Chapter 3

Simple Topic Identification #1

Outline

- Word counts with **bag-of-words**
- Simple text preprocessing

Word counts with bag-of-words

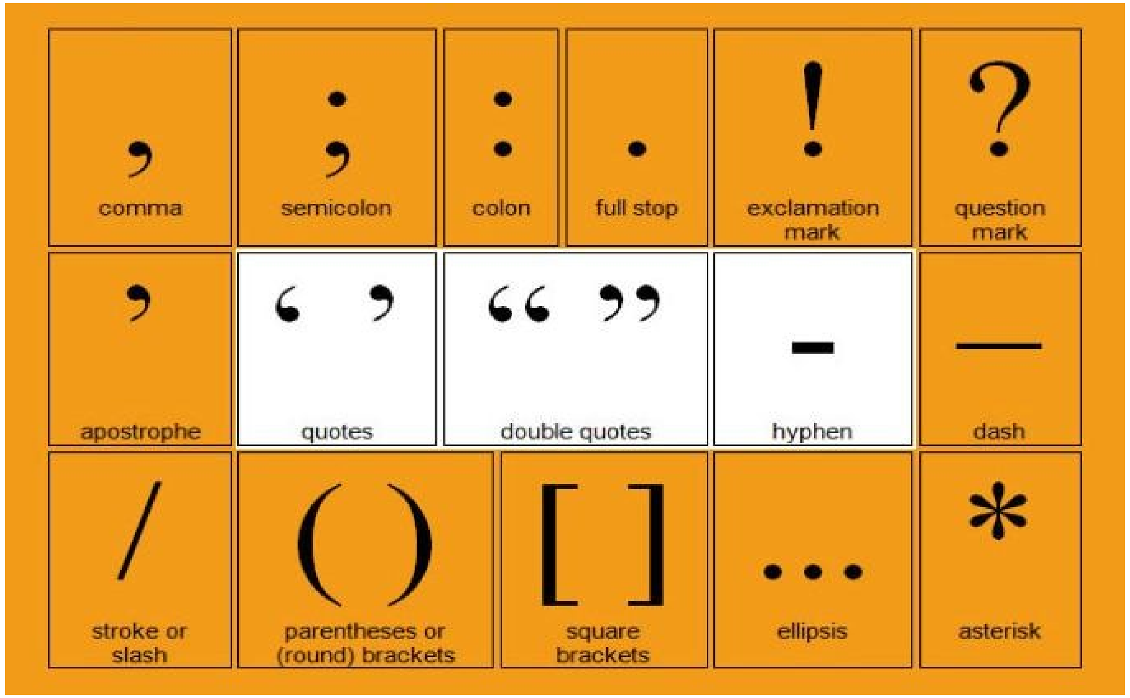
Bag-of-words

- Bag of words is a **basic method** to **finding topics in a text**.
- First, create tokens using tokenization, then count up all the tokens.
- **More frequent** a word, it might be the **more important** in a text.
- Bag of words can be a great way to determine the significant words in a text.

Example of Bag-of-words

- Text: The cat is in the box. The cat likes the box. The box is over the cat.
- Bag of words (stripped punctuation):

Words	Frequency
The	3
cat	3
is	1
in	1
the	3
box	3
likes	1
over	1



If we added a preprocessing step to handle this issue, we could lowercase all of the words, so each word is counted only once.

Bag-of-words in Python

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

```
Counter({'The': 3, 'cat': 3, 'the': 3, 'box': 3, '.': 3, 'is': 2, 'in': 1, 'likes': 1, 'over': 1})
```

```
count = counter.most_common(2)
print(count)
```

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

Let’s practice!

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
  a) ('the', 3), ('box.', 2), ('cat', 2), ('is', 1)
  b) ('The', 3), ('box', 2), ('cat', 2), ('is', 1), ('in', 1), ('.', 1)
  c) ('the', 3), ('cat box', 1), ('cat', 1), ('box', 1), ('is', 1), ('in', 1)
  d) ('The', 2), ('box', 2), ('.', 2), ('cat', 2), ('is', 1), ('in', 1), ('the', 1)

Building a Counter with bag-of-words

- build the first bag-of-words counter using a Wikipedia article.
- Try doing the bag-of-words without looking at the full article text, and guessing what the topic is!
- Import **Counter** from **collections** .

```
# Import Counter
from collections import Counter
```

- Load a Wikipedia article (txt file) as **article**.

```
#Read TXT file
f = open(“wiki_article.txt", "r")
article = f.read()
```

Building a Counter with bag-of-words

- Use **word_tokenize()** to split the article into tokens.

```
# Tokenize the article: tokens
tokens = ________________(_________)
```

- Use a list comprehension with **t** as the iterator variable to convert all the tokens into lowercase. The **.lower()** method converts text into lowercase.

```
# Convert the tokens into lowercase: lower_tokens
lower_tokens = [__.________() for ___ in tokens]
```

Building a Counter with bag-of-words

- Create a bag-of-words counter called **bow_simple** by using **Counter()** with **lower_tokens** as the argument.

```
# Create a Counter with the lowercase tokens: bow_simple
bow_simple = ___________(_____________)
```

- Use the **.most_common()** method of **bow_simple** to print the 10 most common tokens.

```
# Print the 10 most common tokens
print(_______________._____________(___))
```

```
[(',', 151), ('the', 150), ('.', 89), ('of', 81), ("''", 69), ('to', 63), ('a', 60), ('``', 47), ('in', 44), ('and', 41)]
```

Simple text preprocessing

Why preprocess?

- **Text processing** helps make for better input data when performing machine learning or other statistical methods.
- For example:
  - Use Tokenization to create a bag of words
  - Lowercasing words
- **Lemmatization or stemming**: shorten the words to their root stems.
- **Removing** stop words, punctuation, or unwanted tokens.
- Good to experiment with different approaches

Preprocessing Example

- Example of input text:

Dogs, cats, and birds are pets. Fish and rabbit is also pets.

- Output tokens:

dog, cat, bird, fish, rabbit

Text preprocessing with Python

```
import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

text = """The cat is in the box. The cat likes the box. The box is
over the cat in the house."""
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')]
print(Counter(no_stops).most_common(2))
```

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

Let’s practice!

Text preprocessing steps

- Which of the following are useful **text preprocessing steps**?

  Possible Answers

  a) Stems, spelling corrections, lowercase.
  b) Lemmatization, lowercasing, removing unwanted tokens.
  c) Removing stop words, leaving in capital words.
  d) Strip stop words, word endings and digits.

Text preprocessing practice

- Clean up text for better NLP results.
  - remove stop words and non-alphabetic characters, lemmatize, and perform a new bag-of-words on your cleaned text.

```
[(',', 151), ('the', 150), ('.', 89), ('of', 81), ("''", 69), ('to', 63), ('a', 60), ('``', 47),
('in', 44), ('and', 41)]
```

```
[('debugging', 39), ('system', 25), ('bug', 17), ('software', 16), ('problem', 15),
('tool', 15), ('computer', 14), ('process', 13), ('term', 13), ('debugger', 13)]
```

Text preprocessing practice

- Use **lower_tokens** and Counter from last practice.
- Import the **WordNetLemmatizer** class from **nltk.stem**.

```
# Import WordNetLemmatizer
from ____________ import ___________________
```

- Create a list **alpha_only** that contains only alphabetical characters using the **.isalpha()** method.

```
# Retain alphabetic words: alpha_only
alpha_only = [t for t in lower_tokens if t.____________()]
```

Text preprocessing practice

- Create a list called **no_stops** consisting of words from **alpha_only** that are not contained in **english_stops**.

```
# Remove all stop words: no_stops
no_stops = [t for t in _______________ if t not in
_____________._______(____________)]
```

- Initialize a **WordNetLemmatizer** object called **wordnet_lemmatizer**

```
# Instantiate the WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
```

Text preprocessing practice

- Use its **.lemmatize()** method on the tokens in **no_stops** to create a new list called **lemmatized**.

```
# Lemmatize all tokens into a new list: lemmatized
lemmatized = [wordnet_lemmatizer.________(t) for t in ________]
```

- Create a new **Counter** called **bow** with the lemmatized words. Print the 10 most common tokens.

```
# Create the bag-of-words: bow
bow = Counter(__________)

# Print the 10 most common tokens
print(bow.___________(10))
```

Questions

Reference: https://app.datacamp.com/learn