SW800 Advanced Diploma in Artificial Intelligence, NIELIT Calicut

Stack Overflow tag prediction

Members : Praseed M, Nizamudheen T I, Paul M David, Joby John, Kiran K B, Nandu A

Abstract

Natural language processing is a field with wide scope of application in this day and age of data analysis where the availability of data is plentiful. NLP helps us to sift through the gargantuan amount of data generated on the internet (mostly by social media networks) to find patterns and identify human behavior. Until recently, the focus in AI applications in NLP was on knowledge representation, logical reasoning, and constraint satisfaction - first applied to semantics and later to the grammar. In the last decade, a dramatic shift in the NLP research has led to the prevalence of very large scale applications of statistical methods, such as machine learning and data mining. Naturally, this also opened the way to the learning and optimization methods that constitute the core of modern AI, most notably genetic algorithms and neural networks.

Introduction

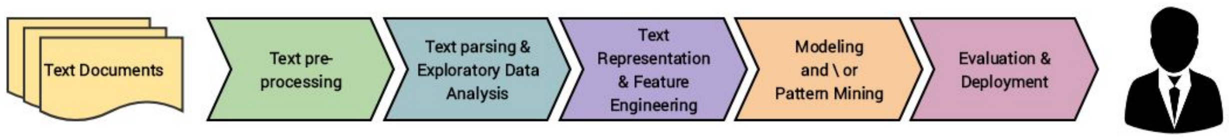
Tagging information is a vital aspect of social networks like Stack Overflow. The intent of our project is to develop a machine learning algorithm to accurately predict the relevant topic(s) present in the questions posted by Stack Overflow users. This process is imperative to ensure that experts of particular fields can easily receive and answer the questions instead of the painstaking process of sifting through the data manually to know if she/he can solve it.

Problem Statement : Predict the relevant topics present in Stack Overflow questions

(Multiple Labels in Questions)

Application : Searching for relevant Questions becomes simple and simplifies answering the questions by experts

NLP Processing Steps


Text Documents
Text pre-processing
Text parsing & Exploratory Data Analysis
Text Representation & Feature Engineering
Modeling and \ or Pattern Mining
Evaluation & Deployment

Tools

The tools we used for our project are:

Coding & Documentation

- Python
- Jupyter Notebook

Natural Language Processing

- NLTK

Model Implementation

- Scikit-learn
- scikit-multilearn

Data Management

- SQLite3
- Pandas
- SQLAlchemy

Plotting and visualization

- Matplotlib
- seaborn

1.Exploratory Data Analysis

Different fields in the Dataset

1. Id-Unique identifier for each question
2.Title - The question title
3.Body - The body of the question
4.Tags - Multiple tags in the question

Dataset Details

Size of Train.csv - 7.3GB
Size of Test.csv - 2.4GB
Number of rows in Train.csv = 6034195

Observations

In order to analyze the data to form a coherent ML prediction algorithm we did an initial scouring of the dataset.Our findings from the same are summarised below.

No.of distinct tags = 42048

Most popular tag : c# (frequency =331505)

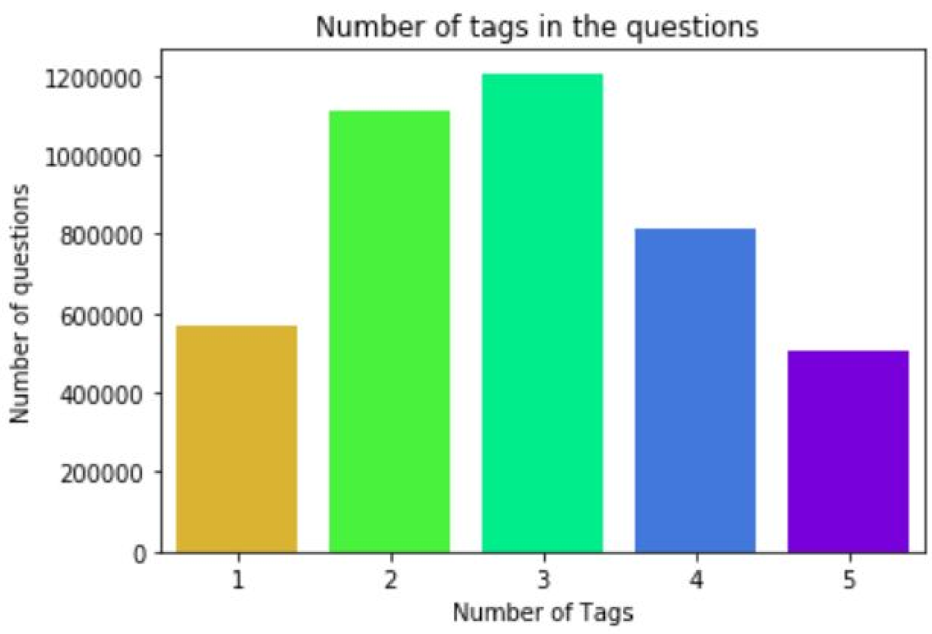
Tags occurings just once : 3065

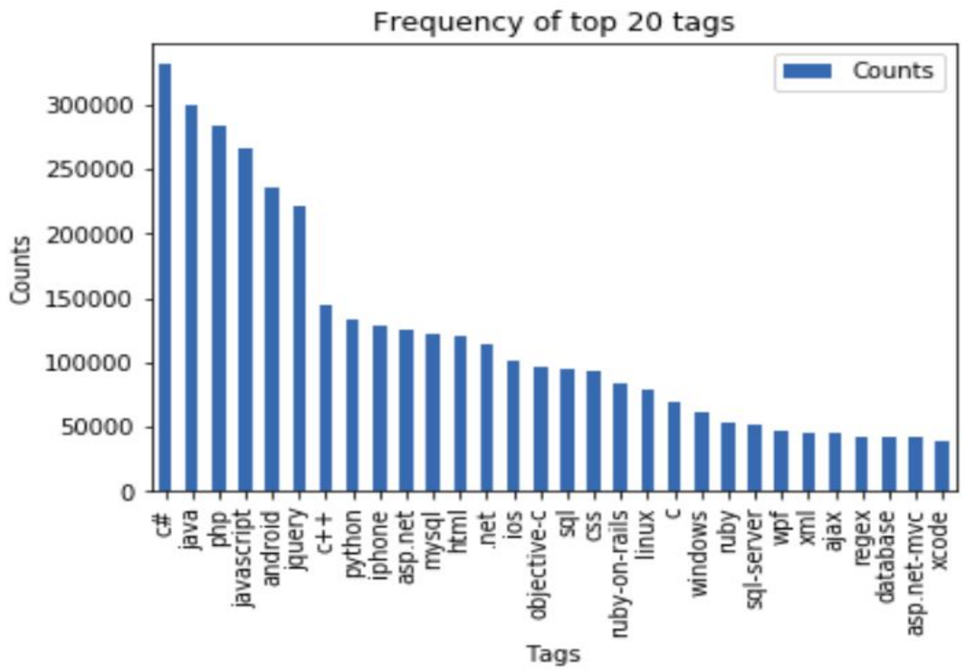
Maximum number of tags per question: 5

Minimum number of tags per question: 1

Average Number of tags per question(rounded up) : 3

Majority of the most frequent tags are **programming languages. C#** was the most frequent tag encountered. **Android, IOS, Linux and windows** are among the top most frequent operating systems tag.





In addition, we observed that only 42.11% of tags appear greater than 24 times.

2.Text Preprocessing

Since, text is the most unstructured form of all the available data, various types of noise are present in it and the data is not readily analyzable without any pre-processing. The entire process of cleaning and standardization of text, making it noise-free and ready for analysis is known as text preprocessing.

It is predominantly comprised of the following steps:

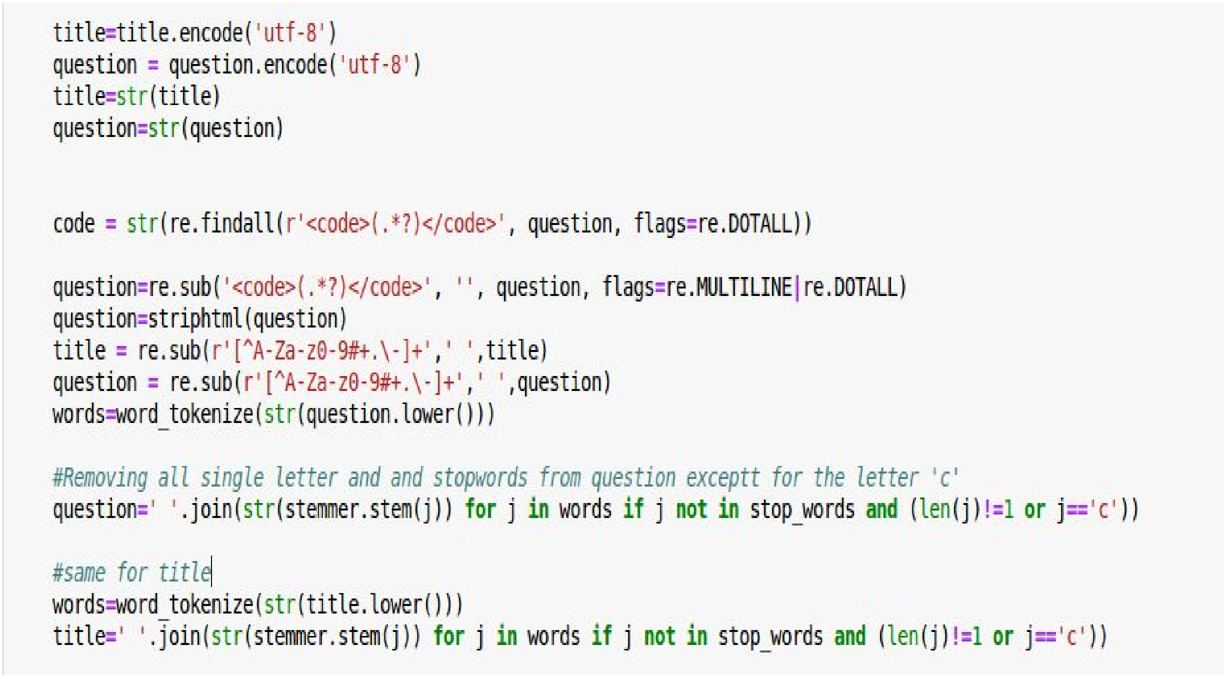
- **Noise Removal:**Any piece of text which is not relevant to the context of the data and the end-output can be specified as the noise. A general approach for noise removal is to prepare a dictionary of noisy entities and iterate the text object by tokens (or by words), eliminating those tokens which are present in the noise dictionary.We used a customized list of stopwords to improve the predictive power of than algorithm.Duplicate rows present in the train.csv dataset was removed.They accounted for 32 percent of the entire dataset.

- **Stemming:** Stemming is a rudimentary rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.We did this using snowball stemmer(NLTK library).

- **Removal of Outlier:**Tags which were present in sparse numbers(less than 24 occurrences) in the entire dataset were removed, treating them as outliers. This helped improve the f1-score significantly.

- **Removal of code snippets:**Code snippets present in the feature ‘body’ of each question is removed to reduce data size.

- **Tag Encoding:** Tags in Train.csv dataset are encode using OneHotEncoder to enable easy classification by the computation algorithm.



```
title=title.encode('utf-8')
question = question.encode('utf-8')
title=str(title)
question=str(question)


code = str(re.findall(r'<code>(.*?)</code>', question, flags=re.DOTALL))

question=re.sub('<code>(.*?)</code>', '', question, flags=re.MULTILINE|re.DOTALL)
question=striphtml(question)
title = re.sub(r'[^A-Za-z0-9#+.\-]+',' ',title)
question = re.sub(r'[^A-Za-z0-9#+.\-]+',' ',question)
words=word_tokenize(str(question.lower()))

#Removing all single letter and and stopwords from question exceptt for the letter 'c'
question=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_words and (len(j)!=1 or j=='c'))

#same for title
words=word_tokenize(str(title.lower()))
title=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_words and (len(j)!=1 or j=='c'))
```

Preprocessing code snippet

The average length of characters in each question(Title+Body) before preprocessing was 1170. After the implementation of multiple preprocessing steps, we were able to limit it to 325 characters per question.

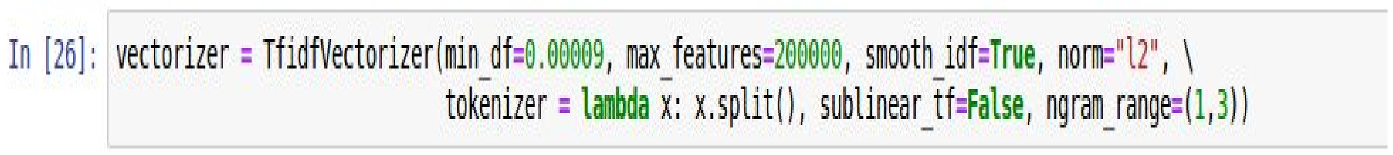
3.Text to Features

To analyse a preprocessed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using an assortment of techniques – tf-idf vectorization of 'title' and 'body' was implemented.

Statistical Features

- **Term Frequency (TF)** – TF for a term "t" is defined as the count of a term "t" in a document "D".

- **Inverse Document Frequency (IDF)** – IDF for a term is defined as logarithm of ratio of total documents available in the corpus and number of documents containing the term T.

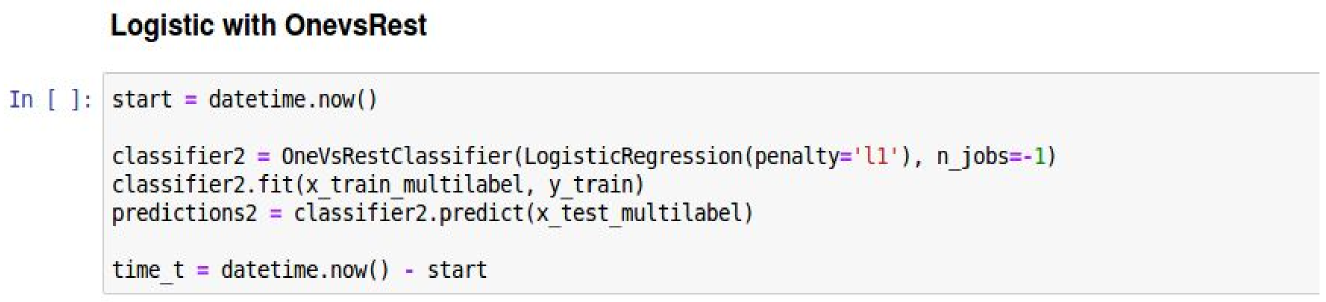


```
In [26]: vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True, norm="l2", \
                                      tokenizer = lambda x: x.split(), sublinear_tf=False, ngram_range=(1,3))
```

Tf-idf vectorizer

4.Model Development

The algorithms we used for training the model are SGDClassifier, Logistic Regression, Support Vector Classifier.



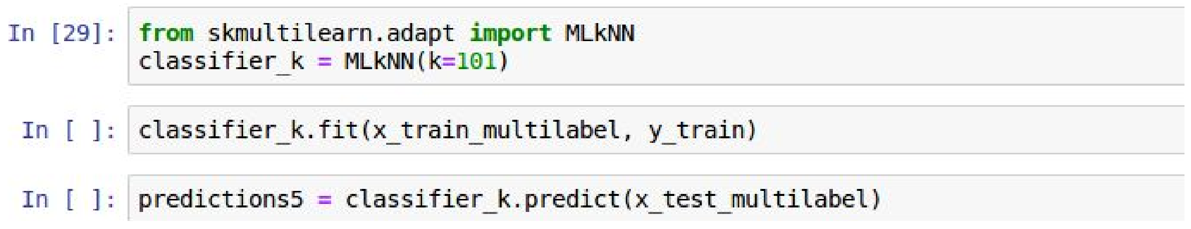
Logistic with OnevsRest

```
In [ ]: start = datetime.now()

        classifier2 = OneVsRestClassifier(LogisticRegression(penalty='l1'), n_jobs=-1)
        classifier2.fit(x_train_multilabel, y_train)
        predictions2 = classifier2.predict(x_test_multilabel)

        time_t = datetime.now() - start
```

Model implementation using Binary Equivalence



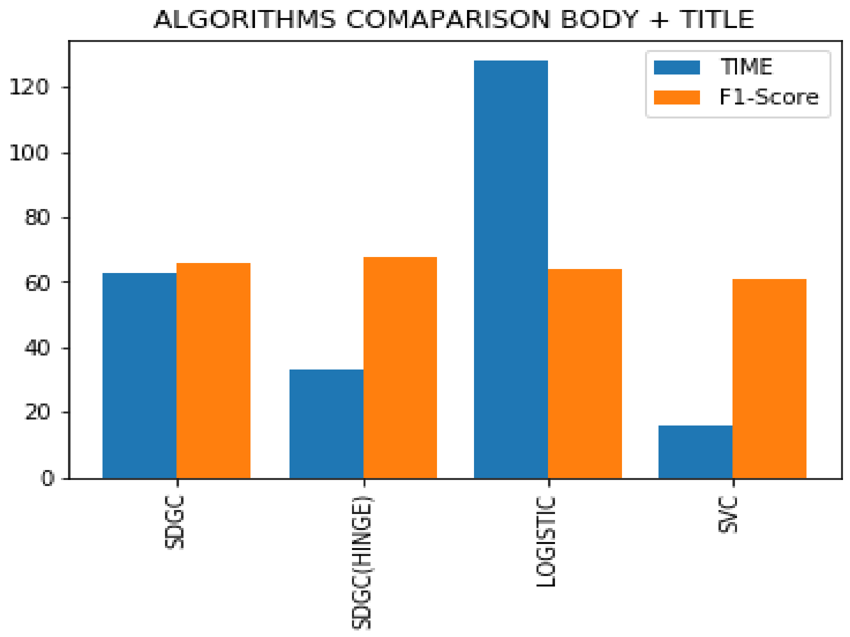
```
In [29]: from skmultilearn.adapt import MLkNN
         classifier_k = MLkNN(k=101)

In [ ]: classifier_k.fit(x_train_multilabel, y_train)

In [ ]: predictions5 = classifier_k.predict(x_test_multilabel)
```

Model Comparison

● Since we are performing a imbalanced multi-label classification, accuracy metric is of lesser relevance than f1-score.



From the above graph it can be inferred that SGDC with hinge in One Vs Rest classifier outperforms all other algorithms in terms of both computational head and f1-score.

Future Work Plan

We have two promising ideas to further improve on our model :

•Using Label encoding can (theoretically,) reduce RAM usage significantly.

•Deploying MultiLabel KNN with optimal hyperparameter values may further improve f1-score.

Data Source

The data required for training and testing our algorithm can be accessed from the link provided below.

https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/

Data size: 3GB (zip-file)

References

[1] https://www.searchtechnologies.com/blog/natural-language-processing-techniques

[2]https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72