**1. MOTIVATION AND PROBLEM STATEMENT.**

The motivation of given task comes from the fact, with radically increment in the measure of information on the web, it is winding up significantly more simple to search for and discover issues identified with a specific field or a territory to set up and investigate upon. In any case, effortlessly of discovering issues, abiding into the arrangements has turned out to be very intense. As now we have ideas identified with a specific issue in a thick way, burrowing profound and finding legitimate arrangements tending to those ideas is presently an obligatory advance.

The objective is to take care of an issue which is close by but somewhat not quite the same as traditional classification problem which typically includes binary or multi class classification, the dataset given falls in Multi Label Classification issue. A Multi Label Classification undertaking includes a dataset in which one instance is appointed with multiple (at least two) labels. For instance, a Multi class arrangement undertaking comprises of grouping if an animal appeared in picture is a Cat, a Dog or perhaps some other animal which we need to order, each example of Multi Class Classification will just have one mutually exclusive label. Aside from that Multi Label Classification fundamentally comprises of an undertaking to relegate various labels to an instance like a given movie that can fall into every one of different classifications as action, adventure, drama, comedy, and so forth.

Assigning multiple labels to a particular instance is a significant troublesome task and our point of this task is to discover reasonable techniques to manage this issue appropriately. Hence we need to deal with the certainties that how the given dataset is structured and organized, how the proportions of the labels are balanced, the measurement and the dimensions it grows to and investigate numerous approaches to manage all these with a yield of ideal evaluation techniques, as the traditional strategies which is typically related to binary or multi class classification, tend not to chip away at this issue.

**2. THE DATASET.**

The dataset given for the present undertaking is fetched from the EUR-Lex repository directly provided by TU Darmstadt. The EUR-Lex is a portal which gives free access to the European Union Law information in around 24 unique dialects. The dataset which we are going to utilize is an accumulation of documented archives in regards to European Union Law in a several categorization plans identified with a label related to a document. The labels allocated to the cases are fundamentally various aspects of European Law.

The dataset consists of text documents in *.html* format, each document is classified with multiple labels with three different categorizations in hierarchy as given:

1. EUROVOC Descriptor
2. Subject Matter
3. Directory Code

We opt-ed for EUROVOC descriptor as our classification category because of only reason that it contains more numbers of labels that the rest two classification categories.

Below we list some statistics related to the current dataset.

* Total number of documents: 19,940
* Total number of labels assigned: 1,03,629
* No. of unique labels assigned: 4,054
* Minimum number of labels assigned to a document: 1
* Maximum number of labels assigned to a document: 24
* Number of labels assigned to majority of documents: 6

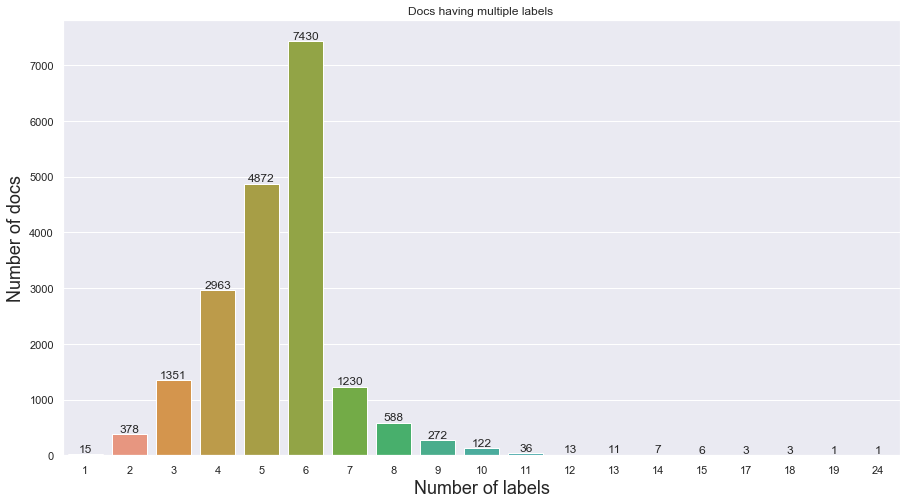


Figure 1: Number of labels distributed over all the documents.

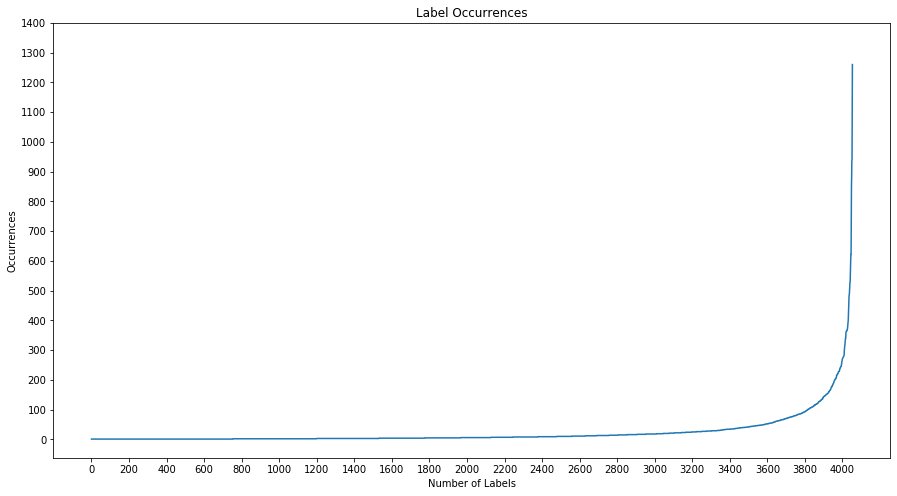


Figure 2: Label occurrences throughout all the documents.

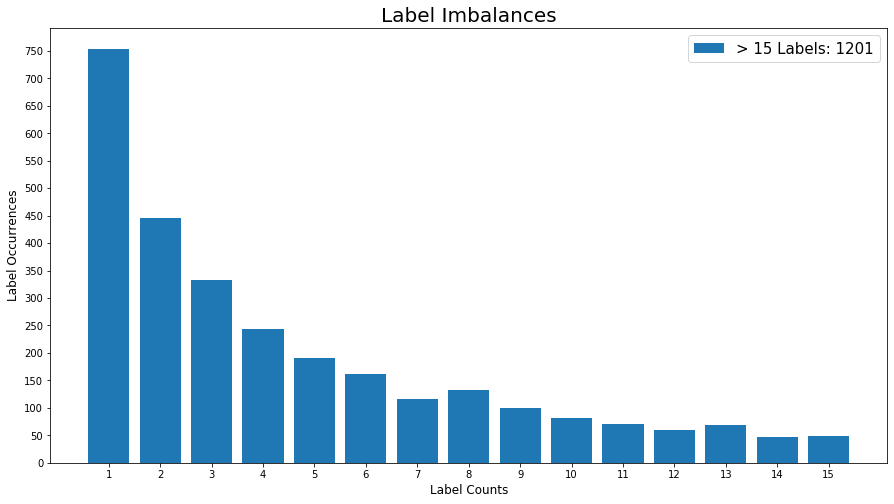


Figure 3: Label imbalances.

**3. CONCEPT**

The aim is to solve the Multi-Label Classification task on the EUR-LEX dataset provided by TU-Darmstadt using two different approaches. Our first approach is using a Neural Network and the other is One vs. Rest Classification using Multinomial Naïve Bayes classifier.

To apply these methods as per our requirement, we first needed to transform the input data into a format that can be used to feed to the corresponding classifier. Therefore we opted to try different approaches. The simplest approach being TF-IDF bag of words representation for the Multinomial Naïve Bayes. Another one is the Bi-Gram bag of words representation which offers some benefits since it catches sequential features like the word order that are basically inherent in textual data. It comes with a price though since it drastically increases the size of the resulting feature vector. For the neural network we use a dictionary to transform every word in the document to an index which we will later use in a look-up table. Since not every document has the same length we apply batch wise padding, always padding to the size of the longest document in the corresponding batch. This representation has the benefit of keeping the complete sequence order and is not increasing the feature vector size. Those feature vectors now need to used for some model.

First we will build a neural network using a LSTM model, this allows us to exploit the sequential nature of text data to a maximum since LSTMs are build for exactly that: sequential data. But we don’t just put the indices into the network but before passing them through an Embedding/Look-Up Layer. This Layer creates a meaningful word embedding feature vector for each word that has much richer information than just the index and hence provides better information for the actual LSTM layer. We then map the output through a *“tanh”* function to a number-of-labels sized output vector which should be equal to the labels. To make that possible we transform the labels to a vector as well. Here we use the one-hot-vector approach or since it is a Multi-Label problem it’s ‘multi-hot-vector’. This means every label is mapped to an index and if a document has that label, so the label vector has a 1 at the corresponding indices and zeros otherwise. Since it is a Multi-Label classification problem we use the Back-Propagation Multi-Label Learning (BPMLL) cost function, (Zhang and Zhou 2006) to address this problem. To take care of the imbalances we additionally penalize the error of not so frequent instances heavier then the error for often occurring instances. For our second model the One vs. Rest using Multinomial Naïve Bayes classifier we first need to augment the dataset to create a binary classification task for each label in the dataset. We then train a binary MNB classifier on each label instance where the positive examples are all documents containing that particular label and the negative examples are the rest. We use a multinomial approach since this is well known to perform for textual data. To infer the labels of an unseen document we run the document through all the classifiers and assign the label to it if the corresponding classifier outputs a 1. To take care of the imbalances we also want to adapt the One vs. Rest approach to a One vs. Some approach. This means we will have a proportional amount of negative examples to positive examples for each MNB classifier. This avoids cases that have 1 positive example and 19100 negative examples which will most likely result in just returning a zero for every possible document.

**4. IMPLEMENTATION**

The dataset is organized in .html format, comprising of its textual contents and labels. We originally scraped the textual contents and the EUROVOC Descriptor labels individually from the documents utilizing tools like *"BeautifulSoup"* accessible in python. At that point we joined the scraped contents and their particular labels to a two dimensional list represented as a solitary instance. This procedure is rehashed until we at long last scraped and consolidated every one of these instances and saved it in a tabular arrangement.

As the majority of the data is in textual format, we opt-ed not to keep any numerical values or any kind of non alphabetic symbols in the feature set just to make the task a little easier for us. We then removed all the english stop-words and then used Porter Stemmer to stem all the individual tokens from the contents by using “*NLTK”* library accessible in python. As doing these will help us in accomplishing improved execution with a marginally less calculation.

As should be obvious from Figure 3. the dataset is profoundly imbalanced, to handle this issue we favoured two situations. In first case we dropped the labels which occurred less in the dataset, and furthermore we utilized oversampling to make copy instances identified with the labels which occurred less and push ahead with both of the strategies to see which one performs all around contrasted with one another.

To transform the preprocessed documents and labels into feature vectors we created dictionaries for all the words occurring in the all the documents and labels. Since the documents are supposed to be English we filtered out all words containing Non-Latin letters. In the dictionaries the keys are the words/labels and the values are the indices that they are assigned to. We also build a Bigram dictionary which is build analog to the word dictionary with the difference that instead of using single words we iterated over all pairs of words that are next to each other. For the Bigram dictionary we threw out all bigrams that only occurred once which reduced the feature vector size from roughly 3M to 1.7M. Which is still enormous compared to the word dictionary size which is around 160K. Furthermore we calculated the IDF for each word and saved it in a dictionary to use it later for the TF-IDF vector representation. For the labels we also calculated a dictionary of penalized values which mapped each label to the inverse of its frequency e.g. if a label occurs in 30 documents the value assigned here would be 1/30. We use this value later for the penalization in the loss function. The actual transformation to the feature vectors is done while the runtime of the model since the feature vectors are the ones that cause trouble when it comes to memory constraints. So far the libraries used are “*pickle”, “re”, “math” and “random”.*

For the implementation of our neural network we used *“tensorflow 1.13.1”* and an implementation of the BP-MLL loss function in tensorflow from this repo:

<https://github.com/vanHavel/bp-mll-tensorflow>

To add the penalization term we had to adapt the BP-MLL function. This function was originally defined as:



where Y is the set of labels that a document x is associated with. being the complementary set to Y and is the prediction of the i-th label based on the instance x.

To implement the penalization, we changed the formula as follows:



Where is the number of occurrences of label i with respect to the complete dataset. By that we reduce the error for labels that occur often and keep the error the same for labels that only occur once in the whole dataset. Thus penalizing rare labels compared to frequent labels. To implement this, we changed the source-code of the bp-mll implementation.

To handle the high dimensional feature vectors with constrained memory resources we build a generator function which only translates the current batch which is fed to the neural network into the feature vector representation. So we only load the whole dataset, where the documents are strings and the labels are a list of strings, into memory and then generate the memory intensive vector representations on demand.

For the Multinomial Naïve Bayes approach we additionally use the libraries “*scikit-learn”****,*** *“random”* and *“itertools”****.*** Again we use a generator approach which only builds the final feature vectors, here TF-IDF vectors, when its needed for training. Next we can choose if we go for the One vs. Rest approach or the One vs. Some approach. The One vs. Rest is the typical text book approach where we just use the documents corresponding to a label as positive samples and all other documents as negative ones. Where as in the One vs. Some approach we count how many positive instances where found for one label and then multiply that number with some scalar. The result is the number of documents we randomly sampled from the dataset that remains when removing the positive instances for that label.

To not run in an out of bound exception we need to be careful that the most occurring label times the scalar value does not exceeds the number of available negative examples. This allows us to assign an equal class distribution to all classifiers where we still manage to represent the skewed ratio of positive to negative samples for each of the labels. We then proceed to train one Multinomial Naïve Bayes classifier for each label in the label set using the dataset created for each label as described before. The trained classifiers are then saved in an array and written to disk. For inference we load the model array and run all ~4000 classifiers on each instance. Since the output is either 0 (negative example) or 1 (positive example) we can concatenate the results of all classifiers to a vector and get the predicted label vector which we can compare to our truth label vector.

**5. EVALUATION**

**6. CONCLUSION**