Text Mining

Primeiros Conceitos

Sumário

Objetivo

O objetivo deste trabalho é apresentar um procedimento para transformar um conjunto de textos, dados não estruturados, em uma estrutura de dados estruturados.

Manipulação de Texto

Manipulação de Texto

O R apresenta em sua instação incial uma série de funções para manipulação de valores do tipo **(**caracter). Por exemplo:

Função	**Ação**
nchar	número de caracteres em um texto
tolower	converte para minusculo
toupper	converte para maiusculo
casefold	converte para maiusculo ou minusculo
chartr	alteracão de caracter
substr	seleciona um trecho do texto
strsplit	separa o texto em palavras

Manipulação de Texto

A função **nchar** conta o número de caracter em um texto.

```
> texto <- "Palavras amigas são doce como o mel ..."
> nchar (texto)
[1] 39
> texto <- c("banana", "laranja", "maça")
> nchar (texto)
[1] 6 7 4 >
```

O espaços em branco são contados.

Manipulação de Texto

As funções **tolower** e **toupper** convertém um texto para minusculo ou maiusculo respectivamente.

```
> texto <- "Vai ter com a formiga, ó preguiçoso;"
> tolower (texto)
[1] "vai ter com a formiga, ó preguiçoso;"
> toupper (texto)
[1] "VAI TER COM A FORMIGA, Ó PREGUIÇOSO;"
>
```

Manipulação de Texto

A função **casefold** pode converter um texto tanto para minusculo como para maiusculo.

```
> casefold (texto, upper=FALSE)
[1] "vai ter com a formiga, ó preguiçoso;"
> casefold (texto, upper=TRUE)
[1] "VAI TER COM A FORMIGA, Ó PREGUIÇOSO;"
>
```

O padrão da função casefold é converte para minusculo.

Manipulação de Texto

A função **chartr** substitui um determinado caracter por um outro.

```
> texto <- "Bênçãos há sobre a cabeça do justo"
> chartr("êáãç","eaac",texto)
[1] "Bencaos ha sobre a cabeca do justo"
```

A função **chartr** tem a seguinte assinatura `chartr(old, new , x )`.

- `old` - caracteres a serem substituídos.
- `new` - caracteres substitutos.
- `x` - texto.

1. old e new devem ter o mesmo número de caracter.
2. o primerio caracter de old será substituído pelo primerio de new, assim por diante.

Manipulação de Texto

A função **substr** pode extrair um trecho do texto ou substituir um treço de texto por outro.

```
> texto <- c("um", "dois", "três")
> substr(texto,2,2)
[1] "m" "o" "r"
> substr(texto,2,2) <- "#"
> texto
[1] "u#" "d#is" "t#ês"
```

A função **substr** tem a assinatura `substr(x, start , stop)`.

- `start` - caracter inicial.
- `new` - caracter final.
- `x` - texto.

start e new devem ser valores inteiros.

Manipulação de Texto

A função **strsplit** sepepara um texto em palavras.

```
> texto <- "Mulher virtuosa quem a achará?
O seu valor muito excede ao de rubis."
> strsplit(texto,)
[[1]]
[1] "Mulher" "virtuosa" "quem" "aachará?" "O" "seu" "valor"
[9] "muito" "excede" "ao" "de" "rubis."endverbatim
```

A função **strsplit** tem a assinatura `strsplit(x, split)`.

- `split` - caracter que separa as palavras.
- `x` - vetor com os textos.

1. A função strsplit retorna uma lista, cada componente contém as palavras de um elemento do vetor texto.
2. A função unlist transforma uma lista em vetor.

Manipulação de Texto

A função **gsub** substitui caracteres por outros em um texto.

```
> texto <- c("Melhor é a repreensão franca do
que o amor encoberto.")
> texto
[1] "Melhor é a repreensão franca do que o amor encoberto."
> gsub("[aeiou]","!",texto)
[1] "M!lh!r é ! r!pr!!nsã! fr!nc! d! q!! ! !m!r !nc!b!rt!."
```

A função **gsub** tem a assinatura `gsub(pattern, replacement, x)`.

- `pattern` - padrão a ser pesquisado.
- `replacement` - o valor que substituirá o padrão.
- `texto` - o texto a ser análisado.

Manipulação de Texto - Exercícios

Amor é fogo que arde sem se ver

Amor é fogo que arde sem se ver; É ferida que dói e não se sente; É um contentamento descontente; É dor que desatina sem doer;

É um não querer mais que bem querer; É solitário andar por entre a gente; É nunca contentar-se de contente; É cuidar que se ganha em se perder;

É querer estar preso por vontade; É servir a quem vence, o vencedor; É ter com quem nos mata lealdade.

Mas como causar pode seu favor Nos corações humanos amizade, Se tão contrário a si é o mesmo Amor?

Luís de Camões

Manipulação de Texto - Exercícios

1. Quantos caracteres tem esse poema?
2. Converta texto para minuscula e use a função cat para conferir as alterações.
3. substitua todos do caracteres acentuados pelo correspondentes sem acento. Use a a função cat para conferir as alterações.
4. Substitua a pontuação (;-.,?\n -) por espaço.
5. Crie uma lista com as palavras desse poema e tranforme essa lista em um vetor.
6. Retire os elementos iguais a " " ou `\n`.
7. Qual a palvra mais frequente?
8. Qual a maior palavra?
9. Use o arquivo `tm_exe_aprs_1.R` que tem o poema já digitado.

Text Mining - Introdução

Text Mining - Introdução

Text Mining, mineração de texto, é o processo de análise de coleções de materiais textuais para capturar conceitos e temas chave e descobrir relacionamentos e tendências ocultas sem exigir que você conheça as palavras ou termos precisos que os autores usaram para expressar esses conceitos.

Fonte:IBM Knowledge Center. Disponível em: <https://www.ibm.com/support/knowledgecenter/en/SS3RA7_17.1.0/ta_guide_ddita/textmining/shared_entities/tm_intro_tm_defined.html>. Acesso em: 11/08/17.

Text Mining - Introdução

No contexto de mineração de textos, tem-se os seguintes conceitos:

- **Documento**: menor elememto de um base de texto.
- **Corpora**: conjunto de documentos a ser analisado.
- **Bag of words**: conjunto de palavras que formam um texto.
- **Dicionário**: conjunto de palavras que ocorrem em todos os documentos.
- **Tokens**: menor elemento com significado em um documento.

Formalmente, uma corpora é representada como um conjunto de dados definidos por n documentos (doc), como em

$$\Psi = \{doc_1, doc_2, \ldots, doc_n\}$$

Cada documento passa a ser definido por um conjunto de d termos na forma

$$doc_i = \{wte_{i1}, wte_{i2}, \ldots, wte_{ij}, \ldots, wte_{id}\}$$

onde wte_{ij} assume algum valor. Em uma representação binária, os valores são 1, se wte_i pertence ao doc_i, e 0 no caso contrário. 0 wte_{i1} pode assumir outros valores, por exemplo a frequencia do termo wte_{i1} no doc_i.

Text Mining - Introdução

Seja a corpora { I need you, I want you, I dore you }. A representação binária Ψ para essa corpora é

	dore	I	need	want	you
doc_1	0	1	1	0	1
doc_2	0	1	0	1	1
doc_3	1	1	0	0	1

É com base em Ψ que se realiza outras análise.

Text Mining - Introdução

Processo para obter Ψ:

- Análise lexical
- Eliminação de termos irrelevantes
- Redução do termo ao radical
- Representação

Text Mining - Introdução

A corpora abaixo será usada para ilustrar todas as fase para obter Ψ.

doc_1	Ambiente agradável e tranquilo. Comida e música com qualidade. Adoramos o Filé à Parmegiana.
doc_2	O Filé à Parmegiana da Cidade. Ambiente agradável e qualidade no atendimeto.
doc_3	O Filé à Parmegiana da com fritas é uma delícia.

Text Mining - Introdução

Análise Lexical tem como objetivo gerar a primeira lista de **tokens** ou de **termos**.

- Eliminar pontuação, acentos e dígitos
- Uniformizar a capitalização das letras (todas maiúsculas ou todas minúsculas).

doc_1	ambiente|agradável|e|tranquilo|comida|e|música| com|qualidade|adoramos|o|filé|à|parmegiana.
doc_2	o|filé|à|parmegiana|da|cidade|ambiente|agradável| e|qualidade|no|atendimeto
doc_3	o|filé|à|parmegiana|com|fritas|é|uma|delícia

Eliminação de Termos Irrelevantes retira da análise termos que isolados não trazem sentido.

- Artigos, preposições, pronomes, numerais, conjuçoes e advébios.
- Compõem a lista de *stopwords*.
- São excluídas da análise

doc_1	ambiente|agradável|tranquilo|comida|música| qualidade|adoramos|filé|parmegiana.
doc_2	filé|parmegiana|cidade|ambiente| agradável|qualidade|atendimeto
doc_3	filé|parmegiana|fritas|delícia

Redução do Termo ao Seu Radical baseia-se no fato que termos com origem no mesmo radical são considerados iguais.

doc_1	ambient|agrad|tranquil|com|músic|qualidad| ador|fíl|parmegian.
doc_2	fil|parmegian|cidad|ambient|agrad|qualidad| atend
doc_3	fil|parmegian|frit|delíci

Dicionário pode ser composto a partir os dos termos reuzidos ao seu radical.

ador	agrad	ambient	atend	cidad
com	delici	fil	frit	music
parmegian	qualidad	tranquil		

Text Mining - Introdução

Representação vetorial binária do corpus é o Ψ

wte_{ij}	doc_1	doc_2	doc_3
ador	1	0	0
agrad	1	1	0
ambient	1	1	0
atend	0	1	0
cidad	0	1	0
com	1	0	0
delici	0	0	1
fil	1	1	1
frit	0	0	1
music	1	0	0
parmegian	1	1	1
qualidad	1	1	0
tranquil	1	0	0

Text Mining - Aplicação

Utilizar o R para gerar uma núvem de palavras a partir de 30 posts do blog **Eight to Late**. Os arquivos com os texto, no fomato txt ,estão na pasta **tm**.

setwd () - determina a ára de trabalho

Fonte: Eight to Late. Disponível em: <https://eight2late.wordpress.com/>. Acesso em 11/08/17.

Text Mining - Aplicação

Os pacotes necessários são

- tm - pacote de mineração de texto.
- SnowballC - necessário para redução de termos ao radical.
- wordcloud - gera uma núvem de palavras.

install.package () - instala pacotes.
library () - prepara pacotes para uso.

Text Mining - Aplicação

Para gerar a **Corpora** ou **Corpus** é são utilizadas as seguintes funções do pacote **tm**:

- Corpus() - gerar um corpus a partir de um conjunto de textos.
- DirSource () - localiza os textos a serem analizados.

Text Mining - Aplicação

Exemplo de código:

```
# Carrega o pacote tm
library(tm)
library(SnowballC)
library(wordcloud)
#Cria Corpus
aux <-"C:/Users/raucelio.rccv/Desktop/tm/textos"
docs <- Corpus(DirSource(aux))
```

Text Mining - Aplicação

Fases a serem executado são:

- Análise lexical
- Eliminação de termos irrelevantes
- Redução do termo ao radical
- Representação

Text Mining - Aplicação

As fases do pré-processamento realizada pelas seguintes funçoes dos pacotes **tm** e **SnowballC**.

Função	**Ação**
removeNumbers	remove dígitos
removePunctuation	remove pontuação
stripWhitespace	remove espaços em branco
content_transformer	transforma conteúdo
removeWords	remove palavras
stemDocument	reduz ao radical

Text Mining - Aplicação

Exemplo de código para análise lexical:

```
toSpace <- content_transformer(function(x, pattern)
             { return (gsub(pattern, " ", x))})
docs <- tm_map(docs, toSpace, "-")
docs <- tm_map(docs, toSpace, ":")
docs <- tm_map(docs, toSpace, "’")
docs <- tm_map(docs, toSpace, "’")
docs <- tm_map(docs, toSpace, "-")
#Remover pontuação - substituir pontuação por ""
docs <- tm_map (docs, removePunctuation)
#Tudo maiuscular
docs <- tm_map (docs, content_transformer (tolower))
```

Text Mining - Aplicação

Retirar termos irrelevantes:

```
#remove stopwords usando a lista padrão em tm
docs <- tm_map (docs, removeWords, stopwords ("english"))
```

Text Mining - Aplicação

Reduz o termo ao radical:

```
Docs <- tm_map (docs, stemDocument)
```

Text Mining - Aplicação

Cria uma repesentação

```
dtm <- DocumentTermMatrix (docs)
```

Text Mining - Aplicação

A nuvem de palavras representa a frequencia das palavras no corpora

```
freq <- colSums (as.matrix(dtm))
# defina uma semente que garante um
# olhar consistente sobre as nuvens
set.seed (42)
#limit palavras especificando min frequency
wordcloud(names(freq),freq,min.freq=70,
          colors=brewer.pal(6,"Dark2))
```

Text Mining - Aplicação

A nuvem de palavras final