**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

**Coding Example**

In following code we can see that every word’s frequency is been counted and that too how?

An counter is used in following which acts as enumerator in an collection of any programming language. Enumerator goes over every word and slices it into words and their frequencies.

In [1]: from nltk.tokenize import word\_tokenize

In [2]: from collections import Counter

In [3]: Counter(word\_tokenize(

"""The cat is in the box. The cat likes the box.

The box is over the cat."""))

Out[3]:

Counter({'.': 3,

'The': 3,

'box': 3,

'cat': 3,

'in': 1,

...

'the': 3})

Counter object has also an important function which is called as most\_common(top n tuples) which returns word with their frequencies. Also note that in parameter we specify that how much words & their frequency we need in view.

In [4]: counter.most\_common(2)

Out[4]: [('The', 3), ('box', 3)]

**Coding exercise**

**Bag-of-words picker**

It's time for a quick check on your understanding of bag-of-words. Which of the below options, with basic nltk tokenization, map the bag-of-words for the following text?

"The cat is in the box. The cat box."

##### Possible Answers

* 

('the', 3), ('box.', 2), ('cat', 2), ('is', 1)

* 

('The', 3), ('box', 2), ('cat', 2), ('is', 1), ('in', 1), ('.', 1)

* 

('the', 3), ('cat box', 1), ('cat', 1), ('box', 1), ('is', 1), ('in', 1)

* ('The', 2), ('box', 2), ('.', 2), ('cat', 2), ('is', 1), ('in', 1), ('the', 1)

**# Import Counter**

**from collections import Counter**

In this exercise, you'll build your first (in this course) bag-of-words counter using a Wikipedia article, which has been pre-loaded as article. Try doing the bag-of-words without looking at the full article text, and guessing what the topic is! If you'd like to peek at the title at the end, we've included it as article\_title. Note that this article text has had very little preprocessing from the raw Wikipedia database entry.

In this exercise, you'll build your first (in this course) bag-of-words counter using a Wikipedia article, which has been pre-loaded as article. Try doing the bag-of-words without looking at the full article text, and guessing what the topic is! If you'd like to peek at the title at the end, we've included it as article\_title. Note that this article text has had very little preprocessing from the raw Wikipedia database entry.

Article : need to be preloaded with text.

word\_tokenize has need to be imported by you.

**# Tokenize the article: tokens**

**tokens = word\_tokenize(article)**

**# Convert the tokens into lowercase: lower\_tokens**

**lower\_tokens = [t.lower() for t in tokens]**

**# Create a Counter with the lowercase tokens: bow\_simple**

**bow\_simple = Counter(lower\_tokens)**

**# Print the 10 most common tokens**

**print(bow\_simple.most\_common(10))**

**Simple text preprocessing**

**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

In [1]: from ntlk.corpus import stopwords

In [2]: text = """The cat is in the box. The cat likes the box.

The box is over the cat."""

In [3]: tokens = [w for w in word\_tokenize(text.lower())

if w.isalpha()]

In [4]: no\_stops = [t for t in tokens

if t not in stopwords.words('english')]

In [5]: Counter(no\_stops).most\_common(2)

Out[5]: [('cat', 3), ('box', 3)]

# Text preprocessing practice

You'll need to remove stop words and non-alphabetic characters, lemmatize, and perform a new bag-of-words on your cleaned text.

# Import WordNetLemmatizer

from nltk.stem import WordNetLemmatizer

from nltk.corpus import stopwords

# Retain alphabetic words: alpha\_only

alpha\_only = [t for t in lower\_tokens if t.isalpha()]

# Remove all stop words: no\_stops

no\_stops = [t for t in alpha\_only if t not in english\_stops]

# Instantiate the WordNetLemmatizer

wordnet\_lemmatizer = WordNetLemmatizer()

# Lemmatize all tokens into a new list: lemmatized

lemmatized = [wordnet\_lemmatizer.lemmatize(t) for t in no\_stops]

# Create the bag-of-words: bow

bow = Counter(lemmatized)

# Print the 10 most common tokens

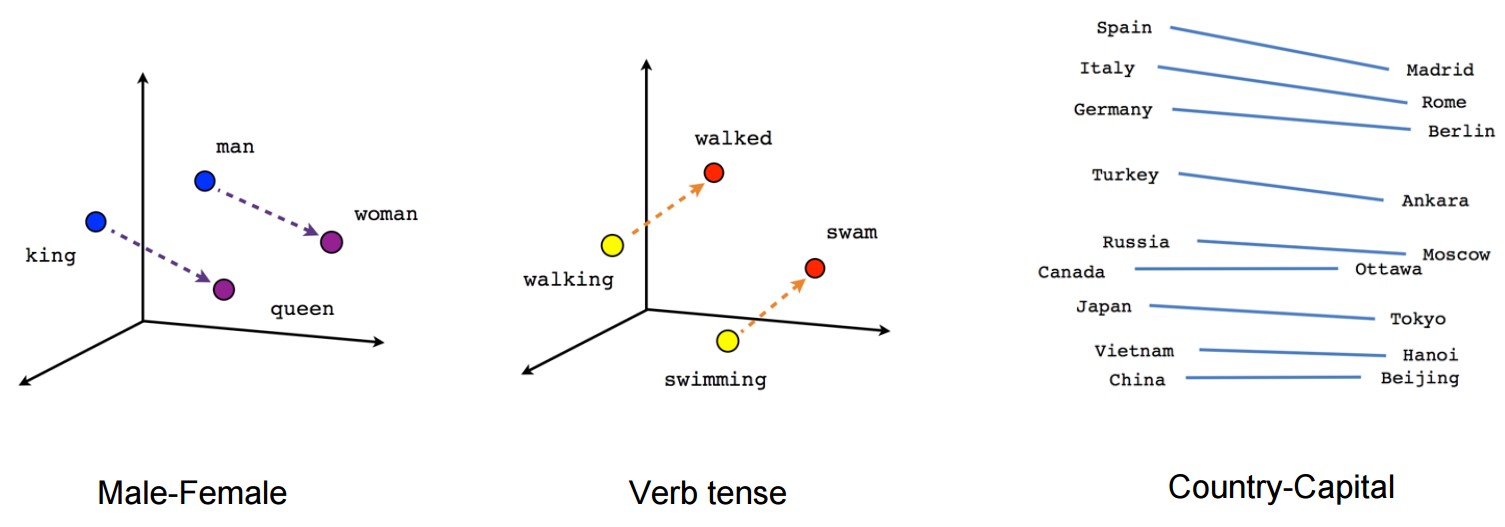
print(bow.most\_common(10))

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



* Import Dictionary from gensim.corpora.dictionary.
* Initialize a gensim Dictionary with the tokens in articles.
* Obtain the id for "computer" from dictionary. To do this, use its .token2id method which returns ids from text, and then chain .get() which returns tokens from ids. Pass in "computer" as an argument to .get().
* Use a list comprehension in which you iterate over articles to create a gensim MmCorpus from dictionary.
  + In the output expression, use the .doc2bow() method on dictionary with article as the argument.

# 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])