Information Retrieval

Learning Outcomes:

1. Read text documents in R and compute the document-term matrix.
2. Use Cosine similarity measure to find similar documents.
3. Given a query, find similar documents
4. Apply k-means to find similar documents.
5. Ranking effectiveness using Recall, Precision and average Precision.
6. Further, computation of NDCG to evaluate the relevancy of search results.
7. Page Rank Computation in R and demonstrate using Pig script.

TF-IDF of documents

Let us take a small dataset to compute the TF-IDF of a document-term matrix

Suppose,

D1: I learnt to compute inverse of a matrix

D2: Matrix inverse is easy to compute

D3: term frequency-inverse document frequency matrix

{"compute, document, easy, frequency, inverse, learnt, matrix, term" }

	compute	document	easy	frequency	inverse	learnt	matrix	term	Total
D1	1	0	0	0	1	1	1	0	4
D2	1	0	1	0	1	0	1	0	4
D3	0	1	0	2	1	0	1	1	6

Use any of the below definitions to compute the weighted matrix.

Term frequency (TF)

a. the number of times that term t occurs in document d
b. Number of times term t appears in a document / (Total number of terms in the document).
c. tf(t,d) = 1 + log $f_{t,d}$
d. for more: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

In R, TF: it is the number of times a term 't' occurs in a document 'd'. If Normalize = t then option 'b' is used.

Inverse document frequency (IDF)

Raw term frequency suffers from a critical problem: all terms are considered equally important when it comes to assessing relevancy on a query. In fact, certain terms have little or no discriminating power in determining relevance.

a. IDF: is a measure of how important a term is. In order to weigh down highly frequent terms that may appear of little importance and scale up the rare terms.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$ (number of documents where the term t appears)

a. the denominator can be : $1 + |\{d \in D : t \in d\}|$
b. log(1 + (max(t,d)/{d ɛ D: t ɛ d)
c. for more: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Log base: 2, 10 or 'e'

In R, it is base 2.

Let us compute TF when normalize = T

	compute	document	easy	frequency	inverse	learnt	matrix	term
D1	0.25	0	0	0	0.25	0.25	0.25	0
D2	0.25	0	0.25	0	0.25	0	0.25	0
D3	0	0.166667	0	0.333	0.166667	0	0.166667	0.166667

Denominator:

Compute	document	Easy	Frequency	inverse	learnt	matrix	term
2	1	1	1	3	1	3	1

TF-IDF, however in R, normalize = F is default.

	compute	document	easy	frequency	Inverse	learnt	matrix	term
D1	0.146241	0	0	0	0	0.396241	0	0
D2	0.146241	0	0.396241	0	0	0	0	0
D3	0	0.26416	0	0.528321	0	0	0	0.26416

In R, DTM <- weightTfIdf(DocTermMatrix(in frequency format), normalize= T)

Refer to different weighting schemes: https://cran.r-project.org/web/packages/tm/tm.pdf

Cosine Similarity of documents

Example:

D1: Raj loves datamining more than Rahul loves datamining
D2: Ravi likes datamining more than Raj loves datamining

Datamining	2	2
Raj	1	1
Rahul	1	0
Ravi	0	1
Like	0	1
Love	2	1
More	1	1
Than	1	1

In 2 dimensions

(Raj, loves, datamining, more, than, Rahul, Ravi, likes)

D1: 1 2 2 1 1 1 0 0

D2: 1 1 2 1 1 0 1 1

To decide how close these two texts are to each other by calculating one function of those two vectors, namely the cosine of the angle between them.

Cos(d1,d2) = d1 . d2/ (|| d1|| . ||d2||)

||d1|| = sqrt($1^2 + 2^2 + 2^2 + 1^2 + 1^2 + 1^2 + 0 + 0$) = 3.4641

||d2|| = sqrt($1^2 + 1^2 + 2^2 + 1^2 + 1^2 + 0 + 1^2 + 1^2$) = 3.1622

d1.d2 = 9

Cos(d1, d2) = 0.82

$\Theta = \cos^{-1}(0.82)$ = 35degrees.

In R,

Distmatrix <- dist(dt_matrix, method = "cosine").

Let us take the first example and suppose the query q = “compute inverse”

Now, to find the similar documents, let us first compute the tf-idf of the query.

The dimension of the TFIDF is 3x8.

q=[1 0 0 0 1 0 0 0]

tf of q = [0.5 0 0 0 0.5 0 0 0]

TFIDF of data

	compute	document	easy	frequency	inverse	learnt	matrix	term
D1	0.146241	0	0	0	0	0.396241	0	0
D2	0.146241	0	0.396241	0	0	0	0	0
D3	0	0.26416	0	0.528321	0	0	0	0.26416

TFIDF of query

	Compute	document	easy	frequency	inverse	learnt	matrix	term
D1	0.146241 *0.5 =0.073	0	0	0	0	0	0	0
D2	0.146241*0.5=.073	0	0	0	0	0	0	0
D3	0	0	0	0	0	0	0	0

Cosine (q, D1) = (0.14624 * 0.0731)/ Sqrt((0.073)^2) *sqrt(0.146241)^2) =

Cosine(q, D2) = (0.14624 * 0.0731)/ Sqrt((0.073)^2) *sqrt(0.146241)^2) =

Exercise: Compute the TFIDF and find which document is similar to the given query.

Document 1: The game of life is a game of everlasting learning
Document 2: The unexamined life is not worth living
Document 3: Never stop learning
Q = life learning

References:

https://courses.cs.washington.edu/courses/cse573/12sp/lectures/17-ir.pdf

https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/

```
########################################
# Using k-means in order to cluster the data
########################################
```

Use the last week's (Lab01) data about speeches.

```
m <- as.matrix(dt_matrix)
rownames(m) <- 1:nrow(m)
### Normalize the data
norm_eucl <- function(m) m/apply(m, MARGIN=1, FUN=function(x) sum(x^2)^.5)
m_norm <- norm_eucl(m)
### cluster into 3 clusters
cl <- kmeans(m_norm, 3)
cl
#cl$centers
# To check how many documents are fallen in a particular cluster
cl$size
table(cl$cluster)
cl$cluster
```

Ranking effectiveness: Precision, Recall and average precision of ranked documents

A is set of relevant documents,
B is set of retrieved documents

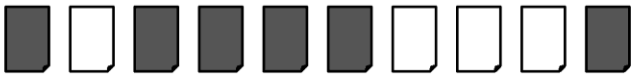
	Relevant	Non-Relevant
Retrieved	$A \cap B$	$\overline{A} \cap B$
Not Retrieved	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$

$$Recall = \frac{|A \cap B|}{|A|}$$

$$Precision = \frac{|A \cap B|}{|B|}$$

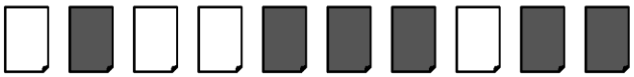
 = the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2



Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

Average precision:

$$\text{Ranking } \#1\text{: } (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

$$\text{Ranking } \#2\text{: } (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$$

Reference: Information retrieval in practice.

Normalized discounted cumulative gain:

It measures the performance of a recommendation system based on the graded relevance of the recommended entities. It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the entities. This metric is commonly used in information retrieval and to evaluate the performance of web search engines.

$$\mathrm{NDCG}_q = Z_q \sum_{j=1}^{L} \frac{2^{r_q(j)} - 1}{\log(1+j)}$$

Example

- 10 ranked documents judged on 0-3 relevance scale:

  3, 2, 3, 0, 0, 1, 2, 2, 3, 0
- For simplicity, let us compute discounted gain: 1/log(rank)

  3, 2/1, 3/1.59, 0, 0, 1/2.59, 2/2.81, 2/3, 3/3.17, 0

  = 3, 2, 1.89, 0, 0, 0.39, 0.71, 0.67, 0.95, 0
- DCG:

  3, 5, 6.89, 6.89, 6.89, 7.28, 7.99, 8.66, 9.61, 9.61

- Perfect ranking:

  3, 3, 3, 2, 2, 2, 1, 0, 0, 0
- ideal DCG values:

  3, 6, 7.89, 8.89, 9.75, 10.52, 10.88, 10.88, 10.88, 10
- NDCG values (divide actual by ideal):

  1, 0.83, 0.87, 0.76, 0.71, 0.69, 0.73, 0.8, 0.88, 0.88
  - NDCG <= 1 at any rank position

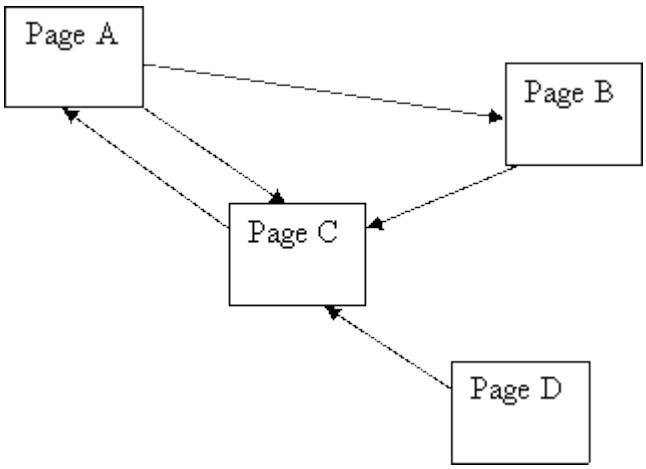
Reference:

https://www.kaggle.com/wiki/NormalizedDiscountedCumulativeGain

https://courses.cs.washington.edu/courses/cse573/12sp/lectures/17-ir.pdf

############## Page Rank #############

In short PageRank is a "vote", by all the other pages on the Web, about how important a page is. A link to a page counts as a vote of support. If there's no link there's no support



Incidence matrix

	A	B	C	D
A	0	0.5	0.5	0
B	0	0	1	0
C	1	0	0	0
D	0	0	1	0

g <- graph(c(1, 2, 1, 3, 2, 3, 3, 1, 4, 3), directed=TRUE)

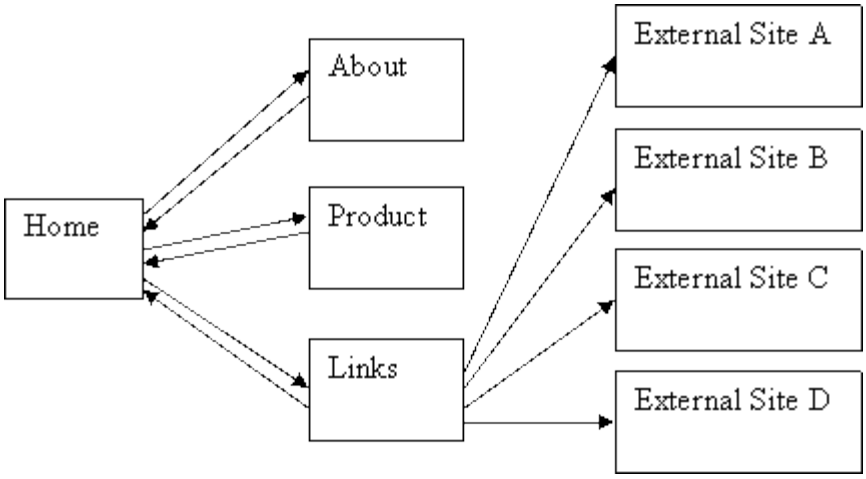
m <-page_rank(g)

m

```
$vector
[1] 0.3725269 0.1958239 0.3941492 0.0375000
```

References: http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm

Exercise:



Answer:

```
[1] 0.32523730 0.14330914 0.14330914 0.14330914 0.08161176 0.08161176 0.08
161176
```