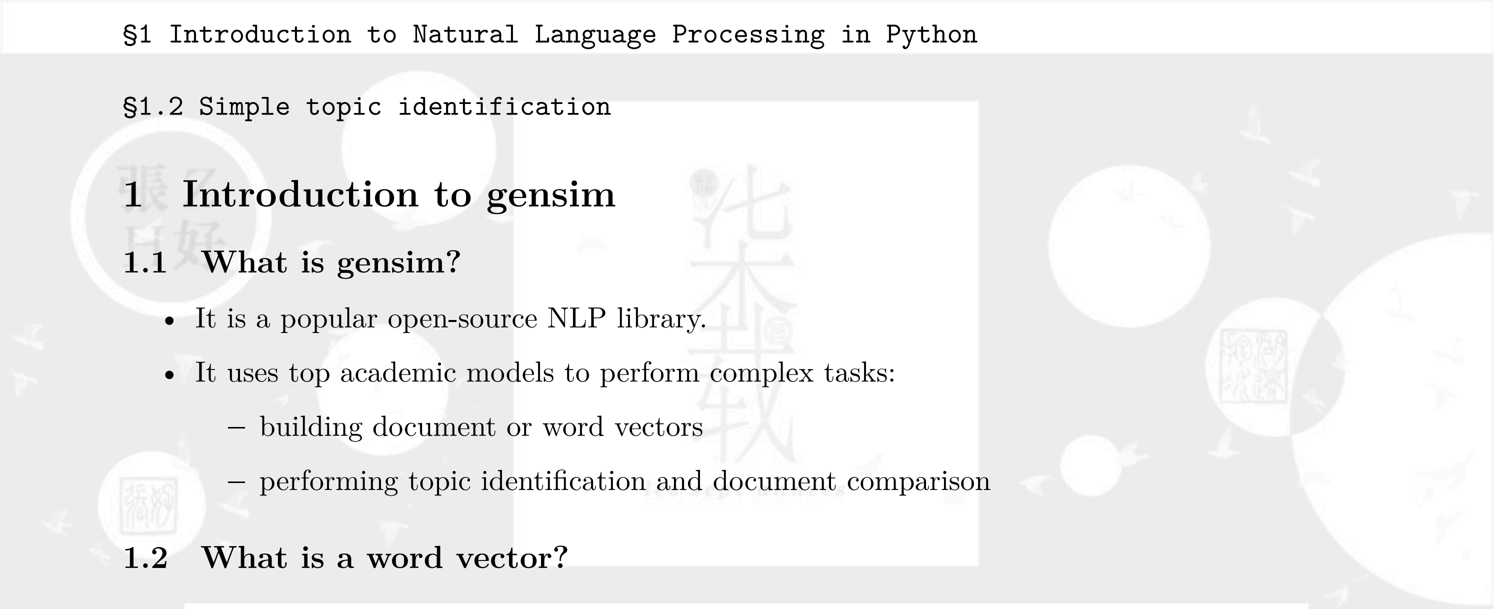
Introduction to gensim

Puteaux, Fall/Winter 2020-2021

```
##############################################
##                                          ##
##  Natural Language Processing in Python  ##
##                                          ##
##############################################
```



§1 Introduction to Natural Language Processing in Python

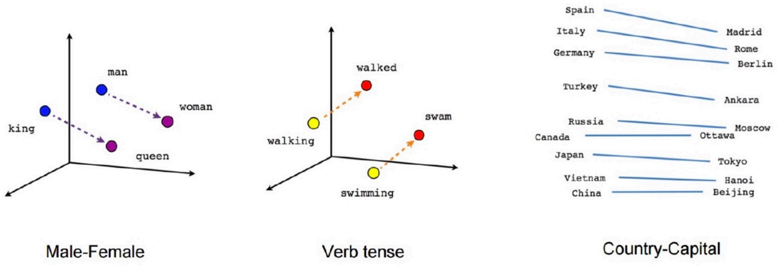
§1.2 Simple topic identification

1 Introduction to gensim

1.1 What is gensim?

- It is a popular open-source NLP library.
- It uses top academic models to perform complex tasks:
  - building document or word vectors
  - performing topic identification and document comparison

1.2 What is a word vector?



Male-Female

Verb tense

Country-Capital

1.3 Code of creating a gensim corpus:

[1]:
```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!',
]
```

[2]:
```
tokenized_docs = [word_tokenize(doc.lower()) for doc in my_documents]

dictionary = Dictionary(tokenized_docs)
dictionary.token2id
```

[2]:
```
{'.': 0,
 'a': 1,
 'about': 2,
 'aliens': 3,
 'and': 4,
 'movie': 5,
 'spaceship': 6,
 'the': 7,
 'was': 8,
 '!': 9,
 'i': 10,
 'liked': 11,
 'really': 12,
 ',': 13,
 'action': 14,
 'awesome': 15,
 'boring': 16,
 'but': 17,
 'characters': 18,
 'scenes': 19,
 'alien': 20,
 'awful': 21,
 'films': 22,
 'hate': 23,
 'cool': 24,
 'is': 25,
 'space': 26,
 'more': 27,
 'please': 28}
```

[3]:
```
corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]
corpus
```

[3]:
```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1)],
 [(5, 1), (7, 1), (9, 1), (10, 1), (11, 1), (12, 1)],
 [(0, 1), (13, 1), (14, 1), (15, 1), (16, 1), (17, 1), (18, 1), (19, 1)],
 [(0, 1),
  (5, 1),
  (7, 1),
  (8, 1),
  (9, 1),
  (10, 1),
  (20, 1),
  (21, 1),
  (22, 1),
  (23, 1)],
 [(0, 1), (5, 1), (7, 1), (9, 1), (10, 1), (11, 1), (24, 1), (25, 1), (26, 1)],
 [(9, 1), (13, 1), (22, 1), (26, 1), (27, 1), (28, 1)]]
```

1.4 What are the advantages of creating a gensim corpus?

- First of all, gensim models can be easily saved, updated, and reused.
- Secondly, the dictionary created can also be updated.
- Lastly, the more advanced and feature-rich bag-of-words can be used in future exercises.

1.5 Practice question for word vectors:

- What are word vectors, and how do they help with NLP?

  ☐ They are similar to bags of words, just with numbers. You use them to count how many tokens there are.

  ☐ Word vectors are sparse arrays representing bigrams in the corpora. You can use them to compare two sets of words to one another.

  ☒ Word vectors are multi-dimensional mathematical representations of words created using deep learning methods. They give us insight into relationships between words in a corpus.

  ☐ Word vectors don’t actually help NLP and are just hype.

1.6 Practice exercises for introduction to gensim:

▶ **Package pre-loading:**

[4]:

```
import zipfile

from nltk import word_tokenize
from nltk.corpus import stopwords
```

▶ **Data pre-loading:**

[5]:

```
file_name = 'ref3. Wikipedia articles.zip'
with zipfile.ZipFile(file_name, 'r') as archive:
    files = [
        archive.read(name) for name in archive.namelist()
        if name.endswith('.txt')
    ]

doc_tokens = [word_tokenize(file.decode("utf-8")) for file in files]

articles = []
english_stops = stopwords.words('english')
for i in range(len(doc_tokens)):
    lower_tokens = [t.lower() for t in doc_tokens[i]]
    alphanumeric_only = [t for t in lower_tokens if t.isalnum()]
    no_stops = [t for t in alphanumeric_only if t not in english_stops]
    articles.append(no_stops)
```

▶ Gensim corpus creating and querying practice:

[6]:
```
# Import Dictionary
from gensim.corpora.dictionary import Dictionary

# Create a Dictionary from the articles: dictionary
dictionary = Dictionary(articles)

# Select the id for "computer": computer_id
computer_id = dictionary.token2id.get("computer")

# Use computer_id with the dictionary to print the word
print(dictionary.get(computer_id))

# Create a MmCorpus: corpus
corpus = [dictionary.doc2bow(article) for article in articles]

# Print the first 10 word ids with their frequency counts from the fifth␣
 ↪document
print(corpus[4][:10])
```

```
computer
[(13, 2), (24, 1), (43, 1), (44, 6), (45, 1), (50, 1), (58, 1), (59, 1), (61,
7), (75, 1)]
```

▶ Package pre-loading:

[7]:
```
from collections import defaultdict
import itertools
```

▶ Gensim bag-of-words practice:

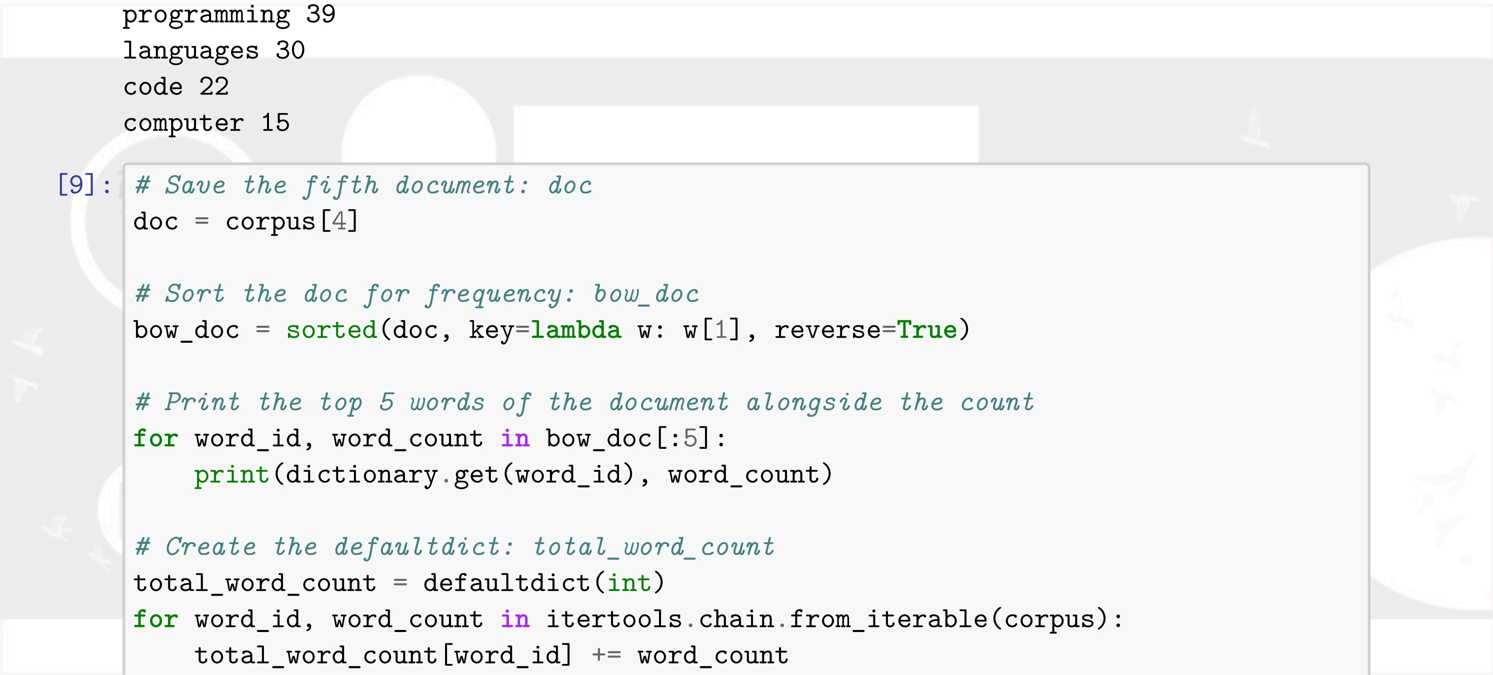
[8]:
```
# Save the fifth document: doc
doc = corpus[4]
```

```
# Sort the doc for frequency: bow_doc
bow_doc = sorted(doc, key=lambda w: w[1], reverse=True)

# Print the top 5 words of the document alongside the count
for word_id, word_count in bow_doc[:5]:
    print(dictionary.get(word_id), word_count)

# Create the defaultdict: total_word_count
total_word_count = defaultdict(int)
for word_id, word_count in itertools.chain.from_iterable(corpus):
    total_word_count[word_id] += word_count
```

```
language 54
programming 39
languages 30
code 22
computer 15
```



```
[9]: # Save the fifth document: doc
doc = corpus[4]

# Sort the doc for frequency: bow_doc
bow_doc = sorted(doc, key=lambda w: w[1], reverse=True)

# Print the top 5 words of the document alongside the count
for word_id, word_count in bow_doc[:5]:
    print(dictionary.get(word_id), word_count)

# Create the defaultdict: total_word_count
total_word_count = defaultdict(int)
for word_id, word_count in itertools.chain.from_iterable(corpus):
    total_word_count[word_id] += word_count

# Create a sorted list from the defaultdict: sorted_word_count
sorted_word_count = sorted(total_word_count.items(),
                           key=lambda w: w[1],
                           reverse=True)

# Print the top 5 words across all documents alongside the count
for word_id, word_count in sorted_word_count[:5]:
    print(dictionary.get(word_id), word_count)
```

```
language 54
programming 39
languages 30
code 22
computer 15
computer 598
software 450
cite 322
ref 259
code 235
```