CS350 Networks of Software and Developers

Riccardo Romio

2022-12-15

Progress So Far

Data Gathering and Cleaning

The central pieces of data needed to create the model are: **Task View packages**, their **description file**, their corresponding (already) **classified Task View** and their **package dependence index** corresponding to all Task Views.

All of this data can be collected by querying CRAN and using tools from packages ctv and cranly.

ctv

Using ctv, it is possible to get all Task Views in a neatly organized list, with all underlying packages and description files. A function has been built to extract this and gives a table with 3 columns: taskView, packages and description. These are all strings.

tm

The next step has currently been performed using the tm (text mining) package, but in the near future will be updated to use the tidytext package, as it allows for a more powerful, faster, and easier handling of text data (this will be following practices from Text Mining with R, A Tidy Approach by Julia Silge and David Robinson).

To convert the description files in usable features for the model, the first requirement is to strip the text data of capital letters, numbers, punctuation, and stop words. Lastly, it is important to stem the text, so that slightly different variations of the same word are counted as the same stem. Using built in functions to the tm package, below it can be seen how the description files are transformed.

```
## [1] "The functions are designed to calculate the most widely-used county-level variables in agricultu
```

```
## [1] "function design calcul widelyus countylevel variabl agricultur product agriculturalclimat weath
```

```
#Creating the bag of words
dtm <- DocumentTermMatrix(corpus)
dtm <- removeSparseTerms(dtm, 0.95)
finalDataSet <- as.data.frame(as.matrix(dtm))
finalDataSet$taskView <- allDescCore$taskView #inserts class of each observation for the model
#perhaps put this after Package dependence index so this is the last column
```

A possible improvement is to change the character "-" to a space, as important words are merged with the next word, as can be seen in the example.

Next, it is necessary to turn the transformed description files into a Document Term Matrix, a sparse matrix that determines the frequency of a word in each "document", or in this context, the frequency of a word in each description file. It is structured so that the columns represent all available words in the constructed corpus, and each row is a different description file. Lastly, terms that have a 95% of empty, or occurring

0 times in a document, are removed from the dataset as they are likely to not give us a lot of information regarding the model. This parameter can be changed perhaps including this as an input in the web-app that will be created. Regardless, this is the dataset that will be used to train and test the classification model.

Model Building and Testing

nnet

Lastly, we split the dataset in 60% train data, 40% test data, and start the model training. Using the `multinom()` function from the nnet package, it is possible create a multionmial logistic regression model training on the test data.

```
sample_size <- floor(0.6 * nrow(finalDataSet))
set.seed(777)

# randomly split data in r
picked <- sample(seq_len(nrow(finalDataSet)),size = sample_size)
train <- finalDataSet[picked,]
test <- finalDataSet[-picked,]

finalDataSet$taskView <- relevel(factor(finalDataSet$taskView), ref = "Agriculture")
model <- multinom(taskView ~ ., data = train, MaxNWts = 10000, maxit = 1000)
```

The model is then tested on the test data, and the class which is predicted with the highest probability is the one that is chosen. Below is a table outlining the model accuracy for each Task View. As can be seen, there is some strong variance between them. Lastly, we can see the overall model accuracy, which is better than a random classification, as there are 42 choices of Task Views, but is nonetheless very low.

```
##       classificationOverview classificationCounter
## 1                Agriculture            0.09090909
## 2                   Bayesian            0.37333333
## 3            CausalInference            0.27868852
## 4                   ChemPhys            0.00000000
## 5             ClinicalTrials            0.05555556
## 6                    Cluster            0.42500000
## 7                  Databases            0.07142857
## 8      DifferentialEquations            0.22222222
## 9               Distributions            0.60824742
## 10              Econometrics            0.11320755
## 11            Environmetrics            0.05000000
## 12              Epidemiology            0.03846154
## 13        ExperimentalDesign            0.23404255
## 14              ExtremeValue            0.09090909
## 15                   Finance            0.12727273
## 16            FunctionalData            0.38461538
## 17           GraphicalModels            0.00000000
## 18  HighPerformanceComputing            0.10810811
## 19                 Hydrology            0.08333333
## 20           MachineLearning            0.11627907
## 21            MedicalImaging            0.00000000
## 22              MetaAnalysis            0.20312500
## 23               MissingData            0.40963855
## 24               MixedModels            0.15000000
## 25           ModelDeployment            0.00000000
## 26 NaturalLanguageProcessing            0.16666667
## 27      NumericalMathematics            0.28260870
```

```
## 28        OfficialStatistics           0.07843137
## 29              Optimization           0.48000000
## 30          Pharmacokinetics           0.28571429
## 31             Phylogenetics           0.02040816
## 32             Psychometrics           0.20238095
## 33      ReproducibleResearch           0.14285714
## 34                    Robust           0.04761905
## 35                   Spatial           0.08955224
## 36            SpatioTemporal           0.15789474
## 37           SportsAnalytics           0.08695652
## 38                  Survival           0.21212121
## 39        TeachingStatistics           0.16666667
## 40                TimeSeries           0.40425532
## 41                  Tracking           0.00000000
## 42           WebTechnologies           0.54385965
```

```
## [1] "Model Accuracy is: 0.0170517051705171"
```

Next steps:

- Tweak parameters such as sparsity
- Improve text cleaning methods with tidytext,
- Introduce parallel programming to improve model training speed (maybe),
- Begin creating web-app with Shiny with basic ui.