TAGS PREDICTION USING NLP

Aditi Gode, Mitali Tavildar, Tanvi Kulkarni

INTRODUCTION

- The developer community has evolved extensively and we notice a wide use of this through websites like Stack Overflow, Stack Exchange, Medium etc.
- The discussions and blogs are a great source of guidance if we are stuck with an issue and trying to resolve it, or maybe need to quickly understand a concept.
- The process of searching for an appropriate discussion thread or perhaps a blog that perfectly approaches our issue is in itself a task; this is where tags come in handy.
- We propose to systemize these blogs by sorting them according to their respective topic of discussions which can be done by predicting the tag for such blogs.

PROBLEM STATEMENT

- The project aims to classify and predict the tags of given blogs/discussions threads.
- We tried to answer the following research questions:
    a. As this is a multi-label prediction problem, do the predicted tags intuitively grasp the context of the blogs and point to the most relevant classes?
    b. Which ML technique works best for blog tag Prediction?
    c. Does the deep learning model work better than the traditional machine learning model?

PRIOR WORK

[1] Bianca Iancu, Gabriele Mazzola, K. Psarakis, Panagiotis Soilis, “Multi-label Classification for Automatic Tag Prediction in the Context of Programming Challenges”, 2019.

[2] Meghashyam Chinta, “Predicting Tags for the Questions in Stack Overflow”, Medium, 2018.

[3] David Ten, “xang1234”, GitHub.

[1] The authors have experimented with various feature engineering techniques like TF-IDF, word2vec, etc. and in combination with various models such as Random Forest, LSTM,RNN

[2] The blog extensively focuses on multi label classification problem for stackoverflow data.

[3] The Github repo presents implementation of most multi-label classification algorithm like MLkNN, Binary Relevance, Classifier Chains, Power Labelset.

APPROACH

1. Utilize data of websites like Stack Overflow from Kaggle
2. Preprocess the data: Data cleaning, tokenization, etc.
3. Feature Engineering: TF-IDF, Countvectorizer
4. Train the models for multilabel classification and test for prediction
5. Compare the performance of the various models to find the best performing model

DESCRIPTION OF DATASET

The dataset has 4 features such as: ID, Title, Body, Tags.

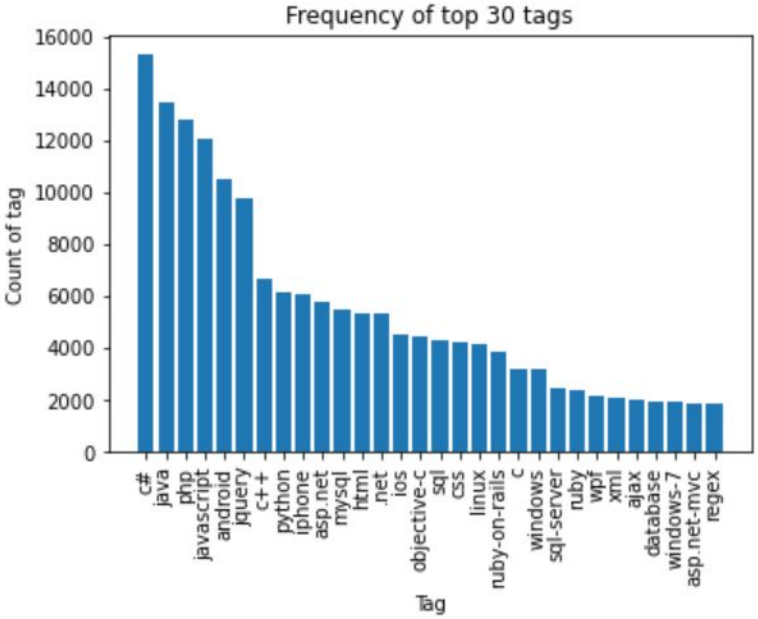
	Id	Title	Body	Tags
0	1	How to check if an uploaded file is an image w...	<p>I'd like to check if an uploaded file is an...	php image-processing file-upload upload mime-t...
1	2	How can I prevent firefox from closing when I ...	<p>In my favorite editor (vim), I regularly us...	firefox
2	3	R Error Invalid type (list) for variable	<p>I am import matlab file and construct a dat...	r matlab machine-learning
3	4	How do I replace special characters in a URL?	<p>This is probably very simple, but I simply ...	c# url encoding
4	5	How to modify whois contact details?	<pre><code>function modify(.....)\n{\n $mco...	php api file-get-contents
5	6	setting proxy in active directory environment	<p>I am using a machine on which active direct...	proxy active-directory jmeter
6	7	How to draw barplot in this way with Coreplot	<p>My image is cannot post so the link is my ...	core-plot
7	8	How to fetch an XML feed using asp.net	<p>I've decided to convert a Windows Phone 7 a...	c# asp.net windows-phone-7

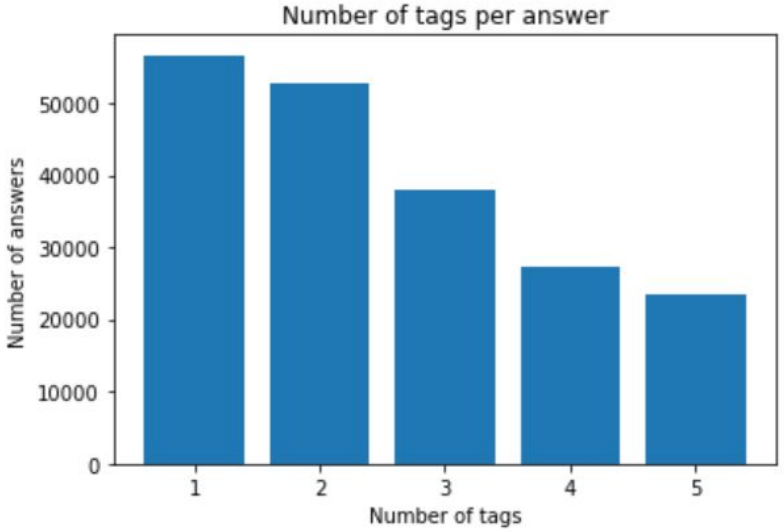
IMPLEMENTATION

A] Data Pre-processing:

- Duplicate values of features ‘Title’ and ‘Tags’ were removed.
- The data was cleaned and rows with null tag values were dropped.
- 200k data points were sampled due to compute and memory limitations.
- HTML Tags were removed using BeautifulSoup from the body and title.
- Special characters were removed from the body and title using regular expressions.
- Body and title were tokenized and stop words were removed except ‘C’ and ‘R’ as they are significant keywords.
- All the words were converted into their root form using WordNetLemmatizer.

B] Exploratory Data Analysis (EDA):





C] Models Implemented:

We built models using various techniques used for multi-label classification such as MLkNN, One Vs Rest and Binary Relevance:

1. **MLkNN**: Lazy learning approach, identified k nearest neighbors.

2. **One Vs Rest**: One model for each class!
   We used the following base classifiers with One Vs Rest:
   a. SGD Classifier using log loss and hinge loss
   b. Multinomial Naive Bayes

C] Models Implemented:

3. **Binary Relevance**: Union of all predicted classes is taken.
   We used the following base classifiers with Binary Relevance:
   a. Random Forest
   b. Gaussian Naive Bayes

D] Evaluation Metrics:

- We have used the following evaluation metrics: Precision, Recall, F1 Score, Hamming Loss
- Why not Accuracy? 🤔

EXPERIMENTS

1. Choosing appropriate number of tags i.e. labels.
2. Hyperparameter tuning
3. Utilizing features ‘body’ and ‘title’ while building models
4. Deep Learning Models

RESULTS

Performance Evaluation of the various models:

		Micro Average			Macro Average			Hamming Loss
		Precision	Recall	F1-score	Precision	Recall	F1-score	
One Vs Rest	SGD with log loss	0.75	0.4	0.52	0.59	0.31	0.39	0.0029
	SGD with hinge loss	0.8	0.41	0.54	0.54	0.31	0.37	0.0027
	Multinomial Naïve Bayes	0.52	0.28	0.36	0.17	0.07	0.08	0.0039
Binary Relevance	Gaussian Naïve Bayes	0.21	0.26	0.23	0.02	0.01	0.01	0.0034
	Random Forest	0.85	0.27	0.41	0.37	0.1	0.13	0.0031
MLkNN		0.3	0.32	0.31	0.3	0.29	0.26	0.0057

RESULTS

Prediction for OneVsRest classifier with Logistic Regression as the base classifier:

In [231]:
```
test_sent = ["select * from table"]
qs = tfidf_body.transform(test_sent)
qs2 = tfidf_title.transform(test_sent)
q = hstack([qs,qs2])
op = classifier_log.predict(q)
print(op)
print("Predicted tags are:")
print(vec.get_feature_names()[300],vec.get_feature_names()[318],vec.get_feature_names()[338],vec.get_feature_names()[
     ,vec.get_feature_names()[380],vec.get_feature_names()[399],vec.get_feature_names()[400],vec.get_feature_names()[
     vec.get_feature_names()[419],vec.get_feature_names()[432])
```

```
  (0, 300)      1
  (0, 318)      1
  (0, 338)      1
  (0, 351)      1
  (0, 380)      1
  (0, 399)      1
  (0, 400)      1
  (0, 402)      1
  (0, 419)      1
  (0, 432)      1
Predicted tags are:
mysql oracle postgresql query select sql sql-server sql-server-2008 table tsql
```

RESULTS

Prediction for MLkNN classifier:

```
In [240]: test_sent = ["<p> body fit here <br>"]
          qs1 = body_vectorizer.transform(test_sent)
          qs2 = title_vectorizer.transform(test_sent)
          q = hstack([qs1,qs2])
          q.shape

Out[240]: (1, 514151)

In [241]: op = mlknn_classifier.predict(q)
          np.where(op.toarray() == 1)

Out[241]: (array([0]), array([97]))

In [242]: vectorizer.inverse_transform(op)

Out[242]: [array(['css'], dtype='<U25')]
```

RESULTS

Prediction for Binary Relevance classifier:

```
test1 = ['problem eclipse regard xml file eclipse complain android scrollbars android fadingedge allow string check android developer sit
e fact accept string xml file relate question pose problem miss android xmlns see code line begin xmlns correct believe complete file con
tent xml version 1 0 encode utf 8 com example todolist todolistitemview xmlns android http schema android com apk res android android lay
out width fill parent android layout height fill parent android pad 10dp android scrollbars verticle android textcolor color notepad text
android fadingedge verticle']
tfidf_test = TfidfVectorizer(min_df=0.00009, max_features=20000, smooth_idf=True, norm="l2",tokenizer = lambda x: x.split(), sublinear_tf
=False,ngram_range=(1,4))
test2 = ['eclipse complain android scrollbars android fadingedge allow string include code']

qs = tfidf_body.transform(test1)
# print(qs.shape)
qs2 = tfidf_title.transform(test2)
# print(qs2.shape)
q = hstack([qs,qs2])
```

Predicted Tags:

```
op = br.predict(q)
print(op)
```

```
  (0, 19)       1
  (0, 133)      1
  (0, 495)      1
```

```
print(vec.get_feature_names()[19])
print(vec.get_feature_names()[133])
print(vec.get_feature_names()[495])
```

```
android
eclipse
xml
```

CONCLUSION

- Research questions answered:
  1) As this is a multilabel prediction problem, do the predicted tags intuitively grasp the context of the blogs and point to the most relevant classes?
     - Models performed with precision values ranging from 0.2 to 0.85

  2) Which ML technique works best for blog tag prediction?
     - OneVsRest using SGD with loss as hinge (SVC) performed the best with a hamming loss of 0.0027

  3) Does the deep learning model work better than the traditional machine learning model?
     - RNN resulted in a low hamming seems promising
     - More data required to train these models. Encountered computational constraints

- Working on multilabel classification was a first-time experience. It was interesting and fun.

- Future scope: Diversifying the models by scraping data from various blogging websites. Will RNN work significantly better than other classifiers?

THANK YOU!