Multi Label Text Classification of News Articles
Data Analytics Project

By Kushal Gohil, Rushil Goyal and Shivani Dahiya

Introduction

We have unlabeled data in various forms - documents, news articles, and countless other forms of text record. Large volume of data is being generated from several online sources such as www, e-mails, news feeds, digital libraries, electronic health records and databases of organizations**.** This data needs to be categorized to prevent loss of information and to enhance faster search and retrieval of data. Text Classification which is a problem in the area computer sciences or library and information sciences is assigning predefined categories to text records – manually or algorithmically[1].

Text is classified with the help of labels, which may come at no or slight cost or may be too costly[2]. So, learning and labelling instances can get arduous and expensive to acquire.

Generally, a single label is not enough to capture all the information to be labelled per instance. This requires a record to have more than one label. Hence, a document may be classified with multi labels, i.e. assigning multiple outputs to text classification request[3]. The applications of multi labelling are not restricted to text categorization but are extended to cataloguing of – image, audio, medical and bio-informatics.

This project takes into consideration the problem of text classification and multi label assignment to text records of news articles. A typical news site, say NY Times, which we considered for our project, classifies news into numerous labels – Business, Politics, World, Sports, Style, etc. Each label has got sub – labels, for the label World – sub labels will be Asia, Americas, Europe, Middle – East etc., Though we did not consider the hierarchy but it is evident that a news article may fall in more than one category.

The main challenge here is to model label dependencies and to do this efficiently is equally important. So accuracy is generally not the most important measure in multi labelling. Multi labelling requires problem transformation and algorithm adaptation methods. The former is done using Binary Relevance, which is also the approach we used to tackle the problem. This

---

[1] Zhang, M. L., & Zhou, Z. H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition*, *40*(7), 2038-2048.

[2] Druck, G., Settles, B., & McCallum, A. (2009, August). Active learning by labeling features. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1 (pp. 81-90). Association for Computational Linguistics.

[3] Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. Machine learning, 85(3), 333-359.

was done by reducing multi label to conventional classification using the BR approach – Binary Relevance where for every classifier a separate binary classifier is trained. The next step was to adapt this single-label algorithm to create multi-label outputs. This step helped in adapting the data to the algorithm.

The two important metrics which we used in evaluating the performance were Hamming Loss and 0/1 Loss which are relevant to multi – labelling.

Motivation and Background

In recent years, multi label classification and text classification are receiving increasing attention, not because they are practically relevant but because of its theoretically interesting point of view. Multi label text classification is ubiquitous in everyday life. For an email it may be a work message having images or strategies. So it may be categorized as Work, Strategy etc. A news article may be categorized as Business, Politics, Economy etc. Any form of text, image, record (text, audio or video) can be described with multiple labels.

A proper algorithm in this domain is strongly recommended as it would not only allow categorizing but also would help in routing, filtering and searching for relevant information. Machine learning algorithms with good learning and categorization capabilities are now available after years of research on improvising text categorization. But most of them are not considering the problem of multi-labelling. The problem now at hand can be handled by the usual learning methods, converting the data to conventional form, but the results will not be optimal as the results are to be predicted after learning all the labels which cannot be done without exploiting their interdependencies and correlations. Therefore, simple transformations and standard classification might not be applied to this problem.

Conventionally, in classification an assumption is made that an instance belongs to one class. Contrary to this, in multi label classification an instance can be associated with multiple labels[4]. Although, it is important to understand that multi label classification is different from multi class problem.

Colossal amount of data is being generated from various online sources such as www, e-mails, news feeds, digital libraries, electronic health records and databases of organizations[5]. Statistical learning algorithms and a variety of open source software with extensions to in-built

[4] Cheng, W., & Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, *76*(2-3), 211-225.

[5] Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine learning*, *39*(2-3), 103-134.

multi labelling capacity have made this task easier as these algorithms can be successfully trained to classify unlabeled documents.
Multi label text classification would allow news articles and web pages[6] to be automatically catalogued with the classifiers being able to automatically learn the interests of their users and perform variety of other tasks.

To understand the problem of multi labelling and using it in the domain of text classification, it was important for us to first understand

1. The kind of data, i.e. the format in which data should be available before the analysis
2. The process of extracting features from the data set
3. The choice of algorithm
4. The metrics to use for evaluating the performance

For the first two points, we built a web crawler to get the latest newsfeed from NY Times and then cleaned and processed it. But for the latter, we used MEKA, multiple label extension of WEKA, and used it for our understanding on a pre-cleaned and pre-processed data set – Enron.ARFF which was the data used by UC Berkeley for their research. This data was collected from Enron and had over 1500 emails spread across 53 categories. After understanding that which algorithm works best on the basis of the metrics identified from MEKA, we tested the classifier we made on the dataset scraped from different sections of NY Times official website. The details for the data procurement and classification phase are explained below. The results are explained in details in the upcoming sections.

Literature Review

To gain a better insight into the subject and the expanse of the project a literature review was conducted. The literature review was done with the help of Google Scholar and Rutgers Online Library using the keywords "review on Multi Label learning", "Study on Multi-label classification", between the range of 2010 and 2015. The following three papers were considered for review as they gave a good understanding of the multi label text classification problem at hand in various scenarios combined in with various approached- *Combining instance-based learning and logistic regression for multi label classification*; *Multi label Classification with Meta-level Features* and *Multi-Label Classification on Tree- and DAG-Structured Hierarchies*.

The first study Cheng *et al.,* 2009 talked about an approach to multi labelling after unifying the instance-based learning and logistic regression[7]. This approach was designed to overcome the

[6] McCallum, A., & Nigam, K. (1998, July). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization* (Vol. 752, pp. 41-48).

limitations of some of the existing approaches and at the same time to highlight the interdependencies among the labels in a better way. The study collected a large amount of datasets from five different domains like – music, vision, biology, multimedia and text to show the performance of their approach. The datasets were either available online or were manually procured. They compared their results with the baseline models and MLKNN which is the state of art instance based learning algorithm for the problem of multi labeling. The following measures were computed to evaluate the performance – hamming loss, one error, coverage, rank loss and average precision. A 10 – fold cross validation was done. To compare the results statistically they also performed the Friedman and Nemenyi tests. The results were good as the method propped in this study outperforms all the models, regardless of the evaluation metric.

The second study Gopal and Yang, 2010 proposed a generic framework that allowed automatic transformations of the data representations (bag of words, tokens per instance) which are conventional into features which are meta-level with the use of SVM-MAP (Mean Averaged Precision)[8]. To evaluate their proposed model they compared their results with MLKNN, Rank – SVM, IBLR and Binary SVM. To provide empirical evidence for the strength of their study they used p – values at the 1% or smaller levels and other significant statistical tests. The study used the following five datasets - *Reuters-21578, Citeseer*, *yeast*, *scene* and *emotion*. They used the following standard metrics for evaluation – Mean Averaged Precision, Ranking Loss, Macro and Micro Averaged F1. The parameters were tuned in a way that the MAP value will be optimized. The tuning was done with the help of five – fold cross validation. Testing the five datasets for four metrics it was found out that SVM – MAP was able to outperform the other algorithms in 18 of the 20 cases.

In the third study Bi and James, 2011, focused on multi labels using the tree-structure hierarchies and directed acyclic graphs (DAG) for nodes with multiple parents[9]. They proposed a problem transformation approach using both the hierarchical structures thus making it more flexible and compatible with all the learners. Their approach used the KDE, Kernel Dependency Estimation framework and also preserving the hierarchy of the label structure. Every label should fall into one of the three categories – if a node is labelled positive, then its parent must be labelled positive; if a node is positive then all its parents must be positive, if a note is labelled positive then at least one of its label must be positive. Experiments were performed on many news datasets. Precision and Recall were used as the performance measures. They

---

[7] Cheng, W., & Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, *76*(2-3), 211-225.

[8] Gopal, S., & Yang, Y. (2010, July). Multilabel classification with meta-level features. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (pp. 315-322). ACM.

[9] Bi, W., & Kwok, J. T. (2011). Multi-label classification on tree-and dag-structured hierarchies. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 17-24).

reported their results in comparison to CLUS – HMC, which is the state-of-the-art algorithm for multi-label classification on DAG-structured hierarchies. The study showed that except for high recalls the proposed method was doing consistently better than the latter method. The AUPRC values also showed the same results.

Approach

Data Procurement:

Procurement and Cleaning Tool:

In the data procurement phase, we proceeded by creating a web-crawler using the Requests and BeautifulSoup libraries in Python programming language. There were a lot of challenges that arose while trying to scrape the web, like JavaScript redirects, advertisements, garbage content, scripted tags, etc. Our objective for this project was to extract valid full-length news articles (more than 120 words), something not possible using RSS feeds or the like. The crawler focused on extracting news articles while minimizing the amount of advertisement taken from the entire HTML chunk returned by the website. The extracted article was filtered to filter out any tags/scripts present in the text content. This cleaned text is then stored in a text file.

Data Source:

The data source that we used for this project was New York Times website, we scraped the NY Times website to get the latest news articles. In order to make the data set representative of the actual news data available on the web, we needed to get news articles from different segments like, Business, Sports, Science, Politics, Health, Real Estate, Food, Technology, Travel, World and arts. We scraped the NYT-Segment specific website in order to obtain the data. (Figure 1)

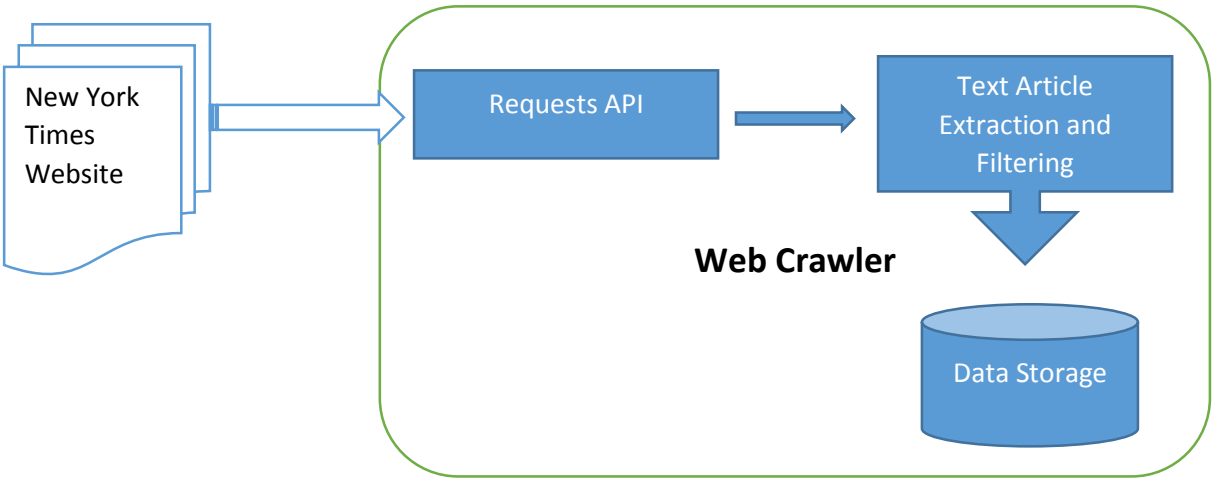


Figure 1: Diagrammatic representation of model

Labeling:

The extracted data was not labeled and was stored in different files, hence, we came up with a tool to help label and organize the data. We created out own custom tool using C# - Winforms to cater to our problem. The tool enabled us to read the text from the article and add labels to it which was stored in a CSV format. This helped us streamline the process of data labeling and provided us with a single CSV file of all the articles and their labels. (Figure 2)

Label Selection and Label Cleaning:

The labels we selected were basically from the different sections on New York Times website, we did not create our custom labels, and we did this in order to keep the labeling procedure simple and consistent. Despite that we did end up with multiple human-errors, we needed to clean this up. We used Microsoft Excel to clean up the inconsistencies in the labeling of the data and took care of the missing labels as well. This gave us a clean set of labeled data, but not necessarily reliable. In order to make the resulting labeled datasets reliable we made sure each article was labeled by two different individuals increasing the reliability of the labels. Out of 6,431 articles scraped we labelled 890 articles.

Classification:

Building the Classifier:

The classifier was built using the famous Open-Source Scikit library in the Python programming language. In order to create a multi-label classifier we selected the One-Vs-Rest classification approach commonly known as BR classifier where each label had an individual classifier, in our case we selected to use Support Vector Machine for these individual classifiers.

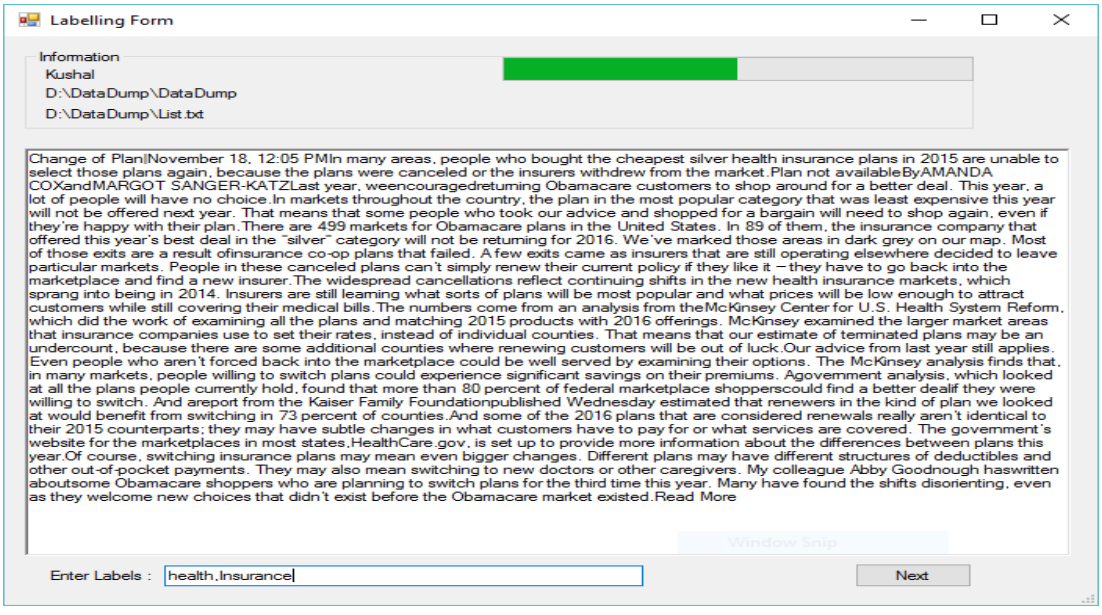


Figure 2: Data Labelling Tool

Text Extraction and Classification:

The text documents were fed into a Pipeline, a concept introduced by the Scikit library where on a given input a sequence of operations are carried out in a sequence where the result of the previous operation is fed as an input to the next operation in the pipeline. The pipeline we constructed had the operations of text count vectorizing (converting text input into a matrix based on the number of occurrence of the words), tf-idf transformation (normalizing the previous matrix) and finally the classification (SVM classifier). (Figure 3)

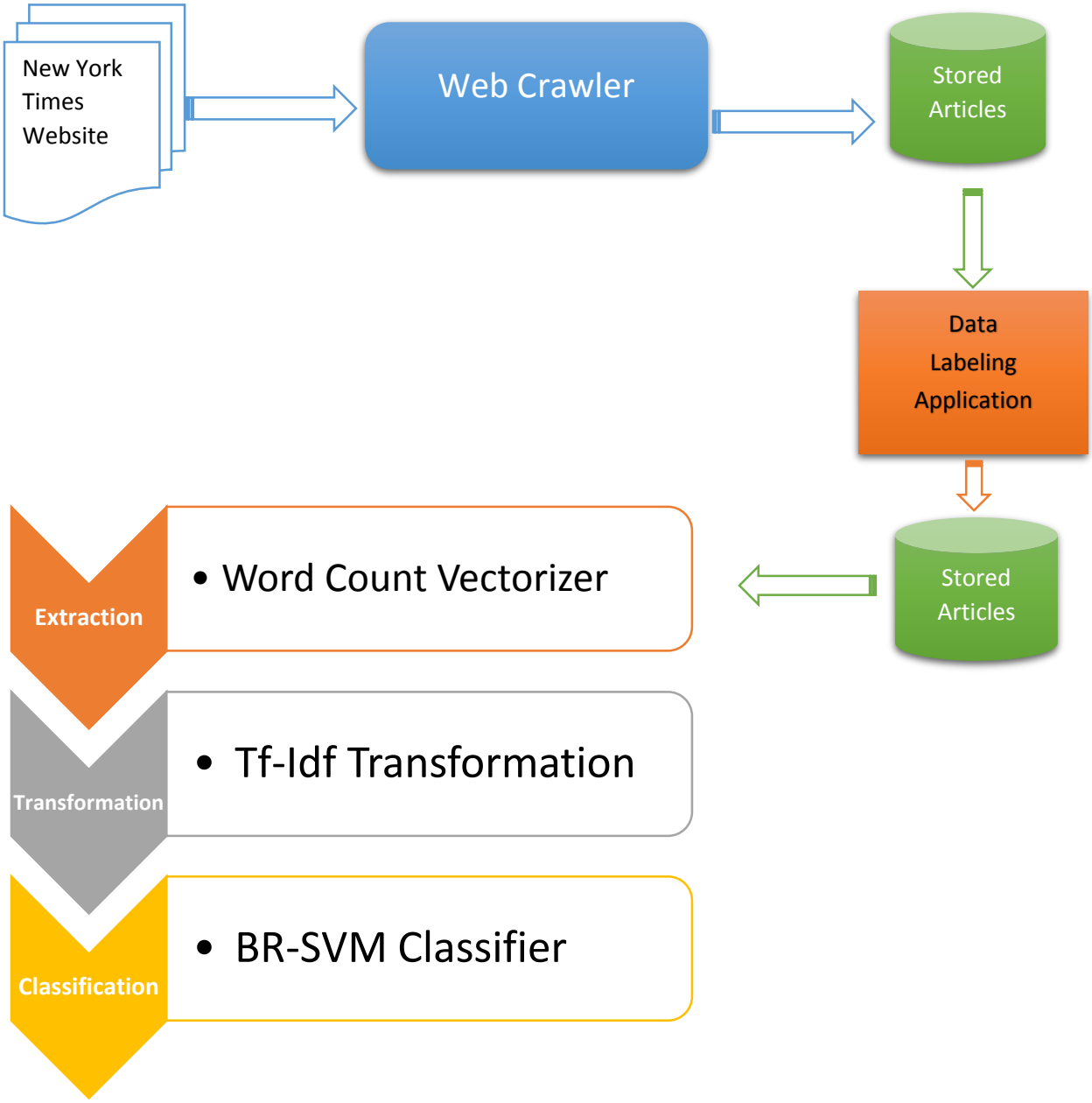


Figure 3: Text Extraction – Transformation – Class

Algorithms:
We used the MEKA tool to identify and test different algorithms for multi-label classification. From our experiments we decided to go with a Binary classifier with the SVM algorithm rather than the J48 or Naive Bayes mainly because of its accuracy and overall performance. So, for our implementation using the SCIKIT toolkit, we used the SVM algorithm in the OneVsRest type of classifier.

Results

In order to see which algorithm performs relatively best, we ran a multi-label text classification problem in MEKA which is an extension of WEKA. MEKA provides an open source implementation of different machine learning algorithms for multi-label learning and evaluation. We used the Enron data which was available online. It was already complied in the ARFF format. Word features were extracted from this dataset using String-To-Word-Vector filter. The data is provided by UC Berkeley Email Analysis project.

MEKA Results:

Representation of a multi-label dataset:

Here, the attributes Y1, Y2, Y3 represents different labels (multi-classification). X2, X3, X4 represents the features corresponding to the dataset (bag of words: generated using feature extraction process) (Figure 6)

Table : Multi-label $Y_1, \ldots, Y_L \in 2^L$

X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4
1	0.1	3	1	0	0	1	1	0
0	0.9	1	0	1	1	0	0	0
0	0.0	1	1	0	0	1	0	0
1	0.8	2	0	1	1	0	0	1
1	0.0	2	0	1	0	0	0	1
0	0.0	3	1	1	?	?	?	?

Figure 4: Example: (binary bag of words)

Dataset: Enron.arff – This dataset used by UC Berkeley for their research. This data was collected from Enron and had over 1500 emails spread across 53 categories. After understanding that which algorithm works best on the basis of the metrics identified from MEKA, we tested the classifier we made on the dataset scraped from different sections of NY Times official website. The details for the data procurement and classification phase are explained below. The results are explained in details in the upcoming sections.

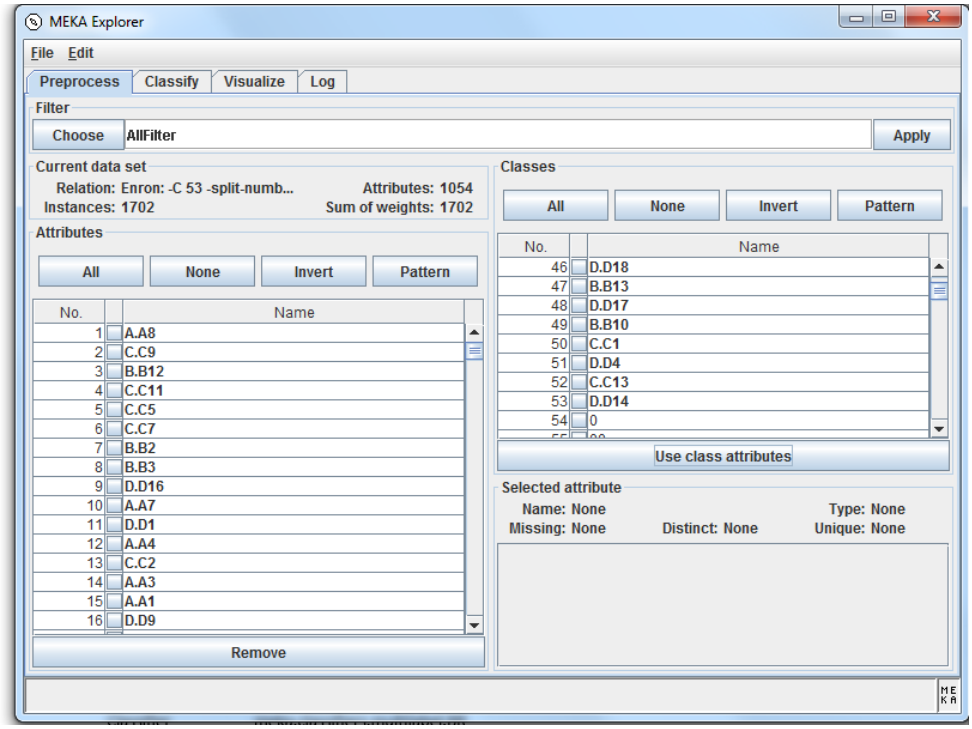
A multi-label dataset with L labels (indexed at the front):

```
@relation  Example_Dataset: -L 3

@attribute Y1 {0,1}
@attribute Y2 {0,1}
@attribute Y3 {0,1}
@attribute X1 {A,B,C}
@attribute X2 {0,1}
@attribute X3 numeric
@attribute X4 numeric

@data
1,0,1,B,1,0.3,0.1
0,1,1,C,0,0.8,0.5
...
```

Feeding the Data:



Implementation of different algorithms using Binary Relevance

Binary Relevance is a popular label-based approach. It is also known as ***One-vs-all*** approach. A multi-label classification is divided into different single-label problem. So, an ensemble of single-label classifiers is trained; one for each class. The target value of each label for a specific instance is its 'membership' or 'non-membership' in that class. (Figure 5)

Binary Relevance (BR)

X	Y_1
$x^{(1)}$	0
$x^{(2)}$	1
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	0

X	Y_2
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	1
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_3
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_4
$x^{(1)}$	0
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	1

Prediction: $\hat{\mathbf{y}} = [h_1(\tilde{\mathbf{x}}), \ldots, h_L(\tilde{\mathbf{x}})]$

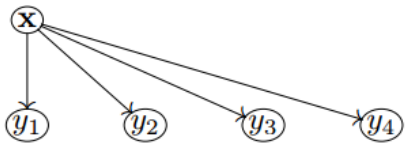
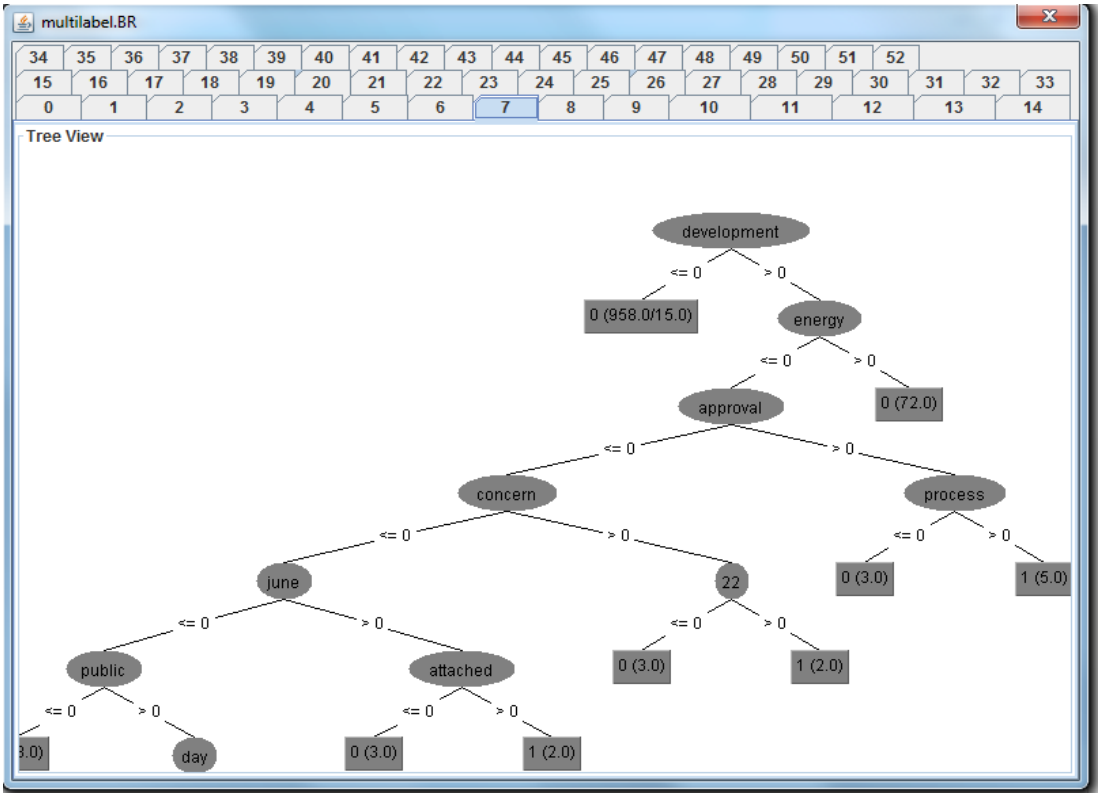


Figure 5: Binary Relevance

Output:

1. **Binary Relevance: J48 Decision Tree:**
   For a particular label 7, the tree is as follows:



Predictive Performance

Accuracy	0.388
Hamming loss	0.06
Zero-One loss	0.969
Avg. precision	0.113
F1 (micro averaged)	0.533
F1 (macro averaged by example)	0.519
F1 (macro averaged by label)	0.157

2. BR: Naïve-Bayes

Predictive Performance

Accuracy	0.205
Hamming loss	0.084
ZeroOne loss	0.991
Avg precision	0.101
F1 (micro averaged)	0.339
F1 (macro averaged by example)	0.301
F1 (macro averaged by label)	0.144

3. BR- Support Vector Machine

Predictive Performance

Accuracy	0.397
Hamming loss	0.06
ZeroOne loss	0.886
Avg precision	0.074
F1 (micro averaged)	0.513
F1 (macro averaged by example)	0.51
F1 (macro averaged by label)	0.218

Based on our literature review and the performance of algorithms in MEKA, we decided to use Support Vector machine for our text classification problem. Its power and suitability for Text Classification is, that it can handle large number of features. For most other classification algorithms a feature selection procedure is done to reduce the number of features. It attempts to find the surface σ_i from all the surfaces from $|T|$-dimensional space, that separates the positive from the negative examples by the widest possible margin. The best decision surface is determined by only a small set of training examples, called the support vectors.

Evaluation Measures:

In order to get an unbiased estimate of the performance of our classifier, it was applied on test set. It is considered to be important in order to avoid overfitting.

We applied different sampling methods to select training and test dataset.

First the classic holdout method was applied in which the data is split into two disjoint sets. We varied the proportion of training and testing set in order to see what works better for us. The limitation to this method was that the classifier was too dependent on the composition of training and test data.

We also applied random subsampling which basically aims to improve the estimation of the classifiers performance by taking the average. This also however suffered from the same limitations as Holdout method.

We have used the following evaluation measures for our study. We noticed that the evaluation measures for Single Classification are usually different than for multi-classification. A logical explanation for that is that in the case of multi-label classification, it's better to predict two of the three labels correctly rather than predicting no labels at all.

Confusion Matrix: In the context of multi-label text classification also, two types of errors are made by the classifier model.

Type 1 Error: When an actual positive example is misclassified, its a False Negative.

Type 2 Error: When an actual negative example is misclassified, its a False Positive.

Class c_i		Predicted	
		positive (c_i)	negative ($\bar{c}_i$)
Actual	positive (c_i)	True Positives (TPs)	False Negatives (FNs)
	negative ($\bar{c}_i$)	False Positives (FPs)	True Negatives (TNs)

1. **Accuracy:** Accuracy is basically the proportion of all the instances predicted correctly including positive and negative. D is the total number of observations.

   TPs+TNs / |D|

Since there are lots of classes who are not as dominant as some other classes, they will contain a lot of negative instances. Accuracy is not a good measure for small classes because always saying no will also be counted as a good accuracy criterion. TN's are a part of Accuracy mathematically. So, always saying no is a strategy which defeats the purpose of building a

classifier. The always “no” classifier will also have 99% accuracy for a class with relative frequency of 1%.

2. **Precision:**

Precision basically measures the estimated probability that a document which is predicted as positive is correctly classified.

TPs/ TPs + FPs

3. **Recall:**

Recall is also called as sensitivity and it expresses the estimated probability that a document whose actual value is positive is correctly predicted.

TPs/ TPs + FNs as shown in Figure 6.

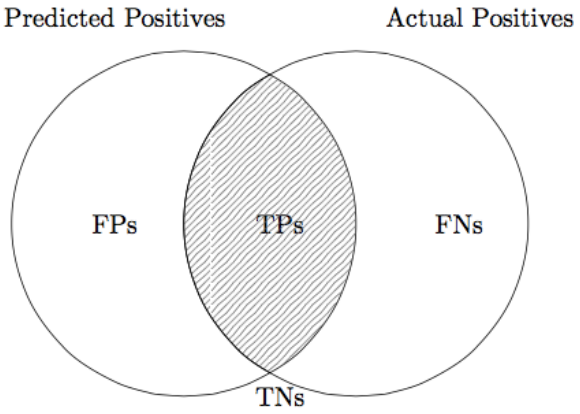


Figure 6: Recall Measure

4. **F-beta Measure:** In multi-label classification, neither precision nor recall can effectively measure the performance of a classifier. In case of a single classification, a classification to the wrong category (decreasing precision) would automatically mean that its not classified to the right category (decreasing recall). So, for single classification, precision or recall can be used for measuring the performance of the classifier.

$$F_{\beta} = ((\beta^2 + 1) \cdot Prc \cdot Rcl) / (\beta_2 \cdot (Pcr+Rcl))$$

This measure combines both precision and recall. The beta value helps to decide what proportion of importance should be given to precision or recall based on the type of application.

5. **Micro-Average F measure:** This is an effective performance evaluating measure for multi-label classification problems. The per-document decisions per classes are pooled together. Because the F-beta measure ignores true negatives and its magnitude is mostly determined by the number of true positives, large classes dominate small classes in micro-averaging. We have chosen to use micro-averaging F1 instead of the macro-averaging F1. This is because the main goal is to predict as many labels from as many previously unseen documents as possible.

6. **Macro-Average F measure**: Macro-averaging computes a simple average over classes. It assign equal weights to each class in contradiction to macro-average F measure which gives equal weight to each per-document decision across classes. To get a sense of effectiveness on small classes, macro-averaged results should be given more importance. If we would adopt macro-averaging, then the categories with very few examples are treated of equally importance as the categories with more examples.

7. **Hamming Loss**:

Hamming Loss basically evaluates how many times an example label pair is misclassified. A label not belonging to a particular document is predicted or a label belonging to an example is not predicted. The smaller value it has, the better the performance of the classifier. Hamming Loss can principally be minimized without taking label dependence into account.

8. **Zero-One Loss**: This is different from the hamming loss in the way that it penalizes the prediction sets that do not strictly match true sets in a multi-label context. Hamming loss on the other hand penalize individual labels. In this case the label dependence must be taken into account. It is usually not possible to minimize both at the same time.

Discussion

We adopted different experimental settings to understand the performance of each setting.

The following settings were implemented and their values by our classifier for each of the evaluation metric we considered are demonstrated below.

1. Uncleaned Label: Full text is used for training but the data here is unclean. Since manual labelling was done, the labels were prone to various mistakes.

```
                    Precision   Recall    F1 Score
          avg / total      0.47      0.31      0.35

Number of Labels:183
hamming loss : 0.019225251076
f-beta(beta=0.5 - biased towards Precision) : 0.406795974241
zero-loss:0.894117647059
Accuracy score:0.105882352941
```

2. Cleaned Labels: Full text is used for training the model for each classifier.

```
                    Precision   Recall    F1 - Score
          avg / total      0.75      0.57      0.63

Number of Labels:78
hamming loss : 0.0259831460674
f-beta(beta=0.5 - biased towards Precision) : 0.6874151515
zero-loss:0.719101123596
Accuracy score:0.280898876404
```

3. Cleaned _ 2T= Here the title has been given two times the weight with an assumption that domain experts are mostly able to recognize the labels just based on the title which is what we want to test with our approach

```
                    Precision   Recall    F1-Score
          avg / total      0.75      0.57      0.63

Number of Labels:78
hamming loss : 0.0258075842697
f-beta(beta=0.5 - biased towards Precision) : 0.689453477451
zero-loss:0.719101123596
Accuracy score:0.280898876404
```

4. Cleaned_2T500= In this we have used both the title and the beginning 500 characters of the remaining text with title given double the weightage as compared to the remaining 500 characters of text. Another advantage of selecting a part of the text is reduction in computational complexity.

```
                    Precision   Recall    F1-Score
          avg / total      0.68      0.33      0.42

Number of Labels:78
hamming loss : 0.029940806681
f-beta(beta=0.5 - biased towards Precision) : 0.526155413799
zero-loss:0.887945670628
Accuracy score:0.112054329372
```

We also used different combinations of Training and Testing Samples"

1. Training: 67.7% Testing 33.3%
2. Training: 80% Testing 20%

We have got best results for a combination of random sampling of 80% Training and 20% testing dataset for 2T_500 combination. This supports our results that the title is more important or has more descriptive nature than the remaining text. Higher micro-F values were observed for this experiment setting relative to others.

The reason because of higher title weight leads to better results because news articles have the most discrete and discriminating for more than one labelled category. The news articles that have distinguishable words more in the full text will have less chances of getting correctly categorized.

Conclusion and Future-Work

There were around 900 instances with multi label in the data set which was built manually. This data set was split with different ratios to see which splitting works better.

1. First setting include the data base being split in 600 instances for training and remaining for testing
2. The second time dataset was split in the ratio of 80% and 20% for training and testing
3. The last splitting was similar to the previous one but only with random sampling.

The evaluation metrics were calculated under all the three settings for data prior to cleaning, after cleaning, giving double preference to the titles and using the first 500 characters.

The results are displayed in Figure 7 and 8.

	Training 300 Testing 600			
	Uncleaned Labels	Cleaned Labels	Cleaned 2T	Cleaned 2T 500
hamming loss	0.014	0.030	0.030	0.030
f-beta	0.308	0.524	0.527	0.526
zero-loss	0.920	0.896	0.893	0.888
Accuracy score	0.080	0.104	0.107	0.112

Training 80% Testing 20%			
Uncleaned Labels	Cleaned Labels	Cleaned 2T	Cleaned 2T 500
0.022	0.036	0.035	0.036
0.258	0.546	0.552	0.547
0.823	0.905	0.905	0.910
0.177	0.095	0.095	0.090

Training- 80% Testing 20% (Random Sampling)			
Uncleaned Labels	Cleaned Labels	Cleaned 2T	Cleaned 2T 500
0.019	0.025	0.025	0.027
0.4	0.687	0.68	0.682
0.89	0.719	0.719	0.73
0.1	0.28	0.28	0.27

Figure 7 – Comparison of performance measures in a tabular form.

Clearly, accuracy is not the best measure while focusing on performance of a multi labeler classifier. Hamming Loss and 0/1 loss are the best for the third setting, where the accuracy has also improved. This is because in previous cases the classifier was picking instances *de novo.* Because of this only instances of certain kind were getting picked. But as the classifier picked instances at random, the metrics improved.

The model could have performed better with more data instances. Removal of inconsistencies from the labels greatly improved the performance.

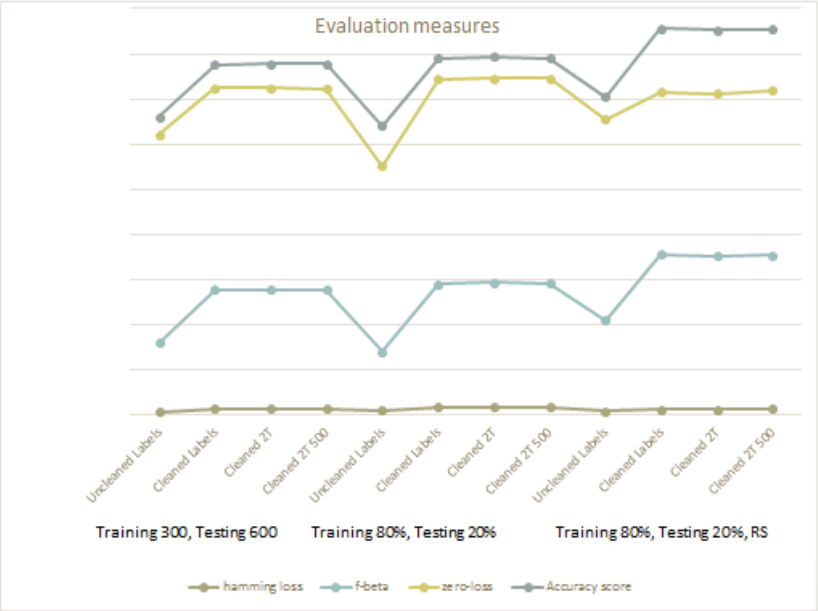


Figure 8 – Comparison of performance measures in a graphical form.

This was the entire representation of the problem of multi labelling in text classification. This problem has a great scope of enhancement. The work can be enhanced by replacing binary relevance with scoring function which will be able to capture the correlation and interdependencies in a better manner.

Though in the domain of text classification, SVMs have shown notable performance, but this performance can be improved after employing better strategies. Hierarchical representation of data and also the use of active learning can help achieve better results.