

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 nltk.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

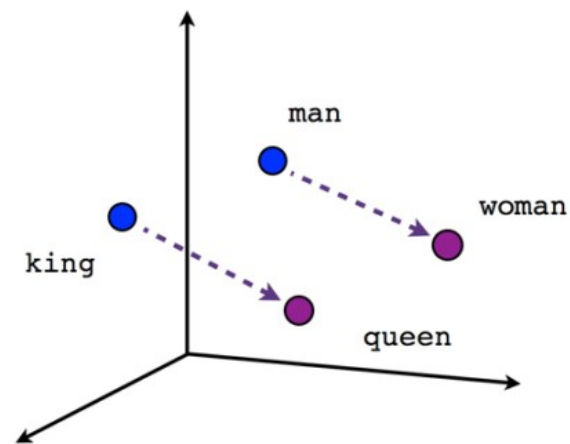


Katharine Jarmul
Founder, kjamistan

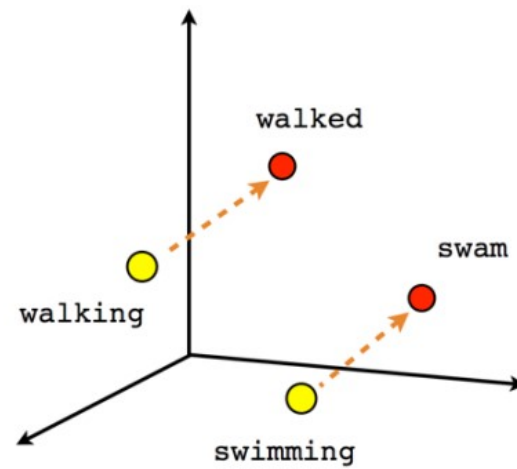
What is gensim?

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

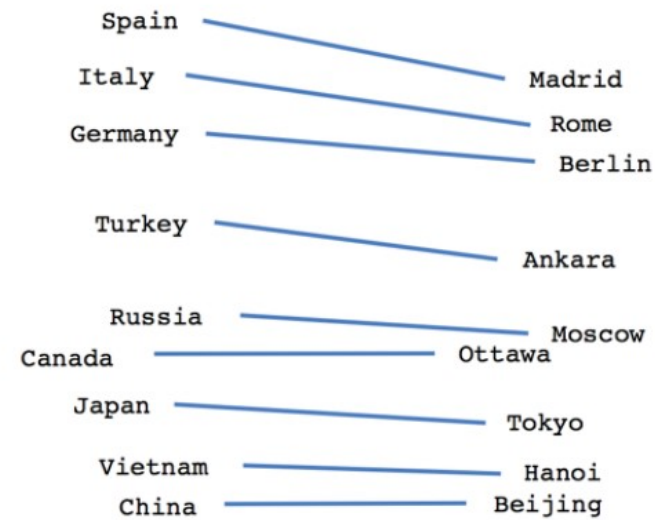
What is a word vector?



Male-Female

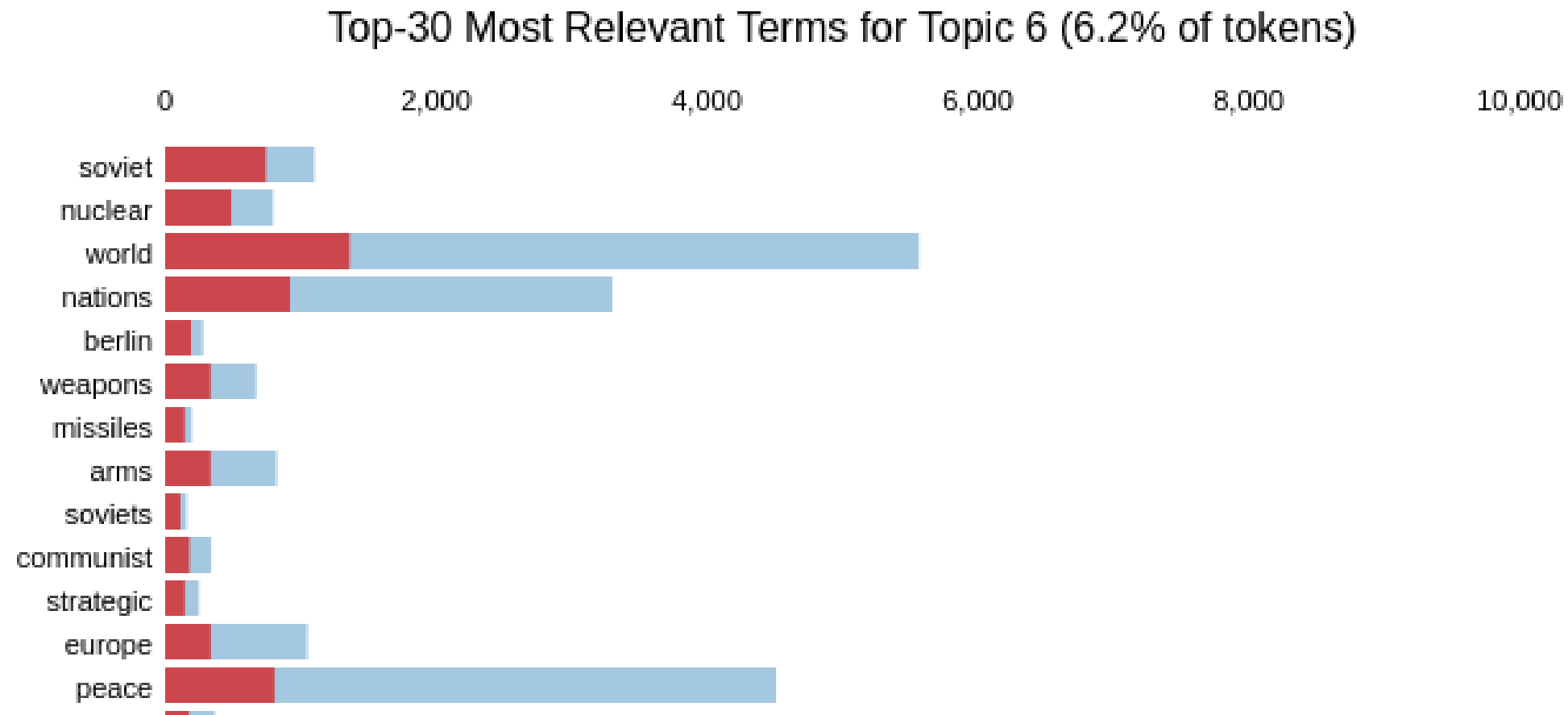


Verb tense



Country-Capital

Gensim example



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

```
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!'],]
```

```
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

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

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

$tf_{i,j}$ = number of occurrences of token i in document j

df_i = number of documents that contain token i

N = total number of documents

the weight will be low if the term doesn't appear often in the document because the tf variable will then be low. However, the weight will also be a low if the logarithm is close to zero, meaning the internal equation is low

Tf-idf with gensim

```
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