Text Mining: classification

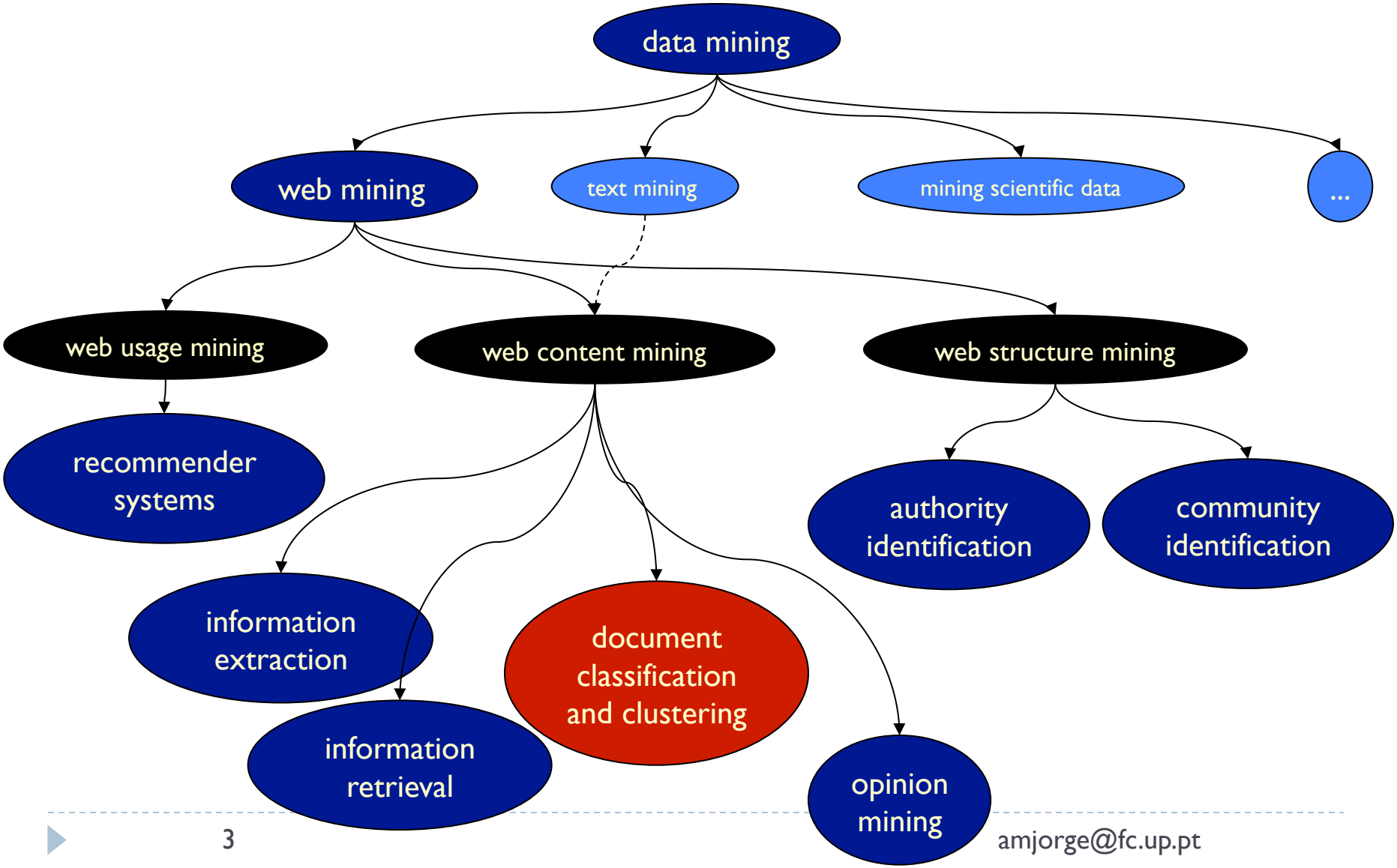
Alípio Jorge, DCC-FC, Universidade do Porto
amjorge@fc.up.pt

Overview

- Classifying documents (categorization)
  - Building a corpus
  - Pre-process the corpus to transform it into a matrix-like representation
    - Each document is "described by a set of features"
  - Add information on the class assigned to each document by some expert on the domain of the documents
  - Build a classification model that can be used to predict the classification of new documents by looking only at its "description"

amjorge@fc.up.pt

Knowledge (sort of) tree



amjorge@fc.up.pt

Classification: one example task

- We have a collection of documents from two different categories (classes) and we want to automatically build a model that is able to assign a new document to one of the classes.

- Our collection:
  - 23 scientific papers on collaborative filtering and 30 scientific papers on image processing.
- Approach:
  - preprocess
  - vectorize
  - build model

Classification

- Loading the data
  - the papers are in two directories named "cf" and "img"
  - we must load the pdfs and transform them to plain texts.
    - some documents were discarded because they gave problems in conversion.
    - we only load the first page of each pdf (to cut down the number of terms)

```
docs1<-Corpus(
  DirSource("C:/temp/cf/"),
  readerControl=list(
            reader=readPDF(control=list(text="-f 1 -l 1")),
            language="en") )
> docs1
A corpus with 23 text documents
```

Classification

- Loading the data
  - now the documents from the other class ("img")

```
docs2<-VCorpus(
  DirSource("C:/temp/img/"),
  readerControl=list(
            reader=readPDF(control=list(text="-f 1 -l 1")),
            language="en")
> docs2
A corpus with 30 text documents
```

Classification

- inspecting
  - some obvious problems can sometimes be seen with visual inspection of the some of the docs

```
> inspect(docs1[[1]])
> inspect(docs2[[1]])
```

Classification

- Now we join the two corpora
  - keep an eye on the number of terms (attributes)

```
> docs<-c(docs1,docs2)
> docs
A corpus with 53 text documents
> DocumentTermMatrix(docs)
A document-term matrix (53 documents, 6047 terms)

Non-/sparse entries: 13451/307040
Sparsity           : 96%
Maximal term length: 2994
Weighting          : term frequency (tf)
```

Classification

- Examples of cleaning up
  - remove whitespaces
  - use lower caps only

```
> docs<-tm_map(docs,stripWhitespace)
> docs<-tm_map(docs,tolower)
> DocumentTermMatrix(docs)
A document-term matrix (53 documents, 6047 terms)

Non-/sparse entries: 13451/307040
Sparsity           : 96%
Maximal term length: 2978
Weighting          : term frequency (tf)
```

Classification

- Further examples of cleaning up
  - remove stopwords
  - stemming

```
> docs<-tm_map(docs,removeWords, stopwords("english"))
A document-term matrix (53 documents, 5599 terms)

Non-/sparse entries: 10749/285998...
Maximal term length: 2889 ...
> docs<-tm_map(docs,stemDocument)
> DocumentTermMatrix(docs)
A document-term matrix (53 documents, 4642 terms)

Non-/sparse entries: 9972/236054
Sparsity           : 96%
Maximal term length: 2887 ...
```

Stemming

- Words have various syntactical forms
  - run, running, runner, ran
  - syntactical variations of the same root form "run" (stem)
- Variety of terms with the same content information degrade the ability to find patterns

- Stemming
  - reducing words to their stems or roots
  - suffix removal
    - computer, compute, computing, computable, computation → comput
  - increases recall
  - reduces number of terms
  - but may introduce errors
    - cope and cop
  - use with caution

Classification

- Building the data set
  - adding a class column

```
dtm <- weightTfIdf(
          DocumentTermMatrix(docs))
data <-
   cbind(data.frame(
        as.matrix(dtm),
        class=c(rep("cf",23),rep("img",30))))

.
```

Preprocessing: Clean-ups

- stopwords
- stemming
- case
- strip white spaces
- remove punctuation
  - corpus<-tm_map(corpus,removePunctuation)
- remove numbers
  - corpus<-tm_map(corpus,removeNumbers)
- remove sparse terms
  - removeSparseTerms(dtm, 0.6)
    - terms with not occuring in at least 60% of the docs

Classification: trying an SVM classifier

- We start by removing sparse terms

```
dtm<-removeSparseTerms(dtm,0.7)
data <- cbind(data.frame(
        as.matrix(dtm),
        class=c(rep("cf",23),rep("img",30))))
library(e1071)
## Splitting the data into a training and test set
randomRows <- sample(1: nrow(data), as.integer(0.7*nrow(data)))  # 70% random rows
train <- data[randomRows, ]
test <- data[-randomRows, ]  # note the "-" to exclude these rows

m <- svm(class ~ ., train)  # obtain the model with the training set
preds <- predict(m, test)  # apply it to obtain predictions for the test set

## now lets calculate the error rate of the predictions
## first the confusion matrix
cm <- table(preds, test$class)
err <- 1- sum(diag(cm)) / sum(cm)
```

Classification example: summary

- Applying classification algorithms to text
- making a corpus out of directories with pdf files
- Using TF-IDF scheme
- Removing stopwords
- and more cleanups
- Using some classification model on the obtained data
  - Which model to use?
    - The model selection problem

Using package performanceEstimation

- Package performanceEstimation can be used to compare and select alternative models
- An example with the previous data

```
library(e1071)
library(performanceEstimation)

compResults <- performanceEstimation(
                 PredTask( class ~., data),
                 workflowVariants(learner="svm", learner.pars = list(cost=c(1,3,10))),
                 EstimationTask(metrics="err", method = CV() )    )

## We can obtain a summary of the results of the comparison doing:
summary(compResults)

## Or visually doing
plot(compResults)
```

A more extended example

- Using the crude and acq sets of documents

```
> data(crude) # 20 docs
> data(acq)   # 50 docs
> docs <- c(crude, acq)
>
> docs <- tm_map(docs, stripWhitespace)
> docs <- tm_map(docs, content_transformer(tolower))
> docs <- tm_map(docs, removeWords, stopwords("english"))
> docs <- tm_map(docs, removePunctuation)
> docs <- tm_map(docs, removeNumbers)
>
> dtm <- DocumentTermMatrix(docs)
> dtm <- removeSparseTerms(dtm, 0.8)
>
> dat <- cbind(
+   data.frame(as.matrix(dtm),
+              class=c(rep("crude",20),rep("acq",50))
+              ))
## removing two constant columns
> dat <- dat[,-which(colnames(dat) %in% c("reuter", "said"))]
```

A more extended example

- Cont.

```
> library(e1071)
> library(performanceEstimation)
>
> exp <- performanceEstimation(
+   PredTask(class ~ ., dat),
+   c( Workflow(learner="naiveBayes"),
+      workflowVariants(learner="svm", learner.pars=list(kernel=c("linear", "radial")))
+      ),
+   EstimationTask(metrics="err", method=CV())
+ )


##### PERFORMANCE ESTIMATION USING  CROSS VALIDATION  #####

** PREDICTIVE TASK :: dat.class

++ MODEL/WORKFLOW :: naiveBayes
Task for estimating  err  using
 1 x 10 - Fold Cross Validation
           Run with seed =  1234
Iteration :**********
...
...
```

A more extended example

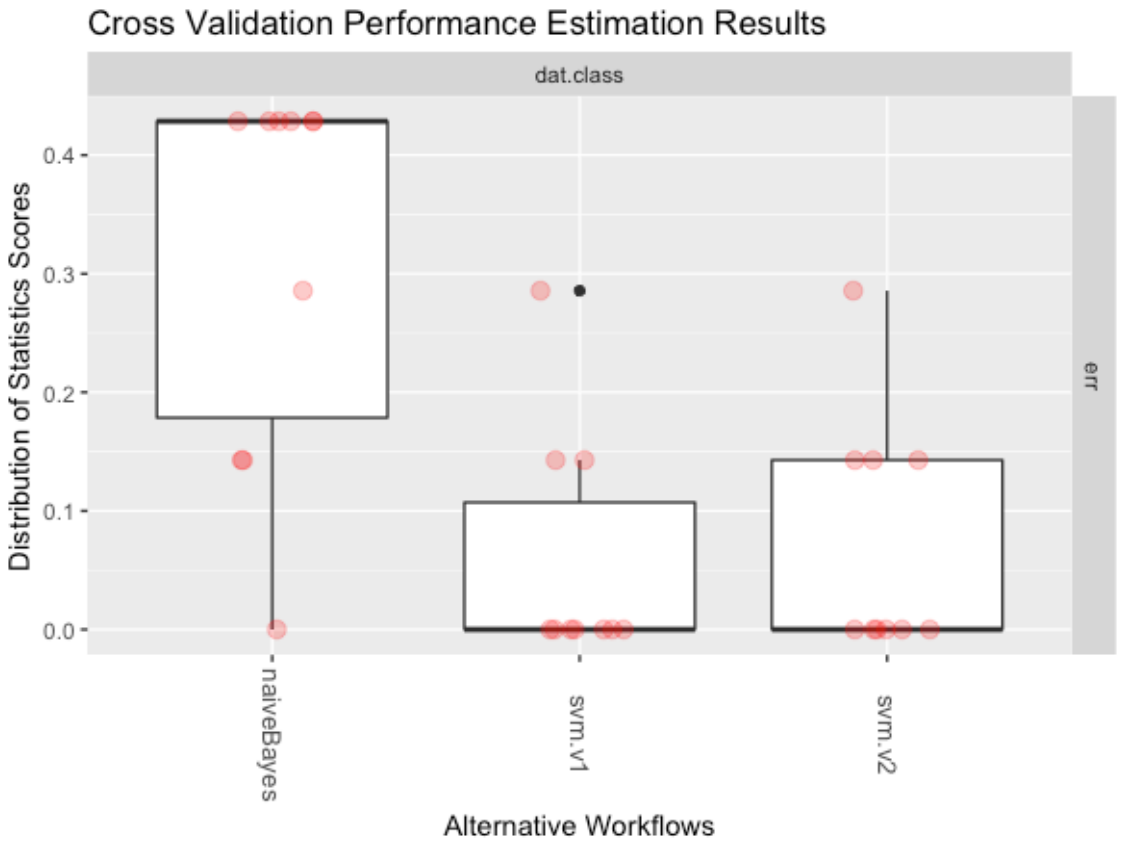
- Cont.

```
> summary(exp)

== Summary of a  Cross Validation Performance Estimation Experiment ==

Task for estimating  err  using
 1 x 10 - Fold Cross Validation
           Run with seed =  1234

* Predictive Tasks ::  dat.class
* Workflows  ::  naiveBayes, svm.v1, svm.v2

-> Task:  dat.class
  *Workflow: naiveBayes
                err
avg     0.3142857
std     0.1621846
med     0.4285714
iqr     0.2500000
min     0.0000000
max     0.4285714
invalid 0.0000000

...
...
```

A more extended example

- Cont.

```
> plot(exp)
```



Classification

- DM algorithms that have given good results with text
  - Naive Bayes
  - K-NN
  - SVM
    - Typically linear kernel
- What is the impact of the number of features in classifier performance?
- To reduce features and sparsity we can
  - Remove sparse terms (with some risk)
  - Use feature reduction techniques: e.g. information gain

Resources

- Books
  - Web Data Mining, Bing Liu, Springer, 2007
  - Mining the World Wide Web, Chang, G., Healey, M., McHugh, J., Wang, J., Kluwer Academic Press, 2001.
- Manuals
  - tm package documentation
    - Ingo Feinerer, "Introduction to the tm package: Text Mining in R", 2011