Team 18

Web Content Extraction using Machine Learning

Karthikeyan K
2018103549
Dhanush Kumar K
2018103022

Web Content Extraction using Machine Learning

Abstract:

The World Wide Web has seen tremendous growth in recent years. With the large amount of information on the Internet, web pages have been the potential source of information retrieval and data mining technology such as commercial search engines, web mining applications. Internet web pages contain several items that cannot be classified as the informative content, e.g., search and filtering panel, navigation links, advertisements, and so on called as noisy parts. Most clients and end-users search for the informative content, and largely do not want the non-informative content. A tool that assists an end-user or application to search and process information automatically ,must separate the “primary or informative content sections” from the other content sections. The content extraction problem has been a subject of study ever since the expansion of the World Wide Web. Its goal is to separate the main content of a web page, such as the text of a news story, from the noisy content, such as advertisements and navigation links. Most content extraction approaches operate at a block level; that is, the web page is segmented into blocks and then each of these blocks is determined to be part of the main content or the noisy content of the web page. The extracted main content is summarized into tabular format.

Keywords:
Web pages, Crawler, Clustering, K-Means, SVM.

Introduction:

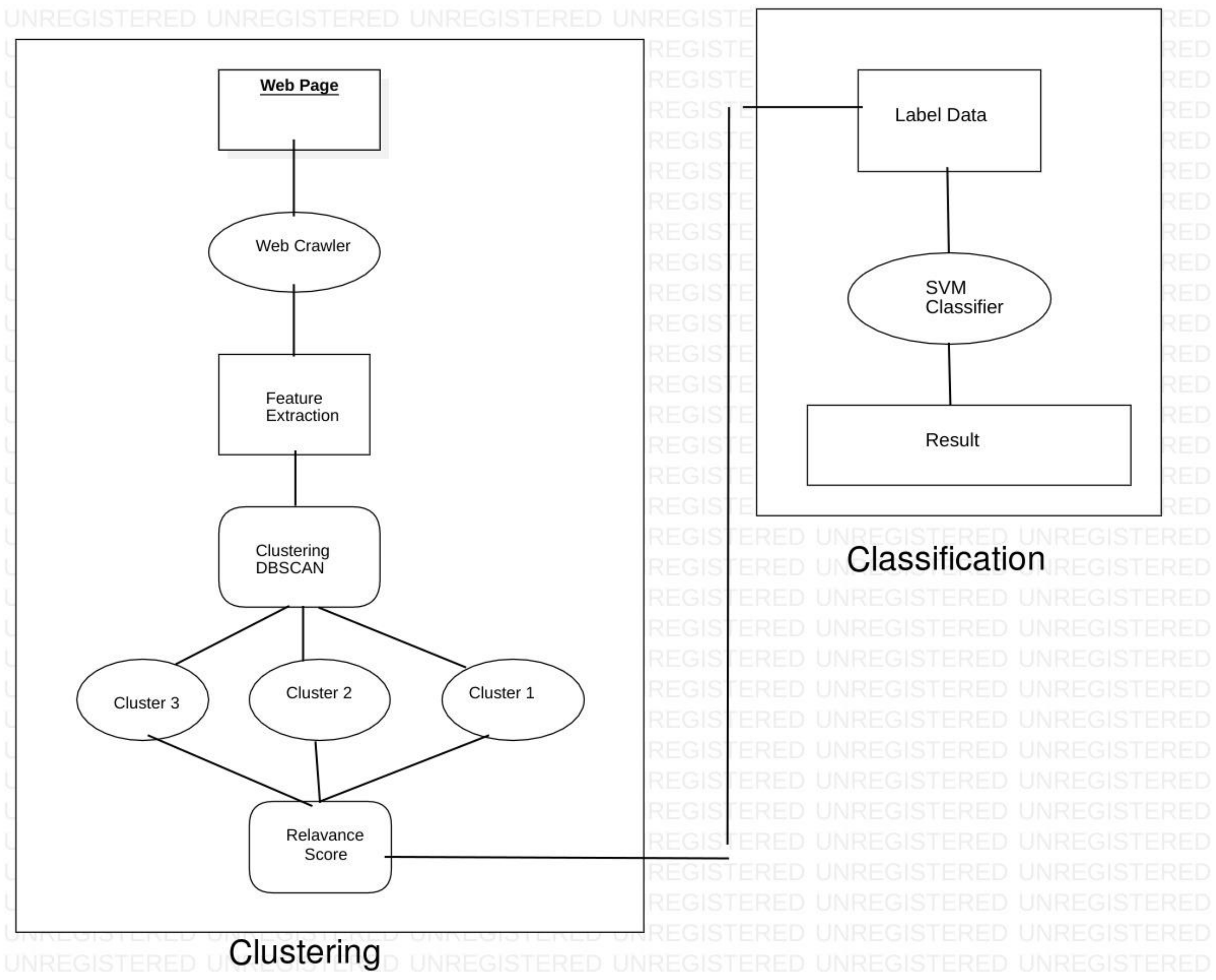
The web pages (also referred to web documents) that constitute the World Wide Web are sources of very diverse categories of information. These include news, reference materials, forum discussions, and commercial product descriptions, just to name a few. Each category of information can in turn have various media formats, such as textual, graphical, or video. This vast amount of information is used by ordinary web users throughout the world, as well as by automated crawlers that traverse the Web for various purposes, such as web mining or web indexing.

In most cases, however, a single web page consists of distinct “parts,” which will be referred to in this thesis as the contents of the web page. Only one type of content, which will be referred to as the main content of the web page, is what makes the web page a useful source of information. Other contents include advertisements, navigation buttons, page settings, and legal notices; these contents will be collectively referred to as the noisy content of the web page. The process of identifying the main content of a web page is called main content extraction, or more briefly content extraction.

The goal of this project is to develop new technique for content extraction using supervised machine learning algorithms based on a sample of web pages with manually labeled contents; that is, the contents of these web pages have been identified as main or noisy by a human annotator. In addition, the content extraction performance under these rules should be evaluated.

Working:

The system contains a main website in which the URL of the web page whose data needs to be crawled is entered and the flow of the system goes as follows.



1) Get URL of a website as input

2) Crawler will crawl the site and extract text data

3) Apply clustering on extracted data to divide data in clusters like text, links, etc.

4) Labeling the data by finding out related data to that page as 1 and unrelated data as 0.

5) Apply SVM to classify data as content and non-content. This process will remove noise, adds, duplicated data.

6) Final output will be the summarized text from website.

Module 1
FEATURE SELECTION

Text Length
A common naive approach to web content extraction is to find the longest contiguous block of text in a webpage. The intuition is that paragraphs in an article are usually long.
To establish some baseline, we’ve experimented with using only text length as feature.

Tag Path
The second approach is to use tag path of the block. Tag path is simply the path it takes to navigate the DOM tree from its root to the text block. It consists of element names for each node along the path.

CSS Selectors
CSS selectors are essentially decorated tag paths. We have incorporated CSS class name in addition to the element name into the CSS selectors.

CSS Properties
The third approach is to use all visual related CSS properties (color, font-size, font-style, line-height etc.) as features.
These features are also treated as discrete features and vectorized.

For each chosen block, a set of features are then extracted, including the size and position of the block, the contained text, the font configurations, color, line height, tag path, etc. In fact, there are over 300 different CSS properties for each block. Our algorithm automatically extracts those with a non-default value. Therefore, the number of features varies from block to block in the data collection phase.

Module 2

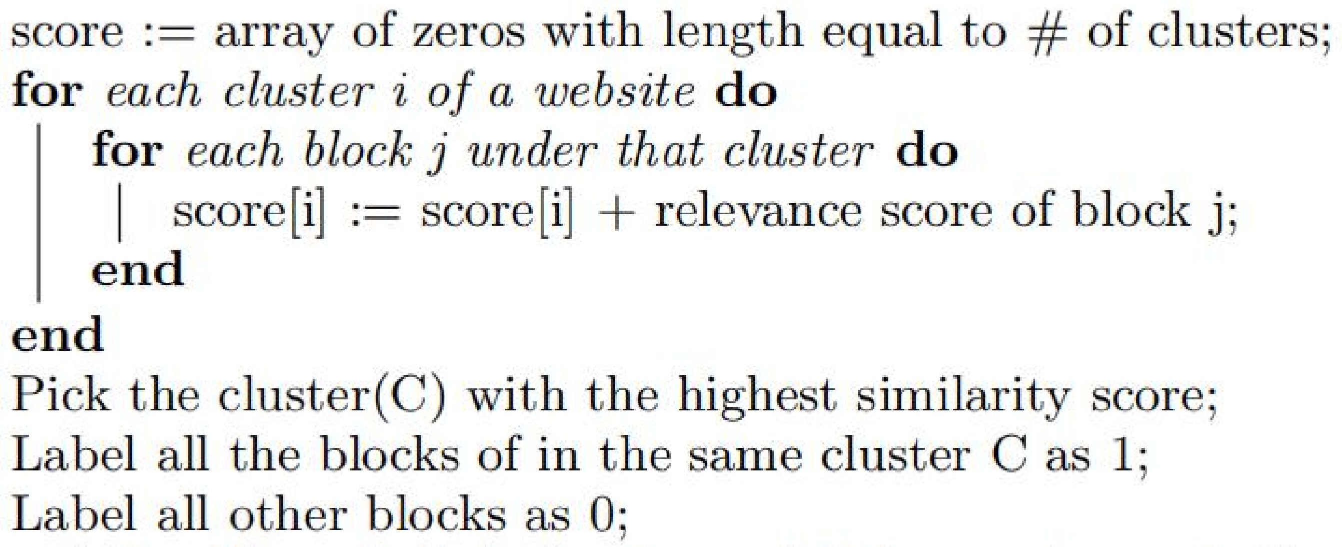
Clustering Blocks

The cluster shape of similar blocks on a webpage (i.e. navigational buttons) are not necessarily well-shaped (i.e. spherical). Also there can be random noises that appears on some of the webpages. Furthermore, it is unclear that how many clusters will there be, since it depends heavily on the design and layout of the webpage. Therefore, we have chosen DBSCAN, density-based spatial clustering of applications with noise, as our clustering algorithm. This algorithm handles the problems of unknown number of clusters, unknown shape of clusters and noise quite nicely. The clustering result shows that, the body text of the articles across multiple webpages, which usually consist of multiple blocks, are clustered together in a single cluster. The navigational links are also clustered together, as well as the links in the footers. Other common elements, like similar sidebar, advertisement, comments, etc., are also successfully clustered. Given all the visual features represented by the CSS properties, the algorithm is quite effective in discerning the different visual elements on the webpages.

Module 3
Labeling Clustered Block

Although all the blocks have been clustered quite precisely based on their visual properties, it is not trivial to find out which cluster contains the content text. The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. Cluster number varies from webpage to webpage, and the cluster containing content can be any of those. For our dataset, the number of output clusters ranges from a few clusters to above 20.

Fortunately, most websites have some metadata stored in meta tags in order to accommodate web crawlers. So we can extract the description of an article by parsing the meta tags in the webpage. The description usually contains a brief summary or just the first few words of the article content. With that, we are able to calculate the similarity between each cluster and the description using the Longest Common Sub-sequence (LCS) algorithm. The longer the common sub-sequence (we call this number “relevance score”), the more likely is the cluster to contain the content text. Note that we used LCS instead of Term Frequency or Inverse Document Frequency to ensure the language independence, as word segmentation varies from language to language. At first we tried to automatically label the blocks by finding the best cluster local to each webpage on a website. After training the SVM on this labeling, we found that for some websites, the precision on the test set was not ideal. On closer examination, we found that on some rare pages, the best cluster according to that webpage’s description is comment instead of the main text. To fix this issue, we implemented Algorithm 1 which scores the blocks across the entire website



```
score := array of zeros with length equal to # of clusters;
for each cluster i of a website do
    for each block j under that cluster do
        score[i] := score[i] + relevance score of block j;
    end
end
Pick the cluster(C) with the highest similarity score;
Label all the blocks of in the same cluster C as 1;
Label all other blocks as 0;
```

Algorithm 1: Labeling from global score in a website

Module 4

SVM Classification

Using the collected and labeled text blocks, the web content extraction problem can be formulated as a classification problem, where the goal content consists of multiple text blocks that are classified as content while the othertext blocks are classified as non-content. We have constructed a support vector classifier with a linear kernel to perform text block classification. Due to the imbalance between the number of content blocks and the number of non-content blocks on a webpage, we applied class weights to give our positive (content) examples higher weights. We have employed three different approaches to the classification problem. For each approach, we have performed 4-fold cross validation to evaluate its performance. In our first approach, we have trained separate classifier for different website.

For each given website, the collected webpages are shuffled then divided into four groups. One of the groups is chosen as evaluation group, while the other three are used as training examples. Intuitively, the support vector classifier in this case tries to learn the underlying structure and template used by the particular website. This approach worked very well. In our second approach, we have trained one classifier model using webpages from multiple websites in the chaos dataset and then tested the model on webpages from the rest of the dataset.

The classifier in this case learns some structures from only a subset of websites then tries to generalize the model on previously unseen websites. This approach produced the worst results. It make sense that specific visual characteristics of one website rarely appears the same in another website. And lastly, we have trained one classifier model using random webpages chosen from the chaos dataset then tested the model on the rest of the dataset. In this approach, since the webpages used in training are picked randomly, our training examples included a wide variety of websites. Therefore, it is likely that the classifier have previously seen at least one example webpage per website for the websites in the evaluation set

CONCLUSION

We have developed a pipeline to extract web content. Our pipeline collects data, labels examples, trains support vector classifier, and evaluates learned model in an automated manner. Our learning algorithm can achieve perfect labeling when trained on a single website, even for websites with multiple different templates. By analyzing features, we have found that some of our features—tag path, CSS selectors—contributed to the near perfect classification results in many websites, but they also fail in some cases. CSS visual properties work particularly well across

Web Content Extraction using Machine Learning

In [116…

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import heapq
import pylcs as LCS

from collections import Counter

import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import f1_score,confusion_matrix

from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
import matplotlib.pyplot as plt

import seaborn as sns
```

Data Retrieval

In [117…

```
URL = "https://www.ndtv.com/india-news/ndtv-news-on-oxygen-supply-cited-by-delhi
page = requests.get(URL)
soup = BeautifulSoup(page.content, 'html5lib')
print(page.content[0:1000])
```

```
b'<!doctype html><html xmlns="http://www.w3.org/1999/xhtml" itemscope itemtype
="http://schema.org/NewsArticle"><head><title>At Delhi Oxygen Hearing, &quot;Bro
ther Judge&quot; Sends NDTV News On WhatsApp</title><meta name="news_keywords" c
ontent="Coronavirus,Delhi High Court,NDTV" itemprop="keywords"/><meta name="desc
ription" content="The full horror of an oxygen shortage in Delhi&#039;s Covid sp
iral sank in before the High Court today when NDTV&#039;s reports on top hospita
ls running out of oxygen were flagged by one of the judges." itemprop="descripti
on"/><meta name="section" content="india" itemprop="articleSection"/><meta name
="url" content="https://www.ndtv.com/india-news/ndtv-news-on-oxygen-supply-cited
-by-delhi-high-court-2418022" itemprop="url"/><link href="https://www.ndtv.com/i
ndia-news/ndtv-news-on-oxygen-supply-cited-by-delhi-high-court-2418022?amp=1&aka
mai-rum=off" rel="amphtml" ><link href="https://plus.google.com/+NDTV" rel="publ
isher" ><link href="android-app://com.july.nd'
```

Extracting the Meta Content

In [118…

```
meta = []
for tag in soup.findAll(True):
    if tag.name == "meta":
        meta.append(tag.attrs)
```

```
metaContent = []
for dic in meta:
    metaContent.append(dic["content"])

metaContent = []
for dic in meta:
    metaContent.append(dic["content"])
```

Removing Noises in the Meta Content

In [119…

```
noise = ["https", ".com", "com.", "www", "@", ":", "=", "#"]
metaContentFinal = []
for content in metaContent:
    if any(i in content for i in noise):
        pass
    else:
        metaContentFinal.append(content)

metaContentStr = ""

for content in metaContentFinal:
    metaContentStr = metaContentStr + content
```

In [120…

```
print(metaContentStr)
```

```
Coronavirus,Delhi High Court,NDTVThe full horror of an oxygen shortage in Delh
i's Covid spiral sank in before the High Court today when NDTV's reports on top
hospitals running out of oxygen were flagged by one of the judges.indiaCoronavir
us,Delhi High Court,NDTVAt Delhi Oxygen Hearing, "Brother Judge" Sends NDTV News
On WhatsApparticle630473The full horror of an oxygen shortage in Delhi's Covid s
piral sank in before the High Court today when NDTV's reports on top hospitals r
unning out of oxygen were flagged by one of the judges.213741912058651NDTV377869
410NDTV390847563NDTV HDsummary_large_imageAt Delhi Oxygen Hearing, "Brother Judg
e" Sends NDTV News On WhatsAppThe full horror of an oxygen shortage in Delhi's C
ovid spiral sank in before the High Court today when NDTV's reports on top hospi
tals running out of oxygen were flagged by one of the judges.newsAt Delhi Oxygen
Hearing, "Brother Judge" Sends NDTV News On WhatsAppNDTV377869410NDTV HDNDTV1003
0NDTV123At Delhi Oxygen Hearing, "Brother Judge" Sends NDTV News On WhatsAppAt D
elhi Oxygen Hearing, "Brother Judge" Sends NDTV News On WhatsAppAt Delhi Oxygen
Hearing, "Brother Judge" Sends NDTV News On WhatsAppT1M20S1200886
```

Feature Extraction

Tags, Texts, Attributes Extraction

In [121…

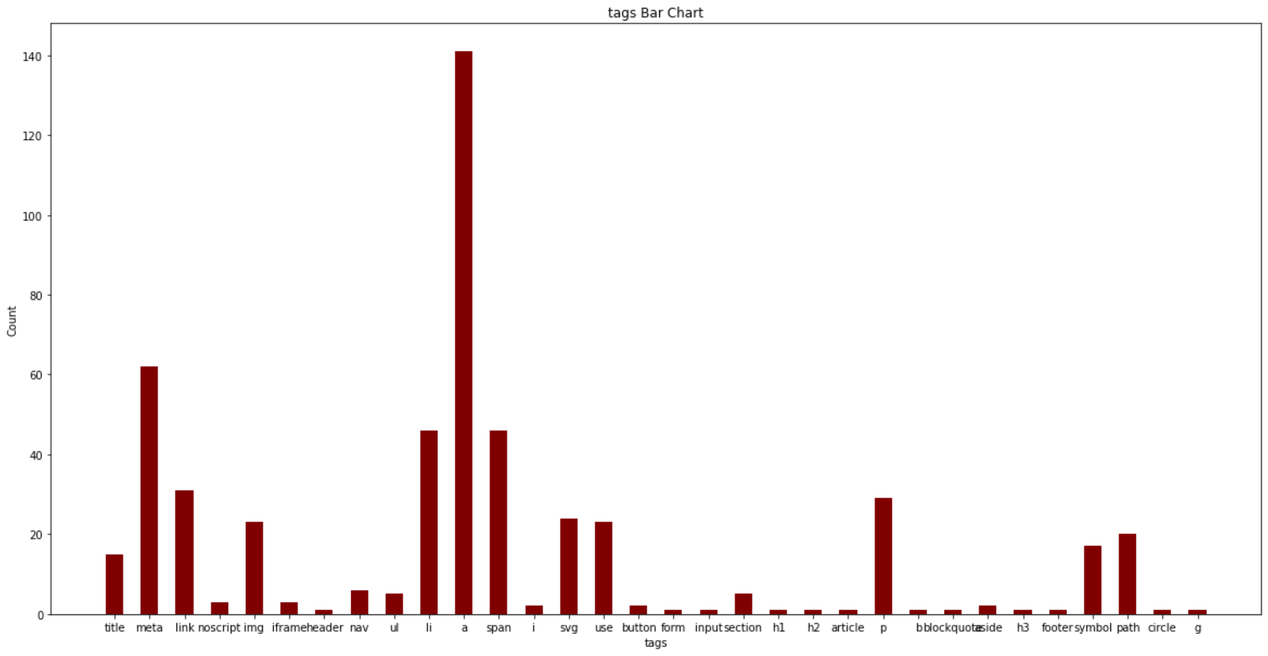
```
tags = []
texts = []
attrs = []

for tag in soup.findAll(True):
    if (tag.name == "style") or (tag.name == "script") or (tag.name == "body") o
        continue
    else:
        tags.append(tag.name)
        texts.append(tag.text)
        attrs.append(tag.attrs)
```

In [122…
```
tags[0:10]
```

Out[122…
```
['title',
 'meta',
 'meta',
 'meta',
 'meta',
 'link',
 'link',
 'link',
 'link',
 'link']
```

In [123…
```
tagsCount = Counter(tags)
uniqueTags = list(tagsCount.keys())
countValues = list(tagsCount.values())
```

In [124…
```
fig = plt.figure(figsize = (20, 10))
plt.bar(uniqueTags, countValues, color ='maroon',width = 0.5)
plt.xlabel("tags")
plt.ylabel("Count")
plt.title("tags Bar Chart")
plt.show()
```



In [125…
```
texts[0:5]
```

Out[125…
```
['At Delhi Oxygen Hearing, "Brother Judge" Sends NDTV News On WhatsApp',
 '',
 '',
 '',
 '']
```

In [126…
```
attrs[0:5]
```

Out[126…
```
[{},
 {'name': 'news_keywords',
  'content': 'Coronavirus,Delhi High Court,NDTV',
  'itemprop': 'keywords'},
 {'name': 'description',
  'content': "The full horror of an oxygen shortage in Delhi's Covid spiral sank
in before the High Court today when NDTV's reports on top hospitals running out
of oxygen were flagged by one of the judges.",
  'itemprop': 'description'},
 {'name': 'section', 'content': 'india', 'itemprop': 'articleSection'},
 {'name': 'url',
  'content': 'https://www.ndtv.com/india-news/ndtv-news-on-oxygen-supply-cited-b
y-delhi-high-court-2418022',
  'itemprop': 'url'}]
```

Feature selection

In [127…
```python
texts = pd.Series(texts)
column = list(set(tags))

data = []

for i in range(0, len(tags)):
    data1 = []
    for j in range(0, len(column)):
        if tags[i] == column[j]:
            data1.append(1)
        else:
            data1.append(0)
    data.append(data1)

initialDf = pd.DataFrame(columns=column, data=data)
```

In [128…
```python
initialDf.head()
```

Out[128…

	svg	title	meta	form	use	noscript	li	blockquote	p	link	...	h1	article	h3	nav	footer	hea
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 32 columns

Normalization

In [129…
```python
def normalize(df):
    std = StandardScaler()

    df = std.fit_transform(df)

    return pd.DataFrame(df)
```

DBSCAN Clustering

In [130…

```
def clusteringDB(df):
    clustering = DBSCAN().fit(df)

    uniqueClusters = set(clustering.labels_)

    df["cluster"] =[i+1 for i in clustering.labels_]

    return uniqueClusters,clustering,df
```

Relavance Score

In [131…

```
def relevanceScore(uniqueClusters,df):
    score = [0 for i in range(len(uniqueClusters))]

    for i in range(0, df.shape[0]):
        score[int(df.loc[i]["cluster"])] = score[int(df.loc[i]["cluster"])] + L(

    return score
```

Finding the cluster with Maximum Score

In [132…

```
def highScore(score):
    maxScoreClusters = heapq.nlargest(2, range(len(score)), key=score.__getitem_

    return maxScoreClusters
```

Marking the Label

In [133…

```
def labelMarking(maxScoreClusters,df):
    label = []

    for i in range(0,df.shape[0]):
        if (int(df.loc[i]["cluster"]) == maxScoreClusters[0]) or (int(df.loc[i][
            label.append(1)
        else:
            label.append(0)

    df["label"] = label

    return df
```

Train and Test Data split

In [134…

```
def trainTestSplit(df):
    df1 = df.drop(columns="cluster")
    X = df1.drop(columns="label")
    y = df1["label"]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, ra

    return X_train, X_test, y_train, y_test
```

SVM Classification and Prediction

In [135…

```
def svmModel(X_train,y_train,X_test):
    svmModel = SVC()
    svmModel.fit(X_train,y_train)

    prediction = svmModel.predict(X_test)

    return prediction
```

F1 Score and Confusion Matrix

In [136…

```
def performance(y_test,prediction):
    f1Score = f1_score(y_test,prediction)
    cf_matrix = confusion_matrix(y_test,prediction)

    print("F1 Score of SVM Model")
    print(f1Score)

    print('Confusion Matrix')
    sns.heatmap(cf_matrix, annot=True)
```

In [ ]:

Text Length as Feature

In [137…

```
textSize = []

for text in texts:
    textSize.append(len(text))

initialDf["textSize"] = textSize
```

In [138…

```
initialDf.head()
```

Out[138…

	svg	title	meta	form	use	noscript	li	blockquote	p	link	...	article	h3	nav	footer	header
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0

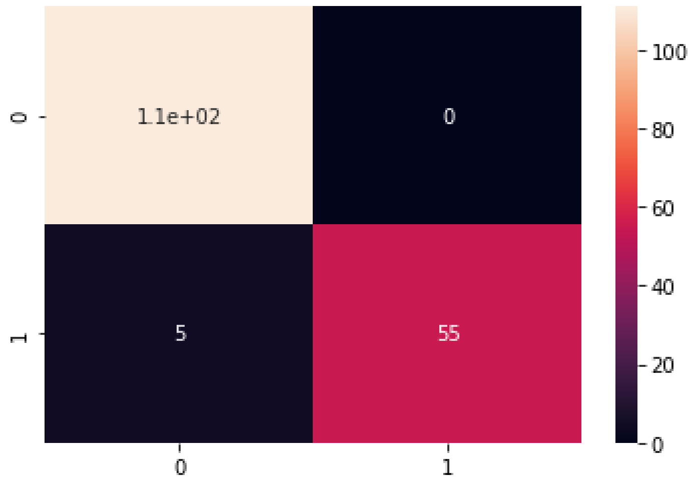
	svg	title	meta	form	use	noscript	li	blockquote	p	link	...	article	h3	nav	footer	header
3	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 33 columns

In [139…

```
df = normalize(df)
uniqueClusters,clustering,df = clusteringDB(df)

score = relevanceScore(uniqueClusters,df)

maxScoreClusters = highScore(score)

df = labelMarking(maxScoreClusters,df)

X_train, X_test, y_train, y_test = trainTestSplit(df)

prediction = svmModel(X_train,y_train,X_test)

performance(y_test,prediction)
```

```
F1 Score of SVM Model
0.9565217391304348
Confusion Matrix
```



HTML Tags Attribute as Feature

In [140…

```
attrsColumn = []
for i in attrs:
    attrsColumn.append(list(i.keys()))

attrsColumn = sum(attrsColumn, [])

attrsColumn = list(set(attrsColumn))

data = []
for i in attrs:
    idx = []
    for j in i.keys():
        for k in range(0, len(attrsColumn)):
            if j == attrsColumn[k]:
                idx.append(k)

    data.append([1 if i in idx else 0 for i in range(0, len(attrsColumn))])

dummyDf = pd.DataFrame(columns=attrsColumn, data=data)

finalDf = pd.concat([initialDf, dummyDf], axis=1, join='inner')

df = pd.concat([initialDf, dummyDf], axis=1, join='inner')
```

In [141…

```
df.head()
```

Out[141…

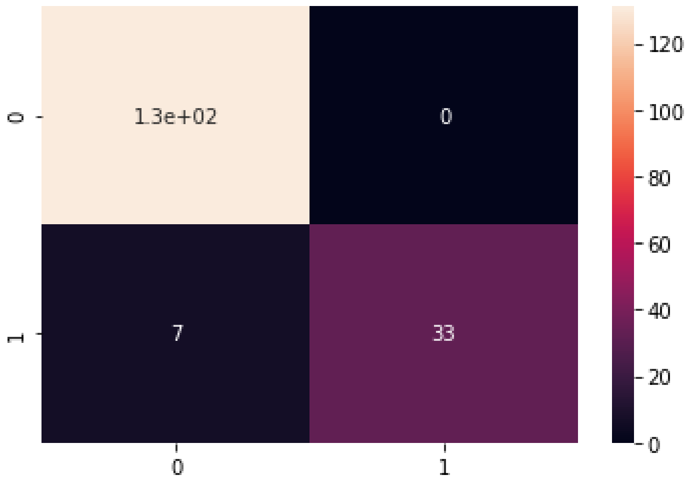
	svg	title	meta	form	use	noscript	li	blockquote	p	link	...	itemscope	dir	target	technolo
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	
1	0	0	1	0	0	0	0	0	0	0	...	0	0	0	
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	
3	0	0	1	0	0	0	0	0	0	0	...	0	0	0	
4	0	0	1	0	0	0	0	0	0	0	...	0	0	0	

5 rows × 91 columns

In [142…

```
df = normalize(df)
uniqueClusters,clustering,df = clusteringDB(df)

score = relevanceScore(uniqueClusters,df)

maxScoreClusters = highScore(score)

df = labelMarking(maxScoreClusters,df)

X_train, X_test, y_train, y_test = trainTestSplit(df)

prediction = svmModel(X_train,y_train,X_test)

performance(y_test,prediction)
```

```
F1 Score of SVM Model
0.9041095890410958
Confusion Matrix
```



JavaScript Keywords as Feature

In [143…

```
keywords = ["await", "break", "case", "catch", "class", "const", "continue", "de
            "default", "delete", "do", "else", "enum", "export", "extends", "fal
            "finally", "for", "function", "if", "implements", "import", "in", "i
            "let", "new", "null", "package", "private", "protected", "public", '
            "static", "throw", "try", "true", "typeof", "var", "void", "while", 
            "(", ")", "{", "}", "]", "[", ";", ".", "\"", "function", "console",
            "window", "href", "\'", "return"]


data=[]
for i in str(texts):
    data1=[]
    for j in keywords:
        c = i.count(j)
        data1.append(c)
    data.append(data1)


dummyDf = pd.DataFrame(columns=keywords, data=data)

finalDf = pd.concat([finalDf, dummyDf], axis=1, join='inner')

df = pd.concat([initialDf, dummyDf], axis=1, join='inner')
```

In [144…

```
df.head()
```

Out[144…

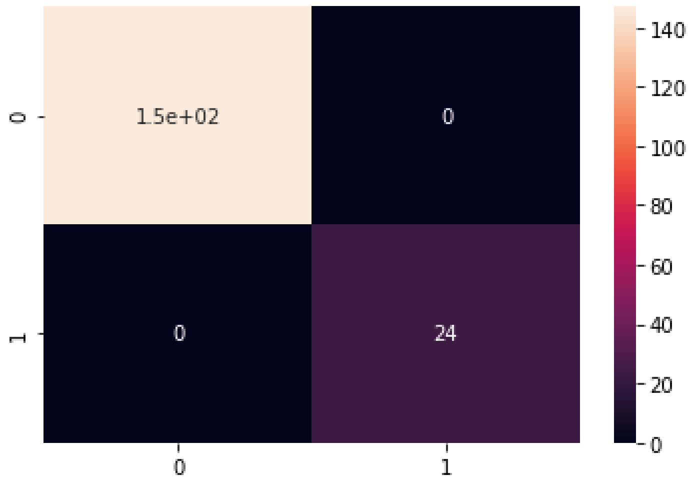
	svg	title	meta	form	use	noscript	li	blockquote	p	link	...	"	function	console	cmd	disp
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	
3	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	
4	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	

5 rows × 96 columns

In [145…

```
df = normalize(df)
uniqueClusters,clustering,df = clusteringDB(df)

score = relevanceScore(uniqueClusters,df)

maxScoreClusters = highScore(score)

df = labelMarking(maxScoreClusters,df)

X_train, X_test, y_train, y_test = trainTestSplit(df)

prediction = svmModel(X_train,y_train,X_test)

performance(y_test,prediction)
```

```
F1 Score of SVM Model
1.0
Confusion Matrix
```



HTML tags CSS Class as attribute

In [146…

```
tagsClass = []

for attr in attrs:
    tagsClass.append(attr.get("class"))

classList = []
for i in tagsClass:
    if i:
        for j in i:
            classList.append(j)

uniqueClass = list(set(classList))

data = []
zeroes = [0 for i in range(len(uniqueClass))]
for iClass in tagsClass:
    row=[]
    if iClass:
        for j in uniqueClass:
            if j in iClass:
                row.append(1)
                else:
                    row.append(0)
            data.append(row)
    else:
        data.append(zeroes)


dummyDf = pd.DataFrame(columns=uniqueClass,data=data)

finalDf = pd.concat([finalDf, dummyDf], axis=1, join='inner')

df = pd.concat([initialDf, dummyDf], axis=1, join='inner')
```

In [147…

```
df.head()
```

Out[147…

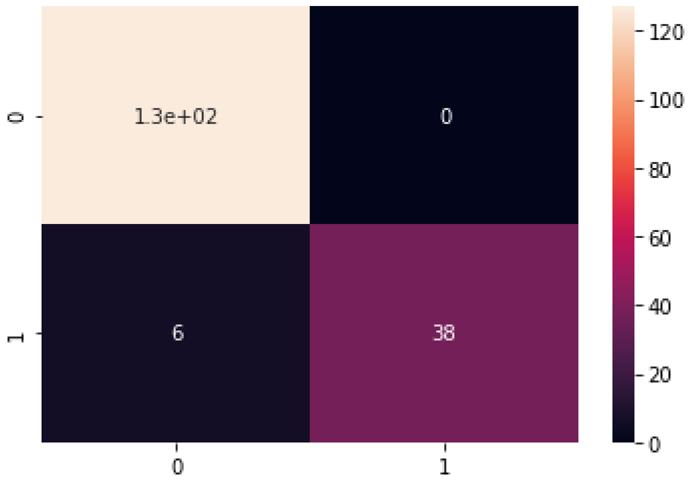
	svg	title	meta	form	use	noscript	li	blockquote	p	link	...	hid_sml-dvc	imgbrd	reddit	adc sectio
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	
1	0	0	1	0	0	0	0	0	0	0	...	0	0	0	
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	
3	0	0	1	0	0	0	0	0	0	0	...	0	0	0	
4	0	0	1	0	0	0	0	0	0	0	...	0	0	0	

5 rows × 121 columns

In [148…

```
df = normalize(df)
uniqueClusters,clustering,df = clusteringDB(df)

score = relevanceScore(uniqueClusters,df)

maxScoreClusters = highScore(score)

df = labelMarking(maxScoreClusters,df)

X_train, X_test, y_train, y_test = trainTestSplit(df)

prediction = svmModel(X_train,y_train,X_test)

performance(y_test,prediction)
```

```
F1 Score of SVM Model
0.9268292682926829
Confusion Matrix
```

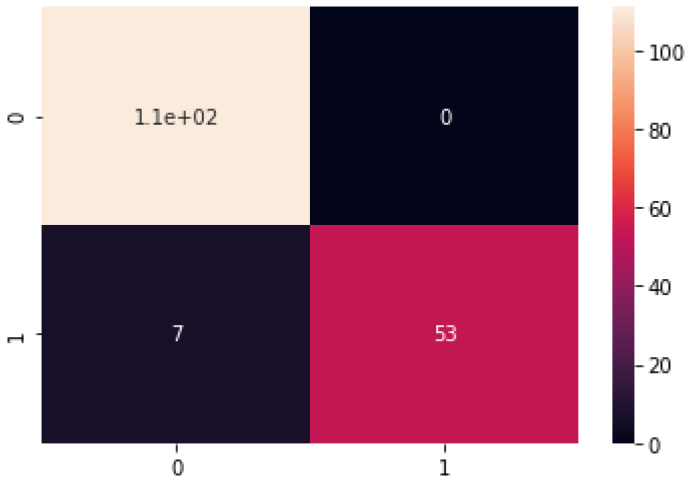


Combined DataSet

In [149…

```
df = normalize(finalDf)
uniqueClusters,clustering,df = clusteringDB(df)

score = relevanceScore(uniqueClusters,df)

maxScoreClusters = highScore(score)

df = labelMarking(maxScoreClusters,df)

X_train, X_test, y_train, y_test = trainTestSplit(df)

prediction = svmModel(X_train,y_train,X_test)

performance(y_test,prediction)
```

```
F1 Score of SVM Model
0.9380530973451328
Confusion Matrix
```



In [ ]:

In [ ]: