DeftEval 2020

[Definition existence text classification]

Wadie Bishoy-5074
Anas Hamed-4989
Nour Shobier-4624

Prof. Dr. Marwan El-Torki

Introduction

Text classification is the process of assigning tags or categories to text according to its content. It's one of the fundamental tasks in Natural Language Processing (NLP) with broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection.

Text classifiers can be used to organize, structure, and categorize pretty much anything. For example, new articles can be organized by topics, support tickets can be organized by urgency, chat conversations can be organized by language, brand mentions can be organized by sentiment, and so on.

As an example, look at the following text below:
"The user interface is quite straightforward and easy to use."

A classifier can take this text as an input, analyze its content, and then and automatically assign relevant tags.

There are many approaches to automatic text classification, which can be grouped into three different types of systems:

- Rule-based systems
- Machine Learning based systems
- Hybrid systems

Our task is to use the machine learning based systems approach in the text classification.

Problem statement

Sentence classification, given a sentence, classify whether it contains a definition. This is a traditional definition extraction task.

Test data will be evaluated in the following format:

- Sentence classification [SENTENCE] [BIN_TAG] where the binary tag is 1 if the sentence contains a definition and 0 if the sentence does not contain a definition.

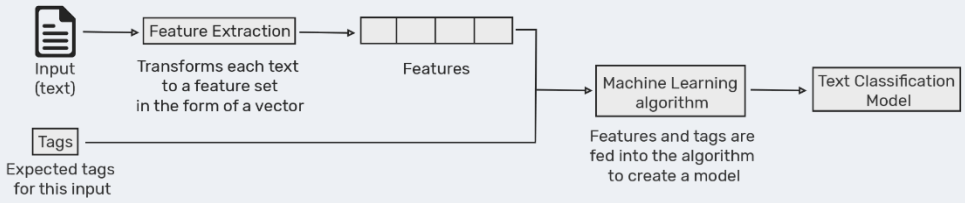
Machine learning based system

Text classification with machine learns to classify on the basis of past observations. A machine learn algorithm can learn the different associations between text pieces and that a specific output by using labeled examples as training data.

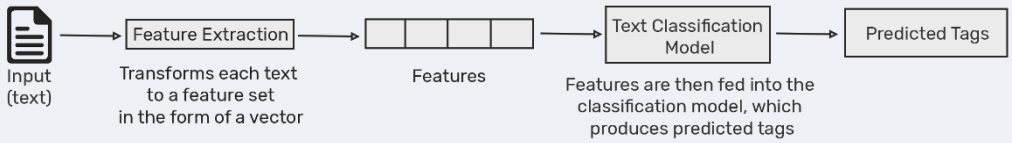
Feature extraction is the first step towards training a classifier with machine learning, which is to transform each text into a numerical representation in the form of a vector. One of the most frequently used approaches is word bag, where a vector in a predefined dictionary of words represents the frequency of a word.

For example, if we have defined our dictionary to have the following words {This, is, the, not, awesome, bad, basketball}, and we wanted to vectorize the text “This is awesome”, we would have the following vector representation of that text: (1, 1, 0, 0, 1, 0, 0).

Then, the machine learning algorithm is fed with training data that consists of pairs of feature sets (vectors for each text example) and tags (e.g. sports, politics) to produce a classification model:



Once it’s trained with enough training samples, the machine learning model can begin to make accurate predictions. The same feature extractor is used to transform unseen text to feature sets which can be fed into the classification model to get predictions on tags (e.g. sports, politics):



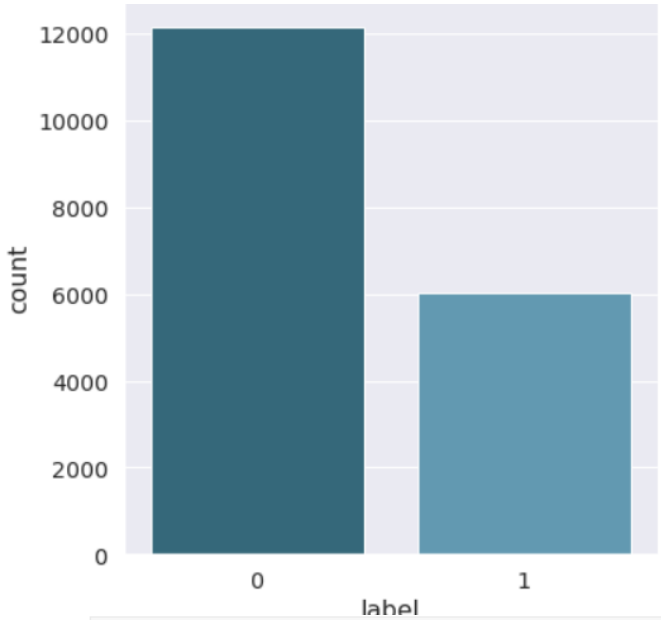
Input data analysis

In [1]:
```
import nltk
import pandas as pd
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
```

In [2]:
```
df = pd.read_table('../input/all_in_one_train.deft', sep='\t', header=None, names=['sentence',
'label'])
```

In [3]:
```
# number of sentences
len(df['label'])

count_1 = 0
count_0 = 0

for k in df['label']:
    if k == 0:
        count_0 +=1
    else:
        count_1 +=1

print(count_0)
print(count_1)

import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="darkgrid")
sns.set(font_scale=1.3)

sns.factorplot(x="label", data=df, kind="count", size=6, aspect=1, palette="PuBuGn_d")
plt.show();
```

```
12143
6014
```



In [4]:
```
# table
df.head()
```

Out[4]:

	sentence	label
0	5 . Science includes such diverse fields as a...	0
1	However , those fields of science related to ...	1
2	Thus , a museum of natural sciences might con...	0
3	8 . In deductive reason , the pattern of thin...	0
4	Deductive reasoning is a form of logical thin...	1

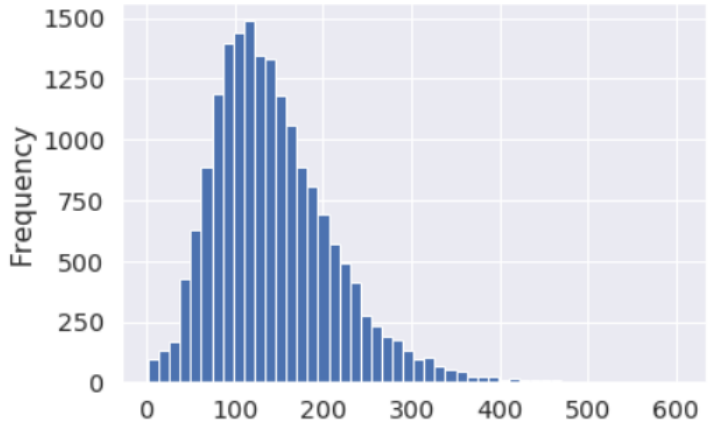
In [5]:
```
# exploratory Data Analysis
df.groupby('label').describe()
```

Out[5]:

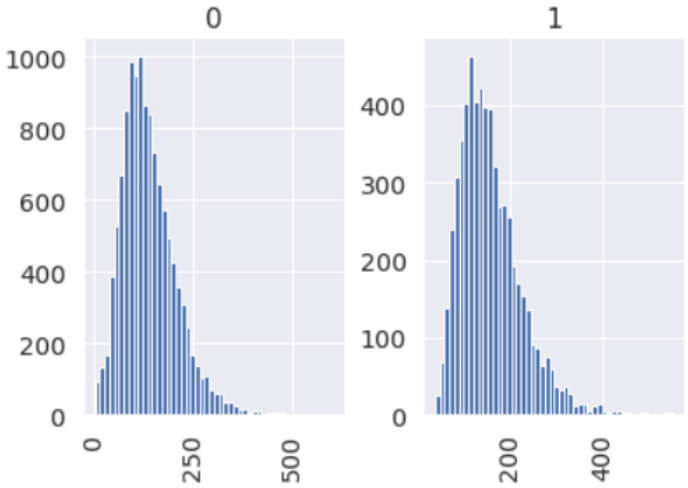
	sentence			
	count	unique	top	freq
label				
0	12143	11590	" >	6
1	6014	5337	Osmoregulation is the process of maintenance ...	4

In [6]:
```
df['length'] = df['sentence'].apply(len)
df.head()
```

In [7]:
```
# Data Visualization
df['length'].plot(bins=50, kind='hist')
```



In [8]:
```
# ploting defined vs undefined
df.hist(column='length', by='label', bins=50)
```



In [9]:
```
df.length.describe()
```

Out[9]:
```
count    18157.000000
mean       144.315581
std         69.424827
min          2.000000
25%         95.000000
50%        133.000000
75%        182.000000
max        603.000000
Name: length, dtype: float64
```

Text classification

- ## Data pre-processing

The steps of pre-processing are needed for transferring text from human language to machine-readable format for further processing. We will discuss text preprocessing tools.

a. Removing stopwards: These are common words that don't really add anything to the categorization, such as a, able, either, else, ever, and so on. So for our purposes, "the election was over" would be "election over" and a "very close game" would be "very close game".

b. Stemming words: the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma.

c. Using n-grams: Instead of counting single words like we did here, we could count sequences of words.

HINT: detailed in 4624-Nour El-din shobier report.

- ## Algorithms

I. Features Extractor

a. CountVectorizer

It provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

You can use it as follows:

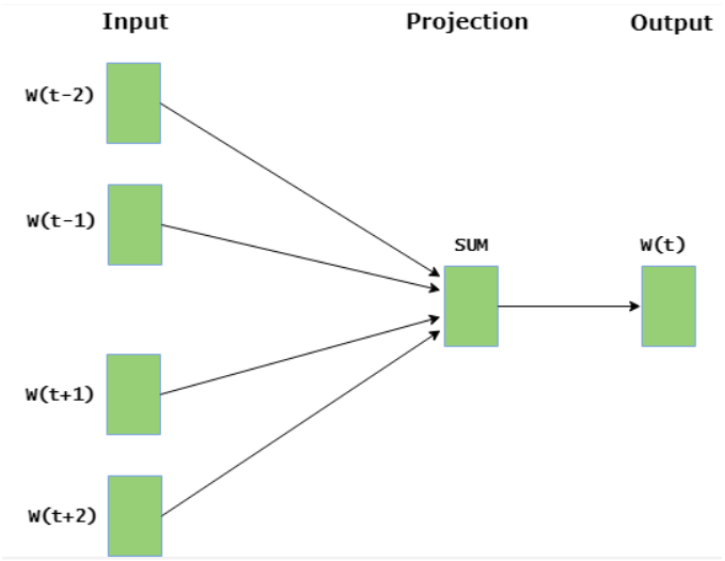
- Create an instance of the CountVectorizer class.
- Call the fit() function in order to learn a vocabulary from one or more documents.
- Call the transform() function on one or more documents as needed to encode each as a vector.

b. Word2Vec

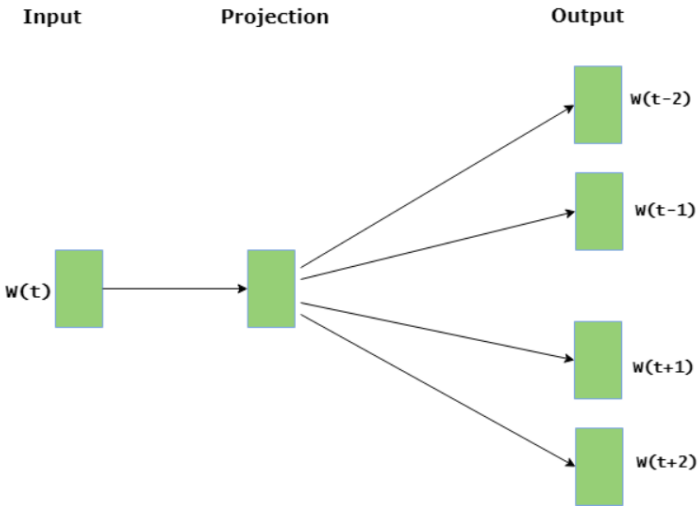
Word Embedding is a language modeling technique used for mapping words to vectors of real numbers. It represents words or phrases in vector space with several dimensions. Word embeddings can be generated using various methods like neural networks, co-occurrence matrix, probabilistic models, etc.

Word2Vec consists of models for generating word embedding. These models are shallow two-layer neural networks having one input layer, one hidden layer and one output layer. Word2Vec utilizes two architectures:

1- CBOW (Continuous Bag of Words): predicts the current word given context words within specific window.



2- Skip Gram: predicts the surrounding context words within specific window given current word.



We found that skip-gram was very slightly slower but produced better results, so we decided to use it.

HINT: detailed in 5074-Wadie Bishoy report.

II. Classifiers

a. Naïve Bayes

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently.

This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

```
P(H|E) = (P(E|H) * P(H)) / P(E)
```

where

- P(H|E) is the probability of hypothesis H given the event E, a posterior probability.
- P(E|H) is the probability of event E given that the hypothesis H is true.
- P(H) is the probability of hypothesis H being true (regardless of any related event), or prior probability of H.
- P(E) is the probability of the event occurring (regardless of the hypothesis).

HINT: detailed in 4624-Nour El-din shobier report.

b. Decision tree

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers the to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surface.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case.

HINT: detailed in 4989-Anas Hamed report.

c. Logistic Regression

Logistics regression is common and is a useful regression method for solving the binary classification problem. Another category of classification is Multinomial classification, which handles the issues where multiple classes are present in the target variable.

It is one of the simplest and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem.

It describes and estimates the relationship between one dependent binary variable and independent variables.

HINT: detailed in 4989-Anas Hamed report.

d. K Nearest Neighbors

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real-world datasets do not follow mathematical theoretical assumptions.

Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

HINT: Is just used for more observation.

Output data analysis

Naive Bayes..

	precision	recall	f1-score	support
0	0.82	0.79	0.80	579
1	0.60	0.65	0.63	286
accuracy			0.74	865
macro avg	0.71	0.72	0.72	865
weighted avg	0.75	0.74	0.75	865

decision tree..

	precision	recall	f1-score	support
0	0.80	0.85	0.82	579
1	0.65	0.56	0.60	286
accuracy			0.75	865
macro avg	0.72	0.71	0.71	865
weighted avg	0.75	0.75	0.75	865

K-Nearest Nieghbors..

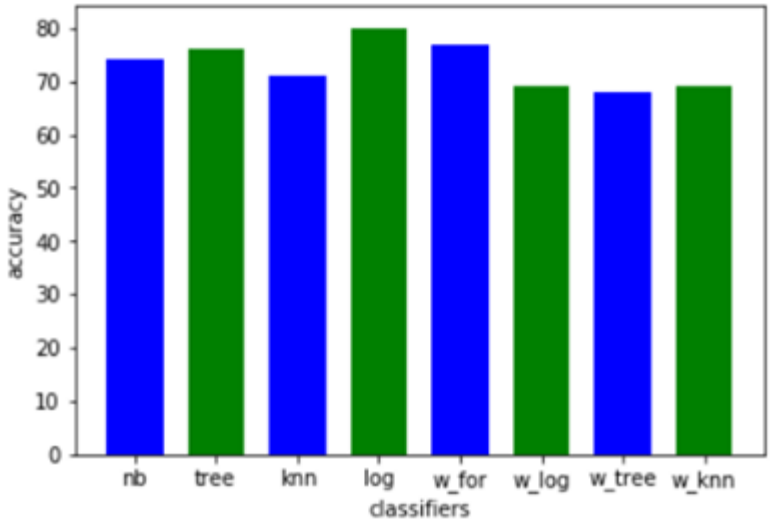
	precision	recall	f1-score	support
0	0.71	0.97	0.82	579
1	0.75	0.19	0.30	286
accuracy			0.71	865
macro avg	0.73	0.58	0.56	865
weighted avg	0.72	0.71	0.65	865

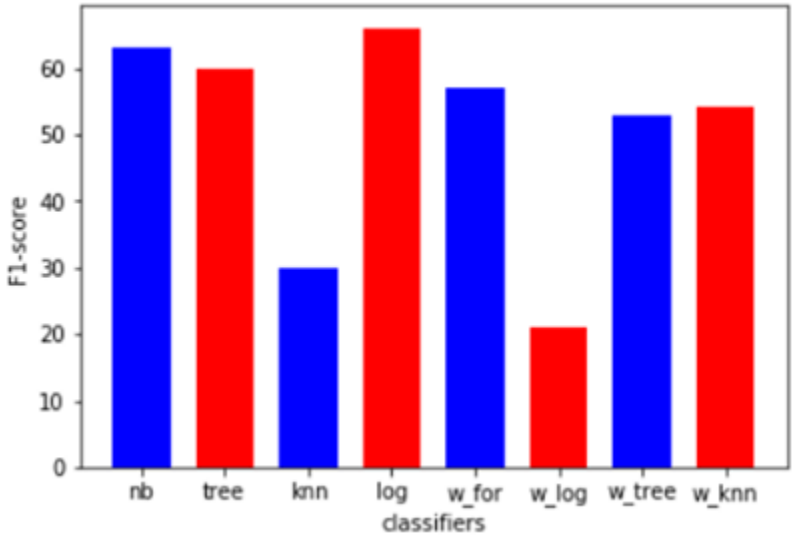
```
Logistic Regression..
              precision    recall  f1-score   support

           0       0.82      0.89      0.85       579
           1       0.74      0.60      0.66       286

    accuracy                           0.80       865
   macro avg       0.78      0.75      0.76       865
weighted avg       0.79      0.80      0.79       865
```

```
--> Using Word2vec
random forest (just for testing)..
              precision    recall  f1-score   support

           0       0.78      0.92      0.84       579
           1       0.75      0.47      0.57       286

    accuracy                           0.77       865
   macro avg       0.76      0.69      0.71       865
weighted avg       0.77      0.77      0.76       865
```

```
decision tree..
              precision    recall  f1-score   support

           0       0.77      0.75      0.76       579
           1       0.52      0.55      0.53       286

    accuracy                           0.68       865
   macro avg       0.64      0.65      0.65       865
weighted avg       0.69      0.68      0.68       865
```

```
K-nearest neighbours..
              precision    recall  f1-score   support

           0       0.77      0.77      0.77       579
           1       0.54      0.54      0.54       286

    accuracy                           0.69       865
   macro avg       0.65      0.66      0.65       865
weighted avg       0.69      0.69      0.69       865
```


accuracy
0
10
20
30
40
50
60
70
80
nb
tree
knn
log
w_for
w_log
w_tree
w_knn
classifiers


F1-score
0
10
20
30
40
50
60
nb
tree
knn
log
w_for
w_log
w_tree
w_knn
classifiers

References

1- https://monkeylearn.com/text-classification/
2- https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn
3- https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/
4- https://syncedreview.com/2017/07/17/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation/
5- https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn
6- https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/
7- https://liferay.de.dariah.eu/tatom/classification_logistic_regression.html
8- https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python