SDSJ 2017 taskA

Konstantin Ivanin

16 СЃРµPSC ,СЦР±СЂСЦ 2017 Pi

Задача A: определение релевантности вопроса

В данной задаче участникам необходимо построить алгоритм, определяющий релевантность поставленных вопросов к параграфу текста. Для решения этой задачи требуется не только понимать, относится ли вопрос к параграфу, но и насколько корректно он поставлен.

Это задача бинарной классификации, в которой целевая переменная target принимает два значения: 0 и 1. Классу 1 соответствуют релевантные вопросы, заданные к параграфу человеком. К классу 0 относятся вопросы, либо заданные человеком к другим параграфам, либо были составлены компьютером. В качестве целевой метрики используется ROC-AUC.

Для решения задачи A участникам дается два файла:

1. Тренировочные 119 399 пар вопросов и параграфов train_taskA.csv, имеющие вид: paragraph_id, question_id, paragraph, question, target.
2. Тестовые 74 295 пар вопросов и параграфов test_taskA.csv, имеющие вид: paragraph_id, question_id, paragraph, question.

В предоставленных тренировочных и тестовых данных релевантные вопросы класса 1 были случайно выбраны из собранных вопросов и ответов. Нерелевантные примеры класса 0, составленные человеком, были получены случайным выбором вопроса к другому параграфу по той же теме. Нерелевантные вопросы класса 0, заданные компьютером, в тренировочных данных отсутствуют. Участникам необходимо самим генерировать такие вопросы для достижения лучшего качества. Также, несмотря на то, что целевая переменная target принимает два значения 0 и 1, в качестве предсказаний можно отправлять вещественные числа.

Решением задачи является .csv файл на основе test_taskA.csv, с заполненным полем target. Файл с решением задачи должен иметь следующий вид: paragraph_id, question_id, target.

Пример решения на Python

Описание метрики ROC-AUC

Материалы соревнования

Text

Оставим в трейне только те параграфы, что есть в тесте:

```
for.resample <- train.data %>% filter(paragraph_id > max(test.data$paragraph_id))
train.data %<>% filter(paragraph_id <= max(test.data$paragraph_id))
```

Баланс классов после удаления неиспользуемых параграфов 0.3158898.

Обрабатываем текст. Удаляем стоп-слова и цифры, приводим к нижнему регистру, делаем стемминг:

```
text_modify <- function(txt_, sw_ = c(), stem_ = FALSE) {
  if (length(sw_) > 0) {
    txt_ %<>% removeWords(sw_)
  }

  txt_ %<>% str_to_lower() %>%
```

```
    str_replace_all('ё', 'е') %>%
    str_replace_all('\\(', ' ( ') %>%
    str_replace_all('\\)', ' ) ') %>%
    str_replace_all('[:digit:]', ' ') %>%
    removePunctuation() %>%
    str_replace_all("\\s+", ' ')

  if (stem_ == TRUE) {
    # позже разобраться с кодировкой
    # txt_ %<>%  enc2utf8() %>%
    #   system(command = 'mystem/mystem.exe -cl',
    #          intern = TRUE,
    #          ignore.stdout = FALSE,
    #          ignore.stderr = FALSE,
    #          wait = TRUE,
    #          input = .,
    #          show.output.on.console = TRUE,
    #          minimized = FALSE,
    #          invisible = TRUE) %>%
    #   str_replace_all('[{}]', '') %>%
    #   str_replace_all('(\\|[^ ]+)', '') %>%
    #   str_replace_all('\\?', '') %>%
    #   str_replace_all('\\s+', ' ')
    txt_ %<>%
      stemDocument('russian')
  }

  return(txt_)
}

sw.url <- 'https://raw.githubusercontent.com/stopwords-iso/stopwords-ru/master/stopwords-ru.txt'
sw <- readr::read_csv(sw.url, col_names = F)$X1
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character()
## )
```

```
train.data$paragraph[1] # оригинал
```

```
## [1] "В отличие от рыб, земноводные (амфибии) и пресмыкающиеся (рептилии или гады) уже имеют два круга кровос
```

```
text_modify(train.data$paragraph[1]) # обработка
```

```
## [1] "в отличие от рыб земноводные амфибии и пресмыкающиеся рептилии или гады уже имеют два круга кровообра
```

```
text_modify(train.data$paragraph[1], sw) # обработка и удаление стоп слов
```

```
## [1] "в отличие рыб земноводные амфибии пресмыкающиеся рептилии гады имеют круга кровообращения сердце тре
```

```
text_modify(train.data$paragraph[1], sw, T) # обработка, удаление стоп слов и стемминг
```

```
## [1] "в отлич рыб земноводн амфиб пресмыка рептил гад имеют круг кровообращен сердц трехкамерн появля межпре
```

```
# отмодифим текстовые данные с полной обработкой
train.data$paragraph %<>% text_modify(sw, T)
train.data$question %<>% text_modify(sw, T)
```

```
test.data$paragraph %<>% text_modify(sw, T)
test.data$question %<>% text_modify(sw, T)
```

Model

Подготовим данные для построения модели:

```
X <- train.data %>%
  select(paragraph_id, par_len, ques_len, len_ratio, jac_sim, cos_sim, jac_dist, cos_dist, subst)
y <- train.data$target
X_pred <- test.data %>%
  select(paragraph_id, par_len, ques_len, len_ratio, jac_sim, cos_sim, jac_dist, cos_dist, subst)
X$paragraph_id <- as.factor(X$paragraph_id)
X_pred$paragraph_id <- as.factor(X_pred$paragraph_id)
X <- ds_toSparseMatrix(X)
X_pred <- ds_toSparseMatrix(X_pred)
```

Параметры модели:

```
k <- 4 #
param <- list(
  eta = 0.2/k,
  nround = 50*k,
  max_depth = 4,
  colsample_bytree = 0.7,
  subsample = 0.7,
  min_child_weight = 6,
  gamma = 4,
  tree_method = 'auto',
  eval_metric = 'auc',
  objective = 'binary:logistic'
)
```

Кроссвалидация и построение модели:

```
# cv.res <- xgb.cv(data = X, label = y, boosting = 'dart',
#                  params = param, nrounds = param$nround, nfold = 5, verbose = 1L)

model <- xgboost(
  data = X,
  label = y,
  params = param,
  boosting = 'dart',
  nrounds = param$nround,
  print_every_n = 50,
  early_stopping_rounds = 100
)
```

```
## [1]  train-auc:0.981410
## Will train until train_auc hasn't improved in 100 rounds.
##
## [51] train-auc:0.990911
## [101]    train-auc:0.991994
## [151]    train-auc:0.992723
## [200]    train-auc:0.993437
```

Предикт и запись в файл:

```
sample.submsission$prediction <- predict(model, X_pred)
readr::write_csv(sample.submsission, 'data/res.csv')
```