URL Categorization

Domantas Meidus

Abstract

In this document we report our proposal for the application of supervised classification methods to the URL Categorization problem. The classifiers that we have selected for the classification tasks were: Logistic regression, Decision Trees, KNeighbors. I have implemented the classification process using the scikit-learn, nltk, numpy, urlib, BeautifulSoup and langdetect libraries. The classifiers is learnt by using the train data and computing the accuracy in the test data. From our results the best accuracy has been produced by the Logistic Regression Classifier.

1 Description of the problem

The task we have to solve is the classification of the "URL categorization", introduced in the project task.

The dataset includes category which each URL belongs to. The goal of the project is to predict the category of URL, among as many categories.

The database has the following characteristics:

- 18 attributes.
- 25 categories:
  1 News_and_Media
  2 Career_and_Education
  3 Internet_and_Telecom
  4 Sports
  5 Games
  6 Books_and_Literature
  7 Law_and_Government
  8 Travel
  9 Finance
  10 Arts_and_Entertainment
  11 Reference
  12 Autos_and_Vehicles
  13 Adult
  14 Business_and_Industry
  15 People_and_Society
  16 Shopping
  17 Health
  18 Pets_and_Animals
  19 Home_and_Garden
  20 Beauty_and_Fitness
  21 Gambling

22 Recreation_and_Hobbies
23 Science
24 Computer_and_Electronics
25 Food_and_Drink

2 Description of our approach

The implementation of the project according to the tasks:

1. Design any preprocessing of the web pages in the dataset.
2. Define and learn the classifier using the training data.
3. Design the validation method to evaluate the accuracy of the proposed classification approach.

2.1 Preprocessing

For preprocessing data these steps have been made:

1. Define setup variables which program needs to start up:
   file: path to the dataset in the local computer.
   limiter: how many lines of dataset we want to preprocess.
   cv_number: determine in how many parts program splits labels for training and testing cases using cross validation method.
   top: a number which represents how many most frequent words is stored for each category
   char_blacklist, stopwords, language_whitelist, domains_whitelist, english_vocab - these variables are for URL filtering.
2. Check if limiter and cv_number meets minimum requirements to prevent cross validation method errors:
   if cv_number $<$ 2: cv_number = 2
   if limiter $<=$ 500:
   if limiter $>=$ 150 and cv_number $>$ 2: cv_number = 2
   if limiter $>=$ 250 and cv_number $<$ 3 or cv_number $>$ 3: cv_number = 3
   if limiter $>=$ 350 and cv_number $<$ 4 or cv_number $>$ 4: cv_number = 4
   if limiter $>=$ 500 and cv_number $<$ 5 or cv_number $>$ 5: cv_number = 5
   if limiter $<$ 150: limiter = 150, cv_number = 2
3. Using urlopen [1] html content of URL is downloaded and with BeautifulSoup tool it is converted to the plain text if URL "Main category" is not 'Not working' and main_category:confidence value is greater than 0.5.
4. Using langdetect tool by plain URL text program detect language which is used in URL content.
5. If detected language is english and URL domain belongs to 'com', 'org', 'net', '.us', '.uk', '.au' or '.ca' domains - text is tokenize by using nltk tool.
   These actions is done for all URL from URL categorization file.
6. After all URL is filtered, our program determinate which categories are compatible with cross validation predict method: if each category is used more than the number of cross validation predict parameter CV, that means that category can be predicted by cross validation predict method. Take note that Cross validation accepts those classes which have minimum 2 of training data.
   These applicable categories are marked as classifier labels.
7. For each category top 50 words which are the most frequently used in URL is stored in list excluding "stop words" and symbols in range 32-64 and 91 - 96 decimal values which represented in ASCII table (http://www.asciitable.com/).

8. Vector with zero values is created for each URL with top 50 words for each category. If each filtered URL top most frequently words contains categories used most frequently words, then value of word position in vector is changed to value 1.

   After this action vector representation of each filtered URL is marked as classifier features.

The data was split into two different sets: train and test, for validation. We use the same train set to learn all the classifiers, and the same test set for evaluating their accuracy.

2.2 Classifiers

We use three different classifiers:

1. Logistic regression
   - C=1.0
   - class_weight=None
   - dual=False
   - fit_intercept=True
   - intercept_scaling=1
   - max_iter=100
   - multi_class='ovr'
   - n_jobs=1
   - penalty='l2'
   - solver='liblinear'
   - tol=0.0001
   - verbose=0
   - warm_start=False
2. Decision trees
   - class_weight=None
   - criterion='gini'
   - max_depth=None
   - max_features=None
   - max_leaf_nodes=None
   - min_impurity_decrease=0.0'
   - min_impurity_split=None
   - min_samples_leaf=1
   - min_samples_split=2
   - min_weight_fraction_leaf=0.0
   - presort=False
   - random_state=None
   - splitter='best'
3. KNeighbors with parameters:
   - algorithm='auto'
   - leaf_size=30
   - metric='euclidean'
   - metric_params=None
   - n_jobs=1
   - n_neighbors=5
   - p=2
   - weights='uniform'

For each classifier, these were the parameters by default of the scikit-learn library.

2.3 Validation

To validate our results we compute the classifier accuracy in the test data. Another possibility was to compute the cross-validation in the complete dataset but we used the split between train and test because it was simpler.

As an additional validation step we computed the confusion matrices for the twenty four classifiers.

3 Implementation

All the project steps were implemented in Python. Libraries that was used in the project: urlib and BeautifulSoup for downloading URL and parsing HTML code into plain text, langdetect for language detection from plain text, numpy for using data structures to store labels and features information and scikit-learn for the classification tasks. Illustration how the implementation works in the Python notebook `URL-categorization-Jupyter-Notebook.ipynb`.

4 Results

The highest score performance comparing of all 3 classifiers has Logistic Regression. The score results are displayed in table 1. The score results for each class displayed for Logistic Regression 1 figure, Decision tree 2 figure, KNeighbors 3 figure.

	Logistic Regression	**Decision tree**	**KNeighbors**
Precision score:	0.40	0.28	0.23
recall:	0.49	0.28	0.11
f1-score:	0.46	0.28	0.09
Accuracy score:	0.49	0.28	0.10

Table 1: Scores produced by Logistic Regression.

	precision	recall	f1-score
0	0.50	0.09	0.15
1	0.48	0.37	0.42
2	0.37	0.56	0.44
3	0.62	0.72	0.67
4	0.61	0.63	0.62
5	0.50	0.20	0.29
6	1.00	0.33	0.50
7	0.00	0.00	0.00
8	0.50	0.42	0.46
9	0.40	0.46	0.43
10	0.33	0.11	0.17
11	0.62	0.67	0.64
12	0.71	0.38	0.50
13	0.37	0.45	0.41
14	0.45	0.57	0.50
15	0.54	0.56	0.55
16	0.57	0.86	0.68
17	0.00	0.00	0.00
18	0.33	0.38	0.36
19	0.46	0.40	0.42
20	0.68	0.62	0.65
21	0.19	0.23	0.21
22	0.30	0.12	0.17
23	0.44	0.41	0.42
24	0.44	0.18	0.26

Figure 1: Logistic Regression scores for each class

	precision	recall	f1-score
0	0.12	0.09	0.10
1	0.19	0.20	0.20
2	0.21	0.26	0.23
3	0.45	0.36	0.40
4	0.31	0.34	0.32
5	0.25	0.20	0.22
6	0.69	0.60	0.64
7	0.00	0.00	0.00
8	0.13	0.12	0.12
9	0.26	0.26	0.26
10	0.12	0.11	0.12
11	0.40	0.43	0.42
12	0.23	0.23	0.23
13	0.24	0.27	0.25
14	0.36	0.34	0.35
15	0.26	0.23	0.24
16	0.51	0.57	0.54
17	0.09	0.10	0.10
18	0.31	0.38	0.34
19	0.10	0.07	0.08
20	0.44	0.52	0.48
21	0.13	0.20	0.16
22	0.08	0.04	0.05
23	0.14	0.09	0.11
24	0.10	0.14	0.11

Figure 2: Decision Tree scores for each class

	precision	recall	f1-score
0	0.00	0.00	0.00
1	0.00	0.00	0.00
2	0.07	0.74	0.12
3	0.77	0.30	0.43
4	0.12	0.06	0.08
5	0.00	0.00	0.00
6	0.00	0.00	0.00
7	0.00	0.00	0.00
8	0.05	0.08	0.06
9	0.27	0.10	0.15
10	0.00	0.00	0.00
11	0.50	0.02	0.04
12	0.00	0.00	0.00
13	0.10	0.02	0.03
14	0.00	0.00	0.00
15	0.67	0.05	0.10
16	0.45	0.06	0.10
17	0.00	0.00	0.00
18	0.07	0.38	0.12
19	0.00	0.00	0.00
20	0.31	0.24	0.27
21	0.12	0.03	0.05
22	0.00	0.00	0.00
23	0.50	0.03	0.06
24	0.00	0.00	0.00

Figure 3: KNeighbors scores for each class

The results of the computation of the confusion matrices for Logistic regression, Decision tree and KNeighbors classifiers are respectively shown in Tables 4, 5 and 6.

Predicted	0	1	2	3	4	5	6	8	9	10	...	16	17	18	19	20	21	22	23	24	All
True											...										
0	2	0	2	6	0	0	0	1	1	0	...	1	0	0	1	0	3	0	1	1	22
1	0	11	0	1	1	0	0	1	0	0	...	4	1	0	0	0	0	2	1	0	30
2	0	0	30	1	0	0	0	0	5	1	...	1	0	0	0	1	4	0	1	0	54
3	0	1	5	48	2	1	0	0	0	0	...	2	0	1	2	0	1	0	1	0	67
4	0	0	2	3	22	0	0	0	1	0	...	2	0	0	0	0	2	1	1	0	35
5	0	1	2	0	1	5	0	1	1	1	...	3	0	1	0	1	1	0	2	0	25
6	0	0	0	0	0	0	5	2	1	0	...	1	0	1	0	0	1	0	0	0	15
7	0	0	1	2	0	0	0	0	1	0	...	0	0	0	0	0	1	0	0	0	10
8	0	1	2	0	0	0	0	11	1	0	...	5	0	0	0	0	1	0	1	0	26
9	1	1	5	1	2	0	0	0	18	0	...	0	0	1	0	0	3	0	0	0	39
10	0	0	2	1	0	2	0	0	1	2	...	1	0	0	1	0	0	1	1	0	18
11	0	0	1	3	1	0	0	2	1	0	...	0	0	1	0	1	2	0	0	1	51
12	0	0	2	0	0	0	0	0	1	1	...	0	0	0	0	0	3	0	0	0	13
13	0	2	5	1	0	1	0	0	0	0	...	3	0	2	1	0	2	0	3	0	49
14	0	2	3	1	0	0	0	1	1	1	...	7	0	0	2	0	3	0	1	0	53
15	0	0	0	1	0	0	0	0	1	0	...	0	0	1	6	0	2	0	1	0	39
16	0	0	1	1	1	0	0	0	0	0	...	77	0	0	4	0	1	0	0	0	90
17	0	0	0	0	0	0	0	0	0	0	...	3	0	0	1	0	1	1	0	1	10
18	0	1	2	0	0	0	0	0	2	0	...	1	0	5	0	0	0	0	0	0	13
19	0	0	3	0	1	1	0	1	2	0	...	7	0	1	17	1	1	0	0	1	43
20	0	0	1	2	1	0	0	0	0	0	...	2	0	0	1	13	0	0	0	0	21
21	1	0	7	2	1	0	0	0	3	0	...	4	0	0	1	0	8	0	1	0	35
22	0	3	1	1	1	0	0	0	1	0	...	6	0	0	0	1	1	3	2	0	25
23	0	0	4	2	1	0	0	0	2	0	...	3	0	0	0	0	0	2	14	1	34
24	0	0	0	0	1	0	0	2	1	0	...	3	1	1	0	1	1	0	1	4	22
All	4	23	81	77	36	10	5	22	45	6	...	136	2	15	37	19	42	10	32	9	839

Figure 4: Logistic Regression confusion matrix

Predicted	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	All
True											...										
0	2	0	2	3	1	0	1	0	1	1	...	2	1	0	1	1	0	2	0	0	22
1	0	6	3	1	1	1	1	0	0	2	...	4	1	0	0	1	0	0	1	1	30
2	1	3	14	3	2	0	1	1	1	2	...	3	0	0	0	1	4	3	3	0	54
3	3	2	4	24	7	1	0	0	2	2	...	1	3	1	4	3	2	0	0	3	67
4	2	1	3	5	12	3	0	0	0	1	...	2	0	0	1	1	0	0	1	1	35
5	1	1	1	1	3	5	0	2	0	1	...	0	0	1	0	0	1	0	1	0	25
6	0	0	0	0	0	0	9	0	0	0	...	1	0	1	0	0	2	0	0	1	15
7	0	0	2	1	0	0	0	0	1	0	...	1	0	0	0	0	1	0	0	0	10
8	0	2	4	0	0	0	0	0	3	0	...	2	0	0	1	0	3	1	0	0	26
9	4	2	0	2	1	1	0	0	1	10	...	2	0	2	1	0	1	0	0	4	39
10	0	1	2	0	0	1	1	0	1	1	...	0	0	0	0	1	0	1	1	1	18
11	0	0	2	1	1	0	0	1	3	2	...	2	1	1	0	1	2	1	2	0	51
12	0	1	2	0	0	0	0	1	0	1	...	0	0	0	1	1	2	0	0	0	13
13	0	0	5	0	0	0	0	1	2	1	...	5	0	2	3	0	4	0	3	3	49
14	1	3	4	3	0	1	0	0	1	3	...	4	0	0	1	0	4	0	0	1	53
15	0	1	2	1	1	0	0	1	1	4	...	3	1	1	2	0	3	0	1	4	39
16	0	2	1	1	0	0	0	1	0	1	...	51	0	0	8	0	3	1	3	1	90
17	0	1	1	1	0	0	0	0	0	0	...	1	1	0	1	0	1	0	0	1	10
18	0	0	0	0	2	0	0	0	0	0	...	0	0	5	0	0	1	0	0	0	13
19	0	1	5	1	2	1	0	1	1	2	...	6	1	1	3	3	2	1	0	3	43
20	0	0	2	2	2	0	0	0	1	1	...	0	0	0	0	11	0	0	0	0	21
21	1	1	3	3	2	2	0	1	1	1	...	1	0	0	0	1	7	0	1	1	35
22	0	2	1	0	0	1	0	1	1	1	...	2	1	0	1	0	5	1	1	2	25
23	2	1	3	0	1	2	0	1	0	2	...	4	0	0	1	0	5	1	3	1	34
24	0	0	2	0	1	1	0	1	2	0	...	3	1	1	2	0	0	0	0	3	22
All	17	31	68	53	39	20	13	13	23	39	...	100	11	16	31	25	53	12	21	31	839

Figure 5: Decision tree confusion matrix

Predicted	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	18	20	21	23	24	All
True																					
0	0	19	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	22
1	0	27	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	30
2	1	40	1	1	0	1	0	2	2	0	0	0	1	0	0	4	0	1	0	0	54
3	0	40	20	0	0	0	0	1	0	0	0	0	0	0	0	4	0	2	0	0	67
4	0	29	0	2	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	35
5	0	18	0	0	0	0	0	2	0	0	0	2	0	0	0	3	0	0	0	0	25
6	0	6	1	0	1	0	0	1	0	0	0	0	0	0	0	5	1	0	0	0	15
7	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	10
8	0	17	0	0	1	0	0	2	1	0	0	1	0	0	0	3	1	0	0	0	26
9	0	28	0	0	0	0	1	1	4	0	0	0	0	0	0	3	2	0	0	0	39
10	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	18
11	0	39	1	0	0	2	0	3	0	1	1	0	0	0	0	4	0	0	0	0	51
12	0	10	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	13
13	0	36	0	2	1	0	0	3	0	0	0	1	0	0	0	5	1	0	0	0	49
14	0	35	0	1	1	0	0	7	3	0	0	0	0	0	2	4	0	0	0	0	53
15	0	22	0	2	0	0	0	2	0	0	1	1	0	2	1	6	0	2	0	0	39
16	0	63	0	4	0	3	0	3	0	2	0	1	0	0	5	7	2	0	0	0	90
17	0	7	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	10
18	0	3	0	0	3	1	0	1	0	0	0	0	0	0	0	5	0	0	0	0	13
19	0	31	0	0	0	0	0	2	1	0	0	1	0	0	1	7	0	0	0	0	43
20	0	10	0	0	0	1	0	3	1	0	0	0	0	0	0	1	5	0	0	0	21
21	0	27	0	1	1	2	0	1	0	0	0	0	0	0	0	1	1	1	0	0	35
22	0	18	0	2	0	0	1	0	0	0	0	0	0	0	1	1	1	1	0	0	25
23	1	21	1	0	0	1	0	1	0	0	0	1	0	1	0	5	0	1	1	0	34
24	0	15	1	0	0	0	0	3	0	0	0	1	0	0	0	2	0	0	0	0	22
All	2	587	26	17	8	13	2	41	15	3	2	10	1	3	11	72	16	8	2	0	839

Figure 6: KNeighbors confusion matrix

5 Conclusions

In our project we have applied Logistic regression, Decision tree and KNeighbors classifiers to the "URL categorization", introduced in [2]. We have computed the accuracy of these classifiers and observed that: **Logistic regression** produces the highest accuracy. We have also observed, from the analysis of the confusion matrices, that Logistic Regression produced poor discrimination between the People_and_Society and an Health categories. Also, it was poorly trained to detect Pets_and_Animals category.
KNeighbors was poorly trained to detect Pets_and_Animals category.
Logistic regression produced poor discrimination between the Health and an Internet_and_Telecom. Also, for this classifier it was unable to define Food_and_Drink category.

References

[1] urllib documentation. "`https://docs.python.org/3/library/urllib.request.html`".

[2] URL categorization. "`https://www.crowdflower.com/data-for-everyone/`", 2015.