Project D

Laura Lonardi, Shounak Ghosh, Viktor Hermann

November 13, 2018

Contents

0 Preliminary Work

0.1 Background and Scope of Work

The first part of the course (14.458 Financial Data Science I), focused on building strategies based on past returns and several other financial metrics. However, in today's business world clean data sets like those are rather the exception than the rule and their availability often lags in time. Most of today's signals are embedded in news articles and push notifications. Even though there is no clean data set to be used, the market immediately prices in this information.

Therefore, having a way of capturing this information in a structured manner is of utmost importance. Project D aims to tackle this issue by introducing Natural Language Processing and by subsequently using this technique to utilize (and possibly trade on) the information extrapolated from all available resources. Even though this technique is most useful with web scraping (i.e. breaking news of CNBC, Bloomberg), our analysis focuses on a pre-cleaned data set to first train our model. We will do this by training (and testing) our model on the data set used in the 2004 paper of Lewis et al.[1]

0.2 Data Preprocessing - Data Source

For the text analytics research we will use the RCV1 corpus. Even though in a real case a text cleaning process would be necessary, this data set has already been cleaned. Therefore, we can go to the data analysis part.

Therefore, we read in both the train as well as test data frames. The train data frame consists of 73,697 observations, whereas the test data frame contain 1,266,713 rows. All of these observations belongs to at least one category, however most of them belong to some sub-categories as well. As a consequence, we sliced both our training and test data sets into 'parent' and 'child' to take this phenomenon into account. Please note that we first had to trim the categories columns because there were still some white spaces included. We then have the following data set sizes:

Table 1: Data set sizes

Data Set	# observations	# columns	# duplicate ID
Train Parent	27,087	3	3,938 (14.54%)
Test Parent	459,113	3	68,497 (14.92%)
Train Child	46,610	3	24,197 (51.92%)
Test Child	807,600	3	428,933 (53.11%)

As seen in Table 1, the test data set is of several magnitudes bigger. This we may take into account when making decisions about the reliability of the predictions.

Furthermore, please note that there are duplicate ID's in each data set. These are ID's that belong to more than 1 category (e.g. one article that deals both with economic as well as governmental issues). These duplicates do not cause a problem for the train data sets. However, if not changed, the model will only predict one category and not two. Therefore, if an ID belongs to 2 categories, the model will always be at least 50% wrong for these ID's as it cannot catch both categories.

In our paper we will run two separate approaches to tackle this issue:

1. Create a model which only predicts one outcome and thus we exclude all duplicate ID's. This approach is not problematic in our case as there are enough data points.
2. Create models that can predict multiple categories at the same time and compare those results with the the whole data set (i.e. duplicates included).

[1] Lewis, D. D., Yang, Y., Rose, T. G., Li, F. (2004), *RCV1: A New Benchmark Collection for Text Categorization Research*, Journal of Machine Learning Research 5 (2004) 361-397.

Finally, please note that for both analyses we will subset our data set.[2] For this we will artificially create a training data set that contains all categories. This way we can make sure that the test set will not include any results that could not have been predicted. Reason for that is mainly our computational constraints and the size of the data set (for running times see Table 2).

However, before jumping into the two approaches, let us first examine 6 of the 9 machine learning algorithms more closely. This way we can familiarize ourselves in advance with which model may be helpful for which approach.

[2]Reason for that is mainly that for the tasks we did not use Amazon servers but our local computers that, however, did not have sufficient computational power to deal with the whole data set at the same time.

1 Algorithm Examination

After having cleaned and prepared our data set, we first focus on training the model. For training purposes we will use the **RTextTools** package. This package allows us to choose from 9 machine learning algorithms.[3]

Based on this paper, we will train and evaluate 6 algorithms:

- Support Vector Machine (SVM)
- Stabilised Linear Discriminant Analysis (SLDA)
- Maximum Entropy (MAXENT)
- Generalized Linear Model with Convex Penalties (GLMNET)
- Bootstep Aggregation (BAGGING)
- Neutral Network(NNET)

As it will be shown, each model has a different purpose and will behave differently in other environments.

1.1 Support Vector Machine (SVM)

The basic idea behind the Support Vector Machine, or SVM, is to separate classes using hyperplanes in multi-dimensional space. The optimal separating hyperplane is the one that maximizes the *margin*, or distance, between the closest points in each of the classes. Points that lie on the boundaries nearest the hyperplanes create what are known as the *support vectors*.

In the simplest case, SVM linearly separates two classes of data. However, as data becomes non-linear, SVM projects the data into multi-dimensional space where the data points can become linearly separable using *kernel transformation*. [4] Consequently, given the inherent multi-dimensionality of language, SVM can be a particularly useful machine learning technique for Natural Language Processing.

Jurka's paper indicated that SVM was one of the strongest performing algorithms in terms of precision, F1 scores and recall. Thus, we felt that it was prudent to include this as one of the six algorithms that should be explored while testing our data set. However, without further changes, this model is incapable of detecting two or more possible categories for one specific ID. Therefore, if we want to use this for model 2, we will need to adapt the algorithm to that environment.

1.2 Scaled Linear Discriminant Analysis (SLDA)

Scaled Linear Discriminant Analysis (SLDA) is a subset of the more common Linear Discriminant Analysis (LDA). LDA is one of the most commonly used algorithms for pattern-classification in machine learning. LDA is similar to Principal Component Analysis, but in addition to finding the component axes that maximize the variance of our data it also find the axes that maximize the variance between multiple classes.

Below are the steps needed to perform the analysis:

- Compute the mean vectors for the classes
- Compute the between and within class matrices and find their eigenvalues and eigenvectors

[3]Jurka, T. P., Collingwood, L., Boydstun, A. E., Grossman, E., von Atteveldt, W. (2013), *RTextTools: A Supervised Learning Package for Text Classification*, The R Journal Vol. 5/1, June.

[4]Refer to Meyer, D. (2018), *Support Vector Machines*, The Interface to libsvm in package e1071.

- Choose the bigger eigenvalues and create a matrix with the corresponding eigenvectors
- Use this to reduce the initial subspace into a smaller one

For SLDA, scaling or normalizing the data does not yield different results. Even though both the eigenvectors and matrices will be different, the eigenvealues will be the same and thus resulting in the same final product.

Please note that SLDA is a very accurate approach to NLP, however, the running time may cause a problem (see Table 2).

1.3 Maximum Entropy (MAXENT)

Maximum entropy states that the prior probability distribution, which best represents the current state of knowledge, is the one with largest entropy.[5] Due to the minimum assumptions that the Maximum Entropy classifier makes, MAXENT is often used when we do not have sufficient information about the prior distributions and when it is unsafe to make any such assumptions. Moreover, Maximum Entropy classifier is used when we cannot assume the conditional independence of the features. This is particularly true in Text Classification problems where our features are usually words which obviously are not independent. MAXENT is also designed to accommodate multi-class problems, which makes it even more well-suited for NLP applications.

The goal of MAXENT is to use contextual information of the document in order to categorize it to a given class (positive/neutral/negative, objective/subjective etc). Following the standard bag-of-words framework, let $\{w_1, \ldots ,w_m\}$ be the m words that can appear in a document. Then each document is represented by a sparse array with 1s and 0s that indicate whether a particular word w_i exists or not in the context of the document.[6] Our target is to construct a stochastic model, which accurately represents the behavior of the random process: take as input the contextual information x of a document and produce the output value y.

The estimate takes the following form[7]:

$$P_{ME}(c|d) = \frac{1}{Z(d)} exp(\sum_i \lambda_{i,c} F_{i,c}(d,c)) \tag{1}$$

where Z(d) is a normalization function. $F_{i,c}$ is a feature/class function for feature f_i and class c, defined as follows:

$$F_{i,c}(d,c^{'}) = \begin{cases} 1, n_i(d) > 0, c^{'} = c \\ 0, otherwise \end{cases} \tag{2}$$

1.4 Generalized Linear Model with Convex Penalties (GLMNET)

Glmnet is a package that fits a generalized linear model via penalized maximum likelihood.[8] The regularization path is computed for the lasso or elasticnet penalty at a grid of values for the regularization parameter lambda. It fits linear, logistic and multinomial, poisson, and Cox regression models.

GLMNET solves the following equation:[9]

$$\min_{\beta,\beta_0} \frac{1}{N} \sum_{i=1}^{N} w_i l(y_i,\beta_0 + \beta^T x_i) + \lambda[\frac{(1-\alpha)||\beta||_2^2}{2} + \alpha||\beta||_1] \tag{3}$$

[5]Berger, A. L., Della Pietra, S. A., Della Pietra, V. J. (1996), *A Maximum Entropy Approach to Natural Language Pro cessing*, Computational Linguistics, (22-1).

[6]Pang, B., Lee, L. (2002) *Thumbs up? Sentiment Classification using Machine Learning Techniques*, url: http://www.cs.cornell.edu/home/llee/papers/sentiment.pdf.

[7]ibid.

[8]See https://web.stanford.edu/ hastie/glmnet/glmnet$_alpha.html$.

[9]https://web.stanford.edu/ hastie/glmnet/glmnet$_alpha.html$.

where l(y,) is the negative log-likelihood contribution for observation i.

The GLMNET algorithms use cyclical coordinate descent, which successively optimizes the objective function over each parameter with others fixed, and cycles repeatedly until convergence.

1.5 Bootstrap Aggregation (BAGGING)

Bootstrap aggregation or BAGGING was first suggested by Breiman to stabilize trees.[10] BAGGING takes a standard training set T of size n (in our case e.g. Train Parent of size 27,087 (see Table 1)) and generates m new training sets T_i each of size $n^{'}$. Like in any bootstrap technique, the samples are replaced so that some observations may be repeated. The aggregation part consists of combining these bootstrap samples and averaging the output.

Initially BAGGING was introduced as a method to improve unstable procedures, such as neural networks or regression trees.[11] As Breiman pointed out, "improvement will occur for unstable procedures where a small change in L can result in large changes in ϕ".[12]

Therefore, before having done the analysis we can hypothesize that BAGGING itself may not yield the best results (as shown in the Jurka paper), but it may be helpful for the *Ensemble* [13] *agreement* where multiple algorithms will be used together to stabilize the classification prediction.[14].

1.6 Neutral Network (NNET)

Neural networks is an optimization technique that seeks to minimize the cost (i.e. residuals) between the input and the output. In essence, neural networks take the data set as an input layer and connect it to our wished output layer. Thereby, it adjusts the weights of each neuron (i.e. input layer particles) has to the output layer. These weights will be adjusted in a manner so that the cost function will be minimized.[15] This basically represents the gradient descent of the cost functions. The fine tuning uses backpropagation.

The layer connection can be described as follows:

$$a_j^i = \sigma(\sum_k w_{jk}^i a_k^{i-1} + b_j^i) \tag{4}$$

where σ is the activation function (often Sigmoid or ReLu functions), w_{jk}^i is the weight from the kth neuron in the (i−1)th layer to the jth neuron in the ith layer, b_j^i is the bias of the jth neuron in the ith layer, and a_j^i represents the activation value of the jth neuron in the ith layer.

In Jurka's paper, NNET performed by far the worst among all the algorithms. It will be an interesting experiment to see if this result will be substantiated by our analysis and then to further contrast this phenomenon.

Furthermore, please note that NNET automatically has the built-in feature to predict more than one category for 1 ID. Therefore, it may be a suitable alternative for the model 2 approach.

[10]Breiman, L. (1996), *Bagging predictors*, Machine Learning, 24, 123-140.
[11]ibid.
[12]ibid.
[13]Ensemble coming from the French 'together'.
[14]For further information see Jurka, RTextTools.
[15]For possible cost functions see https://stats.stackexchange.com/questions/154879/a-list-of-cost-functions-used-in-neural-networks-alongside-applications.

2 Model 1: Subsetting Data Set to Exclude Duplicate ID's

2.1 Model Training

Based on Jurka's paper, we felt that it would be useful to test our model utilizing six of the models (described above) to begin with. We ran these algorithms on a training and test set that were both 1,000 in size. Our decision to set the size to 1,000 stemmed from the training size used by Jurka, and we felt that 1,000 was also a reasonable test size given the expanse of our test data set. Running these algorithms, we derived the following results (see Table 2). As shown, there are 5 main figures to consider:

- Running time: A critical feature of this project is recognizing the computational power required to execute each of the machine learning algorithms. Since we do not know who the user is and what computing equipment they have at their disposal, we felt that it was important to incorporate the running time into our decision-making process. In general, SVM, MAXENT and GLMNET are substantially faster than the other ones (see Table 2).
- Precision: Proportion of items that algorithm predicts belongs to a certain class actually belongs to that class.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{5}$$

- Recall: Proportion of items in a class the algorithm correctly assigns to that class.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{6}$$

- F1 Score: Weighted average of recall and precision.

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

- Cross-validation: Used to estimate how accurately a predictive model will perform in practice. We run cross-validation by partitioning the data set into a training set and a validation set and see how our model performs out of sample. To reduce variability, we ran multiple methods of cross-validation and averaged those over the rounds.[16]

Table 2: Data analytics comparison

Algorithm	Running Time	Precision	Recall	F1 Score	Cross-validation
SVM	4 sec	0.76	0.78	0.76	0.87
SLDA	857 sec	0.58	0.60	0.58	0.79
MAXENT	2 sec	0.74	0.74	0.74	0.88
GLMNET	3 sec	0.74	0.77	0.74	0.76
BAGGING	174 sec	0.72	0.74	0.71	0.80
NNET	769 sec	0.45	0.45	0.40	0.55

Table 3: SVM Multi-Class Model

Class	Precision	Recall	F1 Score
CCAT	0.87	0.81	0.89
ECAT	0.80	0.77	0.78
GCAT	0.90	0.88	0.89
MCAT	0.92	0.91	0.91

[16]Similar to the Jurka paper, we used 4 rounds of cross-validation.

As we can see there are substantial differences in the algorithms and their metrics. Especially the running time and F1 score relation is interesting. Interestingly, the higher the running time, the more inprecise the the prediction. Therefore, when making a decision about the model to be used, we should consider several factors in our decision-making process. Given our computational constraints, we decided to use running time, F1 score and cross-validation as parameters to choose. We used F1 score instead of solely relying on recall, precision or accuracy because we seek to balance between precision and recall.

The second table presents data that was gathered through a separately classified SVM model implementation. The SVM in Table 2 is a vanilla SVM run across all four classes, while the SVM results in Table 4 are derived from an SVM model that was run for each class. This latter implementation, which is also known as a "*One vs. Rest*" implementation, is important to utilize because SVM is inherently a binary classifier, and thus, would produce the most accurate results when running on a per-class basis. As shown in Table 2 and Table 4, this is indeed the result. All the scores are more accurate with the for loop. Interestingly, the model is more much more accurate for MCAT than for ECAT which may be due to the fact that MCAT has more specific words that cannot be found in the other categories. This quick demonstration shows as well that before adapting any machine learning algorithm, one needs to first get acquainted with the algorithm itself. Had we not run the for loop, our results had been more prone to error.[17]

2.2 Model Selection

Based on the previous chapter, we conclude that MAXENT and SVM together ("ensemble") might perform best because they have the highest F1 Score as well as cross-validation results, respectively.

Furthermore, the running time for these two algorithms is by far the lowest, thus using them together should not draw down on our computational constraints either.

However, this does not mean that we expect these algorithms to perform as the most effective ones in all situations. In general, it is very difficult to hone in on an algorithm that performs the best for every problem. The varying structures and sizes of data sets often dictate which algorithm, or combination of algorithms, would yield the optimal results. In our case, we believe that the combination of speed and precision of the MAXENT and SVM ensemble will allow it to yield such results.

Table 4: Ensemble performance MAX-SVM

Ensemble	Coverage	Recall
MAX-SVM	0.89	0.82

2.3 Model Tuning

Given that we opted for an ensemble algorithm of MAX and SVM, we have the following parameters[18] in the *train_model* function which can be further fine-tuned:

- method: Type of implementation of SVM.
- cross: If an integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model: the accuracy rate for classification and the Mean Squared Error for regression
- cost: Cost of constraints violation. It is the 'C'-constant of the regularization term in the Lagrange formulation. It is by default 1.
- kernel: Parameter that helps determine the optimal separating hyperplane in multi-dimensional space.

[17]However, please note that in the the further analysis we will only tune the vanilla SVM because it is substantially faster to run and also gives a good sense about the directional changes.
[18]See R documentation.

- l1_regularizer: Adds an L1 penalty equal to the absolute value of the magnitude of coefficients. In other words, it limits the size of the coefficients.
- l2_regularizer: Adds an L2 penalty equal to the square of the magnitude of coefficients.
- use_sgd: Logical that Stochastic Gradient Descent (SGD) estimation should be used. It is by default *False*.
- set_heldout: An integer specifying the number of document you want to hold out to test against to prevent overfitting.

Even though there are 9 parameters that could hypothetically be tuned, we will not alter method and kernel for SVM because according to the literature review, they should not be altered/are already optimal for NLP.[19]

Furthermore, we will first tune the algorithms separately, then choose the best alternatives and put them together for the ensemble analysis.

SVM:

For the SVM, we first change the cost argument and take a look how that one changes our correction metrics.

Table 5: Data tuning vanilla SVM

Algorithm change	Precision	Recall	F1 Score
Cost $= 10^{-1}$	0.75	0.71	0.71
Cost $= 10^{3}$	0.75	0.77	0.76
Cost $= 10^{5}$	0.75	0.77	0.76
Cross = 5	0.76	0.77	0.76
Cross = 10	0.76	0.78	0.76

From Table 10, we see that increasing the cost restraint actually leads to a decline in the model's F-scores. In addition, altering the cross parameter does not yield any significantly different results than the SVM implemenation with its default parameters.

[19] Meyer, D. (2018),Support Vector Machines, The Interface to libsvm in package e1071.

MAXENT:

As suggested in the paper to the MAXENT function by Jurka and Tsuruoka (2015)[20], we scrutinez 3 main scenarios:

1. Set the l1_regularizer parameter to 1.0, leaving l2_regularizer and set_heldout as default.
2. Set the l2_regularizer parameter to 1.0, leaving l1_regularizer and set_heldout as default.
3. Set the set_heldout parameter to an integer hold-out a portion of your data, leaving l1_regularizer and l2_regularizer as default.

Given that we are also using a large number of training samples, we also set use_sgd parameter to TRUE.

Table 6: Data tuning MAXENT

Algorithm change	Precision	Recall	F1 Score
Tsuruoka 1	0.75	0.78	0.76
Tsuruoka 2	0.76	0.77	0.76
Tsuruoka 3	0.74	0.76	0.74
SGD = TRUE	0.72	0.74	0.72

Here, we find that the second scenario presented by Tsuruoka yielded stronger performance than the default MAXENT model. As a result, we then tested the ensemble of default SVM and Tsuruoka 2 MAXENT.

Table 7: Ensemble performance MAX-SVM (New)

Ensemble	Coverage	Recall
MAX-SVM	0.91	0.83

Training Data Preparation:

Regarding the fine tuning of the training data preparation, we can change two main parameters: sparsity level, term frequency metrics.

Regarding sparsity level the closer the number is to 1, the better. In our data preparation we already used 0.998, therefore it does not make much sense to alter this parameter.[21]

Table 8: Training Data Vanilla Tuning

Algorithm change	Precision	Recall	F1 Score
WeightTF			
SVM	0.76	0.78	0.76
MAXENT	0.74	0.76	0.74
WeightTFidf			
SVM	0.73	0.46	0.42
MAXENT	0.78	0.79	0.78

In changing the preparation of our training data, the most interesting result is that we see that the recall and F-score for SVM drop quite substantially when creating the term matrix with the inverse document frequency

[20]Jurka, T. P., Tsuruoka, Y. (2015), *Low-memory Multinomial Logistic Regression with Support for Text Classification*, url: https://cran.r-project.org/web/packages/maxent/maxent.pdf

[21]Please note that we also tested for a sparsity level of 0.60, but the results were substantially worse, therefore we did not include those in our write-up.

(IDF) weighting. This may sound counter intuitive at first sight as WeightTFidf seeks to tackle the issue of "over-weighting"[22] filling words such as *the, a, an.* These words should not possess predictive power for our analysis. However,

2.4 Preliminary Conclusion

In fine-tuning the parameters of the SVM and MAXENT ensemble, we found that there was only one case that created stronger ensemble performance. Overall, we discovered that altering the parameters either yields the same, or lower, performance in the models, which indicates that the learning algorithms in RTextTools are already well-tuned to accommodate NLP.

However, please note that this method of dealing with the problem of duplicates is only viable when there are sufficient data points at our disposal. We had more than 1 million data points and we subset our analysis, thus we did not face this issue in this assignment. Nonetheless, this possible problem always needs to be borne in mind when doing future analyses.

[22]Over-weighting occurs by equally weighting each word. I.e. that the word *the* weights the same amount as *demand* or *supply* in an economic context.

3 Model 2: Creating Algorithms to Detect More Than One Category

3.1 Background

To create the duplicate data, we performed the same batching as was done for Model 1, but we made sure to check that there were, in fact, duplicate data points. We ran a similar check on the test data as well, and in both cases, we found that 14% of the data was duplicated, which aligns with the overall proportion of duplicate data that exists in the total training and test sets.

3.2 Model Training

In this model, we would like address the issue of articles that can be classified into multiple categories (*binary classification* or *multiclass classification*). This is an important issue when it comes to a number of machine learning models that will classify the ID into one of the categories and then read the ID as being misclassified in the other categories in which it is also included. This consequently impacts the precision, recall and F-score metrics as the number of false negatives in the confusion matrix increases. However, there are two training models of the nine at our disposal- SVM and Neural Networks- which can be tuned to test ID's across all the classes in which they exist. Therefore, the next portion of our study delves into dealing with the multiclass data by using, and potentially further tuning, the two above algorithms.

SVM Multi-Classification Model

First, we decided to enhance the SVM model to accomodate duplicate data points by classifying each category separately, and then running the SVM model on each of the separate classifications.

Table 9: SVM Multi-Class Model

Class	Precision	Recall	F1 Score
CCAT	0.84	0.70	0.73
ECAT	0.78	0.65	0.69
GCAT	0.85	0.77	0.78
MCAT	0.87	0.82	0.84

Precision is the fraction of true positives over the total number of true and false positives; recall is the fraction of true positives over the number of true positive plus the number false negatives. F1 is the harmonic mean of precision and recall. As we can see from the table above, we obtain that the SVM model yields different results for each of the classes. This is due to our sampling procedure. On the one hand for the test data we sampled 250 articles from each of the four classes; on the other hand for the training data sets we sampled 1,000 articles from the overall population. The sampling technique for the training set yields to each of class having a different proportion of each class to be sampled for training purposes. This difference is the main driver in the discrepancy across classes for Precision, Recall, and F1 Score.

Neural Network

Prior machine learning experiments have shown that Neural Networks are one of the algorithms that can appropriately handle binary and multi-class classification. Therefore, we looked into tuning the *nnet* implementation in R in order to test the duplicate data. As can be seen from Table 2, the neural network algorithm had the second highest run time and the lowest F-score and cross-validation metrics. We can thus infer with reasonable certainty that the test on the duplicate data set would be computationally exhaustive and would also yield relatively low performance metrics. In a consultation with the professor, we concluded that neural networks are one of the weaker algorithms within the R environment, and consequently, we feel that while neural networks should be considered when dealing with data that includes multiple classifications, it should not be relied upon for optimal results.

3.3 Model Selection

Dealing with data points that can be classified into multiple classes is an issue that requires one to understand which machine learning algorithms can actually be utilized for testing. As aforementioned, SVM and neural networks are the two such algorithms at our disposal within the context of this project, and of these two, an SVM implementation with a classifier for each class is likely the better option.

Taking a slightly deeper dive into the two models, the separately classified SVM provides a picture of the performance metrics for each class. This is a useful result as we could then fine-tune the SVM model parameters and explore if there was a particular category in which the predictive power could be substantially increased. In contrast, the neural network would yield one result, and given the *"black box"* [23] issue that arises when dealing with neural networks, it would be difficult to pinpoint how the multiple classes were impacting model performance.

3.4 Model Tuning

Since we have selected SVM for the duplicate case, we would look to fine-tune the same parameters that we discussed in section 2.3: Cost and Cross. As we had seen, altering the cost and cross parameters either did not impact, or slightly lowered, our F-scores. Consequently, we could expect to see a similar outcome in the fine-tuning of SVM for this model as well.

Table 10: Data tuning SVM

Algorithm change	Precision	Recall	F1 Score
Cost = 10^{-1}	0.75	0.71	0.71
Cost = 10^{3}	0.75	0.77	0.76
Cost = 10^{5}	0.75	0.77	0.76
Cross = 5	0.76	0.77	0.76
Cross = 10	0.76	0.78	0.76

3.5 Preliminary Conclusion

This second model seeks to understand how to handle the data set just as we extracted it i.e. with multi-classified data. Having sample points that fall into two or more classes introduces complexities that can be managed either by "upgrading" certain models, such as SVM, or by applying algorithms that were designed to deal with the multi-classification case, such as neural networks.

Outside of utilizing SVM and neural networks, it may be helpful to look into other algorithms that are able to deal with multiple classifications, including Naive Bayes and Decision Trees, and compare the results.

[23] The black box issue refers to the difficulty in understanding how exactly your neural network came up with a certain output

4 Further Study

The RCV1 data set includes five hierarchies of classification for the articles that are included within it. The lower-hierarchy classifications (H2,H3,H4,H5) more specifically identify the articles, and thus, it would be prudent to run algorithms on these lower levels in order to gain more detailed class separation. From a broad perspective, there are few components of this hierarchical data that must be taken into account:

- Number of Classes: As we move down in the hierarchy, we see that the number of classes increases substantially, especially in H2 and H3. This will lead to a few results:
  - The run time for each algorithm will increase, as the class separation will become more complex.
  - Algorithms which are better equipped to handle a large number of classes will perform better.

To do the lower hierarchies, we would create a map with the hierarchy levels and then let the machine run through it and doing a one vs. rest approach everywhere. However, here we faced the difficulty of running time as the number of cases increases with lower hierarchy levels. Furthermore, other techniques that can predict multiple outputs at the same time (e.g. NNET) were also not considered as their predictive power was among the worst and the running time was a concern as well.

5 Conclusion

The goal of this assignment was twofold:

1. Familiarize ourselves with natural language processing techniques and understand which algorithm may be the best in a given scenario.
2. Look at two different approaches of tackling the same problem and understanding the effects of sample bias.

As we could show, depending on what are constraints and goals are, we will need to use different algorithms to tackle the issues. There is no unique one-size-fits-all approach. As a conclusion, before starting with any analysis, it is always of utmost importance to determine what are objectives are and also examine the data to know what the constraints are.