Package ‘smdc’

February 20, 2015

Type Package

Title Document Similarity

Version 0.0.2

Date 2013-02-16

Author Masaaki TAKADA

Maintainer Masaaki TAKADA <tkdmah@gmail.com>

Description This package provides similarity among documents.

License BSD

Depends proxy,tm

NeedsCompilation no

Repository CRAN

Date/Publication 2013-02-15 19:45:47

R topics documented:

smdc-package .................................................. 2
conv2Freq ......................................................... 3
normalize ........................................................ 4
simDic ............................................................ 4
simDoc ............................................................ 6
simSum ............................................................ 7
simSyn ............................................................ 8
uniform ........................................................... 9

Index 11
**smdc-package**  

**Document Similarity**

**Description**

This package provide functions that calculate similarity among documents.

**Details**

<table>
<thead>
<tr>
<th>Package</th>
<th>smdc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>0.0.2</td>
</tr>
<tr>
<td>Date</td>
<td>2013-02-16</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
</tbody>
</table>

**Author(s)**

Masaaki TAKADA  
Maintainer: Masaaki TAKADA <tkdmah@gmail.com>

**Examples**

```r
# Load text mining package 'tm' for English.  
# 'RMecab' is available for Japanese.  
# install.packages('tm')  
library('tm')

# Read corpus data.  
crudeDir <- system.file("texts", "crude", package = "tm")  
crude <- Corpus(DirSource(crudeDir))  
docMatrix1 <- t(as.matrix(DocumentTermMatrix(crude)))  
acqDir <- system.file("texts", "acq", package = "tm")  
acq <- Corpus(DirSource(acqDir))  
docMatrix2 <- t(as.matrix(DocumentTermMatrix(acq)))

# Create score dictionary.  
words <- unique(c(rownames(docMatrix1), rownames(docMatrix2)))  
scores <- runif(length(words), -1, 1)  
dict <- data.frame(word = words, score = scores)

# Calculate similarity.  
sim1 <- simDoc(docMatrix1, docMatrix2, norm = TRUE)  
sim2 <- simDic(docMatrix1, docMatrix2, dict, norm = TRUE)  
sim <- simSyn(list(sim1, sim2), c(0.5, 0.5))
```
**conv2Freq**

*Conversion from Matrix to Frequency Distribution*

**Description**

This function converts a matrix to a frequency distribution.

**Usage**

```r
conv2Freq(tmpmatrixL, wordclassL, breaks)
```

**Arguments**

- `tmpMatrix` Matrix.
- `wordClass` Classes of each row of matrix.
- `breaks` Class division vector.

**Value**

Frequency distribution matrix.

**Author(s)**

Masaaki TAKADA

**Examples**

```r
### The function is currently defined as
function (tmpMatrix, wordClass, breaks)
{
    freqDist <- matrix(0L, nrow = length(breaks) - 1, ncol = ncol(tmpMatrix))
    for (tmp in rownames(tmpMatrix)) {
        cat <- wordClass[tmp]
        if (!is.na(cat)) {
        }
    }
    colnames(freqDist) <- colnames(tmpMatrix)
    if (!is.null(names(breaks))) {
        rownames(freqDist) <- names(breaks)[2:length(breaks)]
    }
    return(freqDist)
}
```
normalize  

Normalization of Similarity Matrix

Description
This function normalizes similarity matrix.

Usage
normalize(sim)

Arguments
sim  
Similarity matrix.

Value
Normalized similarity matrix.

Author(s)
Masaaki TAKADA

Examples

## The function is currently defined as
function (sim)
{
  meanVec <- apply(sim, 1, mean, na.rm = TRUE)
  sdVec <- apply(sim, 1, sd, na.rm = TRUE)
  sim <- t(scale(t(sim), meanVec, sdVec))
  return(sim)
}

simDic  

Document Similarity using Dictionary

Description
This function calculates the similarity between documents and documents by using dictionary.

Usage
simDic(docMatrix1, docMatrix2, scoreDict, breaks = seq(-1, 1, length = 11), norm = FALSE, method = "cosine").
Arguments

- `docMatrix1`: Document matrix whose rows represent feature vector of one document. This matrix must satisfy the following: `colnames(docMatrix1)` denote feature names, `rownames(docMatrix1)` denote document names, every element is numerical.

- `docMatrix2`: Document matrix whose rows represent feature vector of one document. This matrix must satisfy the following: `colnames(docMatrix2)` denote feature names, `rownames(docMatrix2)` denote document names, every element is numerical.

- `scoreDict`: Dictionary matrix which converts features to numbers. This matrix must be a k * 2 matrix: 1st column represents features and 2nd column represents corresponding numbers. Similarity is calculated according to the number.

- `breaks`: Range vector of frequency distribution. Each element must be ascending order.

- `norm`: Whether normalize similarity matrix or not.

- `method`: Method to calculate similarity.

- `scoreFunc`: Function of scoring from dictionary.

Value

Similarity Matrix whose rows represent documents of `docMatrix1` and whose columns represent documents of `docMatrix2`. This matrix is an n * m matrix where n=ncol(docMatrix1) and m=ncol(docMatrix2), and satisfy the following: `rownames(returnValue)=colnames(docMatrix1)`, `colnames(returnValue)=colnames(docMatrix2)`.

Author(s)

Masaaki TAKADA

Examples

```r
## The function is currently defined as
function (docMatrix1L, docMatrix2L, scoreDictL, breaksL, normL = FALSEL, methodL = "cosineL", scoreFuncL = mean) {

library("proxy")
wordsL <- unique(rbind(matrix(rownames(docMatrix1)), matrix(rownames(docMatrix2))))
wordsL <- words[order(words)]
wordScoresL <- rep(NA, length(words))
for (i in 1:length(words)) {
  condL <- (scoreDict[, 1] == words[i])
  valueL <- scoreDict[cond, 2]
  if (length(value) != 0) {
    wordScores[i] <- scoreFunc(value, na.rm = TRUE)
  }
}
names(breaks) <- cut(breaks, breaks)
wordClassL <- cut(wordScores, breaks)
names(wordClass) <- words
docFreq1L <- conv2Freq(docMatrix1, wordClass, breaks)
docFreq2L <- conv2Freq(docMatrix2, wordClass, breaks)
colnames(docFreq1L) <- paste("r_{-}", colnames(docMatrix1), sep = ")
```
colnames(docFreq2) <- paste("c_", colnames(docMatrix2), sep = "")
sim <- as.matrix(simil(t(cbind(docFreq1, docFreq2)), method = method))[colnames(docFreq1),
  colnames(docFreq2)]
rownames(sim) <- colnames(docMatrix1)
colnames(sim) <- colnames(docMatrix2)
if (norm) {
  sim <- normalize(sim)
}
return(sim)
}

simDoc

**Document Similarity**

**Description**
This function calculates the similarity between documents and documents.

**Usage**
simDoc(docMatrix1, docMatrix2, norm = FALSE, method = "cosine")

**Arguments**
- **docMatrix1**: Document matrix whose rows represent feature vector of one document. This matrix must satisfy the following: colnames(docMatrix1) denote feature names, rownames(docMatrix1) denote document names, every element is numerical.
- **docMatrix2**: Document matrix whose rows represent feature vector of one document. This matrix must satisfy the following: colnames(docMatrix2) denote feature names, rownames(docMatrix2) denote document names, every element is numerical.
- **norm**: Whether normalize similarity matrix or not.
- **method**: Method to calculate similarity.

**Value**
Similarity Matrix whose rows represent documents of docMatrix1 and whose columns represent documents of docMatrix2. This matrix is n * m matrix where n=ncol(docMatrix1) and m=ncol(docMatrix2), and satisfy the following: rownames(returnValue)=colnames(docMatrix1), colnames(returnValue)=colnames(docMatrix2).

**Author(s)**
Masaaki TAKADA
Examples

```r
## The function is currently defined as
function (docMatrix1L, docMatrix2L, norm = FALSE, method = "cosine")
{
  library("proxy")
  exDocMatrix1 <- uniform(docMatrix1L, docMatrix2L)
  exDocMatrix1L <- exDocMatrix1[[1]]
  exDocMatrix2L <- exDocMatrix1[[2]]
  colnames(exDocMatrix1L) <- paste("r_", colnames(docMatrix1L),
    sep = "")
  colnames(exDocMatrix2L) <- paste("c_", colnames(docMatrix2L),
    sep = "")
  sim <- as.matrix(simil(t(cbind(exDocMatrix1L, exDocMatrix2L)),
    method = method))[colnames(exDocMatrix1L), colnames(exDocMatrix2L)]
  rownames(sim) <- colnames(docMatrix1L)
  colnames(sim) <- colnames(docMatrix2L)
  if (norm) {
    sim <- normalize(sim)
  }
  return(sim)
}
```

---

**simSum**

*Summary of Document Similarity*

**Description**

This function summarize the calculation of similarity.

**Usage**

```r
simSum(sim)
```

**Arguments**

- `sim` Similarity matrix.

**Value**

List of similar documents to each documents. The number of list equals to ncol(sim).

**Author(s)**

Masaaki TAKADA
Examples

```r
## The function is currently defined as
function (sim)
{
  results <- rep(0, ncol(sim))
  names(results) <- colnames(sim)
  scores <- rep(0, ncol(sim))
  for (i in 1:ncol(sim)) {
    scores[i] <- max(sim[, i])
    results[i] <- rownames(sim)[which.max(sim[, i])]
  }
  summary <- as.list(NULL, length = nrow(sim))
  for (i in 1:nrow(sim)) {
    cond <- results == rownames(sim)[i]
    summary[[i]] <- names(which(cond[order(-scores)]))
  }
  names(summary) <- rownames(sim)
  return(summary)
}
```

---

**simSyn**

**Synthesis of Document Similarity**

**Description**

This function synthesize the similarity.

**Usage**

`simSyn(sims, weight)`

**Arguments**

- **sims**: List of similarity matrix.
- **weight**: Weight vector of similarity matrix.

**Value**

Weighted sum of similarity matrix

**Author(s)**

Masaaki TAKADA
Examples

```r
## The function is currently defined as
function (sims, weight)
{
  len <- length(sims)
  if (len != length(weight)) {
    stop(message = "different lengths between sims and weight")
  }
  for (i in 2:len) {
    sim <- sim + weight[i] * sims[i]
  }
  return(sim)
}
```

**Description**

This function uniform the row number of two similarity matrices according to each row names.

**Usage**

`uniform(matrix1, matrix2)`

**Arguments**

- `matrix1` Similarity matrix.
- `matrix2` Similarity matrix.

**Value**

List of two uniformed similarity matrices.

**Author(s)**

Masaaki TAKADA

**Examples**

```r
## The function is currently defined as
function (matrix1, matrix2)
{
  words <- unique(rbind(matrix(rownames(matrix1)), matrix(rownames(matrix2)))))
  words <- words[order(words)]
  exMatrix1 <- matrix(0, nrow = length(words), ncol = ncol(matrix1))
```
exMatrix2 <- matrix(0, nrow = length(words), ncol = ncol(matrix2))
rownames(exMatrix1) <- words
rownames(exMatrix2) <- words
colnames(exMatrix1) <- colnames(matrix1)
colnames(exMatrix2) <- colnames(matrix2)
for (word in rownames(matrix1)) {
  exMatrix1[word] <- matrix1[word]
}
for (word in rownames(matrix2)) {
  exMatrix2[word] <- matrix2[word]
}
return(list(exMatrix1, exMatrix2))
Index

*Topic \textasciitilde kwd1
  convRfreq, 3
  normalize, 4
  simDic, 4
  simDoc, 6
  simSum, 7
  simSyn, 8
  uniform, 9

*Topic \textasciitilde kwd2
  convRfreq, 3
  normalize, 4
  simDic, 4
  simDoc, 6
  simSum, 7
  simSyn, 8
  uniform, 9

*Topic package
  smdc-package, 2
conv2Freq, 3
normalize, 4
simDic, 4
simDoc, 6
simSum, 7
simSyn, 8
smdc (smdc-package), 2
smdc-package, 2
uniform, 9