

# Hw01-Tutorial-TextProcessing

## Tutorial: Bag-of-Words for Text Processing

We have the following objectives in this practicum:

1. This tutorial aims to practice one application in the field of natural language processing (NLP), called the sentiment analysis. This is also the text classification problem, similar to other applications such as spam email filter (i.e., detecting spam email).
2. Train a machine learning model to classify the text/messages into Positive (+1) or Negative (0).
3. To train the model, we need to prepare the raw text to get training dataset.



**Before the tutorial, it is highly recommended to review the following course materials:**

**Slide:** [Lecture8\\_TextProcessing\\_FeatureExtraction.pdf](#)

(<https://canvas.slu.edu/courses/68524/files/6037962?wrap=1>)

**Lecture note:** [Lecture8\\_TextProcessing\\_FeatureExtraction\\_notes.pdf](#)


(<https://canvas.slu.edu/courses/68524/files/6037963?wrap=1>)

## Step 1: Objective Definition

Data source:

<https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences>

(<https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences>)



## Sentiment Labeled Sentences

Donated on 5/29/2015

The dataset contains sentences labelled with positive or negative sentiment.


Dataset Characteristics	Subject Area	Associated Tasks
Text	Other	Classification

Feature Type	# Instances	# Features
-	3000	-

**1 citations**

**23883 views**

**Creators**

 Dimitrios Kotzias

**DOI**


10.24432/C57604

**License**

**Dataset Information**

**Additional Information**

This dataset was created for the Paper 'From Group to Individual Labels using Deep Features', Kotzias et. al., KDD 2015  
Please cite the paper if you want to use it :)

The repository contains three manually labeled dataset of reviews from the IMDb, Amazon, and Yelp datasets. Details for each dataset has been described in [From Group to Individual Labels using Deep Features](#) 

([https://dl.acm.org/doi/pdf/10.1145/2783258.2783380?casa\\_token=G3b2uvz-TqsAAAAA:JGqry1jYbYKWQk0TVVgEERZkEREI\\_CeH6T-MKnhTyWwCTB7sSKqknm0wZV2Ki5MiMbHaHeelGID-](https://dl.acm.org/doi/pdf/10.1145/2783258.2783380?casa_token=G3b2uvz-TqsAAAAA:JGqry1jYbYKWQk0TVVgEERZkEREI_CeH6T-MKnhTyWwCTB7sSKqknm0wZV2Ki5MiMbHaHeelGID-))

**Yelp Example:** refers to the dataset from the Yelp dataset challenge

([http://www.yelp.com/dataset\\_challenge](http://www.yelp.com/dataset_challenge)  ([http://www.yelp.com/dataset\\_challenge](http://www.yelp.com/dataset_challenge)))

from which we extracted the restaurant reviews. **Scores are on an integer scale from 1 to 5. We again considered reviews with scores 4 and 5 to be positive, and 1 and 2 to be negative.** We randomly generated a 50-50 training and testing split, which led to approximately 300,000 documents for each set.


## Step 2: Prepare the dataset

You can download the dataset from the UCI website:

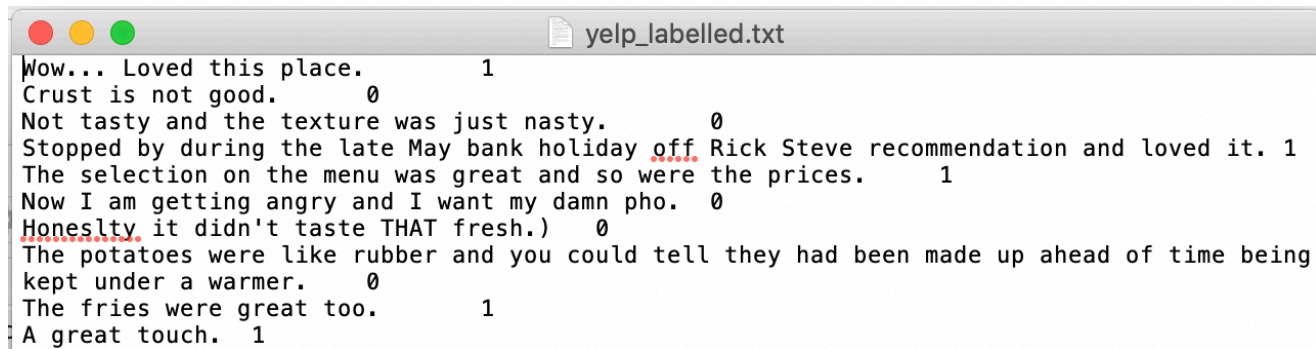
<https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences> 

(<https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences>).

For this tutorial, you can directly download the Yelp Text Data from Canvas

**Dataset link:** [yelp\\_labelled.txt \(https://canvas.slu.edu/courses/68524/files/5736295?wrap=1\)](https://canvas.slu.edu/courses/68524/files/5736295?wrap=1)  ([https://canvas.slu.edu/courses/68524/files/5736295/download?download\\_frd=1](https://canvas.slu.edu/courses/68524/files/5736295/download?download_frd=1))

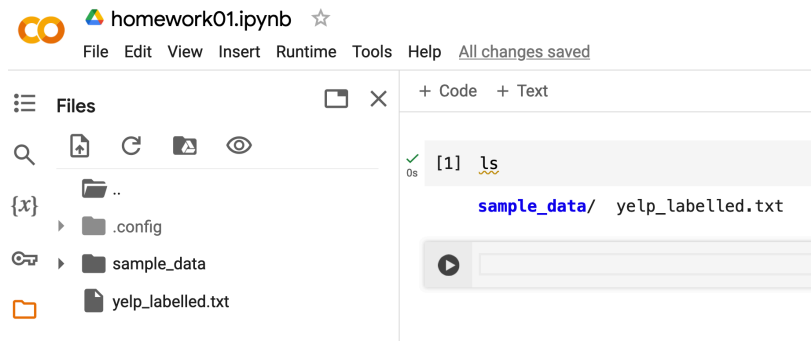
## Requirement: Upload the text file to Google Colab or Google Drive folder



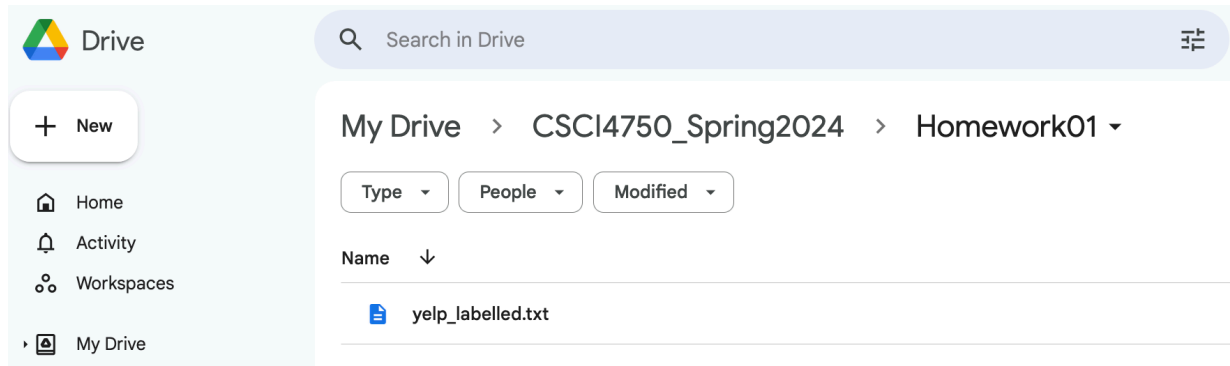
```
Wow... Loved this place. 1
Crust is not good. 0
Not tasty and the texture was just nasty. 0
Stopped by during the late May bank holiday off. Rick Steve recommendation and loved it. 1
The selection on the menu was great and so were the prices. 1
Now I am getting angry and I want my damn pho. 0
Honeslty, it didn't taste THAT fresh.) 0
The potatoes were like rubber and you could tell they had been made up ahead of time being
kept under a warmer. 0
The fries were great too. 1
A great touch. 1
```

## Practice: Check if you can view the uploaded text files:

(The following commands are applied on google colab to verify the file you upload)



The recommended approach is uploading your file to Google Drive Folder and mount Google Colab to Google Drive to access the file as follows:



```
# Load the Drive helper and mount
```

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
%cd /content/drive/MyDrive/CSCI4750_Spring2025/Homework01
```

## Step 3: View data format

### Step 3.1: View YELP reviews (use Yelp data as example)

```
## View first 10 rows
!head yelp_labelled.txt
```

```
[9] ## View first 10 rows
!head yelp_labelled.txt
```

```
Wow... Loved this place.      1
Crust is not good.           0
Not tasty and the texture was just nasty.      0
Stopped by during the late May bank holiday off Rick Steve r
The selection on the menu was great and so were the prices.
Now I am getting angry and I want my damn pho.  0
Honeslty it didn't taste THAT fresh.)          0
The potatoes were like rubber and you could tell they had be
The fries were great too.      1
A great touch.                 1
```

```
## Step 3.1: load yelp dataset
```

```
import pandas as pd
yelp_df = pd.read_csv('yelp_labelled.txt', names=['sentence',
'sentence', 'label'], sep='\t')
yelp_df
```

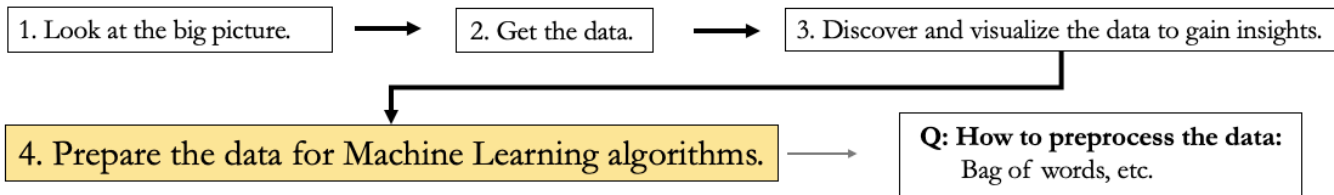
	sentence	label
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1
...	...	...
995	I think food should have flavor and texture an...	0
996	Appetite instantly gone.	0
997	Overall I was not impressed and would not go b...	0
998	The whole experience was underwhelming, and I ...	0
999	Then, as if I hadn't wasted enough of my life ...	0

1000 rows × 2 columns

## Step 4: Apply Bag-of-Words for Text Data Processing to get Word Frequency Table (Review: [Lecture8\\_TextProcessing\\_FeatureExtraction.pdf](#)

(<https://canvas.slu.edu/courses/68524/files/6037962?wrap=1>).\_)

## End-to-End Data Science Pipeline – Text Classification



### ➤ Bag of words (Feature extraction technique for text data)

- ✓ **Feature extraction:** Extract numerical features from text content
- ✓ **Vectorization:** general process of turning a collection of text documents into numerical feature vectors.
- ✓ **Bag of words:** Documents/sentences are described by word occurrences (ignoring the relative word position)

terms

	4	10		2	1		2
documents	1	7	7	6		5	
	...						

(Bag of words model for text corpus)

- **Row:** each document
- **Column:** distinct words
- **Value:** number of occurrences of each word in each document

## End-to-End Data Science Pipeline – Text Classification

### ➤ Bag of words (Feature extraction technique for text data)

#### Step 1: Clean data / Preprocessing

- ✓ **Lowercasing:** Lowercasing all text data

Raw text	Lowercased
Machine, <u>MACH</u> ine, MACHINE	machine
Learning, LEARNING	learning

- ✓ **Stemming / Lemmatize data:** reduce all words to root form.

Raw text	Stemming
train, <u>train</u> ing, <u>train</u> s, <u>train</u> ed	train
classif <u>er</u> s, classifier	classifier

Raw text	Lemmatize
optimize, <u>optim</u> izing, optimized	optimize
trouble, <u>troubl</u> ing	trouble

- ✓ **Remove stop words:** remove commonly used but low information words, focus more on important words used for classification. (i.e., a, an, the, is, are, of, etc)

Raw text	Stop words removal
<u>This</u> <u>is</u> a Machine Learning	Machine Learning
<u>We</u> <u>are</u> learning text classification	learning text classification

## End-to-End Data Science Pipeline – Text Classification

### ➤ Bag of words (Feature extraction technique for text data)

#### Step 2: Tokenization:

- ✓ **split** strings into pieces by using white-spaces and punctuation as token separators.
- ✓ **sorting** all possible tokens (word) alphabetically, giving an integer id for each possible token

**Token:** an individual instance in sentence, i.e., a word, etc.

Raw text	Tokenization
We are learning first classification topic.	['We', 'are', learning', 'first', 'classification', 'topic']
The first topic is the text classification.	['the', 'first', 'topic', 'is', 'the', 'text', 'classification']
The topic is the text processing.	['the', 'topic', 'is', 'the', 'text', 'processing']

Index	0	1	2	3	4	5	6	7	8	9
Unique Tokens	are	classification	first	is	learning	processing	text	the	topic	we
Total Counts	1	2	2	2	1	1	2	4	3	1

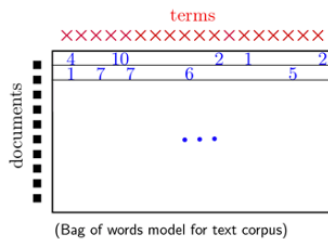
[https://scikit-learn.org/stable/modules/feature\\_extraction.html](https://scikit-learn.org/stable/modules/feature_extraction.html)

## End-to-End Data Science Pipeline – Text Classification

### ➤ Bag of words (Feature extraction technique for text data)

**Step 3: Word Frequency: Counting** the occurrences of tokens in each document

	0	1	2	3	4	5	6	7	8	9
Raw text	are	classification	first	is	learning	processing	text	the	topic	we
We are learning first classification topic.	1	1	1	0	1	0	0	0	1	1
The first topic is the text classification.	0	1	1	1	0	0	1	2	1	0
The topic is the text processing.	0	0	0	1	0	1	1	2	1	0
<b>Total Word Counts</b>	1	2	2	2	1	1	2	4	3	1



- **Row:** each document
- **Column:** distinct words
- **Value:** number of occurrences of each word in each document

[https://scikit-learn.org/stable/modules/feature\\_extraction.html](https://scikit-learn.org/stable/modules/feature_extraction.html)

### Step 4 Bag-of-words for Text processing & Feature extraction from the text:

Represent the whole sequence as a vector. and Drive Frequency of occurrence of each word

#### Step 4.a. Creating 'corpus': collecting all texts for training a ML model

```
# Step 4.a. get corpus from Yelp dataset
yelp_sentence = yelp_df['sentence'].to_list()
```

**Practice:** print out all documents (corpus), and write codes to count how many sentences we have in this yelp dataset?

**Step 4.b. Creating vocabulary:** collecting all unique words in all sentences. Create a vocabulary of all unique words by assigning index to each word.

**Building vocabulary dictionaries:** When creating Machine Learning models, we need to transform the text data into a sequence numbers, which can be the indexes of the tokens in the array of unique tokens.

```
from sklearn.feature_extraction.text import CountVectorizer
# # Step 4.b. creating vocabulary to map words into their index
# using CountVectorizer provided by the scikit-learn library
```

```
vectorizer = CountVectorizer(min_df=0, lowercase=False, stop_
ords='english') # will also remove punctuation or stop words

print("Test sequence: ",yelp_sentence[0:2])
vectorizer.fit(yelp_sentence[0:2]) # test the vocabulary on fi
rst two sentences
print("Vocabulary: ",vectorizer.vocabulary_)
```

```
Test sequence: ['Wow... Loved this place.', 'Crust is not good.']
Vocabulary: {'Wow': 2, 'Loved': 1, 'place': 4, 'Crust': 0, 'good': 3}
```

Note:

1. The vocabulary consists of all **five** words in our sentences, each representing one word in the vocabulary.
2. **The vocabulary dictionary uses the words as keys and indexes as values. We can use it to transform the text into numerical values.**
3. CountVectorizer performs **tokenization** which separates the sentences into a set of tokens. It additionally **removes punctuation and special characters** and can apply other preprocessing to each word.

**Practice: print out your vocabulary (tokens), and describe what do those key and values in the printed vocabulary represent.**

#### Step 4.c. Vectorize the sequence:

1. With the created vocabulary dictionaries, we can prepare transform the sequence into numbers for machine learning algorithm.
2. Create a feature vector with a length of the vocabulary. **Each element is the count of each word from the vocabulary** that can be found in the sentence (if not found, labeled as 0).

```
## Step 4.c.
print("Sequence: ",yelp_sentence[0:2])

vectorizer.fit(yelp_sentence[0:2]) # test the vocabulary on fi
rst two sentences
vectorizer.transform(yelp_sentence[0:2]).toarray()
```

```
Sequence: ['Wow... Loved this place.', 'Crust is not good.']
array([[0, 1, 1, 0, 1],
       [1, 0, 0, 1, 0]])
```

**Practice: Practice Vectorizer other sentences such as**

```
vectorizer.transform(["Loved the place. Loved the place. "]).toarray()
```

**Explain what do the values in vector represent. If the value equals to 0, what does it mean?**

**Step 5: Vectorize the training/testing dataset****a. Generate training/testing data from the full dataset**

Reference: [https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

[learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.htm](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

[I ↗ \(https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

[learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html\)](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

```
from sklearn.model_selection import train_test_split
sentences_train, sentences_test, y_train, y_test = train_test_split
```

**Practice: Check the size of sentences\_train, sentences\_test, y\_train, y\_test**

**Step 5.a. Generate vocabulary from the training data**

```
## Step 5: Learn Vocabulary & Vectorize the training dataset
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=0, lowercase=False, stop_words='english') # will also remove punctuation or stop words
vectorizer.fit(sentences_train) ### fit on the whole dataset
```

**Step 5.b. Check the vocabulary**

```
## Step 5.2: check the vocabulary
print("Vocabulary: ", vectorizer.vocabulary_)
print("Vocabulary words: ", vectorizer.vocabulary_.keys())
print("Vocabulary index: ", vectorizer.vocabulary_.values())
```

```
Vocabulary: {'Our': 255, 'server': 1376, 'nice': 1128, 'attentive': 438, 'serving'
Vocabulary words: dict_keys(['Our', 'server', 'nice', 'attentive', 'serving', 'sta
Vocabulary index: dict_values([255, 1376, 1128, 438, 1381, 1442, 304, 1152, 1519,
```

## Step 5.c. Create the feature vectors for each sentence of the training and testing set

```
## Step 5.3:create feature vector for each sentence
X_train = vectorizer.transform(sentences_train).toarray()
X_test = vectorizer.transform(sentences_test).toarray()
print("Training matrix shape", X_train.shape)
print("Testing matrix shape", X_test.shape)
```

Training matrix shape (670, 1651)

Testing matrix shape (330, 1651)

**practice 1:** Describe what do the rows and columns represent in variables `X_train` and `X_test`?

**practice 2:** How many features do the dataset have? For instance, if the total number of columns equals to 1651, could you explain what does 1651 mean? Why vectorization step return 1651 columns?

**Note:** For all the sequences in the dataset, we can treat their vector representations as feature vectors for machine learning models for classification (e.g., KNN, LDA, QDA, Perceptron, logistic regression).

**Tutorial Practice:** Export the final feature count table for each dataset as well as the their actual class labels into csv file (`Yelp_BoW.csv`) using Pandas.

We can directly use this new dataset (numeric count table, and label) for ML model training.

Sample format:

`Yelp_BoW.csv`

Instance	Word 1	Word 2	.....	Word K	Class Label
1					
2					
3					
...					
N					