

## Week 03 Handson - Data Preprocessing #02 (Text)

To use a learning model, we need to input numerical data to the model. However, we often get non-numerical data as input, e.g., text data. Thus, to use text as input to the learning model, we need to do pre-processing and convert it to numerical data.

Steps below are typical pre-processing steps for text data.

1. Tokenization
2. Normalization
3. Cleaning
4. Lemmatization/stemming

### Tokenization

```
In [1]: # please install nltk in your python env.
# if you use pip, you type: pip install nltk
import nltk
from nltk.tokenize import word_tokenize

# we have two raw texts here that we want to pre-process
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',
'that', 'the', 'film', 'is', 'really', 'fantastic', '#', 'brilliant',
'.']

tokens2:
['Foods', 'sold', 'there', 'are', 'little', 'bit', 'pricy', ',', 'meanwh
ile', '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',
'that', 'the', 'film', 'is', 'really', 'fantastic', '#', 'brilliant',
'.']

normalized_words2:
['foods', 'sold', 'there', 'are', 'little', 'bit', 'pricy', ',', 'meanwh
ile', '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', '', 'meanwhi
le', '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', 'tha
t', 'the', 'film', 'is', 'really', 'fantastic', 'brilliant']

isalpha_words2:
['foods', 'sold', 'there', 'are', 'little', 'bit', 'pricy', 'meanwhile',
'the', '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', 'f
antastic', 'brilliant']

stopWords_removed2:
['foods', 'sold', 'little', 'bit', 'pricy', 'meanwhile', 'taste', 'delic
ious', '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', 'fantas
t', 'brilliant']

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

## 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', 'fan
tastic', 'brilliant']

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

## Example of Converting Preprocessed Text into Numerical Features

```
In [ ]: 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 by the model,
        e.g., for 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)
```

## Question 01 (Q01)

- What are tokenization, normalization and cleaning?
- What is/are the difference(s) between stemming and lemmatization?

**Answer:**

[write your answers here, can use Bahasa]

## 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 example: you can write  $\log_2(P_i)$  by using `$\log_{2}(P_{i})$`.

**Answer:**

[write your answer here]

## (Bonus) Q03



What are other methods that can be used to convert "preprocessed text" to "numerical features" other than TF-IDF? From what you mention, what are methods that keep the semantic?

**Answer:**

[write your answer here]