Text preprocessing

- NLTK: natural language toolkit

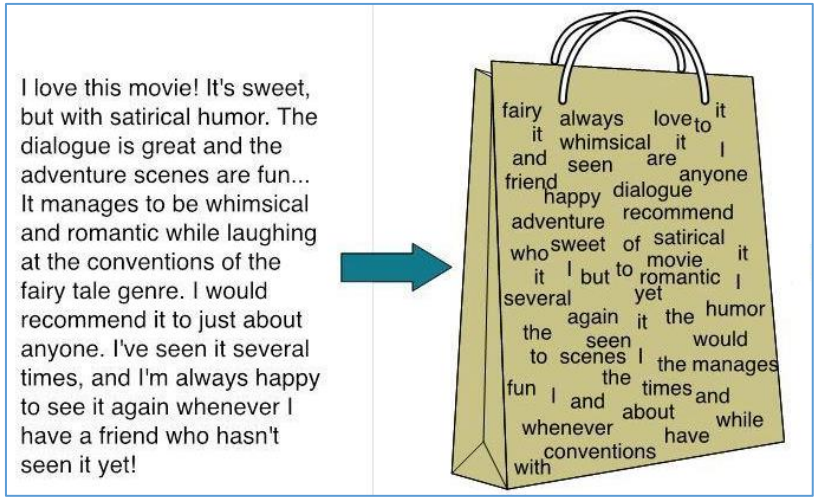
Basic Preprocessing Text

- Sentence Tokenization
- Word Tokenization
- Text Lemmatization and Stemming
- Stop Words
- Regex

Text to Vector Representations

Bag-of-words

A **bag-of-words** model describes the occurrence of each word within a document.



To use this model, we need to:

- Design a **vocabulary** of known words (also called **tokens**)
- Choose a **measure of the presence** of known words
- Any information about **the order** or **structure** of words **is discarded**.

- unigram (1-gram):

a	swimmer	likes	swimming	thus	he	swims

- bigram (2-gram):

a swimmer	swimmer likes	likes swimming	swimming thus	...

- trigram (3-gram):

a swimmer likes	swimmer likes swimming	likes swimming thus	...

A bag-of-n-grams model represents a text document as an unordered collection of its n-grams.

Word frequency

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

```
corpus = [ 'This is the first document.',  'This document is the second
document.',  'And this is the third one.',  'Is this the first document?']
```

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
vectorizer.get_feature_names()
```

CountVectorizer: convert a collection of text documents to a matrix of token counts

```
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

```
X.toarray()
```

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

sklearn.feature_extraction.text.**CountVectorizer**
(*lowercase=True, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word',max_df, min_df, vocabulary=None*)

TF-IDF

One problem with **scoring word frequency** is that the most frequent words in the document start to have the highest scores. These frequent words may not contain as much "**informational gain**" to the model compared with some rarer and domain-specific words.

TF-IDF, short for **term frequency-inverse document frequency**

$$TF(\boldsymbol{term}) = \frac{Number\ of\ times\ \boldsymbol{term}\ appears\ in\ a\ document}{Total\ number\ of\ items\ in\ the\ document}$$

$$IDF(\boldsymbol{term}) = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ \boldsymbol{term}\ in\ it}\right)$$

$$TFIDF(\boldsymbol{term}) = TF(\boldsymbol{term}) * \mathrm{IDF}(\boldsymbol{term})$$

Parameters are similar to CounVectorizer

sklearn.feature_extraction.text.**TfidfVectorizer**()

Examples of Text Classification

Examples of text classification

- ✓ Topic identification (text classification): Is this news article about Politics, Sports, or Technology?
- ✓ Spam detection: Is this email a spam or not?
- ✓ Sentiment analysis: Is this movie review positive or negative?