Machine Learning BITS F464

Assignment #1 Report

BITS Pilani

September 30, 2015

Problem Statement: #5 Classifications of Computer Science Research Articles

Submitted By: Abhinav Agarwal, 2013A7PS124P

1 Introduction

The problem is to classify given set of Computer Science papers to different fields of study based on the body of the research papers. This particular problem identifies to Multi Label Classification because multiple target labels i.e. categories must be assigned to each instance. For example, a paper describing a obstacle avoiding robot which learns on its paths can come under the umbrella categories of ‘Machine Learning’, ‘Artificial Intelligence’ and ‘Robotics’ but a paper describing a modification to an existing SVM algorithm may only be classified under ‘Machine Learning’. A list of the labels can be found in the following section.
Tools used: R language, RStudio, R Packages (tm, plyr, class, randomForest, MASS, party), MS Excel, Linux shell scripting.

1.1 Assumptions

- The labels are considered as mutually exclusive classes. A paper is mapped to only one class, its primary class. So this problem is restricted to a multi class problem and not a multi label problem.
- It is assumed that only the body of the research paper classifies the paper. Other factors that may improve classification like author’s previous collaborations, researcher’s personal experience areas etc. are not considered.
- The pdf format is a container format. Conversion of pdf to txt file involves Optical Character Recognition and may not be perfect. It is assumed that not enough error is introduced which may hamper classification. However, it is expected that most of errors are eliminated during text clean up in preprocessing.

2 Data

2.1 Source

All data was sourced from the pre-print online repository arxiv.org accessed through the Computing Research Repository arxiv.org/corr

A script was written to scrape the repository for the recent articles and their pdfs were downloaded and their classification and unique identifiers were added to a csv file.

2.2 Preprocessing

A total of 212 files were downloaded out of which 75% = 159 were put in the training set and 25% = 53 in the testing set.
The library 'xpdf' was used to convert these pdfs to text files. A simple command which automates running the library for all documents in linux was used:

```
for file in *.pdf; do pdftotext "$file"; done
```

To split the training data and testing data into their respective directories another such command was used:

```
for file in 'cat names.txt'; do mv "$file" test; done
```

The csv file created contains 30 columns with the first column being the name of the file and the rest 29 being the labels i.e. the fields of study. These 29 labels are:

cs.AI	*Artificial Intelligence*
cs.CL	*Computation and Language*
cs.CC	*Computational Complexity*
cs.CG	*Computational Geometry*
cs.GT	*Computer Science and Game Theory*
cs.CV	*Computer Vision and Pattern Recognition*
cs.CY	*Computers and Society*
cs.CR	*Cryptography and Security*
cs.DS	*Data Structures and Algorithms*
cs.DB	*Databases*
cs.DL	*Digital Libraries*
cs.DM	*Discrete Mathematics*
cs.DC	*Distributed, Parallel, and Cluster Computing*
cs.FL	*Formal Languages and Automata Theory*
cs.AR	*Hardware Architecture*
cs.HC	*Human-Computer Interaction*
cs.IR	*Information Retrieval*
cs.IT	*Information Theory*
cs.LG	*Machine Learning*
cs.LO	*Logic in Computer Science*
cs.NI	*Network and Internet Architecture*
cs.OS	*Operating Systems*
cs.OH	*Other*
cs.PF	*Performance*
cs.PL	*Programming Languages*

cs.RO	*Robotics*
cs.SI	*Social and Information Networks*
cs.SE	*Software Engineering*
cs.SY	*System and Control*

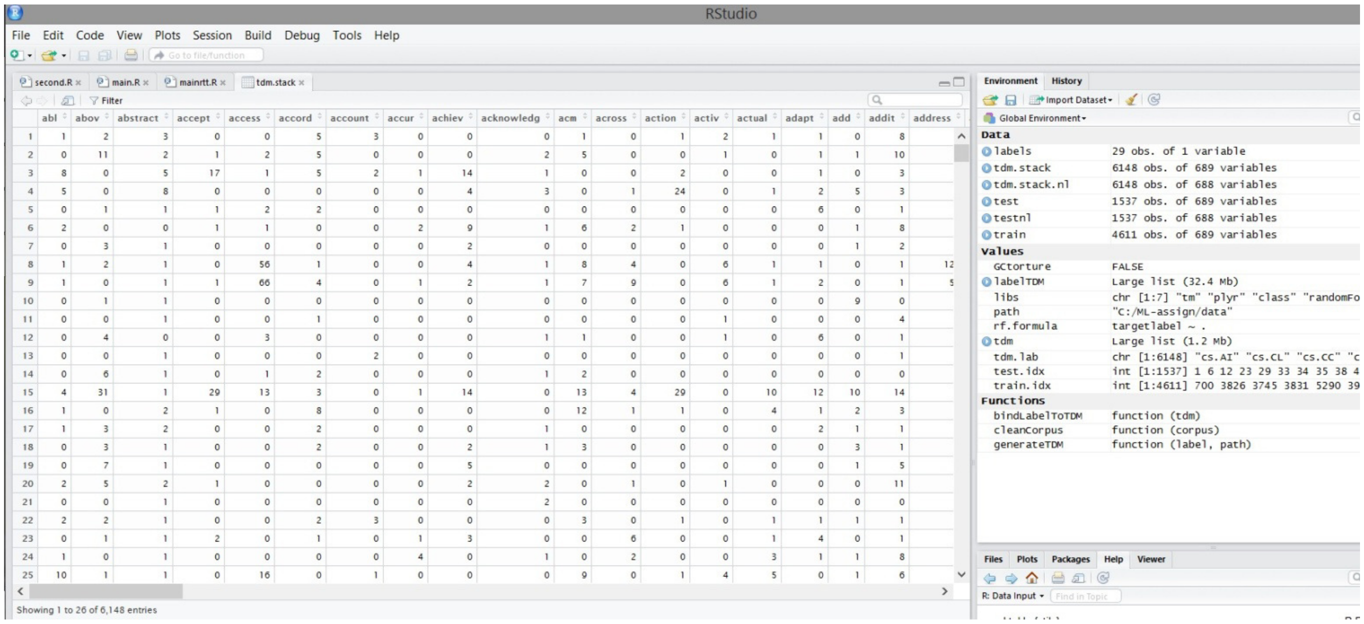
CORR describes a total of 40 labels but 11 were removed because of very less frequency of documents classified to those labels.
Papers which were cross-listed to non-CS fields of study were removed. An example of such articles is say, something which describes applications machine learning to astronomical data. Such a paper would be classified to Astro-ph as well as cs.LG.

After writing the train.csv file a script was written to create subdirectories in the data directory with names as the names of the labels and the papers were put into these directories as per their class values. We now have 212 text files distributed in these subdirectories.

The data is handled by creating corpuses of each document and successively its term document matrix. A term document matrix contains information about frequency of word i.e. term occurrence. The term document matrixes of all documents of a particular class i.e. all the files in that particular subdirectory are created after corpus cleanup and put into a list along with the label of that class. *cleanCorpus()* involves:

- Removal of punctuation characters
- Removal of numbers because they do not contribute to classification
- Stripping of extra whitespace which is excess of the regular blanks between the words
- All terms are converted to lower case because semantically 'intelligence' is equivalent to 'Intelligence'
- Removal of stopwords i.e. common words which are of little relevance like 'and', 'this', 'the' etc.
- Stemming of terms using Porter's stemming algorithm. This involves reducing inflected or derived words to their word stem, base or root stem. For example, 'swims', 'swimming', 'swimmer' have same stem 'swim'.
- Removal of non ASCII characters.

After the corpus is cleaned, the data is clubbed into TDMs as described above, bound with class labels, split into testing (30%) and training (70%) datasets. This concludes preprocessing. The below image illustrates a selection of a cleaned TDM:


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
second.R main.R mainrtt.R tdm.stack
Filter
Environment History
Import Dataset
Global Environment
Files Plots Packages Help Viewer
R: Data Input
Find in Topic

	abl	abov	abstract	accept	access	accord	account	accur	achiev	acknowledg	acm	across	action	activ	actual	adapt	add	addit	address
1	1	2	3	0	0	5	3	0	0	0	1	0	1	2	1	1	0	8	
2	0	11	2	1	2	5	0	0	0	2	5	0	0	1	0	1	1	10	
3	8	0	5	17	1	5	2	1	14	1	0	0	2	0	0	1	0	3	
4	5	0	8	0	0	0	0	0	4	3	0	1	24	0	1	2	5	3	
5	0	1	1	1	2	2	0	0	0	0	0	0	0	0	0	6	0	1	
6	2	0	0	1	1	0	0	2	9	1	6	2	1	0	0	0	1	8	
7	0	3	1	0	0	0	0	0	2	0	0	0	0	0	0	0	1	2	
8	1	2	1	0	56	1	0	0	4	1	8	4	0	6	1	1	0	1	12
9	1	0	1	1	66	4	0	1	2	1	7	9	0	6	1	2	0	1	5
10	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	
11	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	4	
12	0	4	0	0	3	0	0	0	0	1	1	0	0	1	0	6	0	1	
13	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	1	
14	0	6	1	0	1	2	0	0	0	1	2	0	0	0	0	0	0	0	
15	4	31	1	29	13	3	0	1	14	0	13	4	29	0	10	12	10	14	
16	1	0	2	1	0	8	0	0	0	0	12	1	1	0	4	1	2	3	
17	1	3	2	0	0	2	0	0	0	1	0	0	0	0	0	2	1	1	
18	0	3	1	0	0	2	0	0	2	1	3	0	0	0	0	0	3	1	
19	0	7	1	0	0	0	0	0	5	0	0	0	0	0	0	0	1	5	
20	2	5	2	1	0	0	0	0	2	2	0	1	0	1	0	0	0	11	
21	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	
22	2	2	1	0	0	2	3	0	0	0	3	0	1	0	1	1	1	1	
23	0	1	1	2	0	1	0	1	3	0	0	6	0	0	1	4	0	1	
24	1	0	1	0	0	0	0	4	0	1	0	2	0	0	3	1	1	8	
25	10	1	1	0	16	0	1	0	0	0	9	0	1	4	5	0	1	6	

Showing 1 to 26 of 6,148 entries

Data	
labels	29 obs. of 1 variable
tdm.stack	6148 obs. of 689 variables
tdm.stack.nl	6148 obs. of 688 variables
test	1537 obs. of 689 variables
testnl	1537 obs. of 688 variables
train	4611 obs. of 689 variables
Values	
GCtorture	FALSE
labelTDM	Large list (32.4 Mb)
libs	chr [1:7] "tm" "plyr" "class" "randomFo
path	"C:/ML-assign/data"
rf.formula	targetlabel ~ .
tdm	Large list (1.2 Mb)
tdm.lab	chr [1:6148] "cs.AI" "cs.CL" "cs.CC" "c
test.idx	int [1:1537] 1 6 12 23 29 33 34 35 38 4
train.idx	int [1:4611] 700 3826 3745 3831 5290 39
Functions	
bindLabelToTDM	function (tdm)
cleanCorpus	function (corpus)
generateTDM	function (label, path)

3 Algorithms

Multi Label classification approaches are broadly two: problem-transformation and algorithm-adaptation. Problem Transformation involves transforming the multi-label problem into single-label problems and then using any off-the-shelf single-label classifier to suit requirements i.e. adapting data to the algorithm. Algorithm Adaptation involves adapting a single-label algorithm to produce multi-label outputs i.e. adapting algorithm to the data.

Since we have reduced the problem to a multi class problem instead we chose an three algorithms which were suitable for such a situation:

Random Forests: These are ensemble learning methods for classification that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. They correct over fitting in decision trees.

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

Conditional Inference Trees: CTree is a non-parametric class of regression trees embedding tree-structured regression models into a well defined theory of conditional inference procedures.

Linear Discriminant Analysis: This is a generalization of Fisher's Linear Discriminant. It may have poor predictive power where there are complex forms of dependence on the explanatory factors and variables. In cases where it is effective, it has the virtue of simplicity.

These three algorithms were tested on the data and the results are discussed below.

3.1 Code

```
# init
libs <- c("tm", "plyr", "class", "randomForest", "lda", "MASS", "party") # load requred R packages
lapply(libs, require, character.only = TRUE)

# set options
options(stringsAsFactors = FALSE)

# set parameters
labels <- read.table('labels.txt') # read classes
path <- paste(getwd(), "/data", sep="") # get directory path of data
```

```
# clean text
cleanCorpus <- function(corpus) {
  corpus.tmp <- tm_map(corpus, removePunctuation) # remove punctuation
  corpus.tmp <- tm_map(corpus.tmp, removeNumbers) # remove numbers
  corpus.tmp <- tm_map(corpus.tmp, stripWhitespace) # strip extra whitespace
  corpus.tmp <- tm_map(corpus.tmp, content_transformer(tolower)) # convert all text to lowercase
  corpus.tmp <- tm_map(corpus.tmp, stemDocument, language = "english") # stem words using porter's
stemming algorithm
  corpus.tmp <- tm_map(corpus.tmp, removeWords, stopwords("english")) # remove stopwords
  corpus.tmp <- tm_map(corpus.tmp, function(x) iconv(x, "latin1", "ASCII", sub="")) # remove non ASCII
characters
  corpus.tmp <- tm_map(corpus.tmp, PlainTextDocument) # ensure that corpus is plaintextdocument
  return(corpus.tmp)
}

# build TDM
generateTDM <- function(label, path) {
  s.dir <- sprintf("%s/%s", path, label) # create directory path by appending label to master path
  s.cor <- Corpus(DirSource(directory = s.dir), readerControl = list(language = "en")) # create corpus
of all docs in directory
  s.cor.cl <- cleanCorpus(s.cor) # call cleancorpus function to clean text
  s.tdm <- TermDocumentMatrix(s.cor.cl) # create tdm
  s.tdm <- removeSparseTerms(s.tdm, 0.7) # remove sparsely occuring words
  return(list(name = label, tdm = s.tdm))
}

tdm <- lapply(labels, generateTDM, path = path) # fetch list of TDMs for all classes

# attach name
bindLabelToTDM <- function(tdm) { # binds class name to appropriate TDM
  s.mat <- t(data.matrix(tdm[["tdm"]]))
  s.df <- as.data.frame(s.mat, stringsAsFactors = FALSE)
  s.df <- cbind(s.df, rep(tdm[["name"]], nrow(s.df)), row.names = NULL)
  colnames(s.df)[ncol(s.df)] <- "targetlabel"
  return(s.df)
}

labelTDM <- lapply(tdm, bindLabelToTDM) # calls binding function

# stack
tdm.stack <- do.call(rbind.fill, labelTDM) # stacks similar TDMs together
tdm.stack[is.na(tdm.stack)] <- 0 # replace NA values with zero

# hold-out
train.idx <- sample(nrow(tdm.stack), ceiling(nrow(tdm.stack) * 0.75)) # split data into test and train
test.idx <- (1:nrow(tdm.stack)) [- train.idx]

tdm.lab <- tdm.stack[, "targetlabel"]
tdm.stack.nl <- tdm.stack[, !colnames(tdm.stack) %in% "targetlabel"] # same dataset without class values
train <- tdm.stack[train.idx, ] # train dataset
test <- tdm.stack[test.idx, ] # test dataset with class values
testnl <- tdm.stack.nl[test.idx, ] # test dataset without class values

set.seed(1234)
```

```
# Random FOrrest
rf.train <- train
rf.test <- rf.test
rf.testnl <- rf.testnl
rf.formula <- as.formula("targetlabel ~ .")
environment(rf.formula) <- environment()
rf.train$targetlabel <- as.factor(rf.train$targetlabel)
rf <- randomForest(rf.formula, data = rf.train, ntree = 5000, mtry = 15, importance = TRUE)
rf.testnl$targetlabel = predict(rf, rf.testnl, type = "response")
table(rf.test$targetlabel, rf.testnl$targetlabel)
prop.table(table(rf.test$targetlabel, rf.testnl$targetlabel), 1)

# Conditional Inference Tree
ctree.train <- train
ctree.testnl <- testnl
ctree.train$targetlabel <- as.factor(ctree.train$targetlabel)
ctree <- ctree(targetlabel ~ ., data = ctree.train)
ctree.testnl$targetlabel <- predict(ctree, ctree.testnl, type = "response")
ctree.conf.mat <- table(ctree.test$targetlabel, ctree,testnl$targetlabel)
ctree.prop.table <- prop.table(table(ctree.test$targetlabel, ctree.testnl$targetlabel), 1)

# Linear Discriminant Analysis
lda.train <- train
lda.testnl <- testnl
lda.test <- test
lda <- lda(targetlabel ~ ., data = lda.train)
k <- as.data.frame(predict(lda, lda.testnl, type = "response"))
lda.testnl$targetlabel = k
lda.testnl$targetlabel = k$class
lda.conf.mat <- table(lda.test$targetlabel, lda.testnl$targetlabel)
lda.prop.table <- prop.table(table(lda.test$targetlabel, lda.testnl$targetlabel), 1)
```

4 Results

The accuracy of the algorithm can be calculated from the confusion matrix.
Accuracies of the different algorithms were as follows:

CTree	0.6803937
randomForest	0.7560629
LDA	0.7983464

Accuracy order was found to be:

`CTree < randomForest < LDA`

Hence, LDA was found to be most accurate as expected.

The problem was successfully solved and research papers classified with appreciable accuracy.

5 Further Scope

Results can be improved vastly by having much more and better quality data.
The accuracy can be improved by providing all labels with same number of training examples. More algorithms can be tried to judge best accuracy and recall. The dataset can also be improved by better formatting like putting the data in arff files which is ideal for such multi label datasets.

6 References

[1] www.arxiv.org

[2] www.arxiv.org/corr/home

[3] Multi-label Classification by Jesse Read, *Summer School on Machine Learning and Knowledge Discovery in Databases (MLKDD)* July 14-17, 2013, Sao Carlos, SP, Brazil.
Part 1: https://users.ics.aalto.fi/jesse/talks/Multilabel-Part01.pdf
Part 2: https://users.ics.aalto.fi/jesse/talks/Multilabel-Part02.pdf

[4] http://scikit-learn.org/stable/modules/multiclass.html

[5] http://blog.ideas2it.com/multi-label-classification-with-r/

[6] https://cran.r-project.org/web/packages/partykit/vignettes/ctree.pdf

[7] http://www.r-bloggers.com/my-intro-to-multiple-classification-with-random-forests-conditional-inference-trees-and-linear-discriminant-analysis/

[8] http://maths-people.anu.edu.au/~johnm/courses/dm/math3346/2008/pdf/r-exercisesVI.pdf

[9] http://www.bios.unc.edu/~dzeng/BIOS740/randomforest.pdf

[10] https://cran.r-project.org/web/packages/randomForest/randomForest.pdf