Mineraração de dados em um conjunto de transações de uma padaria

Detalhes

Universidade Federal de Santa Maria (UFSM)
Centro de Tecnologia (CT)
Disciplina de Mineração de dados (ELC1098)
Professor Dr. Joaquim Assunção
Autores: Bruno Perussatto & Gabriel Vinicius Schmitt Caetano
Trabalho prático 1
peso: 6

Etapas do processo

Seleção

Partimos da premissa que já recebemos os dados corretamente selecionados e ignoramos esta etapa.

Pré-processamento

Para o pré-processamento, apenas uma pequena correção manual no arquivo e a limpeza dos dados foram necessários para o objetivo do trabalho.

Usando um formatador json identificamos e corrigimos um erro na gramática do arquivo adicionando uma vírgula no final da linha 737. Carregamos o arquivo json com a lib jsonlite para o script R.

```
json <- fromJSON('padaria_trab.json')
```

Limpeza dos dados (cleaning)

Para a limpeza dos dados executamos um script para filtrar os produtos específicos e obter a lista de tipos únicos ("Café", "Pão", "Presunto", "Queijo", "Pastel", "Doce", "Refri") da lista de compras.

```
products = unique(unlist(json[2]))
get_first_word <- function(x) strsplit(x, " ")[[1]][1]
types <- unique(sapply(products, get_first_word))
```

Transformação

Considerando que será feito o uso do algoritmo a priori para encontrar as regras de associação, transformamos a lista de transações em uma matriz transação x produto.
Para isso criamos duas matrizes (**mat_p** para produtos únicos e **mat_t** para tipos de produtos) contendo o número de transações em linhas e o número de produtos/tipos como colunas totalmente preenchida com zeros.

```
mat_p <- matrix(0, nrow = nrow(json), ncol = length(types))
mat_t <- matrix(0, nrow = nrow(json), ncol = length(types))
```

Definimos o nome das colunas de **mat_p** conforme os produtos únicos identificados e as colunas de **mat_t** conforme os tipos de produtos identificados.

```
colnames(mat_p) <- products
colnames(mat_t) <- types
```

E então preenchemos as matrizes com os valores conforme as transações do arquivo importado. Para cada transação, as colunas dos produtos comprados recebem o valor 1, os demais são mantidos como 0 conforme definido na criação da matriz.

```
for (i in 1:nrow(json)) {
 p <- json$produtos[[i]]
 mat_p[i,p] <- 1
 pt <- unique(sapply(p, get_first_word))
 mat_t[i,pt] <- 1
}
```

Assim obtemos duas matrizes uma com:
[quantidade de compras] linhas x [quantidade de produtos específicos] colunas

e outra com**:**

[quantidade de compras] linhas x [quantidade de tipos de produtos únicos] colunas
com valores 0 ou 1, que permitirá a aplicação do algoritmo *apriori*

Mineração de dados

Nessa etapa usamos o algoritmo apriori com o parâmetro minlen = 2 para ignorar associações entre conjuntos vazios e ajustando os parâmetros de suporte e confiança para encontrar os resultados esperados.
Após alguns testes chegamos aos seguintes resultados:

```
rules_p = apriori(mat_p, parameter = list(supp = 0.06, conf = 0.2, minlen = 2))
rules_p <- sort(rules_p, by = "confidence", decreasing = TRUE)
```

encontramos apenas 6 regras válidas para **produtos específicos** ordenadas pela confiança

```
    lhs                    rhs                 support    confidence coverage  lift     count
[1] {Pão Francês}       => {Queijo Mussarela}  0.13008130 0.5714286  0.2276423 1.802198 16
[2] {Presunto Perdigão} => {Pão Francês}       0.06504065 0.5000000  0.1300813 2.196429  8
[3] {Queijo Mussarela}  => {Pão Francês}       0.13008130 0.4102564  0.3170732 1.802198 16
[4] {Presunto Sadia}    => {Queijo Mussarela}  0.06504065 0.4000000  0.1626016 1.261538  8
[5] {Pão Francês}       => {Presunto Perdigão} 0.06504065 0.2857143  0.2276423 2.196429  8
[6] {Queijo Mussarela}  => {Presunto Sadia}    0.06504065 0.2051282  0.3170732 1.261538  8
```

Dessa forma identificamos que as 5 principais regras deste conjunto são:

1. quem compra Pão Francês tem grande chance de comprar Queijo Mussarela
2. quem compra Presunto Perdigão tem grande chance de comprar Pão Francês
3. quem compra Queijo Mussarela tem grande chance de comprar Pão Francês
4. quem compra Presunto Sadia tem grande chance de comprar Queijo Mussarela
5. quem compra Pão Francês tem grande chance de comprar Presunto Perdigão

E o produto específico mais influente 1 para 1 é o Pão Francês => Queijo Mussarela.

Para **tipos** de produtos únicos encontramos 6 regras válidas também

```
rules_t = apriori(mat_t, parameter = list(supp = 0.2, conf = 0.4, minlen = 2))
rules_t <- sort(rules_t, by = "confidence", decreasing = TRUE)
```

```
    lhs            rhs         support   confidence coverage  lift     count
[1] {Pão}       => {Queijo}    0.2845528 0.6250000  0.4552846 1.260246 35
[2] {Presunto}  => {Pão}       0.2520325 0.5849057  0.4308943 1.284704 31
[3] {Queijo}    => {Pão}       0.2845528 0.5737705  0.4959350 1.260246 35
[4] {Presunto}  => {Queijo}    0.2439024 0.5660377  0.4308943 1.141355 30
[5] {Pão}       => {Presunto}  0.2520325 0.5535714  0.4552846 1.284704 31
[6] {Queijo}    => {Presunto}  0.2439024 0.4918033  0.4959350 1.141355 30
```

Dessa forma identificamos que as 5 principais regras deste conjunto são:

1. quem compra Pão tem grande chance de comprar Queijo
2. quem compra Presunto tem grande chance de comprar Pão
3. quem compra Queijo tem grande chance de comprar Pão
4. quem compra Presunto tem grande chance de comprar Queijo
5. quem compra Pão tem grande chance de comprar Presunto

E o tipo de produto mais influente 1 para 1 é o Pão => Queijo.

Para as regras que implicam a compra de doce foram aplicados os seguintes parâmetros com filtro sobre os doces específicos identificados:

```
rules_p = apriori(mat_p, parameter = list(supp = 0.03, conf = 0.1, minlen = 2))
filtered = subset(rules_p, rhs %in% "Doce Goiabada" | rhs %in% "Doce Leite" | rhs
%in% "Doce Amendoim")
```

```
rules_t = apriori(mat_t, parameter = list(supp = 0.1, conf = 0.2, minlen = 2))
filtered = subset(rules_t, rhs %in% "Doce")
inspect(filtered)
```

Para **doces específicos** chegamos nos seguintes resultado:

```
    lhs                  rhs             support    confidence coverage  lift      count
[1] {Café Melita}     => {Doce Leite}    0.03252033 0.2352941  0.1382114 2.2262443 4
[2] {Presunto Sadia}  => {Doce Amendoim} 0.03252033 0.2000000  0.1626016 1.6400000 4
[3] {Pão Francês}     => {Doce Amendoim} 0.03252033 0.1428571  0.2276423 1.1714286 4
[4] {Refri - Fanta}   => {Doce Goiabada} 0.04065041 0.2941176  0.1382114 2.0098039 5
[5] {Café Melita}     => {Doce Goiabada} 0.03252033 0.2352941  0.1382114 1.6078431 4
[6] {Queijo Mussarela} => {Doce Goiabada} 0.04065041 0.1282051  0.3170732 0.8760684 5
```

Para o **tipo "Doce"** chegamos nos seguintes resultados:

```
    lhs            rhs     support   confidence coverage  lift      count
[1] {Café}      => {Doce} 0.1382114 0.3953488  0.3495935 1.0571284 17
[2] {Refri}     => {Doce} 0.1463415 0.3600000  0.4065041 0.9626087 18
[3] {Presunto}  => {Doce} 0.1219512 0.2830189  0.4308943 0.7567678 15
[4] {Pastel}    => {Doce} 0.1626016 0.3389831  0.4796748 0.9064112 20
[5] {Pão}       => {Doce} 0.1463415 0.3214286  0.4552846 0.8594720 18
[6] {Queijo}    => {Doce} 0.1463415 0.2950820  0.4959350 0.7890235 18
```