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

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

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

```
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 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

```
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()`

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

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

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

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

```
re.match('cd', 'abcde')
re.search('cd', 'abcde')
```

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

Let's practice!

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 `[]`

```
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

<table>
<tr><th>pattern</th><th>matches</th><th>example</th></tr>
<tr><td>[A-Za-z]+</td><td>upper and lowercase English alphabet</td><td>'ABCDEFghijk'</td></tr>
<tr><td>[0-9]</td><td>numbers from 0 to 9</td><td>9</td></tr>
<tr><td>[A-Za-z\-\.]+</td><td>upper and lowercase English alphabet, - and .</td><td>'My-Website.com'</td></tr>
<tr><td>(a-z)</td><td>a, - and z</td><td>'a-z'</td></tr>
<tr><td>(\s+|,)</td><td>spaces or a comma</td><td>', '</td></tr>
</table>

Character range with `re.match()`

```
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

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!

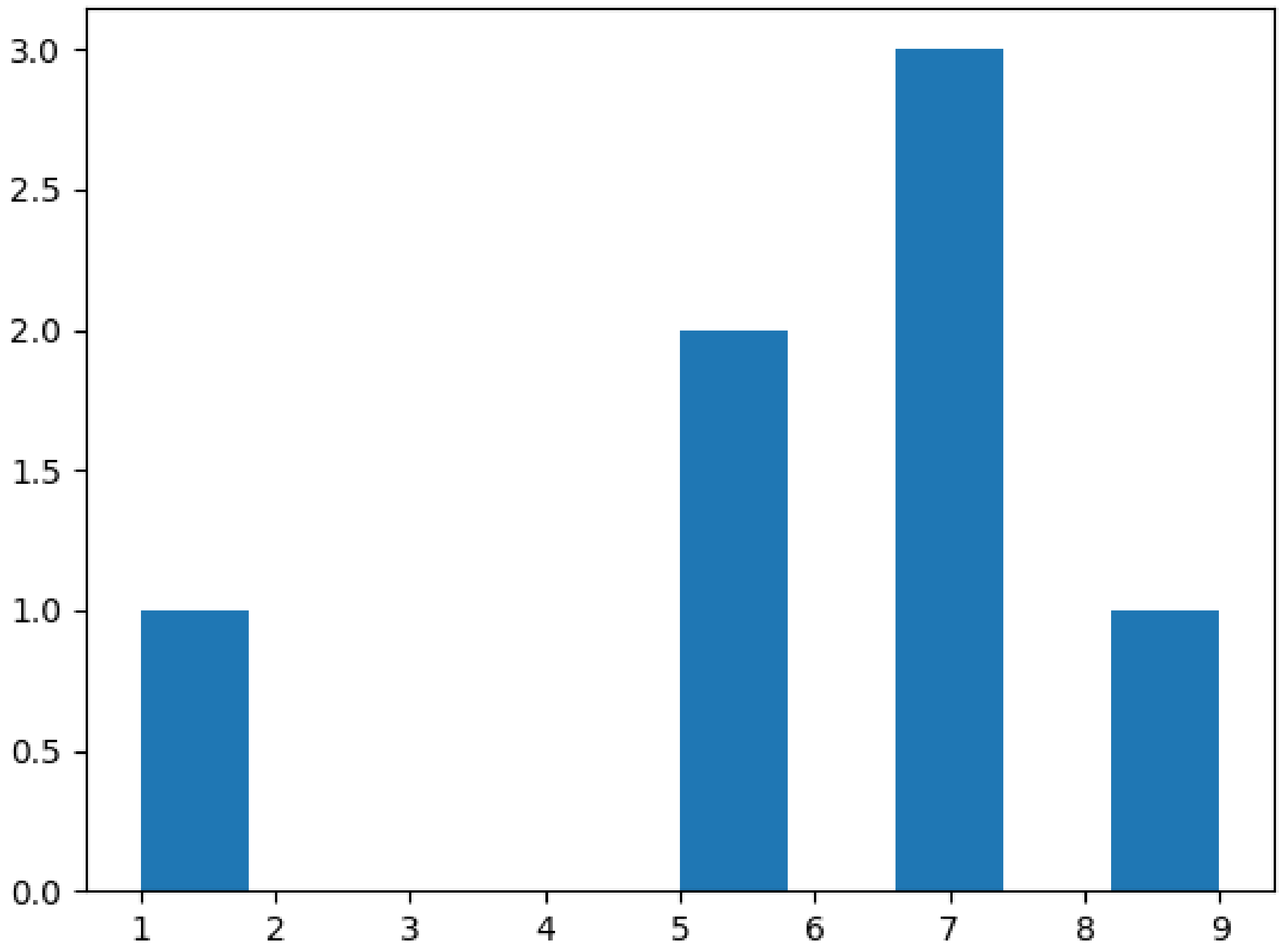
Plotting a histogram with matplotlib

```
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>)
```

```
plt.show()
```

Generated histogram



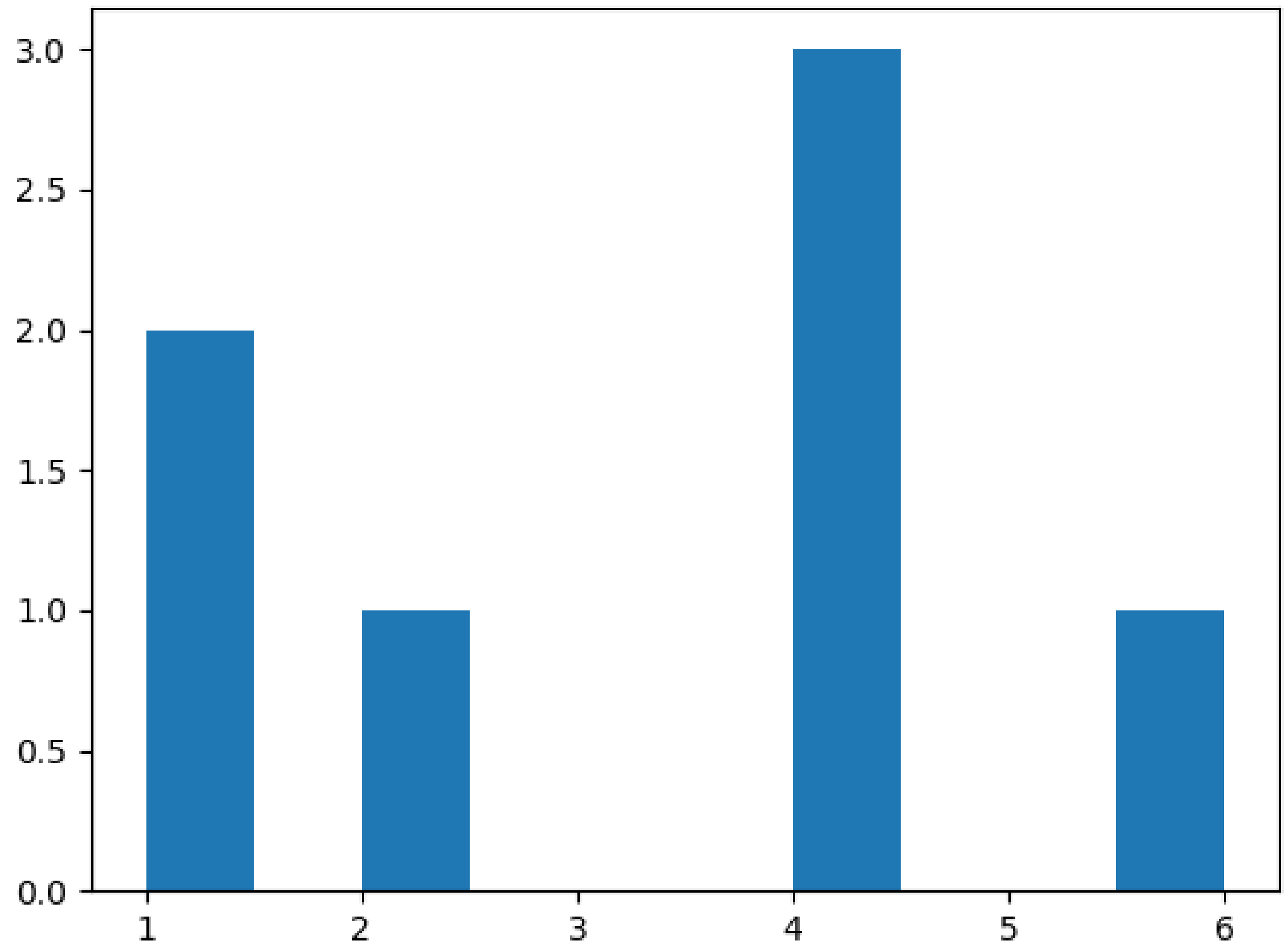
Combining NLP data extraction with plotting

```
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>)
```

```
plt.show()
```

Word length histogram



Let's practice!

Word counts with bag-of-words

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

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

```
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})
```

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

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

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Simple text preprocessing

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

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

```
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 gensim

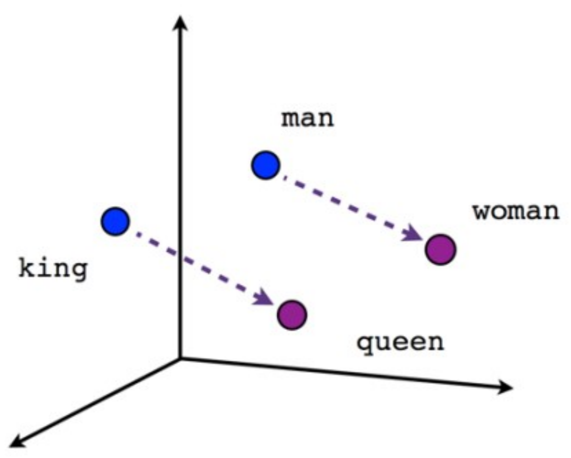
INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

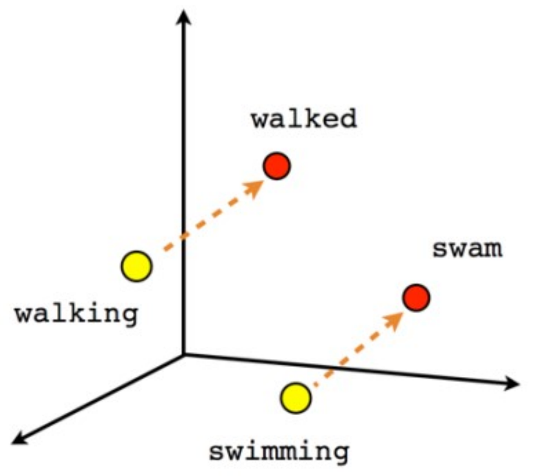
What is gensim?

- Popular open-source NLP library
- Uses top academic models to perform complex tasks
  - Building document or word vectors
  - Performing topic identification and document comparison

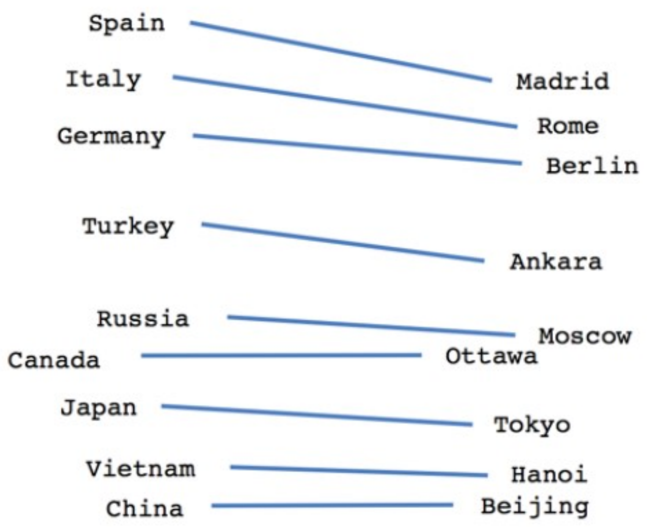
What is a word vector?



Male-Female

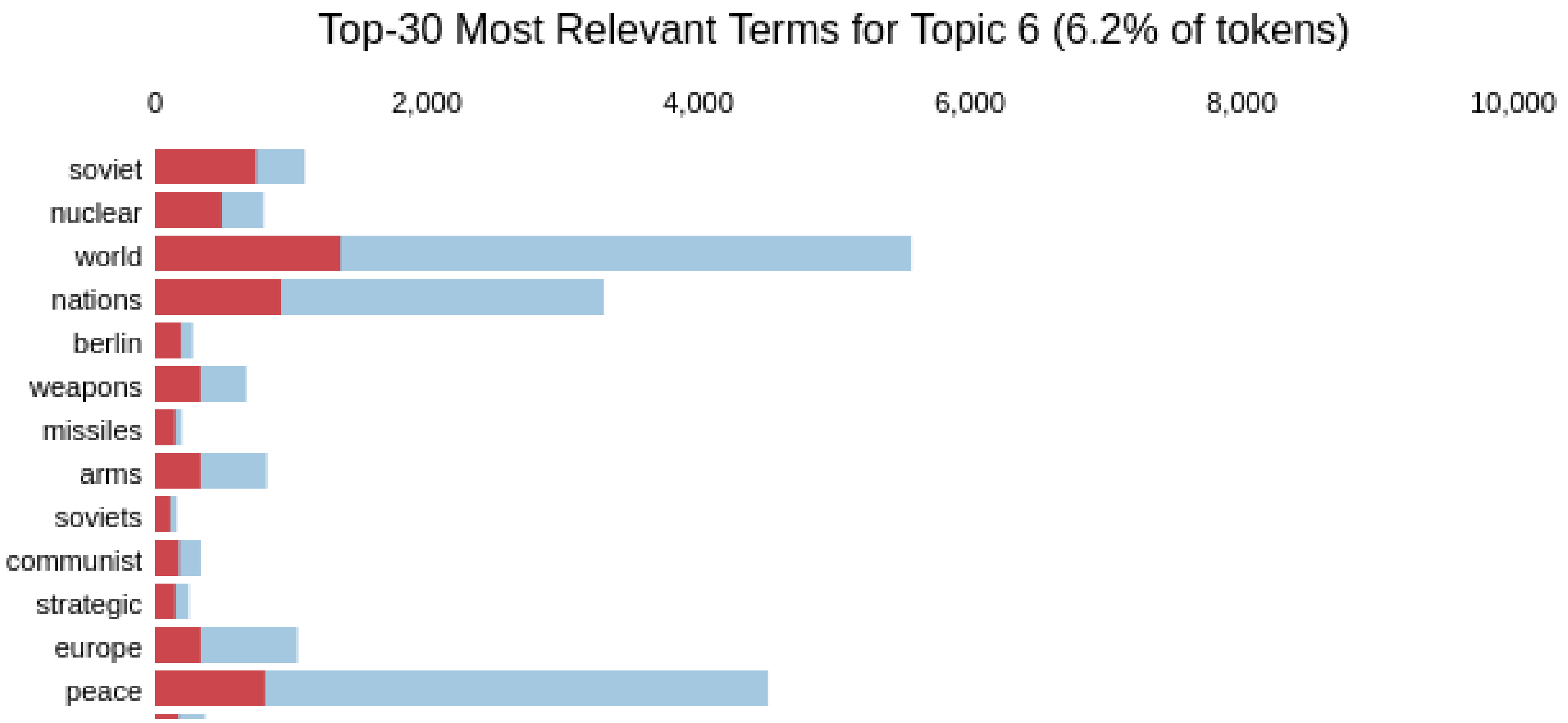


Verb tense



Country-Capital

Gensim example



(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!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

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 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

```
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

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/**)*

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

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

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

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

```python
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       ينديا
beat        بيت
Bermuda     بيرمودا
in          ين
Port        بورت
of          وف
Spain       سباين
in          ين
2007
,
which       ويكه
was         واس
equalled    يكالليد
five        فيفي
days        دايس
ago         اغو
by          بي
South       سووث
Africa      افريكا
in          ين
their       ثير
victory     فيكتوري
over        وفير
West        ويست
Indies      يندييس
in          ين
Sydney      سيدني
.
```

Spanish NER with polyglot

```
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!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

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!

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!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Training and testing a classification model with scikit-learn

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

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

```
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

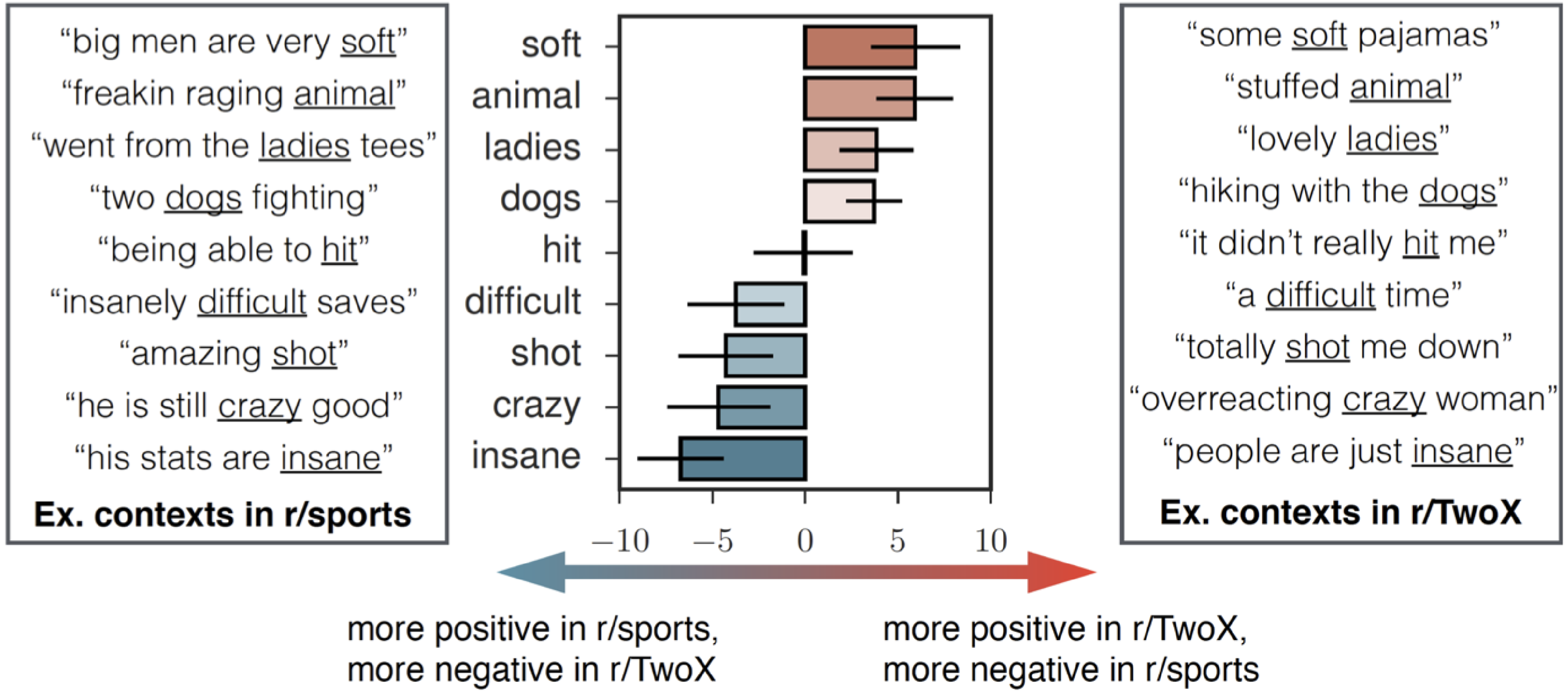
INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Katharine Jarmul
Founder, kjamistan

Translation

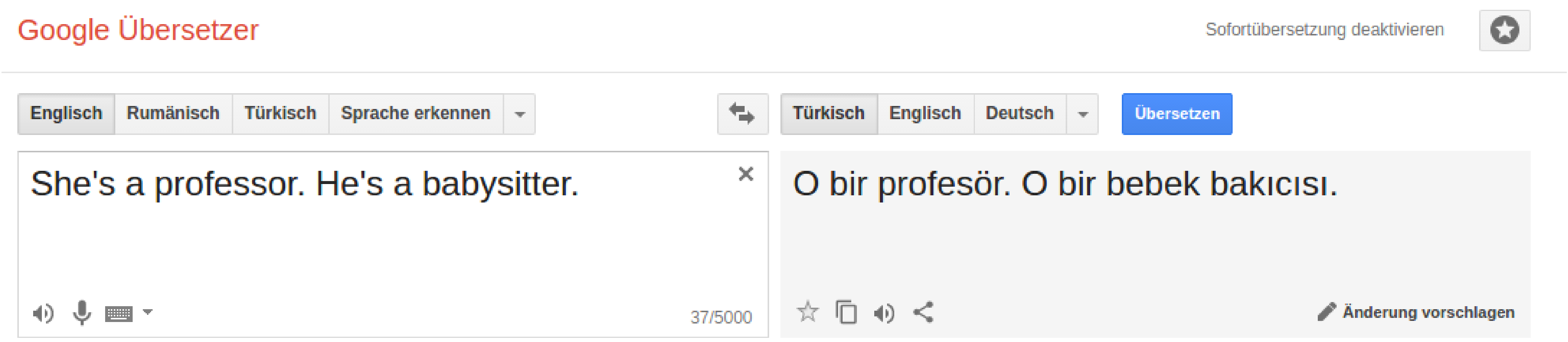
source:
(https://twitter.com/Lupintweets/status/865533182455685121)

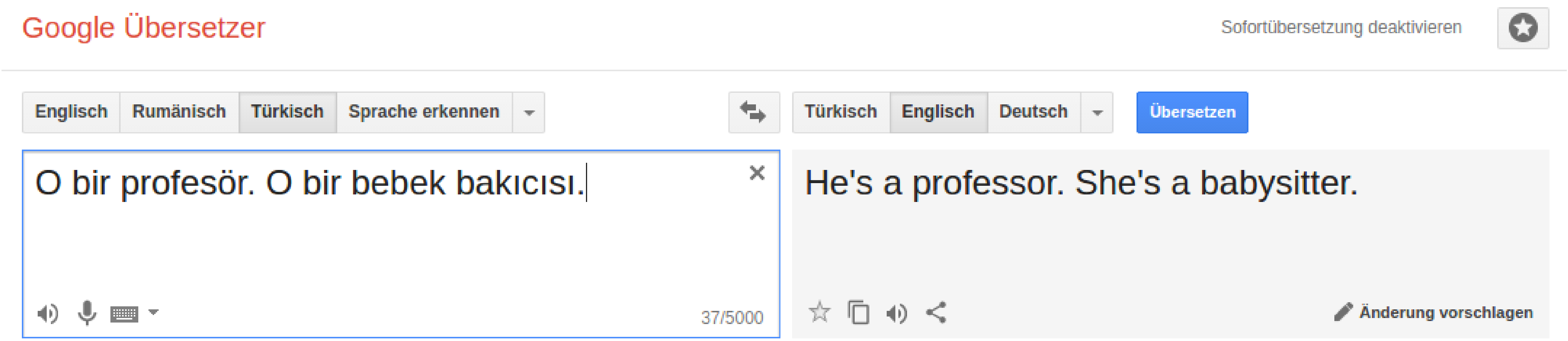
Sentiment analysis



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

Language biases





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

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Introduction to NLP feature engineering

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Numerical data

Iris dataset

sepal length	sepal width	petal length	petal width	class
6.3	2.9	5.6	1.8	Iris-virginica
4.9	3.0	1.4	0.2	Iris-setosa
5.6	2.9	3.6	1.3	Iris-versicolor
6.0	2.7	5.1	1.6	Iris-versicolor
7.2	3.6	6.1	2.5	Iris-virginica

One-hot encoding

sex
female
male
female
male
female
...

One-hot encoding

sex	one-hot encoding
female	→
male	→
female	→
male	→
female	→
...	...

One-hot encoding

sex	one-hot encoding	sex_female	sex_male
female	→	1	0
male	→	0	1
female	→	1	0
male	→	0	1
female	→	1	0
...	...	...	...

One-hot encoding with pandas

```python
# Import the pandas library
import pandas as pd

# Perform one-hot encoding on the 'sex' feature of df
df = pd.get_dummies(df, columns=['sex'])
```

Textual data

Movie Review Dataset

review	class
This movie is for dog lovers. A very poignant...	positive
The movie is forgettable. The plot lacked...	negative
A truly amazing movie about dogs. A gripping...	positive

Text pre-processing

- Converting to lowercase
  - **Example:** `Reduction` to `reduction`
- Converting to base-form
  - **Example:** `reduction` to `reduce`

Vectorization

review	class
This movie is for dog lovers. A very poignant...	positive
The movie is forgettable. The plot lacked...	negative
A truly amazing movie about dogs. A gripping...	positive

Vectorization

0	1	2	...	n	class
0.03	0.71	0.00	...	0.22	positive
0.45	0.00	0.03	...	0.19	negative
0.14	0.18	0.00	...	0.45	positive

Basic features

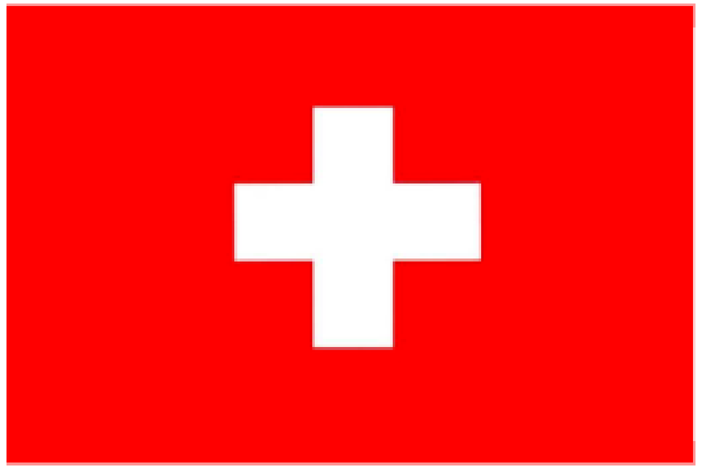
- Number of words
- Number of characters
- Average length of words
- Tweets

POS tagging

Word	POS
I	Pronoun
have	Verb
a	Article
dog	Noun

Named Entity Recognition

- Does noun refer to person, organization or country?

Noun	NER
Brian	**Person**
DataCamp	**Organization**

Concepts covered

- Text Preprocessing
- Basic Features
- Word Features
- Vectorization

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Basic feature extraction

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Number of characters

```
"I don't know." # 13 characters
```

```
# Compute the number of characters
text = "I don't know."
num_char = len(text)

# Print the number of characters
print(num_char)
```

```
13
```

```
# Create a 'num_chars' feature
df['num_chars'] = df['review'].apply(len)
```

Number of words

```
# Split the string into words
text = "Mary had a little lamb."
words = text.split()

# Print the list containing words
print(words)
```

```
['Mary', 'had', 'a', 'little', 'lamb.']
```

```
# Print number of words
print(len(words))
```

```
5
```

Number of words

```
# Function that returns number of words in string
def word_count(string):
    # Split the string into words
    words = string.split()

    # Return length of words list
    return len(words)
```

```
# Create num_words feature in df
df['num_words'] = df['review'].apply(word_count)
```

Average word length

```python
#Function that returns average word length
def avg_word_length(x):
    # Split the string into words
    words = x.split()
    # Compute length of each word and store in a separate list
    word_lengths = [len(word) for word in words]
    # Compute average word length
    avg_word_length = sum(word_lengths)/len(words)
    # Return average word length
    return(avg_word_length)
```

Average word length

```
# Create a new feature avg_word_length
df['avg_word_length'] = df['review'].apply(doc_density)
```

Special features

DataCamp @DataCamp

Big Data Fundamentals via PySpark by @upendra_35! This course covers the fundamentals of #BigData via #PySpark. #Spark is a “lightning fast cluster computing" framework for big data. datacamp.com/courses/big-da...

Hashtags and mentions

```python
# Function that returns number of hashtags
def hashtag_count(string):
    # Split the string into words
    words = string.split()
    # Create a list of hashtags
    hashtags = [word for word in words if word.startswith('#')]
    # Return number of hashtags
    return len(hashtags)
```

```python
hashtag_count("@janedoe This is my first tweet! #FirstTweet #Happy")
```

```
2
```

Other features

- Number of sentences
- Number of paragraphs
- Words starting with an uppercase
- All-capital words
- Numeric quantities

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Readability tests

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Overview of readability tests

- Determine readability of an English passage
- Scale ranging from primary school up to college graduate level
- A mathematical formula utilizing word, syllable and sentence count
- Used in fake news and opinion spam detection

Readability text examples

- Flesch reading ease
- Gunning fog index
- Simple Measure of Gobbledygook (SMOG)
- Dale-Chall score

Readability test examples

- **Flesch reading ease**
- **Gunning fog index**
- Simple Measure of Gobbledygook (SMOG)
- Dale-Chall score

Flesch reading ease

- One of the oldest and most widely used tests
- Dependent on two factors:
- **Greater the average sentence length, harder the text is to read**
    - "This is a short sentence."
    - "This is longer sentence with more words and it is harder to follow than the first sentence."
- **Greater the average number of syllables in a word, harder the text is to read**
    - "I live in my home."
    - "I reside in my domicile."
- Higher the score, greater the readability

Flesch reading ease score interpretation

Reading ease score	Grade Level
90-100	5
80-90	6
70-80	7
60-70	8-9
50-60	10-12
30-50	College
0-30	College Graduate

Gunning fog index

- Developed in 1954
- Also dependent on average sentence length
- Greater the percentage of complex words, harder the text is to read
- Higher the index, lesser the readability

Gunning fog index interpretation

Fog index	Grade level
17	College graduate
16	College senior
15	College junior
14	College sophomore
13	College freshman
12	High school senior
11	High school junior

Fog index	Grade level
10	High school sophomore
9	High school freshman
8	Eighth grade
7	Seventh grade
6	Sixth grade

The textatistic library

```
# Import the Textatistic class
from textatistic import Textatistic

# Create a Textatistic Object
readability_scores = Textatistic(text).scores

# Generate scores
print(readability_scores['flesch_score'])
print(readability_scores['gunningfog_score'])
```

```
21.14
16.26
```

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Tokenization and Lemmatization

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Text sources

- News articles
- Tweets
- Comments

Making text machine friendly

- `Dogs`, `dog`
- `reduction`, `REDUCING`, `Reduce`
- `don't`, `do not`
- `won't`, `will not`

Text preprocessing techniques

- Converting words into lowercase
- Removing leading and trailing whitespaces
- Removing punctuation
- Removing stopwords
- Expanding contractions
- Removing special characters (numbers, emojis, etc.)

Tokenization

```
"I have a dog. His name is Hachi."
```

Tokens:

```
["I", "have", "a", "dog", ".", "His", "name", "is", "Hachi", "."]
```

```
"Don't do this."
```

Tokens:

```
["Do", "n't", "do", "this", "."]
```

Tokenization using spaCy

```python
import spacy
# Load the en_core_web_sm model
nlp = spacy.load('en_core_web_sm')
# Initiliaze string
string = "Hello! I don't know what I'm doing here."
# Create a Doc object
doc = nlp(string)
# Generate list of tokens
tokens = [token.text for token in doc]
print(tokens)
```

```
['Hello','!','I','do',"n't",'know','what','I',"'m",'doing','here','.']
```

Lemmatization

- Convert word into its base form
  - `reducing` , `reduces` , `reduced` , `reduction` → `reduce`
  - `am` , `are` , `is` → `be`
  - `n't` → `not`
  - `'ve` → `have`

Lemmatization using spaCy

```
import spacy

# Load the en_core_web_sm model
nlp = spacy.load('en_core_web_sm')
# Initiliaze string
string = "Hello! I don't know what I'm doing here."
# Create a Doc object
doc = nlp(string)

# Generate list of lemmas
lemmas = [token.lemma_ for token in doc]
print(lemmas)
```

```
['hello','!','-PRON-','do','not','know','what','-PRON','be','do','here', '.']
```

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Text cleaning

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Text cleaning techniques

- Unnecessary whitespaces and escape sequences
- Punctuations
- Special characters (numbers, emojis, etc.)
- Stopwords

isalpha()

```
"Dog".isalpha()
```

```
True
```

```
"3dogs".isalpha()
```

```
False
```

```
"12347".isalpha()
```

```
False
```

```
"!".isalpha()
```

```
False
```

```
"?".isalpha()
```

```
False
```

A word of caution

- Abbreviations: `U.S.A`, `U.K`, etc.
- Proper Nouns: `word2vec` and `xto10x`.
- Write your own custom function (using regex) for the more nuanced cases.

Removing non-alphabetic characters

```
string = """
OMG!!!! This is like    the best thing ever \t\n.
Wow, such an amazing song! I'm hooked. Top 5 definitely. ?
"""
import spacy

# Generate list of tokens
nlp = spacy.load('en_core_web_sm')
doc = nlp(string)
lemmas = [token.lemma_ for token in doc]
```

Removing non-alphabetic characters

```
...
...
# Remove tokens that are not alphabetic
a_lemmas = [lemma for lemma in lemmas
            if lemma.isalpha() or lemma == '-PRON-']

# Print string after text cleaning
print(' '.join(a_lemmas))
```

```
'omg this be like the good thing ever wow such an amazing song -PRON- be hooked top definitely'
```

Stopwords

- Words that occur extremely commonly
- Eg. articles, be verbs, pronouns, etc.

Removing stopwords using spaCy

```
# Get list of stopwords
stopwords = spacy.lang.en.stop_words.STOP_WORDS
string = """
OMG!!!! This is like    the best thing ever \t\n.
Wow, such an amazing song! I'm hooked. Top 5 definitely. ?
"""
```

Removing stopwords using spaCy

```
...
...
# Remove stopwords and non-alphabetic tokens
a_lemmas = [lemma for lemma in lemmas
            if lemma.isalpha() and lemma not in stopwords]
# Print string after text cleaning
print(' '.join(a_lemmas))
```

```
'omg like good thing wow amazing song hooked definitely'
```

Other text preprocessing techniques

- Removing HTML/XML tags
- Replacing accented characters (such as é)
- Correcting spelling errors

A word of caution

Always use only those text preprocessing techniques that are relevant to your application.

Let's practice!

Part-of-speech tagging

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Applications

- Word-sense disambiguation
  - `"The bear is a majestic animal"`
  - `"Please bear with me"`
- Sentiment analysis
- Question answering
- Fake news and opinion spam detection

POS tagging

- Assigning every word, its corresponding part of speech.

```
"Jane is an amazing guitarist."
```

- **POS Tagging:**
  - `Jane` → **proper noun**
  - `is` → **verb**
  - `an` → **determiner**
  - `amazing` → **adjective**
  - `guitarist` → **noun**

POS tagging using spaCy

```
import spacy

# Load the en_core_web_sm model
nlp = spacy.load('en_core_web_sm')
```

```
# Initiliaze string
string = "Jane is an amazing guitarist"
```

```
# Create a Doc object
doc = nlp(string)
```

POS tagging using spaCy

```
...
...
# Generate list of tokens and pos tags
pos = [(token.text, token.pos_) for token in doc]
print(pos)
```

```
[('Jane', 'PROPN'),
 ('is', 'VERB'),
 ('an', 'DET'),
 ('amazing', 'ADJ'),
 ('guitarist', 'NOUN')]
```

POS annotations in spaCy

- `PROPN` → proper noun
- `DET` → determinant
- spaCy annotations at **https://spacy.io/api/annotation**

POS	DESCRIPTION	EXAMPLES
`ADJ`	adjective	big, old, green, incomprehensible, first
`ADP`	adposition	in, to, during
`ADV`	adverb	very, tomorrow, down, where, there
`AUX`	auxiliary	is, has (done), will (do), should (do)
`CONJ`	conjunction	and, or, but
`CCONJ`	coordinating conjunction	and, or, but
`DET`	determiner	a, an, the

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Named entity recognition

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Applications

- Efficient search algorithms
- Question answering
- News article classification
- Customer service

Named entity recognition

- Identifying and classifying named entities into predefined categories.
- Categories include person, organization, country, etc.

```
"John Doe is a software engineer working at Google. He lives in France."
```

- **Named Entities**
- `John Doe` → person
- `Google` → organization
- `France` → country (geopolitical entity)

NER using spaCy

```python
import spacy
string = "John Doe is a software engineer working at Google. He lives in France."

# Load model and create Doc object
nlp = spacy.load('en_core_web_sm')
doc = nlp(string)

# Generate named entities
ne = [(ent.text, ent.label_) for ent in doc.ents]
print(ne)
```

```
[('John Doe', 'PERSON'), ('Google', 'ORG'), ('France', 'GPE')]
```

NER annotations in spaCy

- More than 15 categories of named entities
- NER annotations at **https://spacy.io/api/annotation#named-entities**

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.

A word of caution

- Not perfect
- Performance dependent on training and test data
- Train models with specialized data for nuanced cases
- Language specific

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Building a bag of words model

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Recap of data format for ML algorithms

For any ML algorithm,

- Data must be in tabular form
- Training features must be numerical

Bag of words model

- Extract word tokens
- Compute frequency of word tokens
- Construct a word vector out of these frequencies and vocabulary of corpus

Bag of words model example

Corpus

```
"The lion is the king of the jungle"
```

```
"Lions have lifespans of a decade"
```

```
"The lion is an endangered species"
```

Bag of words model example

Vocabulary → `a`, `an`, `decade`, `endangered`, `have`, `is`, `jungle`, `king`, `lifespans`, `lion`, `Lions`, `of`, `species`, `the`, `The`

```
"The lion is the king of the jungle"
```

```
[0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 2, 1]
```

```
"Lions have lifespans of a decade"
```

```
[1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0]
```

```
"The lion is an endangered species"
```

```
[0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1]
```

Text preprocessing

- `Lions`, `lion` → `lion`
- `The`, `the` → `the`
- No punctuations
- No stopwords
- Leads to smaller vocabularies
- Reducing number of dimensions helps improve performance

Bag of words model using sklearn

```
corpus = pd.Series([
    'The lion is the king of the jungle',
    'Lions have lifespans of a decade',
    'The lion is an endangered species'
])
```

Bag of words model using sklearn

```
# Import CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
# Create CountVectorizer object
vectorizer = CountVectorizer()
# Generate matrix of word vectors
bow_matrix = vectorizer.fit_transform(corpus)
print(bow_matrix.toarray())
```

```
array([[0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 3],
       [0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0],
       [1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1]], dtype=int64)
```

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Building a BoW Naive Bayes classifier

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Spam filtering

message	label
WINNER!! As a valued network customer you have been selected to receive a $900 prize reward! To claim call 09061701461	spam
Ah, work. I vaguely remember that. What does it feel like?	ham

Steps

1. Text preprocessing
2. Building a bag-of-words model (or representation)
3. Machine learning

Text preprocessing using CountVectorizer

CountVectorizer arguments

- `lowercase` : `False` , `True`
- `strip_accents` : `'unciode'` , `'ascii'` , `None`
- `stop_words` : `'english'` , `list` , `None`
- `token_pattern` : `regex`
- `tokenizer` : `function`

Building the BoW model

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

# Create CountVectorizer object
vectorizer = CountVectorizer(strip_accents='ascii', stop_words='english', lowercase=False)

# Import train_test_split
from sklearn.model_selection import train_test_split

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df['message'], df['label'], test_size=0.25)
```

Building the BoW model

```
...
...
# Generate training Bow vectors
X_train_bow = vectorizer.fit_transform(X_train)

# Generate test BoW vectors
X_test_bow = vectorizer.transform(X_test)
```

Training the Naive Bayes classifier

```
# Import MultinomialNB
from sklearn.naive_bayes import MultinomialNB

# Create MultinomialNB object
clf = MultinomialNB()

# Train clf
clf.fit(X_train_bow, y_train)

# Compute accuracy on test set
accuracy = clf.score(X_test_bow, y_test)
print(accuracy)
```

```
0.760051
```

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Building n-gram models

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

BoW shortcomings

review	label
`'The movie was good and not boring'`	positive
`'The movie was not good and boring'`	negative

- Exactly the same BoW representation!
- Context of the words is lost.
- Sentiment dependent on the position of 'not'.

n-grams

- Contiguous sequence of n elements (or words) in a given document.
- n = 1 → bag-of-words

```
'for you a thousand times over'
```

- n = 2, n-grams:

```
[
'for you',
'you a',
'a thousand',
'thousand times',
'times over'
]
```

n-grams

```
'for you a thousand times over'
```

- n = 3, n-grams:

```
[
'for you a',
'you a thousand',
'a thousand times',
'thousand times over'
]
```

- Captures more context.

Applications

- Sentence completion
- Spelling correction
- Machine translation correction

Building n-gram models using scikit-learn

Generates only bigrams.

```
bigrams = CountVectorizer(ngram_range=(2,2))
```

Generates unigrams, bigrams and trigrams.

```
ngrams = CountVectorizer(ngram_range=(1,3))
```

Shortcomings

- Curse of dimensionality
- Higher order n-grams are rare
- Keep n small

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Building tf-idf document vectors

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

n-gram modeling

- Weight of dimension dependent on the frequency of the word corresponding to the dimension.
  - Document contains the word `human` in five places.
  - Dimension corresponding to `human` has weight `5`.

Motivation

- Some words occur very commonly across all documents
- Corpus of documents on the universe
  - One document has `jupiter` and `universe` occurring 20 times each.
  - `jupiter` rarely occurs in the other documents. `universe` is common.
  - Give more weight to `jupiter` on account of exclusivity.

Applications

- Automatically detect stopwords
- Search
- Recommender systems
- Better performance in predictive modeling for some cases

Term frequency-inverse document frequency

- Proportional to term frequency
- Inverse function of the number of documents in which it occurs

Mathematical formula

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

$$w_{i,j} \rightarrow \text{weight of term } i \text{ in document } j$$

Mathematical formula

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

$$w_{i,j} \rightarrow \text{weight of term } i \text{ in document } j$$

$$tf_{i,j} \rightarrow \text{term frequency of term } i \text{ in document } j$$

Mathematical formula

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

$w_{i,j} \rightarrow$ weight of term i in document j

$tf_{i,j} \rightarrow$ term frequency of term $i$in document j

$N \rightarrow$ number of documents in the corpus

$df_i \rightarrow$ number of documents containing term i

Mathematical formula

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

$$w_{i,j} \rightarrow \text{weight of term } i \text{ in document } j$$

$$tf_{i,j} \rightarrow term\ frequency\ of\ term\ i\ in\ document\ j$$

$$N \rightarrow number\ of\ documents\ in\ the\ corpus$$

$$df_i \rightarrow number\ of\ documents\ cotaining\ term\ i$$

Example:

$$w_{library,document} = 5 \cdot log(\frac{20}{8}) \approx 2$$

tf-idf using scikit-learn

```
# Import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
# Create TfidfVectorizer object
vectorizer = TfidfVectorizer()
# Generate matrix of word vectors
tfidf_matrix = vectorizer.fit_transform(corpus)
print(tfidf_matrix.toarray())
```

```
[[0.         0.         0.         0.         0.25434658 0.33443519
  0.33443519 0.         0.25434658 0.         0.25434658 0.
  0.76303975]
 [0.         0.46735098 0.         0.46735098 0.         0.
  0.         0.46735098 0.         0.46735098 0.35543247 0.
  0.        ]
...
```

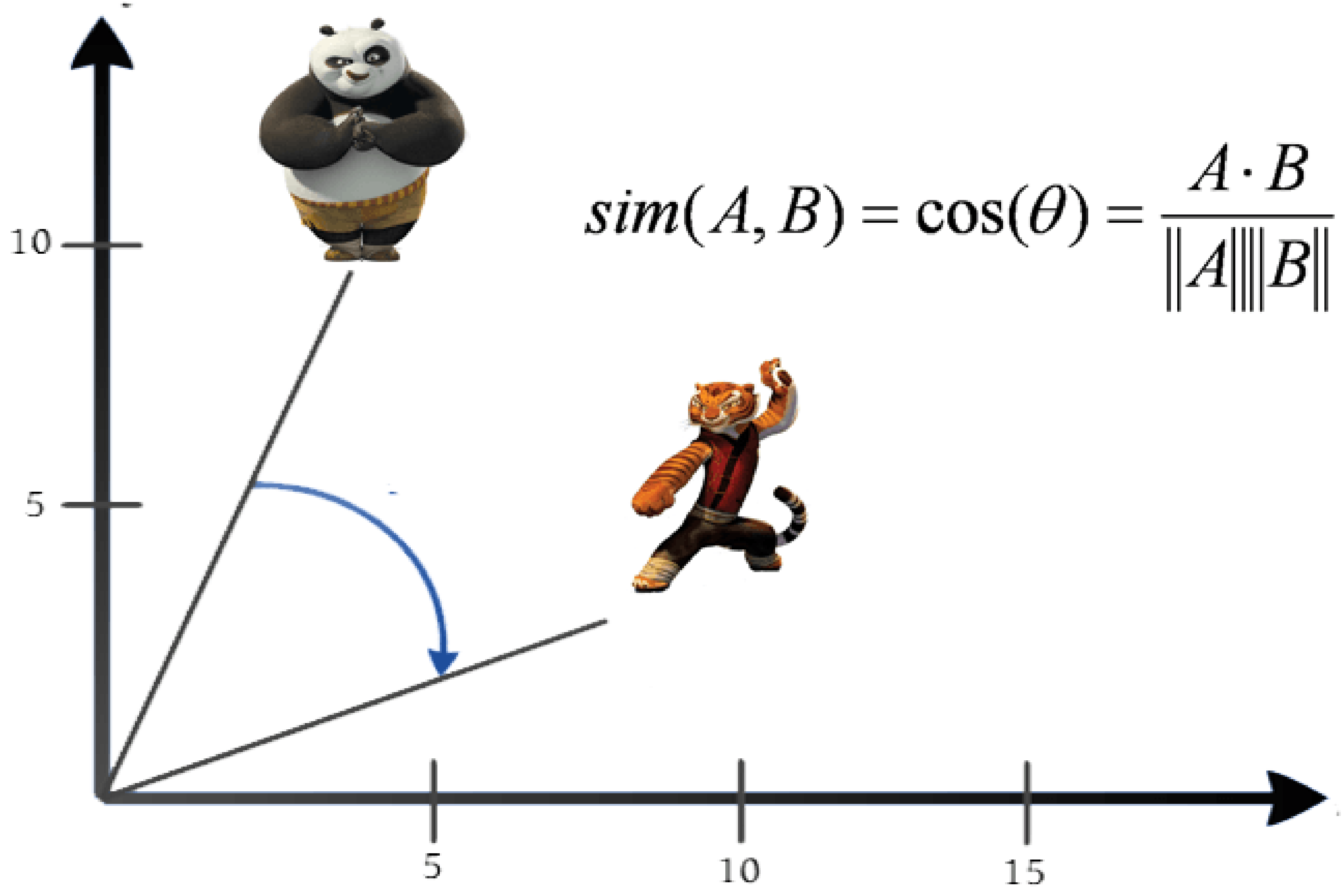
Let's practice!

Cosine similarity

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Cosine Similarity



$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

[1] Image courtesy techninpink.com

The dot product

Consider two vectors,

$$V = (v_1, v_2, \cdots, v_n), W = (w_1, w_2, \cdots, w_n)$$

Then the dot product of V and W is,

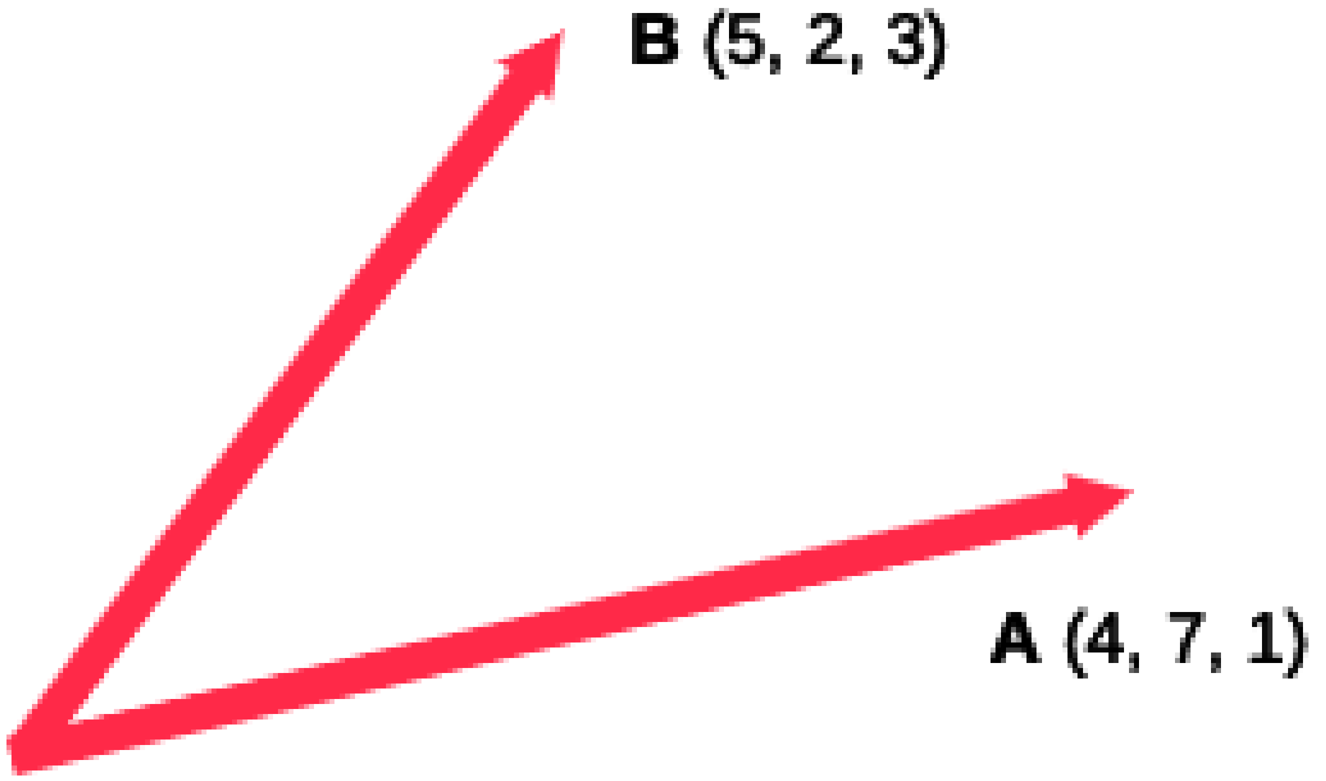
$$V \cdot W = (v_1 \times w_1) + (v_2 \times w_2) + \cdots + (v_n \times w_n)$$

Example:

$$A = (4, 7, 1)\ ,\ B = (5, 2, 3)$$

$$A \cdot B = (4 \times 5) + (7 \times 2) + \cdots (1 \times 3)$$

$$= 20 + 14 + 3 = 37$$



Magnitude of a vector

For any vector,

$$V = (v_1, v_2, \cdots, v_n)$$

The magnitude is defined as,

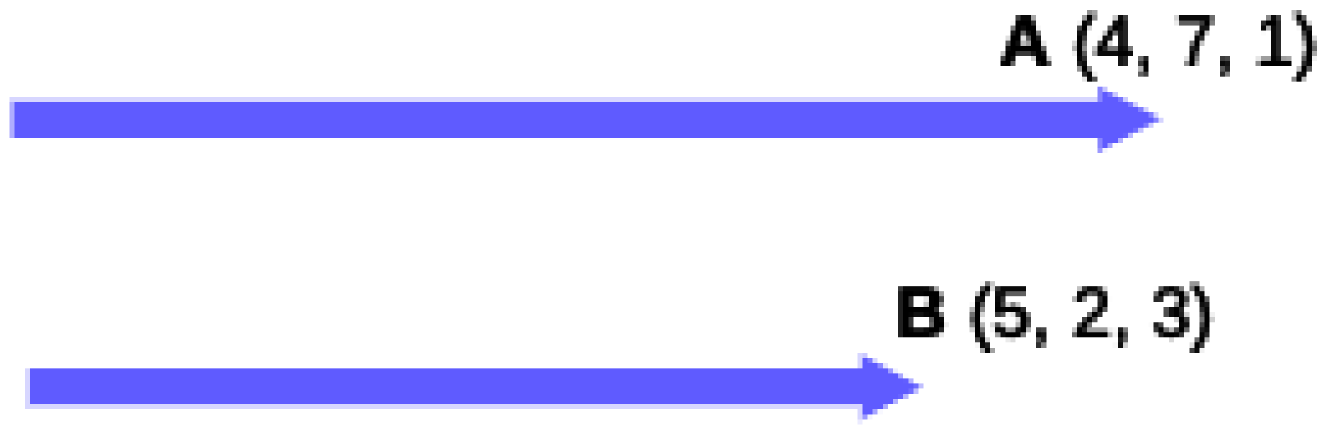
$$||\mathbf{V}|| = \sqrt{(v_1)^2 + (v_2)^2 + ... + (v_n)^2}$$

Example:

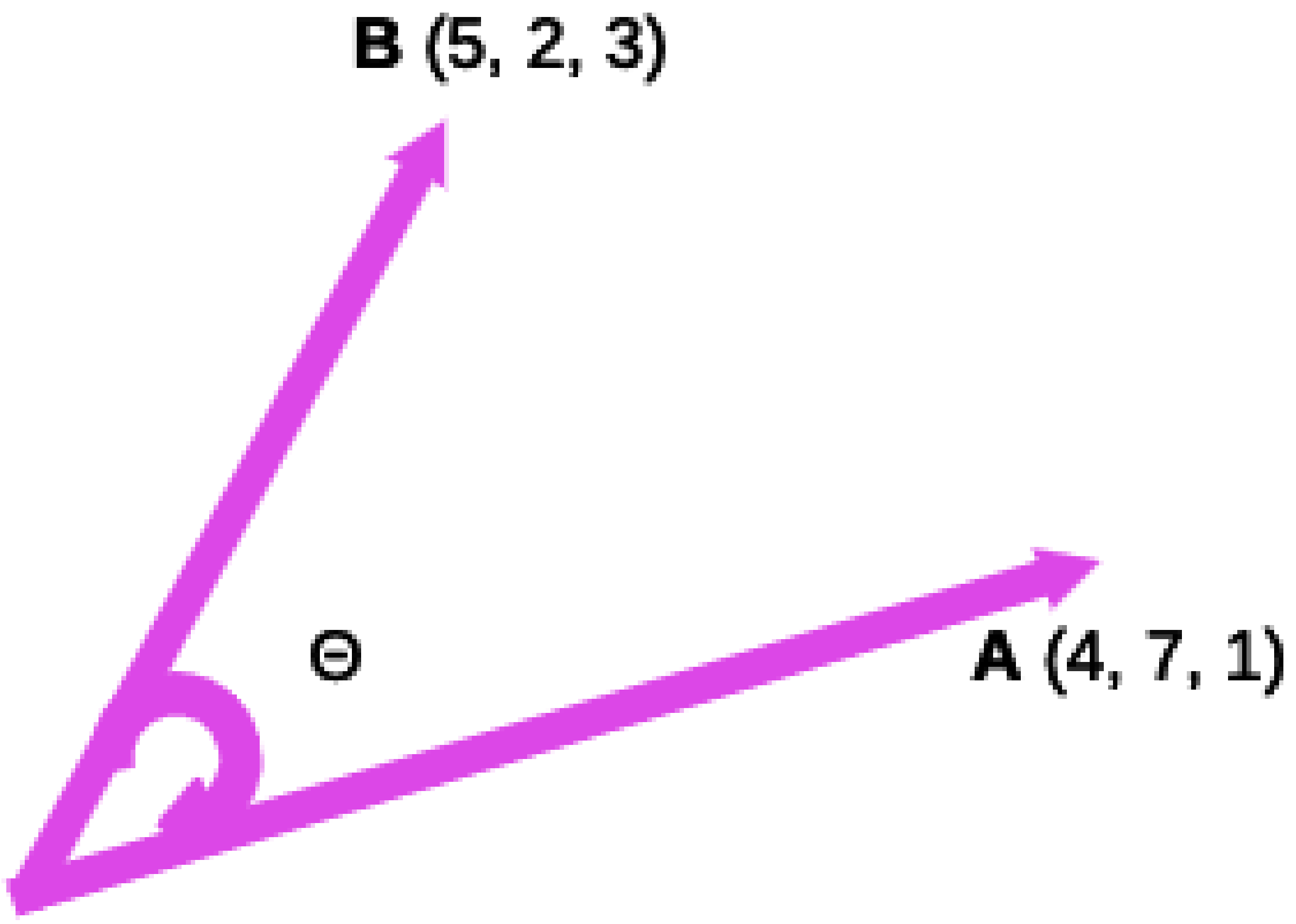
$$A = (4, 7, 1)\ ,\ B = (5, 2, 3)$$

$$||\mathbf{A}|| = \sqrt{(4)^2 + (7)^2 + (1)^2}$$

$$= \sqrt{16 + 49 + 1} = \sqrt{66}$$



The cosine score



$$A : (4, 7, 1)$$

$$B : (5, 2, 3)$$

The cosine score,

$$cos(A, B) = \frac{A \cdot B}{|A| \cdot |B|}$$

$$= \frac{37}{\sqrt{66} \times \sqrt{38}}$$

$$= 0.7388$$

Cosine Score: points to remember

- Value between -1 and 1.
- In NLP, value between 0 and 1.
- Robust to document length.

Implementation using scikit-learn

```
# Import the cosine_similarity
from sklearn.metrics.pairwise import cosine_similarity
```

```
# Define two 3-dimensional vectors A and B
A = (4,7,1)
B = (5,2,3)

# Compute the cosine score of A and B
score = cosine_similarity([A], [B])

# Print the cosine score
print(score)
```

```
array([[ 0.73881883]])
```

Let's practice!

Building a plot line based recommender

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

datacamp

Movie recommender

Title	Overview
Shanghai Triad	A provincial boy related to a Shanghai crime family is recruited by his uncle into cosmopolitan Shanghai in the 1930s to be a servant to a ganglord's mistress.
Cry, the Beloved Country	A South-African preacher goes to search for his wayward son who has committed a crime in the big city.

Movie recommender

```
get_recommendations("The Godfather")
```

```
1178                 The Godfather: Part II
44030    The Godfather Trilogy: 1972-1990
1914                The Godfather: Part III
23126                           Blood Ties
11297                     Household Saints
34717                    Start Liquidation
10821                             Election
38030                           Goodfellas
17729                    Short Sharp Shock
26293                   Beck 28 - Familjen
Name: title, dtype: object
```

Steps

1. Text preprocessing
2. Generate tf-idf vectors
3. Generate cosine similarity matrix

The recommender function

1. Take a movie title, cosine similarity matrix and indices series as arguments.
2. Extract pairwise cosine similarity scores for the movie.
3. Sort the scores in descending order.
4. Output titles corresponding to the highest scores.
5. Ignore the highest similarity score (of 1).

Generating tf-idf vectors

```
# Import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

# Create TfidfVectorizer object
vectorizer = TfidfVectorizer()

# Generate matrix of tf-idf vectors
tfidf_matrix = vectorizer.fit_transform(movie_plots)
```

Generating cosine similarity matrix

```
# Import cosine_similarity
from sklearn.metrics.pairwise import cosine_similarity

# Generate cosine similarity matrix
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

```
array([[1.        , 0.27435345, 0.23092036, ..., 0.        , 0.        ,
        0.00758112],
       [0.27435345, 1.        , 0.1246955 , ..., 0.        , 0.        ,
        0.00740494],
       ...,
       [0.00758112, 0.00740494, 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

The linear_kernel function

- Magnitude of a tf-idf vector is 1
- Cosine score between two tf-idf vectors is their dot product.
- Can significantly improve computation time.
- Use `linear_kernel` instead of `cosine_similarity` .

Generating cosine similarity matrix

```
# Import cosine_similarity
from sklearn.metrics.pairwise import linear_kernel

# Generate cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

```
array([[1.        , 0.27435345, 0.23092036, ..., 0.        , 0.        ,
        0.00758112],
       [0.27435345, 1.        , 0.1246955 , ..., 0.        , 0.        ,
        0.00740494],
       ...,
       [0.00758112, 0.00740494, 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

The get_recommendations function

```
get_recommendations('The Lion King', cosine_sim, indices)
```

```
7782                          African Cats
5877        The Lion King 2: Simba's Pride
4524                             Born Free
2719                              The Bear
4770         Once Upon a Time in China III
7070                            Crows Zero
739                       The Wizard of Oz
8926                        The Jungle Book
1749                     Shadow of a Doubt
7993                          October Baby
Name: title, dtype: object
```

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Beyond n-grams: word embeddings

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

The problem with BoW and tf-idf

```
'I am happy'
```

```
'I am joyous'
```

```
'I am sad'
```

Word embeddings

- Mapping words into an n-dimensional vector space
- Produced using deep learning and huge amounts of data
- Discern how similar two words are to each other
- Used to detect synonyms and antonyms
- Captures complex relationships
  - `King` - `Queen` → `Man` - `Woman`
  - `France` - `Paris` → `Russia` - `Moscow`
- Dependent on spacy model; independent of dataset you use

Word embeddings using spaCy

```
import spacy

# Load model and create Doc object
nlp = spacy.load('en_core_web_lg')
doc = nlp('I am happy')
```

```
# Generate word vectors for each token
for token in doc:
  print(token.vector)
```

```
[-1.0747459e+00  4.8677087e-02  5.6630421e+00  1.6680446e+00
 -1.3194644e+00 -1.5142369e+00  1.1940931e+00 -3.0168812e+00
 ...
```

Word similarities

```
doc = nlp("happy joyous sad")
for token1 in doc:
  for token2 in doc:
    print(token1.text, token2.text, token1.similarity(token2))
```

```
happy happy 1.0
happy joyous 0.63244456
happy sad 0.37338886
joyous happy 0.63244456
joyous joyous 1.0
joyous sad 0.5340932
...
```

Document similarities

```
# Generate doc objects
sent1 = nlp("I am happy")
sent2 = nlp("I am sad")
sent3 = nlp("I am joyous")
```

```
# Compute similarity between sent1 and sent2
sent1.similarity(sent2)
```

```
0.9273363837282105
```

```
# Compute similarity between sent1 and sent3
sent1.similarity(sent3)
```

```
0.9403554938594568
```

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Congratulations!

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

Review

- Basic features (characters, words, mentions, etc.)
- Readability scores
- Tokenization and lemmatization
- Text cleaning
- Part-of-speech tagging & named entity recognition
- n-gram modeling
- tf-idf
- Cosine similarity
- Word embeddings

Further resources

- Advanced NLP with spaCy
- Deep Learning in Python

Thank you!

FEATURE ENGINEERING FOR NLP IN PYTHON