein Manifesto**, which highlighted the danger of nuclear weapons. Einstein was affiliated with the **Institute for Advanced Study** in **Princeton**, **New Jersey**, until his death in **1955**.

Tag colours:
LOCATION **TIME** **PERSON** **ORGANIZATION** **MONEY** **PERCENT** **DATE**

(Source: Europeana Newspapers (**http://www.europeana-newspapers.eu**))

nltk and the Stanford CoreNLP Library

- The Stanford CoreNLP library:
  - Integrated into Python via `nltk`

  - Java based

  - Support for NER as well as coreference and dependency trees

Using nltk for Named Entity Recognition

```python
import nltk
sentence = '''In New York, I like to ride the Metro to
              visit MOMA and some restaurants rated
              well by Ruth Reichl.'''
tokenized_sent = nltk.word_tokenize(sentence)
tagged_sent = nltk.pos_tag(tokenized_sent)
tagged_sent[:3]
```

```
[('In', 'IN'), ('New', 'NNP'), ('York', 'NNP')]
```

```
print(nltk.ne_chunk(tagged_sent))
```

```
(S
  In/IN
  (GPE New/NNP York/NNP)
  ,/,
  I/PRP
  like/VBP
  to/TO
  ride/VB
  the/DT
  (ORGANIZATION Metro/NNP)
  to/TO
  visit/VB
  (ORGANIZATION MOMA/NNP)
  and/CC
  some/DT
  restaurants/NNS
  rated/VBN
  well/RB
  by/IN
  (PERSON Ruth/NNP Reichl/NNP)
  ./.)
```

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Introduction to SpaCy

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

datacamp

What is SpaCy?

- NLP library similar to `gensim` , with different implementations

- Focus on creating NLP pipelines to generate models and corpora

- Open-source, with extra libraries and tools
  - Displacy

Displacy entity recognition visualizer

In `New York GPE` , I like to ride the Metro to visit `MOMA ORG` and some restaurants rated well by `Ruth Reichl PERSON`

(source: https://demos.explosion.ai/displacy-ent/)

```
import spacy
nlp = spacy.load('en')
nlp.entity
```

```
<spacy.pipeline.EntityRecognizer at 0x7f76b75e68b8>
```

```
doc = nlp("""Berlin is the capital of Germany;
              and the residence of Chancellor Angela Merkel.""")
doc.ents
```

```
(Berlin, Germany, Angela Merkel)
```

```
print(doc.ents[0], doc.ents[0].label_)
```

```
Berlin GPE
```

Why use SpaCy for NER?

- Easy pipeline creation

- Different entity types compared to `nltk`

- Informal language corpora
  - Easily find entities in Tweets and chat messages

- Quickly growing!

Let's practice!

Multilingual NER with polyglot

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON
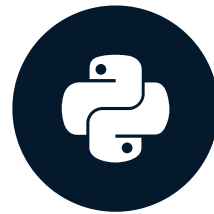
Katharine Jarmul
Founder, kjamistan

What is polyglot?

- NLP library which uses word vectors

- Why `polyglot`?
  - Vectors for many different languages
  - More than 130!

```
which          ويكه
India          يندبا
beat           بيت
Bermuda        بيرمودا
in             ين
Port           بورت
of             وف
Spain          سبابن
in             ين
2007
,
which          ويكه
was            واس
equalled       يكاللبد
five           فيفي
days           دابس
ago            اغو
by             بي
South          سووث
Africa         افريكا
in             ين
their          ثبر
victory        فيكتوري
over           وفبر
West           ويست
Indies         يندبيس
in             ين
Sydney         سبدني
.
```

Spanish NER with polyglot

```python
from polyglot.text import Text
?ext = """"El presidente de la Generalitat de Cataluña,
                Carles Puigdemont, ha afirmado hoy a la alcaldesa
                de Madrid, Manuela Carmena, que en su etapa de
                alcalde de Girona (de julio de 2011 a enero de 2016)
                hizo una gran promoción de Madrid."""
ptext = Text(text)
ptext.entities
```
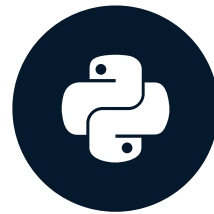
```
[I-ORG(['Generalitat', 'de']),
 I-LOC(['Generalitat', 'de', 'Cataluña']),
 I-PER(['Carles', 'Puigdemont']),
 I-LOC(['Madrid']),
 I-PER(['Manuela', 'Carmena']),
 I-LOC(['Girona']),
 I-LOC(['Madrid'])]
```

Let's practice!

Bag-of-words

- Basic method for finding topics in a text

- Need to first create tokens using tokenization

- ... and then count up all the tokens

- The more frequent a word, the more important it might be

- Can be a great way to determine the significant words in a text

Bag-of-words example

- Text: "The cat is in the box. The cat likes the box. The box is over the cat."

- Bag of words (stripped punctuation):
  - "The": 3, "box": 3
  - "cat": 3, "the": 3
  - "is": 2
  - "in": 1, "likes": 1, "over": 1

Bag-of-words in Python

```python
from nltk.tokenize import word_tokenize
from collections import Counter
Counter(word_tokenize("""The cat is in the box. The cat likes the box.
                The box is over the cat."""))
```

```
Counter({'.': 3,
        'The': 3,
        'box': 3,
        'cat': 3,
        'in': 1,
        ...
        'the': 3})
```

```python
counter.most_common(2)
```

```
[('The', 3), ('box', 3)]
```

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

datacamp

Simple text preprocessing

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

datacamp
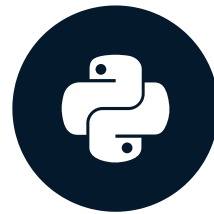
Why preprocess?

- Helps make for better input data
  - When performing machine learning or other statistical methods

- Examples:
  - Tokenization to create a bag of words

  - Lowercasing words

- Lemmatization/Stemming
  - Shorten words to their root stems

- Removing stop words, punctuation, or unwanted tokens

- Good to experiment with different approaches

Preprocessing example

- Input text: Cats, dogs and birds are common pets. So are fish.

- Output tokens: cat, dog, bird, common, pet, fish

Text preprocessing with Python

```python
from ntlk.corpus import stopwords
text = """The cat is in the box. The cat likes the box.
             The box is over the cat."""
tokens = [w for w in word_tokenize(text.lower())
                if w.isalpha()]
 no_stops = [t for t in tokens
                    if t not in stopwords.words('english')]
Counter(no_stops).most_common(2)
```

```
[('cat', 3), ('box', 3)]
```

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Introduction to gensim

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
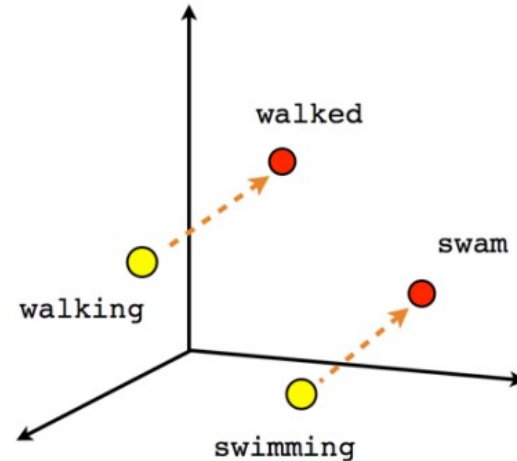Founder, kjamistan

What is gensim?

- Popular open-source NLP library

- Uses top academic models to perform complex tasks
  - Building document or word vectors

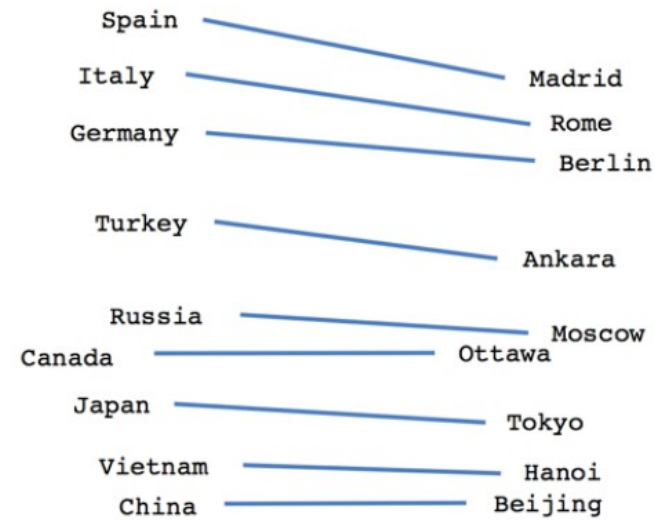 - Performing topic identification and document comparison
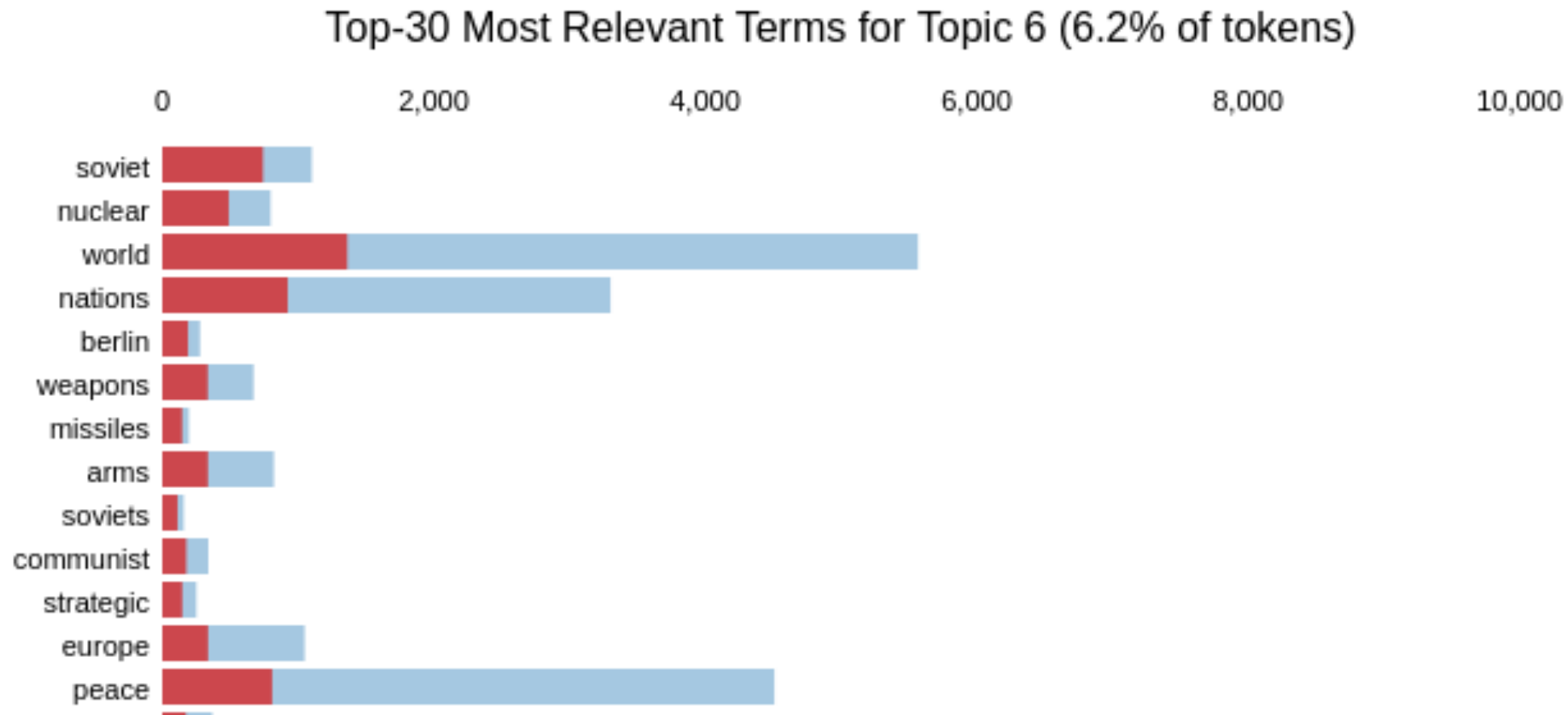
What is a word vector?
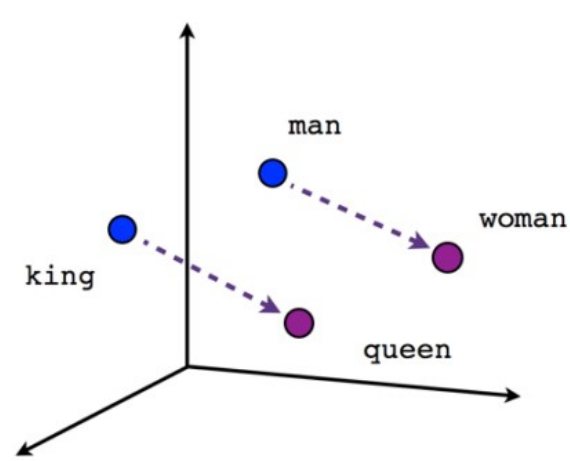
Male-Female

Verb tense

Country-Capital

Gensim example

Top-30 Most Relevant Terms for Topic 6 (6.2% of tokens)

(Source: **http://tlfvincent.github.io/2015/10/23/presidential-speech-topics**)

```python
from gensim.corpora.dictionary import Dictionary
from nltk.tokenize import word_tokenize
my_documents = ['The movie was about a spaceship and aliens.',
                'I really liked the movie!',
                'Awesome action scenes, but boring characters.',
                'The movie was awful! I hate alien films.',
                'Space is cool! I liked the movie.',
                'More space films, please!',]
```

```python
tokenized_docs = [word_tokenize(doc.lower())
                  for doc in my_documents]
dictionary = Dictionary(tokenized_docs)
dictionary.token2id
```

```
{'!': 11,
 ',': 17,
 '.': 7,
 'a': 2,
 'about': 4,
...}
```

Creating a gensim corpus

```
corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]
corpus
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1)],
 [(0, 1), (1, 1), (9, 1), (10, 1), (11, 1), (12, 1)],
...]
```
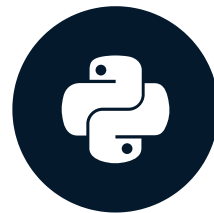
- `gensim` models can be easily saved, updated, and reused

- Our dictionary can also be updated

- This more advanced and feature rich bag-of-words can be used in future exercises

Let's practice!

datacamp

Tf-idf with gensim

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

What is tf-idf?

- Term frequency - inverse document frequency

- Allows you to determine the most important words in each document

- Each corpus may have shared words beyond just stopwords

- These words should be down-weighted in importance

- Example from astronomy: "Sky"

- Ensures most common words don't show up as key words

- Keeps document specific frequent words weighted high

Tf-idf formula

$$w_{i,j} = tf_{i,j} * \log(\frac{N}{df_i})$$

$w_{i,j} = \text{tf-idf weight for token } i \text{ in document } j$

$tf_{i,j} = \text{number of occurences of token } i \text{ in document } j$

$df_i = \text{number of documents that contain token } i$

$N = \text{total number of documents}$

Tf-idf with gensim

```python
from gensim.models.tfidfmodel import TfidfModel
tfidf = TfidfModel(corpus)
tfidf[corpus[1]]
```
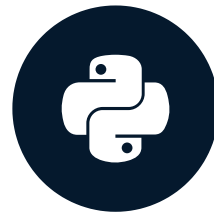
```
[(0, 0.1746298276735174),
 (1, 0.1746298276735174),
 (9, 0.29853166221463673),
 (10, 0.7716931521027908),
...
]
```

Let's practice!

Introduction to regular expressions

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

What is Natural Language Processing?

- Field of study focused on making sense of language
  - Using statistics and computers

- You will learn the basics of NLP
  - Topic identification

  - Text classification

- NLP applications include:
  - Chatbots

  - Translation

  - Sentiment analysis

  - ... and many more!

What exactly are regular expressions?

- Strings with a special syntax

- Allow us to match patterns in other strings

- Applications of regular expressions:

→ Find all web links in a document

→ Parse email addresses

→ Remove/replace unwanted characters

```python
import re
re.match('abc', 'abcdef')
```

```
<_sre.SRE_Match object; span=(0, 3), match='abc'>
```

```python
word_regex = '\w+'
re.match(word_regex,
        'hi there!')
```

```
<_sre.SRE_Match object; span=(0, 2), match='hi'>
```

Common regex patterns

pattern	matches	example
\w+	word	'Magic'

Common regex patterns (2)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9

Common regex patterns (3)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '

Common regex patterns (4)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'

Common regex patterns (5)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'
+ or *	greedy match	'aaaaaa'

Common regex patterns (6)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'
+ or *	greedy match	'aaaaaa'
\S	**not** space	'no_spaces'

Common regex patterns (7)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'
+ or *	greedy match	'aaaaaa'
\S	**not** space	'no_spaces'
[a-z]	lowercase group	'abcdefg'

Python's re module

- `re` module

- `split` : split a string on regex

- `findall` : find all patterns in a string

- `search` : search for a pattern

- `match` : match an entire string or substring based on a pattern

- Pattern first, and the string second

- May return an iterator, string, or match object
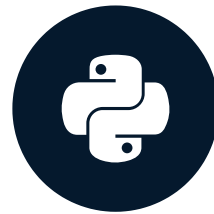
```
re.split('\s+', 'Split on spaces.')
```

```
['Split', 'on', 'spaces.']
```

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

datacamp

Introduction to tokenization

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

What is tokenization?

- Turning a string or document into **tokens** (smaller chunks)

- One step in preparing a text for NLP

- Many different theories and rules

- You can create your own rules using regular expressions

- Some examples:
  - Breaking out words or sentences

  - Separating punctuation

  - Separating all hashtags in a tweet

nltk library

- `nltk` : natural language toolkit

```python
from nltk.tokenize import word_tokenize
word_tokenize("Hi there!")
```

```
['Hi', 'there', '!']
```

Why tokenize?

- Easier to map part of speech

- Matching common words

- Removing unwanted tokens

- "I don't like Sam's shoes."

- "I", "do", "n't", "like", "Sam", "'s", "shoes", "."

Other nltk tokenizers

- `sent_tokenize` : tokenize a document into sentences

- `regexp_tokenize` : tokenize a string or document based on a regular expression pattern

- `TweetTokenizer` : special class just for tweet tokenization, allowing you to separate hashtags, mentions and lots of exclamation points!!!

More regex practice

- Difference between `re.search()` and `re.match()`

```python
import re
re.match('abc', 'abcde')
```

```
<_sre.SRE_Match object; span=(0, 3), match='abc'>
```

```python
re.search('abc', 'abcde')
```

```
<_sre.SRE_Match object; span=(0, 3), match='abc'>
```
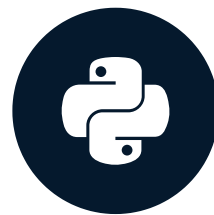
```python
re.match('cd', 'abcde')
re.search('cd', 'abcde')
```

```
<_sre.SRE_Match object; span=(2, 4), match='cd'>
```

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Advanced tokenization with regex

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

Regex groups using or "|"

- OR is represented using |

- You can define a group using ()

- You can define explicit character ranges using []

```python
import re
 match_digits_and_words = ('(\d+|\w+)')
re.findall(match_digits_and_words, 'He has 11 cats.')
```

```
['He', 'has', '11', 'cats']
```

Regex ranges and groups

pattern	matches	example
[A-Za-z]+	upper and lowercase English alphabet	'ABCDEFghijk'
[0-9]	numbers from 0 to 9	9
[A-Za-z\-\.]+	upper and lowercase English alphabet, - and .	'My-Website.com'
(a-z)	a, - and z	'a-z'
(\s+l,)	spaces or a comma	', '

Character range with `re.match()`

```python
import re
my_str = 'match lowercase spaces nums like 12, but no commas'
re.match('[a-z0-9 ]+', my_str)
```

```
<_sre.SRE_Match object;
        span=(0, 42), match='match lowercase spaces nums like 12'>
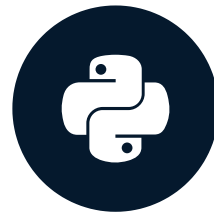```

# Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

# Charting word length with nltk

## INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

**Katharine Jarmul**
Founder, kjamistan

datacamp

# Getting started with matplotlib

- Charting library used by many open source Python projects

- Straightforward functionality with lots of options
  - Histograms

  - Bar charts

  - Line charts

  - Scatter plots

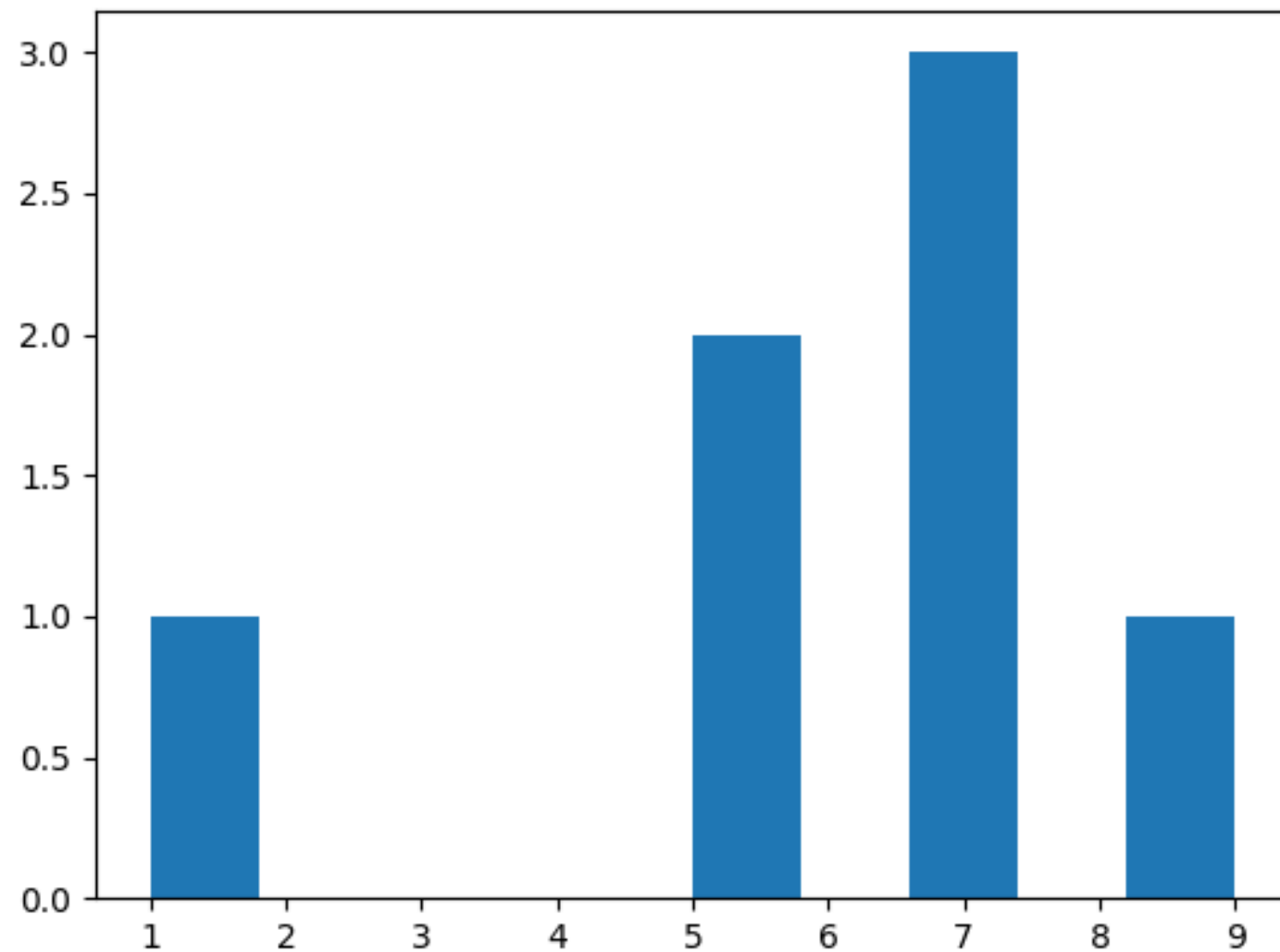- ... and also advanced functionality like 3D graphs and animations!

# Plotting a histogram with matplotlib

```python
from matplotlib import pyplot as plt
plt.hist([1, 5, 5, 7, 7, 7, 9])
```

```
(array([ 1., 0., 0., 0., 0., 2., 0., 3., 0., 1.]),
 array([ 1., 1.8, 2.6, 3.4, 4.2, 5., 5.8, 6.6, 7.4, 8.2, 9.]),
        <a list of 10 Patch objects>)
```

```python
plt.show()
```
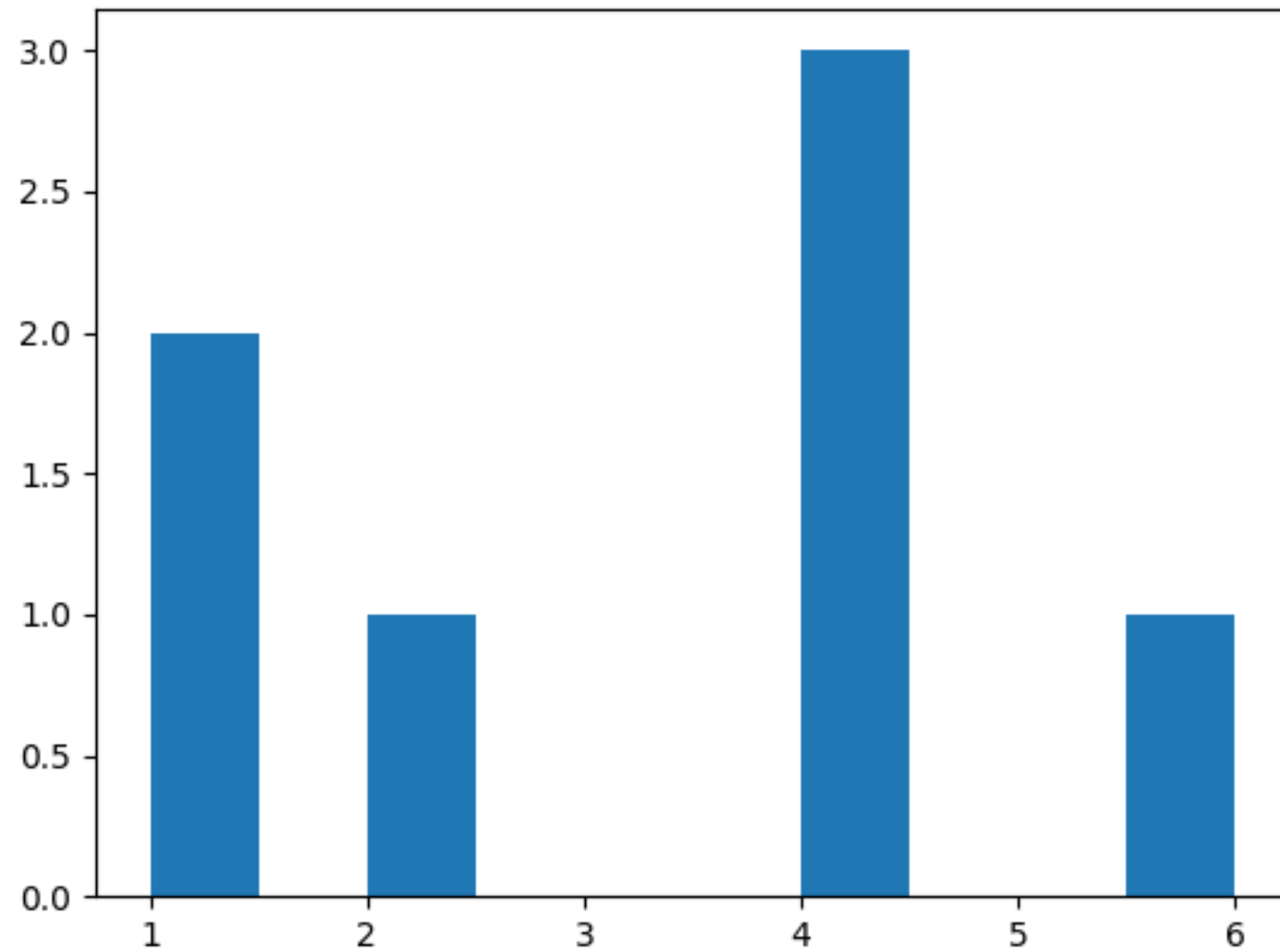
Generated histogram

Combining NLP data extraction with plotting

```python
from matplotlib import pyplot as plt
from nltk.tokenize import word_tokenize
words = word_tokenize("This is a pretty cool tool!")
word_lengths = [len(w) for w in words]
plt.hist(word_lengths)
```

```
(array([ 2., 0., 1., 0., 0., 0., 3., 0., 0., 1.]),
 array([ 1., 1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6.]),
        <a list of 10 Patch objects>)
```

```python
plt.show()
```

Word length histogram

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON