# Typical Pre-processing for Text Data ¶

1. **Tokenization**: given a text, this will separate it into individual words.
2. **Normalization**: convert text into all lowercase, spelling mistake correction, etc.
3. **Cleaning**: remove unwanted parts, e.g., punctuation, stop words, etc.
4. **Lemmatization/stemming**: convert individual words to the corresponding 'root word'. There is a difference between 'lemmatization' & 'stemming', you may check in some references if you want to know further.

## Tokenization

In [1]:
```
import nltk
from nltk.tokenize import word_tokenize

text1 = "After watching two hours non stop, \
        he says that the film is really fantastic #brilliant."
text2 = "Foods sold there are little bit pricy, \
        meanwhile the taste is not delicious #notrecommended."

tokens1 = word_tokenize(text1)
print("tokens1:\n", tokens1)

tokens2 = word_tokenize(text2)
print("\n\ntokens2:\n", tokens2)
```

```
tokens1:
 ['After', 'watching', 'two', 'hours', 'non', 'stop', ',', 'he', 'says', 'tha
t', 'the', 'film', 'is', 'really', 'fantastic', '#', 'brilliant', '.']


tokens2:
 ['Foods', 'sold', 'there', 'are', 'little', 'bit', 'pricy', ',', 'meanwhile',
'the', 'taste', 'is', 'not', 'delicious', '#', 'notrecommended', '.']
```

## Normalization

In this block of code, we try one of normalization processes: converting to lowercase.

In [2]:
```
# convert to lower case
normalized_words1 = [w.lower() for w in tokens1]
print("normalized_words1:\n", normalized_words1)

normalized_words2 = [w.lower() for w in tokens2]
print("\n\nnormalized_words2:\n", normalized_words2)
```

```
normalized_words1:
 ['after', 'watching', 'two', 'hours', 'non', 'stop', ',', 'he', 'says', 'tha
t', 'the', 'film', 'is', 'really', 'fantastic', '#', 'brilliant', '.']


normalized_words2:
 ['foods', 'sold', 'there', 'are', 'little', 'bit', 'pricy', ',', 'meanwhile',
'the', 'taste', 'is', 'not', 'delicious', '#', 'notrecommended', '.']
```

## Cleaning 01: remove punctuation

In [3]:
```
# remove punctuation from each word
import string
table = str.maketrans('', '', string.punctuation)
punc_removed1 = [w.translate(table) for w in normalized_words1]
print("punc_removed1:\n", punc_removed1)

punc_removed2 = [w.translate(table) for w in normalized_words2]
print("\n\npunc_removed2:\n", punc_removed2)
```

```
punc_removed1:
 ['after', 'watching', 'two', 'hours', 'non', 'stop', '', 'he', 'says', 'that',
'the', 'film', 'is', 'really', 'fantastic', '', 'brilliant', '']


punc_removed2:
 ['foods', 'sold', 'there', 'are', 'little', 'bit', 'pricy', '', 'meanwhile',
'the', 'taste', 'is', 'not', 'delicious', '', 'notrecommended', '']
```

## Cleaning 02: remove not alphabetic

In [4]:
```
# remove remaining tokens that are not alphabetic
isalpha_words1 = [word for word in punc_removed1 if word.isalpha()]
print("isalpha_words1:\n", isalpha_words1)

isalpha_words2 = [word for word in punc_removed2 if word.isalpha()]
print("\n\nisalpha_words2:\n", isalpha_words2)
```

```
isalpha_words1:
 ['after', 'watching', 'two', 'hours', 'non', 'stop', 'he', 'says', 'that', 'th
e', 'film', 'is', 'really', 'fantastic', 'brilliant']


isalpha_words2:
 ['foods', 'sold', 'there', 'are', 'little', 'bit', 'pricy', 'meanwhile', 'th
e', 'taste', 'is', 'not', 'delicious', 'notrecommended']
```

## Cleaning 03: remove stop words

In [5]:
```
# filter out stop words
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
#print("stop_words:\n", stop_words, "\n") #print this if you want
                                          #to see stop word examples

stopWords_removed1 = [w for w in isalpha_words1 if not w in stop_words]
print("stopWords_removed1:\n", stopWords_removed1)

stopWords_removed2 = [w for w in isalpha_words2 if not w in stop_words]
print("\n\nstopWords_removed2:\n", stopWords_removed2)
```

```
stopWords_removed1:
 ['watching', 'two', 'hours', 'non', 'stop', 'says', 'film', 'really', 'fantast
ic', 'brilliant']


stopWords_removed2:
 ['foods', 'sold', 'little', 'bit', 'pricy', 'meanwhile', 'taste', 'delicious',
'notrecommended']
```

## Stemming

In [6]:
```
from nltk.stem import PorterStemmer
ps = PorterStemmer()

stemmed_word1 = [ps.stem(w) for w in stopWords_removed1]
print("stemmed_word1:\n", stemmed_word1)

stemmed_word2 = [ps.stem(w) for w in stopWords_removed2]
print("\n\nstemmed_word2:\n", stemmed_word2)
```

```
stemmed_word1:
 ['watch', 'two', 'hour', 'non', 'stop', 'say', 'film', 'realli', 'fantast', 'b
rilliant']


stemmed_word2:
 ['food', 'sold', 'littl', 'bit', 'prici', 'meanwhil', 'tast', 'delici', 'notre
commend']
```

## Lemmatization

In [7]:
```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

lemmatized_words1 = [lemmatizer.lemmatize(w) for w in stopWords_removed1]
print("lemmatized_words1:\n", lemmatized_words1)

lemmatized_words2 = [lemmatizer.lemmatize(w) for w in stopWords_removed2]
print("\n\nlemmatized_words2:\n", lemmatized_words2)
```

```
lemmatized_words1:
 ['watching', 'two', 'hour', 'non', 'stop', 'say', 'film', 'really', 'fantasti
c', 'brilliant']


lemmatized_words2:
 ['food', 'sold', 'little', 'bit', 'pricy', 'meanwhile', 'taste', 'delicious',
'notrecommended']
```

# Example of Converting Preprocessed Text into Numerical Features

In [8]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

# merge two texts into one list (you may also try to use the stemmed_word)
two_preprocessed_text = [lemmatized_words1, lemmatized_words2]

# define the tfidf vectorizer
def dummy(doc):
    return doc
tfidf = TfidfVectorizer(
    analyzer='word',
    tokenizer=dummy,
    preprocessor=dummy,
    token_pattern=None)

# train / learn from the given data
model = tfidf.fit(two_preprocessed_text)
# transform to numerical features using the trained model
numerical_features = model.transform(two_preprocessed_text).toarray()
""" --> these numerical features can then be used for mathematical model,
        e.g., classification to sentiment class: positive and negative.
"""

print("numerical_features of text1:\n", numerical_features[0],
      "; shape:", numerical_features[0].shape)
print("\n\nnumerical_features of text2:\n", numerical_features[1],
      "; shape:", numerical_features[1].shape)
```

```
numerical_features of text1:
 [0.         0.31622777 0.         0.31622777 0.31622777 0.
 0.31622777 0.         0.         0.31622777 0.         0.
 0.31622777 0.31622777 0.         0.31622777 0.         0.31622777
 0.31622777] ; shape: (19,)


numerical_features of text2:
 [0.33333333 0.         0.33333333 0.         0.         0.33333333
 0.         0.33333333 0.33333333 0.         0.33333333 0.33333333
 0.         0.         0.33333333 0.         0.33333333 0.
 0.        ] ; shape: (19,)
```

# Question 01 (Q01)

What is/are the difference(s) between stemming and lemmatization?

**Answer:**
[write your answer here, can use Bahasa]

# Question 02 (Q02)

Please explain what TF-IDF is!
***Note***: (i) you can insert picture (if you want) in the answer, and then upload all the materials (this ipynb file and the pictures) into one zip file to the course portal, (ii) you can also use mathematical

equation here, for exampe: you can write $log_2(P_i)$ by using `$log_{2}(P_{i})$` .

**Answer:**
[write your answer here]

## (Bonus) Question 03 (Q03)

What are other methods that can be used to convert "preprocessed text" to "numerical features" other than TF-IDF?

**Answer:**
[write your answer here]

In [ ]: