Project

Statistical NLP

Author Features Prediction

Description

Classification is probably the most popular task that you would deal with in real life.
Text in the form of blogs, posts, articles, etc. is written every second. It is a challenge to predict the information about the writer without knowing about him/her.

We are going to create a classifier that predicts multiple features of the author of a given text.
We have designed it as a Multilabel classification problem.

Dataset

Blog Authorship Corpus
Over 600,000 posts from more than 19 thousand bloggers

The Blog Authorship Corpus consists of the collected posts of 19,320 bloggers gathered from blogger.com in August 2004. The corpus incorporates a total of 681,288 posts and over 140 million words - or approximately 35 posts and 7250 words per person.

Each blog is presented as a separate file, the name of which indicates a blogger id# and the blogger's self-provided gender, age, industry, and astrological sign. (All are labelled for gender and age but for many, industry and/or sign is marked as unknown.)

All bloggers included in the corpus fall into one of three age groups:

8240 "10s" blogs (ages 13-17),

8086 "20s" blogs(ages 23-27)

2994 "30s" blogs (ages 33-47)

For each age group, there is an equal number of male and female bloggers.

Each blog in the corpus includes at least 200 occurrences of common English words. All formatting has been stripped with two exceptions. Individual posts within a single blogger are separated by the date of the following post and links within a post are denoted by the label urllink.

Link to dataset: https://www.kaggle.com/rtatman/blog-authorship-corpus

Steps & Approach

1. Load the dataset (5 points)
    a. Tip: As the dataset is large, use fewer rows. Check what is working well on your machine and decide accordingly.
2. Preprocess rows of the "text" column (7.5 points)
    a. Remove unwanted characters
    b. Convert text to lowercase
    c. Remove unwanted spaces
    d. Remove stopwords
3. As we want to make this into a multi-label classification problem, you are required to merge all the label columns together, so that we have all the labels together for a particular sentence (7.5 points)
    a. Label columns to merge: "gender", "age", "topic", "sign"
    b. After completing the previous step, there should be only two columns in your data frame i.e. "text" and "labels" as shown in the below image

	text	labels
0	info found pages mb pdf files wait untill team...	[male, 15, Student, Leo]
1	team members drewes van der laag urllink mail ...	[male, 15, Student, Leo]
2	het kader van kernfusie op aarde maak je eigen...	[male, 15, Student, Leo]
3	testing testing	[male, 15, Student, Leo]
4	thanks yahoo toolbar capture urls popups means...	[male, 33, InvestmentBanking, Aquarius]

4. Separate features and labels, and split the data into training and testing (5 points)
5. Vectorize the features (5 points)
    a. Create a Bag of Words using count vectorizer
        i. Use ngram_range=(1, 2)
        ii. Vectorize training and testing features
    b. Print the term-document matrix

6. Create a dictionary to get the count of every label i.e. the key will be label name and value will be the total count of the label. Check below image for reference (5 points)

```
{'male': 2272,
 '15': 299,
 'Student': 403,
 'Leo': 55,
 '33': 94,
 'InvestmentBanking': 70,
 'Aquarius': 286,
 'female': 728,
```

7. Transform the labels - (7.5 points)
   As we have noticed before, in this task each example can have multiple tags. To deal with such kind of prediction, we need to transform labels in a binary form and the prediction will be a mask of 0s and 1s. For this purpose, it is convenient to use MultiLabelBinarizer from sklearn
   a. Convert your train and test labels using MultiLabelBinarizer
8. Choose a classifier - (5 points)
   In this task, we suggest using the One-vs-Rest approach, which is implemented in OneVsRestClassifier class. In this approach k classifiers (= number of tags) are trained. As a basic classifier, use LogisticRegression. It is one of the simplest methods, but often it performs good enough in text classification tasks. It might take some time because the number of classifiers to train is large.
   a. Use a linear classifier of your choice, wrap it up in OneVsRestClassifier to train it on every label
   b. As One-vs-Rest approach might not have been discussed in the sessions, we are providing you with the code for that

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(solver='lbfgs')
clf = OneVsRestClassifier(clf)
```

9. Fit the classifier, make predictions and get the accuracy (5 points)
   a. Print the following
      i. Accuracy score
      ii. F1 score
      iii. Average precision score
      iv. Average recall score
      v. Tip: Make sure you are familiar with all of them. How would you expect the things to work for the multi-label scenario? Read about micro/macro/weighted averaging
10. Print true label and predicted label for any five examples (7.5 points)

Support

You can clarify your queries by dropping a mail to Olympus

Happy Learning!