Working with Text Data

Dr. Yuzana Win (Nagasaki University, Japan)
Lecturer
Department of Computer Engineering and Information Technology

Lecture Objectives

- To introduce
  - Feature extraction using text data
  - Pre-processing of text data
  - Performing Sentiment Analysis using Text Classification

Feature Extraction

- Use twitter sentiment dataset from the datahack platform

```
train = pd.read_csv('train_E6oV3lV.csv')
```

- Number of words
  - Extract the number of words in each tweet by using split function in python

```
train['word_count'] = train['tweet'].apply(lambda x: len(str(x).split(" ")))
train[['tweet','word_count']].head()
```

	tweet	word_count
0	@user when a father is dysfunctional and is s...	21
1	@user @user thanks for #lyft credit i can't us...	22
2	bihday your majesty	5
3	#model i love u take with u all the time in ...	17
4	factsguide: society now #motivation	8

Feature Extraction

- Number of characters
  - By calculating the length of the tweet

```
train['char_count'] = train['tweet'].str.len() ## this also includes spaces
train[['tweet','char_count']].head()
```

	tweet	char_count
0	@user when a father is dysfunctional and is s...	102
1	@user @user thanks for #lyft credit i can't us...	122
2	bihday your majesty	21
3	#model i love u take with u all the time in ...	86
4	factsguide: society now #motivation	39

Feature Extraction

- Average Word Length
  - Sum of the length of all the words and divide it by the total length of the tweet

```
def avg_word(sentence):
  words = sentence.split()
  return (sum(len(word) for word in words)/len(words))

train['avg_word'] = train['tweet'].apply(lambda x: avg_word(x))
train[['tweet','avg_word']].head()
```

	tweet	avg_word
0	@user when a father is dysfunctional and is s...	4.555556
1	@user @user thanks for #lyft credit i can't us...	5.315789
2	bihday your majesty	5.666667
3	#model i love u take with u all the time in ...	4.928571
4	factsguide: society now #motivation	8.000000

CEIT
Think smart, be different.

Feature Extraction

- Number of stopwords
  - Use nltk toolkit which is a basic NLP library in python

```
from nltk.corpus import stopwords
stop = stopwords.words('english')


train['stopwords'] = train['tweet'].apply(lambda x: len([x for x in x.split() if x
in stop]))
train[['tweet','stopwords']].head()
```

	tweet	stopwords
0	@user when a father is dysfunctional and is s...	10
1	@user @user thanks for #lyft credit i can't us...	4
2	bihday your majesty	1
3	#model i love u take with u all the time in ...	5
4	factsguide: society now #motivation	1

Feature Extraction

- Number of special characters
  - Extract from a tweet is calculating the number of hashtags or mentions present in it
  - Helps in extra information from text data

```
train['hastags'] = train['tweet'].apply(lambda x: len([x for x in x.split() if x.st
artswith('#')]))
train[['tweet','hastags']].head()
```

	tweet	hastags
0	@user when a father is dysfunctional and is s...	1
1	@user @user thanks for #lyft credit i can't us...	3
2	bihday your majesty	0
3	#model i love u take with u all the time in ...	1
4	factsguide: society now #motivation	1

Feature Extraction

- Number of numerics
  - Calculate the number of numerics which are present in the tweets

```
train['numerics'] = train['tweet'].apply(lambda x: len([x for x in x.split() if x.i
sdigit()]))
train[['tweet','numerics']].head()
```

	tweet	numerics
0	@user when a father is dysfunctional and is s...	0
1	@user @user thanks for #lyft credit i can't us...	0
2	bihday your majesty	0
3	#model i love u take with u all the time in ...	0
4	factsguide: society now #motivation	0

CEIT
Think smart, be different.

Feature Extraction

- Number of Uppercase Words
  - Makes this is necessary operation to identify those words

```
train['upper'] = train['tweet'].apply(lambda x: len([x for x in x.split() if x.isup
per()]))
train[['tweet','upper']].head()
```

	tweet	upper
0	@user when a father is dysfunctional and is s...	0
1	@user @user thanks for #lyft credit i can't us...	0
2	bihday your majesty	0
3	#model i love u take with u all the time in ...	0
4	factsguide: society now #motivation	0

CEIT
Think smart, be different.

Pre-processing

- Lower case
  - To avoid having multiple copies of the same words

```
train['tweet'] = train['tweet'].apply(lambda x: " ".join(x.lower() for x in x.split
()))
train['tweet'].head()
```

```
0    @user when a father is dysfunctional and is so...
1    @user @user thanks for #lyft credit i can't us...
2                                  bihday your majesty
3    #model i love u take with u all the time in ur...
4                  factsguide: society now #motivation
Name: tweet, dtype: object
```

CEIT
Think smart, be different.

Pre-processing

- Removing Punctuation
  - Remove all unnecessary instances will help to reduce the size of the training

```
train['tweet'] = train['tweet'].str.replace('[^\w\s]','')

train['tweet'].head()
```

```
0    user when a father is dysfunctional and is so ...
1    user user thanks for lyft credit i cant use ca...
2                                  bihday your majesty
3    model i love u take with u all the time in urð...
4                  factsguide society now motivation
Name: tweet, dtype: object
```

CEIT
Think smart, be different.

Pre-processing

- Removal of Stopwords
  - Stopwords means function words (i.e., punctuation, pronoun, interjection words and so on) should be removed from the text data.
  - We can create a list of stopwords as you wish

```
from nltk.corpus import stopwords

stop = stopwords.words('english')

train['tweet'] = train['tweet'].apply(lambda x: " ".join(x for x in x.split() if x

not in stop))

train['tweet'].head()
```

```
0    user father dysfunctional selfish drags kids d...
1    user user thanks lyft credit cant use cause do...
2                                      bihday majesty
3                model love u take u time urð ðððð ðð
4                        factsguide society motivation
Name: tweet, dtype: object
```

CEIT
Think smart, be different.

Pre-processing

- Common word removal
  - Remove commonly occurring words in a general sense

```
freq = pd.Series(' '.join(train['tweet']).split()).value_counts()[:10]
freq
```

```
> user     17473
love       2647
ð          2511
day        2199
â          1797
happy      1663
amp        1582
im         1139
u          1136
time       1110
dtype: int64
```

```
freq = list(freq.index)
train['tweet'] = train['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
train['tweet'].head()
```

```
0    father dysfunctional selfish drags kids dysfun...
1    thanks lyft credit cant use cause dont offer w...
2                                     bihday majesty
3                          model take urð ðððð ðð
4                     factsguide society motivation
Name: tweet, dtype: object
```

CEIT
Think smart, be different.

Pre-processing

- Rare words removal
  - Remove rarely occurring words from the text

```
freq = pd.Series(' '.join(train['tweet']).split()).value_counts()[-10:]
freq
```

```
> tvperfect          1
oau                  1
850am                1
semangatpagi         1
kindestbravest       1
moodyah              1
downhill             1
loreal               1
ohwhatcoulditbe      1
maannnn              1
dtype: int64
```

```
freq = list(freq.index)
train['tweet'] = train['tweet'].apply(lambda x: " ".join(x for x in x.split() if x
not in freq))
train['tweet'].head()
```

```
0    father dysfunctional selfish drags kids dysfun...
1    thanks lyft credit cant use cause dont offer w...
2                                      bihday majesty
3                              model take urð ðððð ðð
4                         factsguide society motivation
Name: tweet, dtype: object
```

CEIT
Think smart, be different.

Pre-processing

- Tokenization
  - Tokenization is the first step in text analysis
  - Tokenization refers to dividing the text into a sequence of words or sentences.

Sentence Tokenization

```
from nltk.tokenize import sent_tokenize
text="""Hello Mr. Smith, how are you doing today? The weather is great, and city is awesome.
The sky is pinkish-blue. You shouldn't eat cardboard"""
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['Hello Mr. Smith, how are you doing today?', 'The weather is great, and city is awesome.',
```

CEIT
Think smart, be different.

Pre-processing

- Word Tokenization
  - Word tokenizer breaks text paragraph into words

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

- Output

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather',
'is', 'great', ',', 'and', 'city', 'is', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '
'.', 'You', 'should', "n't", 'eat', 'cardboard']
```

CEIT
Think smart, be different.

Pre-processing

- Stemming
  - Is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes

```
# Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize


ps = PorterStemmer()


stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))


print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)
```

Pre-processing

- Output

```
Filtered Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?']
Stemmed Sentence: ['hello', 'mr.', 'smith', ',', 'today', '?']
```

CEIT
Think smart, be different.

Pre-processing

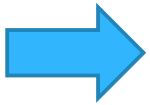
- Lemmatization
  - Reduces words to their base word, which is linguistically correct lemmas
  - Transfoms root word with the use of vocabulary and morphological analysis
  - Lemmatization is usually more sophisticated than stemming

```
#Lexicon Normalization
#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()

word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))
```



```
Lemmatized Word: fly
Stemmed Word: fli
```

CEIT
Think smart, be different.

Pre-processing

- POS-Tagging
  - Part-of-Speech (POS) tagging is to identify the grammatical group of a given word
  - It is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc
  - POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word

```
sent = "Albert Einstein was born in Ulm, Germany in 1879."

tokens=nltk.word_tokenize(sent)
print(tokens)
```

```
['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879', '.']
```

CEIT
Think smart, be different.

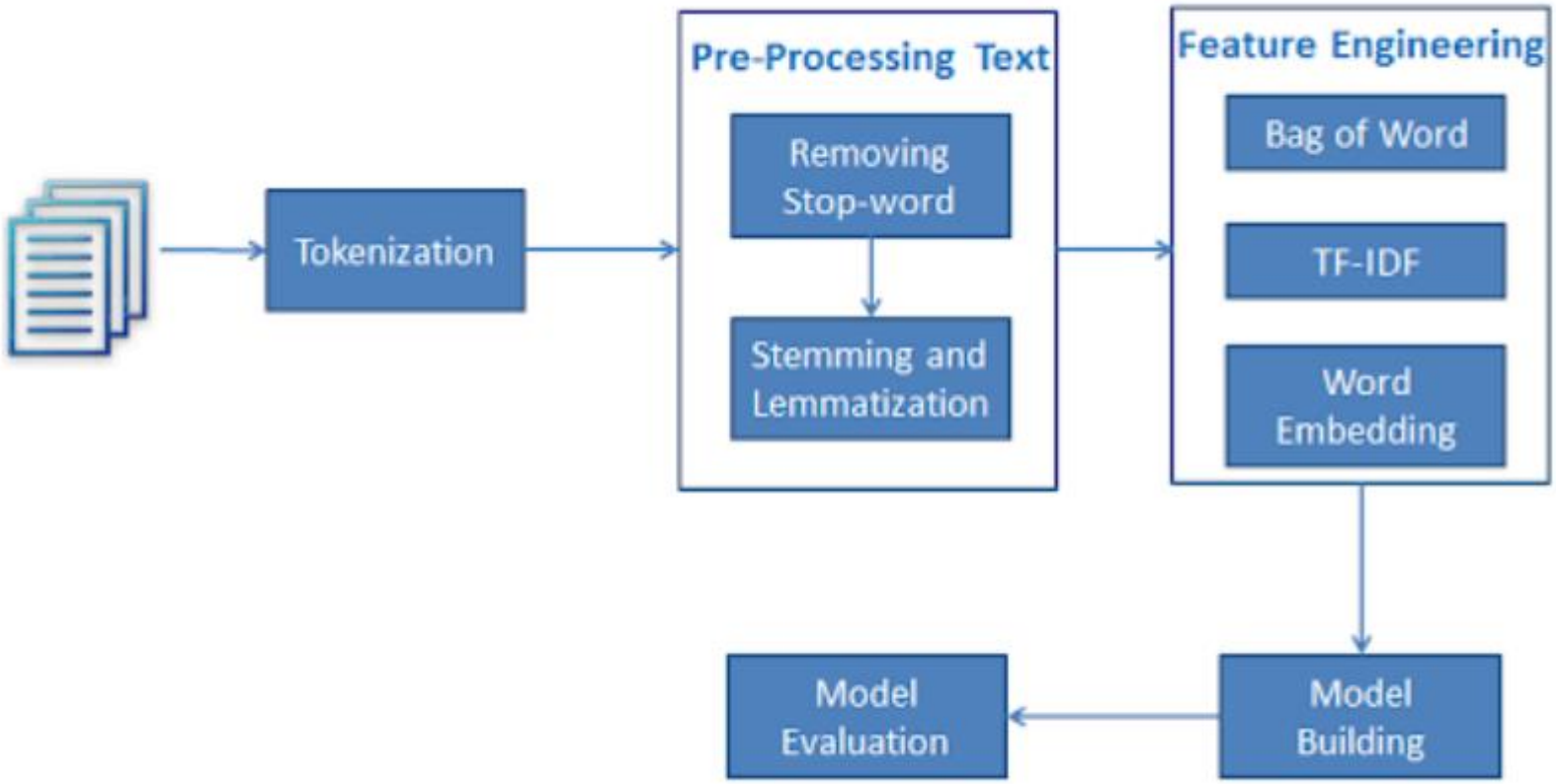
Pre-processing

- POS-Tagging

```
nltk.pos_tag(tokens)
```

```
[('Albert', 'NNP'),
 ('Einstein', 'NNP'),
 ('was', 'VBD'),
 ('born', 'VBN'),
 ('in', 'IN'),
 ('Ulm', 'NNP'),
 (',', ','),
 ('Germany', 'NNP'),
 ('in', 'IN'),
 ('1879', 'CD'),
 ('.', '.')]
```

Performing Sentiment Analysis using Text Classification



Performing Sentiment Analysis using Text Classification

- Loading Data
  - use the "Sentiment Analysis of Movie, Reviews" dataset available on Kaggle
  - This data has 5 sentiment labels:
    - 0 - negative 1 - somewhat negative 2 - neutral 3 - somewhat positive 4 - positive

```
# Import pandas

import pandas as pd

data=pd.read_csv('train.tsv', sep='\t')

data.head()
```

Performing Sentiment Analysis using Text Classification

- Output

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

CEIT
Think smart, be different.

Performing Sentiment Analysis using Text Classification

```
data.info()
```

- Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156060 entries, 0 to 156059
Data columns (total 4 columns):
PhraseId      156060 non-null int64
SentenceId    156060 non-null int64
Phrase        156060 non-null object
Sentiment     156060 non-null int64
dtypes: int64(3), object(1)
memory usage: 4.8+ MB
```

```
data.Sentiment.value_counts()
```

- Output

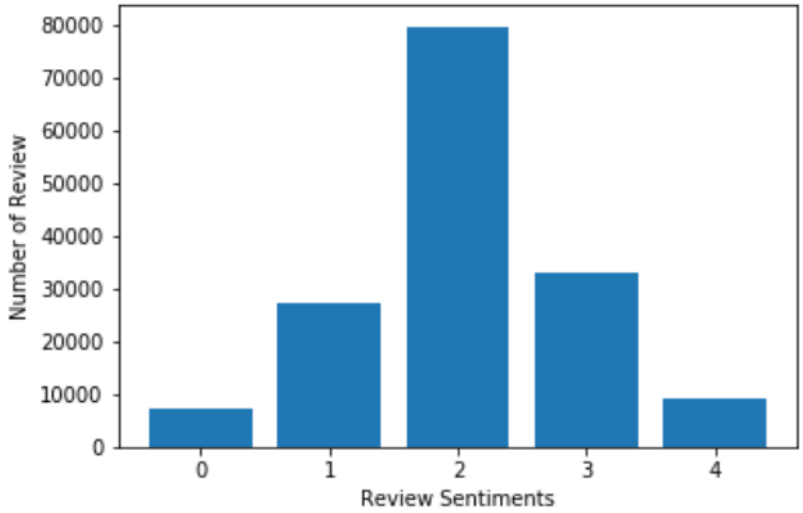
```
2    79582
3    32927
1    27273
4     9206
0     7072
Name: Sentiment, dtype: int64
```

CEIT
Think smart, be different.

Performing Sentiment Analysis using Text Classification

```
Sentiment_count=data.groupby('Sentiment').count()
plt.bar(Sentiment_count.index.values, Sentiment_count['Phrase'])
plt.xlabel('Review Sentiments')
plt.ylabel('Number of Review')
plt.show()
```

- Output



References

- https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/
- https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk