Multi-label Classification on EUR-Lex Dataset

Rakesh Devalapally, Praneeth Chedella, Ravi Gunti, Karthik Kadajji, Rabi Kumar

18th June 2019

Abstract

This report details the work done by our team on the task of multi-label classification on EUR-Lex data set. The dataset contains class imbalance problem. The goal is to be able to find suitable methods to recognize the labels on the documents.We considered Multinomial Naive Bayes, Convoultuion Neural Network and Stochastic Gradient Descent (with hinge loss functions, linear SVM) for implementation and evaluation of the EUR-Lex dataset. We evaluated our models using F1-score and hamming loss.

1 Motivation and Problem Statement

We look at ways to tackle multi-label classification on the given dataset. This is different from a multi-class classification problem. A Multiclass classification means a classification task with more than two classes; e.g., classify a set of images of fruits which may be oranges, apples, or pears. Multiclass classification makes the assumption that each sample is assigned to one and only one label: a fruit can be either an apple or a pear but not both at the same time. Multi-Label classification on the other hand assigns to each sample, a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A text might be about any of religion, politics, finance or education at the same time or none of these.

The goal of this project is to find one or more suitable methods to automatically recognize the multiple labels on the documents. The data is high dimensional and there are potentially several labels that are assigned to each document which makes it difficult. Considering options such as features selection, handling the size of dataset processed at a time so as to not to run into memory errors is important. Also, since it is a Multi-Label classification problem, measures that are used for classic classification problems like accuracy, precision and recall cannot be used to evaluate the performance.

2 Data Set:

The Data set is taken from the repository of TU-Darmstadt which contains the EUR-Lex data that is gathered from the EUR-Lex/CELEX (Communitatis Europeae Lex) Site which is publicly accessible repository for European Union law texts. EUR-Lex dataset is a collection of the documents of European Union laws and this data set contains documents such as treaties, legislation, proposals that are indexed and labeled which makes it ideal data set for performing different Multi-Label classification techniques.

2.1 Dataset Description:

The documents have been labelled in 3 ways: the EUROVOC descriptors, Directory Codes and Subject maters. We considered EUROVOC descriptors as it contains more than 4000 labels.

- Number of files in the dataset: 19940
- Number of files without text: 337
- Number of files without english text: 189

- Number of files / instances considered: 19414
- Number of labels: 4071

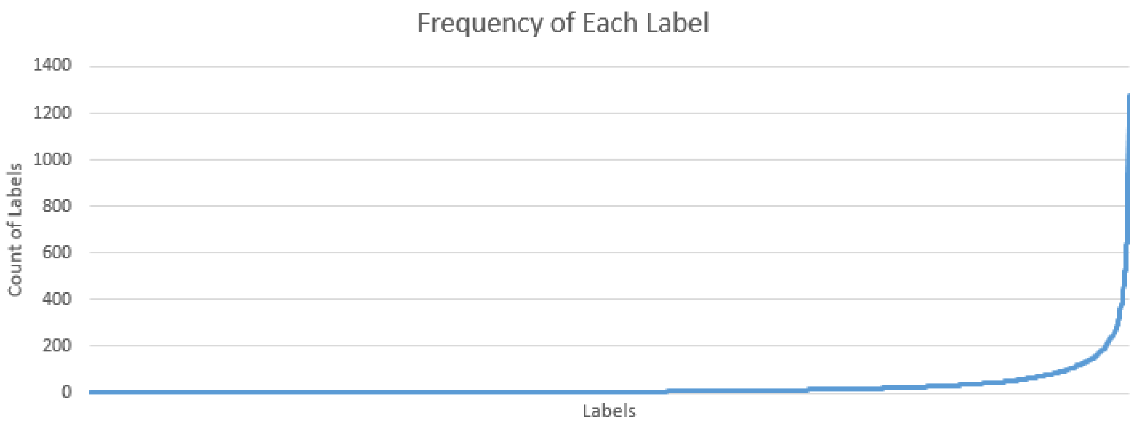
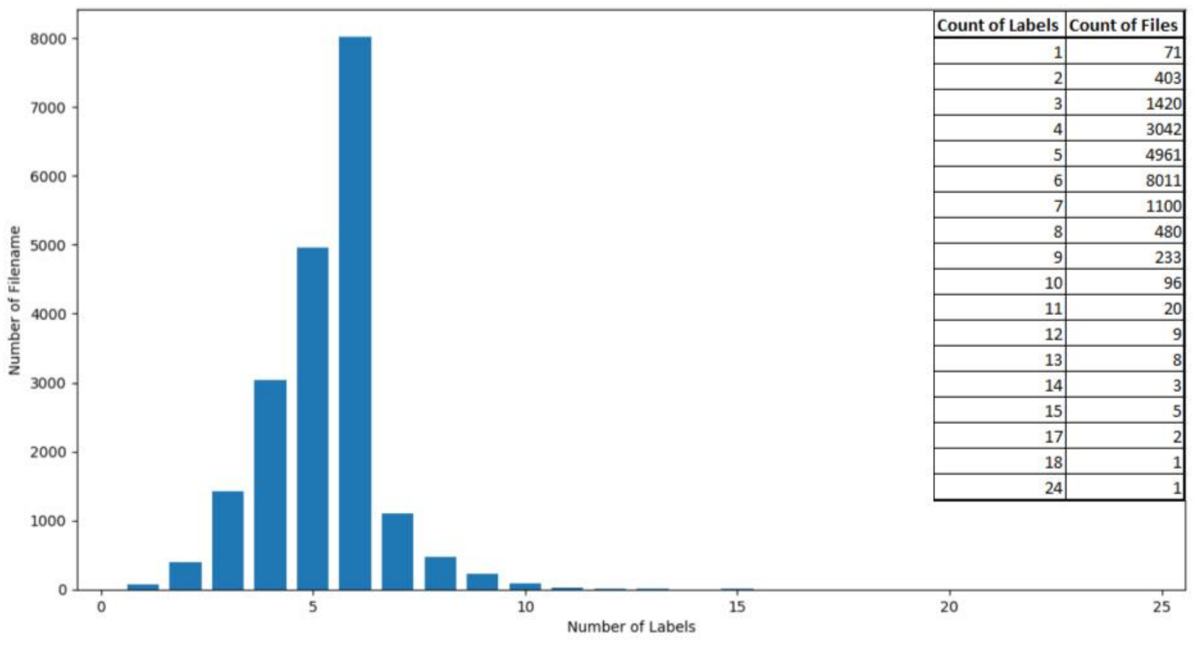


Figure 1: Plot showing number of files each label is assigned to

Figure 1 does not clearly show how the labels are distributed between the files. Figure 2 gives a better image of the number of files that have 1, 2, 3..n labels.



Count of Labels	Count of Files
1	71
2	403
3	1420
4	3042
5	4961
6	8011
7	1100
8	480
9	233
10	96
11	20
12	9
13	8
14	3
15	5
17	2
18	1
24	1

Figure 2: Plot showing number of labels each file is assigned

From the above, we can see that 71 files have only 1 label, 403 files have 2 labels and so on. The highest number of labels a file has is 24. Figure 3 shows the number of times a label appears in the dataset. So, it can be inferred that 741 labels appear only once, 445 labels appear twice, and so on, in the dataset.

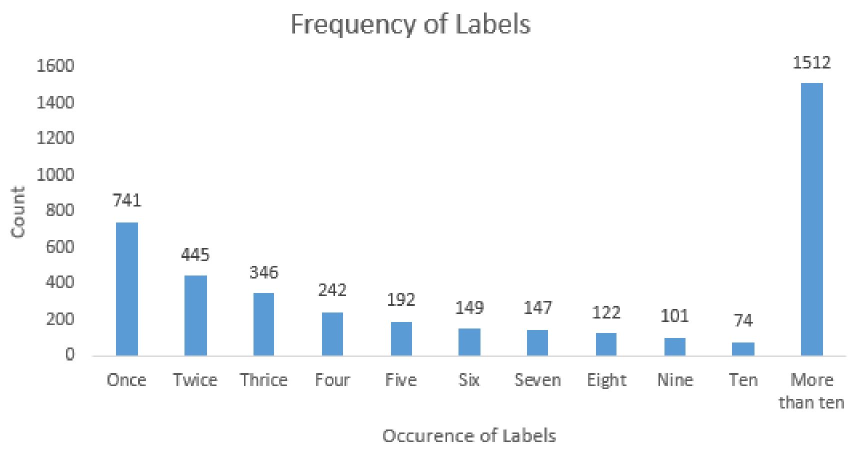


Figure 3

3 Concept

3.1 Data Preprocessing and feature extraction:

We cleansed the data by performing stop word removal, tokenization, stemming and lemmatization on the text of EUR-Lex documents. For this we used *NLTK English stop word* list to remove stop words. *Porter Stemmer* algorithm is used to stem the words further.

* An approach considered - We checked the number of labels that were assigned to only one instance and found out that there are 741 of them. We brought in a synthetic label to group all the instances into one label. However, after checking the correlation matrix, it was clear that labels were hierarchical.

* Label preprocessing - We checked the correlation of labels with one another. The label that has the lower number of instances, in a label-pair which had a correlation greater than 0.9, was dropped.

* Feature Extraction: We used CountVectorizer to get the term frequency with these hyper parameters *max_ df = 0.9 and min_ df = 0.001*. This means, we dropped those terms with frequency less than .001 and greater than 0.9. After this, we used the TfidfTransformer to get the tf-idf vectors. At this stage, we have 12000 features in a matrix and this matrix is sparse.

To tackle the curse of dimensionality of 12000+ features, we used Singular Value Decomposition (LSI / LSA) with top 10000 features. Despite the labels having high dimensionality / sparsely populated, we didn't apply SVD because the interpretability of the labels is lost.

4 Implementation

Data is present in HTML documents and with the help of the package *BeautifulSoup* we scraped the File Name, EUROVOC Descriptors and text from each file. We extracted the text and labels into separate files and then later merged them using pandas frames. The preprocessing and feature extraction is done on this merged data frame using *scikit* library and the models we considered are also part of *scikit* library. We were able to run the whole code on *Google Colab*. We also could run it on a local machine with 16GB of RAM.

4.1 Algorithms we considered:

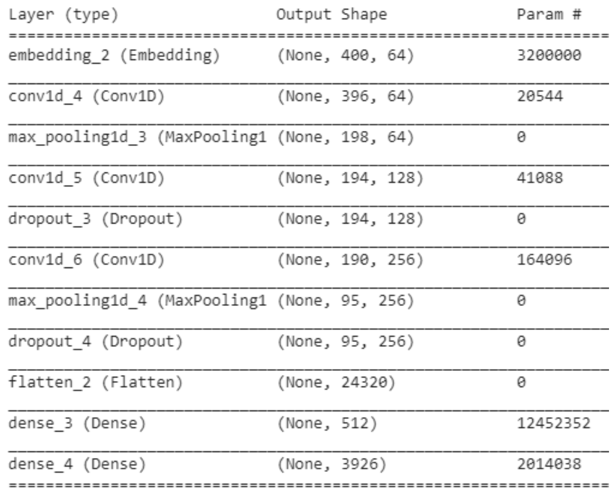
- **Linear SVM** - We implemented Support Vector Machine in *OnevsRestClassifier* setting. Due to the high dimensionality of features and large number of labels, we used Stochastic Gradient Descent (*sklearn.linear_ model.SGDClassifier*), since this algorithm has the ability to easily handle 100,000 features and training examples. Some of the hyper parameters we considered were :
  - loss function: hinge loss
  - learning rate: optimal *((1.0 / alpha * (t + t0))*

  - class_weights: balanced / unbalanced. We considered both.
  - Penalty: L2

  From Figure 2 and 3, it can be noticed that there exists a class imbalance issue in the data set. To address this, we tried oversampling the minority class labels. Since this was a multi-labeled data set the approach wasn't straight forward since there are many combinations/sets of class labels to be considered. Hence we inadvertently were struck with the memory issue. Also under-sampling the majority class would have lead to loss of features. Hence we decided to weigh the class labels differently as in a "balanced" setting, the weights of the classes are adjusted inversely proportional to class frequencies, whereas in "unbalanced" all the class labels are given equal weights.

- **Multinomial Naive Bayes**: We also ran Multinomial Naive Bayes OnevsRest Classifier. Train to test data ratio was 80:20.

- **Convolution Neural Network** The architecture from fig 4, was used for classifying the datasets. We vectorized the entire corpus by changing text to sequence of numbers. The frequent 50000 words were selected for this. We built a Embedding layer where the dense embedding is of 400 dimensions. Drop out layers with 50% was added to inhibit 50 percent of neurons randomly. Since our dataset was a multilabel dataset we tried both sigmoid and softmax activation fuction in the final layer. The model was trained with the loss function 'binary-crossentropy'.



```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 400, 64)           3200000
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 396, 64)           20544
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 198, 64)           0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 194, 128)          41088
_________________________________________________________________
dropout_3 (Dropout)          (None, 194, 128)          0
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 190, 256)          164096
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 95, 256)           0
_________________________________________________________________
dropout_4 (Dropout)          (None, 95, 256)           0
_________________________________________________________________
flatten_2 (Flatten)          (None, 24320)             0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               12452352
_________________________________________________________________
dense_4 (Dense)              (None, 3926)              2014038
=================================================================
```

Figure 4: Architecture of CNN

5 Evaluation

Since classic evaluation metrics like accuracy and precision are not suitable, we considered the following metrics:

- *F1-Score*: It is a weighted average of precision and recall.
  - Micro: metrics are calculated globally by aggregating the total true positives, false negatives and false positives.
  - Macro: metrics are calculated for each label and their unweighted mean are found.

- *Hamming loss*: Evaluates how many times an instance–label pair is misclassified, i.e. a label not belonging to the instance is predicted or a label belonging to the instance is not predicted. The performance is perfect when loss is zero. The smaller the Hamming loss, the better the performance.

5.1 Evaluation Results

The following table gives the results of our evaluation of SGD Classifiers and the parameters we considered:

Alpha	Max Iterations	Hamming Loss	F1 Score Micro	F1 Score Macro	F1 Score Weighted	Jaccard Score
0.001	100 Iterations	0.054521798	0.037745249	0.158469658	0.378756982	0.048726486
0.001	50 Iterations	0.062914564	0.033359423	0.148064615	0.370698606	0.025566791
0.001	5 Iterations	0.11162597	0.020421319	0.077472608	0.301186453	0.014306846
0.001	10 Iterations	0.087416693	0.025279912	0.098190915	0.328340805	0.019156422
0.0001	10 Iterations	0.069867418	0.030666186	0.097082006	0.361080965	0.034785641
0.0001	5 Iterations	0.091983722	0.024242526	0.070818683	0.306249441	0.019903835

Figure 5: SGD Classifier Evaluation Results

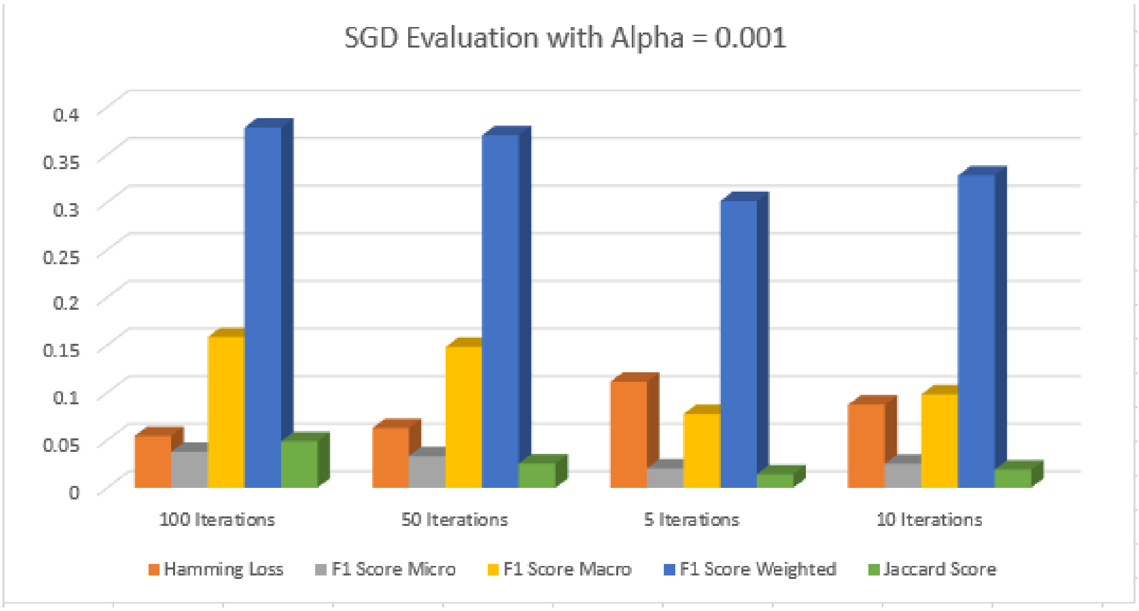


Figure 6: BarGraph showing the evaluation results for different settings of SGD Classifier

Max Iterations	Output Dimensions	Hamming Loss	F1 Score Micro	F1 Score Macro	Jaccard Score
20	30 Dimensions	0.002149167	0.255582337	0.057435979	0.1702351
20	64 Dimensions	0.002098356	0.273182046	0.064479798	0.183144835

Figure 7: CNN Evaluation Results

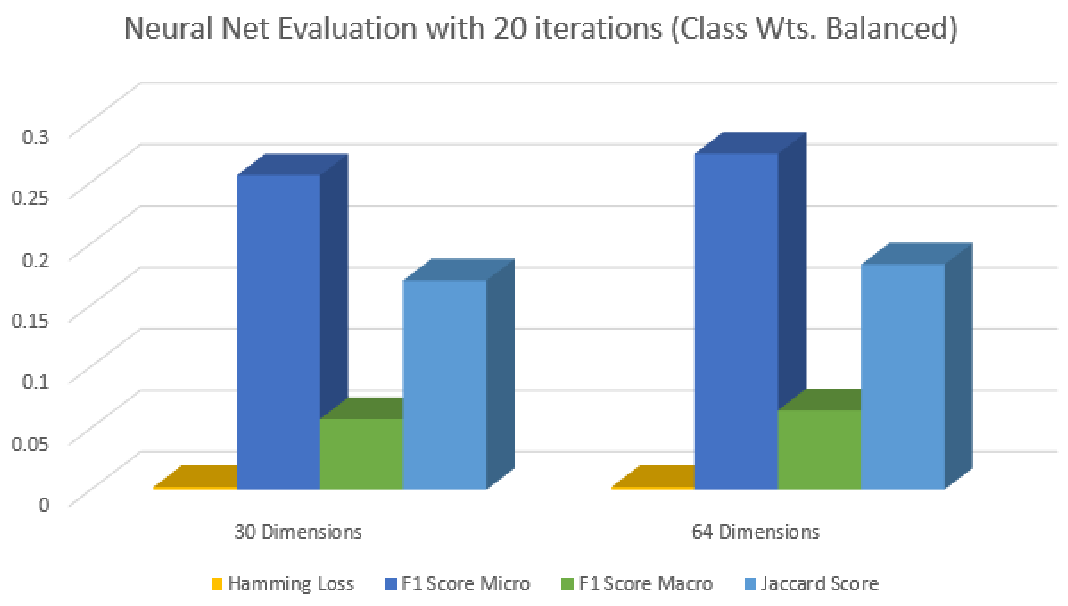


Figure 8: BarGraph showing the evaluation results for two settings of CNN

Without balancing the weights of class labels in the classifier we observed a hamming loss of 0.05797 which suggests that approximately 5.8 percent of the class labels were incorrectly classified. However this measure is highly influenced by the class labels that appears most frequently hence 0.127150 should be considered as correct hamming loss score.

- For Multinomial Naive Bayes, the hamming loss was 0.001588

- *CNN*

  The final layer was sorted with a top K values where K was fixed at 6. Only these values were made to 1 and other Labels-k were made to 0. K was chosen as 6 due to the mean labels of each datapoint being 6.

6 Conclusion

In this project, we attempted to tackle a multi-label classification problem. The dataset is multidimensional and the class labels are sparse which was difficult to deal with. We implemented a One vs Rest Classifier with balanced and unbalanced weights. With the Google colab setting where only 12GB of RAM was available, even calculation of SVD for matrix of 12000 features was long and almost crashed the session. However, if we had more memory at our disposal, we would have preferred to have more than 20,000 features and then reduce the dimensions to 10,000(top 10000 singular values). Another major overhead for our approach was also the correlation matrix of 4000*4000 matrix, which was a long running task of about 3hrs, not to mention the time it took for models to run. Few of the bottlenecks were addressed by saving the required transformed training datasets directly, rather than compiling every time.

With-regard to CNN,there should have been a better way of choosing K, rather than just a fixed number because dataset had a range of labels from 1-24 labels for each document/instance. Hence, a mechanism should be developed to decide K dynamically for each predictions. since most of the labels for an instance would be 0, this would lead to very low hamming loss, if K is low.

Another approach we would have liked to consider is to use prelearned Embedding layer (GLOVE) rather than learning the weights of the embedding layer using our network.