Latent Semantic Indexing application of Ranked Retrieval model

Azza M. N. Abdalghani

Department of Mathematics and Geo-sciences, University of Trieste

Information Retrieval course
Final Project $\sim$ July 2021

Table of Contents

Latent Semantic Indexing

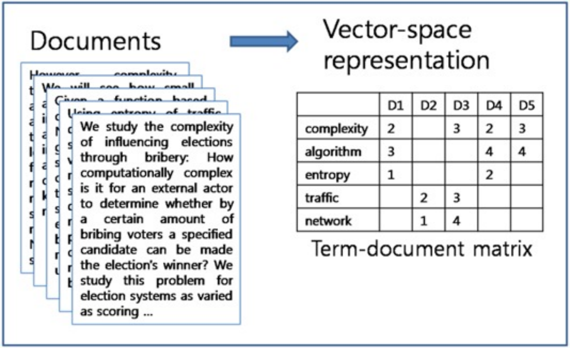
- Proposed by Deerwester, 1990 .
- **Vector Space** approach for modeling documents .
- Has achieved up to 30% better retrieval performance than lexical searching techniques .
- Advantage : Overcome the problems with term matching retrieval, mainly :
  - **Polysemy** : a term has more than one meaning .
  - **Synonymy** : same topic use different vocabulary.
- Based on a mathematical technique termed **Singular Value Decomposition (SVD)**

LSI : Main Idea

- **Documents** and **Queries** are represented as vectors in **t**-dimensional space.
- Map documents and queries into a lower dimensional space : **SVD** and **Dimensionality Reduction**
- Map each **term** into some **concepts**.
- Map **concepts** into **documents**.
- Compute **Similarity** between the documents and queries and return the top-k high ranked documents

Recall : Vector Space Model

- Represent documents as **vectors** of terms $d = (t_1, t_2, ..., t_n)$ where $t_i (1 \leqslant i \leqslant n)$ denoting the single or multiple occurrences of term i in document $d \Longrightarrow$ **Term-Document Matrix** .
- Thus, each term corresponds to a dimension in the space.



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Recall : Singular Value Decomposition

- Goal : **Decompose** the term-document matrix into a product of matrices.

$$M = USV^T$$

  - M : original matrix
  - U : a term by r matrix
  - S : a singular value matrix $(r \times r)$
  - V^T : a document by r matrix
  - where r is the rank of the term-document matrix

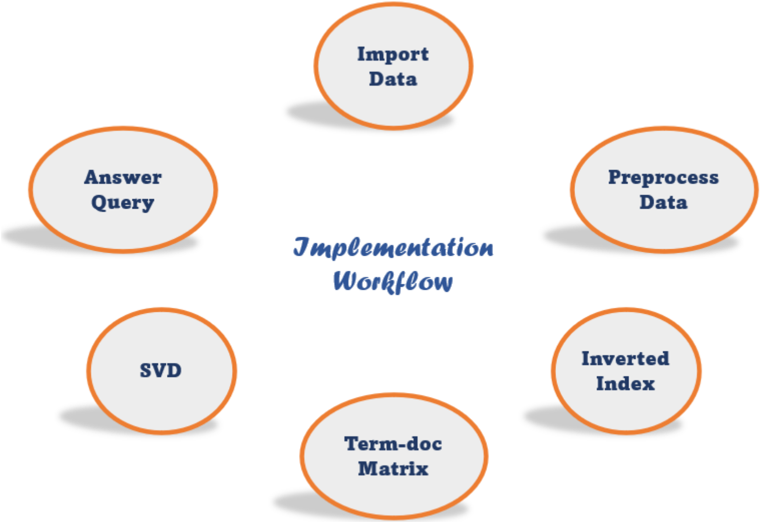
Table of Contents

Data Overview

- It is a collection that consists of articles extracted from the magazine Time.
- It includes 5 main files :
  - **README** : Explanation of files
  - **time.all** : the documents
  - **time.que** : the queries
  - **time.rel** : relevance assessments
  - **time.stp** : list of stop words

Table of Contents

Implementation Workflow



Importing and Preprocessing data

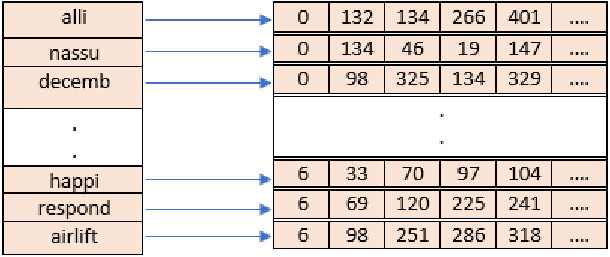
- Import **TIME** dataset : Read TIME.ALL and TIME.STP files.
- Preprocessing data :
  - Tokenization
  - Remove stop-words
  - Remove Punctuation marks
  - Stemming : PorterStemmer()

Term	Term after preprocessing
EMERGING	emerg
CANCELLATION	cancel
FRANCE	franc
DELIVERY	deliveri
ANGLOUS	anglou

- In total, there is 422 articles(documents) and 14952 terms.

Build Inverted-Index

- Mapping terms to the related documents.
- Build a **dictionary** of terms, connected to a list of **documents ids** that contain the corresponding term.



Build Term-Document Matrix

- Build a matrix of **M x N**, where rows are terms and columns are documents
- Elements of term-document matrix are **tf-idf** weights

$$TF - IDF_{t,d} = tf_d * idf_t$$

where $tf_d = \frac{\#term\_occurrences}{|d|}$ and $idf_t = \log \frac{N}{df_t + 1}$

	0	1	…	412	….	421
alli	0.003114	0.0	….	0.001233	…	0.0
nassau	0.005462	0.0	….	0.0	….	0.0
decemb	0.001149	0.0	….	0.0	….	0.0
help	0.002240	0.0	….	0.000887	….	0.0
dictatori	0.0	0.0	….	0.0	….	0.008709
….	….		….	…	….	…

[14952 rows x 422 columns]

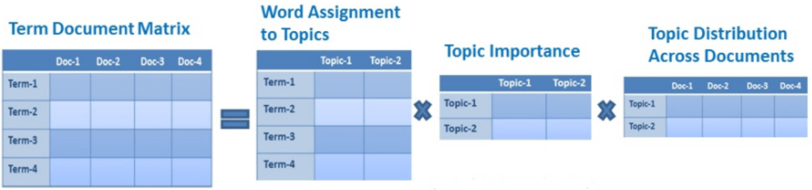
Matrix Decomposition : SVD

- First, decompose the term-document Matrix into 3 matrices :

$$M = U.\Sigma.V^T$$

  - U : Left Singular vectors, called **term-concept matrix**
  - Σ : contains Singular values where each value represents a weight of a concept, called **concept matrix**
  - V^T : Right Singular vectors, called **document-concept matrix**
- Then, reduce document dimensions by choosing the number of important concepts **k**.

$$M = U_{Mxk}.\Sigma_{kxk}.V^T_{kxN}$$



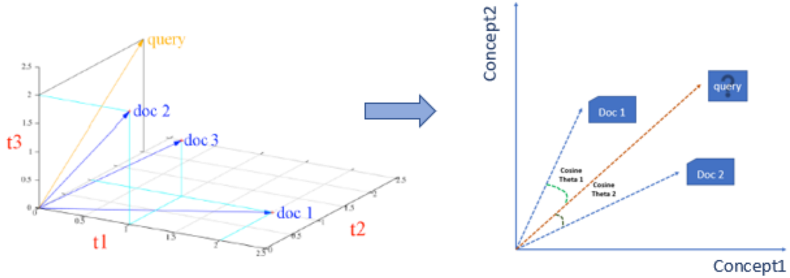
How to select the value of k?

- Determining the number of concepts (k) is fully dependent on the corpus.
- In this project, many random values for k were taken, and generate the results for each one of them.
- Based on the provided queries and relevance assessments, best k that gives better performance was chosen .

$$k = 215$$

In order to answer Query :

- Preprocessing query
- Represent query as a vector in terms dimensions
- Remap query in concept space : $\hat{q} = \Sigma^{-1} U^T q$
- Compute the similarity between each document and the query and assign a score for each document



How to compute Similarity?

By **Cosine Similarity** : cosine angle between two vectors (query and document), can be computed as the inner product of the two vectors normalized to both have length 1 :

$$sim(v_1, v_2) = \vec{v_1}.\vec{v_2}$$

Thus, the score of similarity of each document with query can be :

$$score(q, d_j) = \vec{q}.\vec{d_j}$$

Finally, Scores get sorted , higher score means more relevant document (collinear with query, e.i point to the same direction), return the top k ranked documents.

Table of Contents

Results

- For testing our implementation, a set of queries were taken with their corresponding answers from **time.que** and **time.rel**.
- A list of relevant documents and their scores were retrieved as an answer to each query.
- Finally, as a performance metric, **R-Precision** was computed for each query and at the end compute the **Average R-Precision (ARP)** for the set of queries.

$$\Longrightarrow ARP = \sum_{n} RP_n$$

where n is number of queries

Experimental Results

```
#69  70 100 115 121 139 159 194 210 224 234 309 379 388
query_1 = ["THE BAATH (RENAISSANCE) PARTY FOUNDED BY MICHEL AFLAK, WHICH HAS GAINED CONTROL OF SYRIA AND IRAQ AND AIMS TO UNITE 
M_decomposition = LSI(M, 215)
relevant_docs_1 = answer_query(query_1,M_decomposition,terms,13)
print(relevant_docs_1)
```

```
[[377, 0.9074748457722953], [386, 0.7648667535593023], [309, 0.5648373281418607], [194, 0.5316742024010941], [115, 0.4927082195
7710237], [159, 0.4702934871295355], [210, 0.3849730137900559], [100, 0.35748794415113494], [121, 0.3377839712475548], [70, 0.3
3242266716334884], [234, 0.326790886761082], [139, 0.291839556073581], [224, 0.21395117397533003]]
```

It is observed that 11 of the retrieved documents are relevant

$$\rightarrow R - Precision_{q_1} = \frac{11}{13} = 0.84615$$

```
#12  61 155 156 242 269 339 358
query_6 = ["CONTROVERSY BETWEEN INDONESIA AND MALAYA ON THE PROPOSED FEDERATION OF MALAYSIA, WHICH WOULD UNITE FIVE TERRITORIES
M_decomposition = LSI(M, 215)
relevant_docs_6 = answer_query(query_6,M_decomposition,terms,7)
print(relevant_docs_6)
```

```
[[61, 0.6785103101149405], [358, 0.6234520083076952], [156, 0.60534689432005], [155, 0.5143008007463842], [303, 0.4370222262841
8297], [269, 0.36305323227580283], [339, 0.3591404003504791]]
```

It is observed that 6 of the retrieved documents are relevant

$$\rightarrow R - Precision_{q_6} = \frac{6}{7} = 0.85714$$

Experimental Results

```
#39  22  73 173 189 219 265 277 360 396
query_3 = ["COALITION GOVERNMENT TO BE FORMED IN ITALY BY THE LEFT-WING SOCIALISTS, THE REPUBLICANS, SOCIAL DEMOCRATS, AND CHRIS
M_decomposition = LSI(M, 215)
relevant_docs_3 = answer_query(query_3,M_decomposition,terms,8)
print(relevant_docs_3)
```

```
[[277, 0.7656766098060409], [360, 0.7491591189751454], [394, 0.6779197006972426], [265, 0.5678370396247605], [219, 0.4940555543
654217], [189, 0.44341632087842253], [134, 0.40568774969918425], [22, 0.39216211767542447]]
```

$$\rightarrow R - Precision_{q_3} = \frac{6}{8} = 0.75$$

```
#27 272 295 306
query_8 = ["BRITISH PROPOSAL FOR NEW HIGH LEVEL NEGOTIATIONS WITH RUSSIA OR A FOUR-POWER SUMMIT MEETING ."]
M_decomposition = LSI(M, 215)
relevant_docs_8 = answer_query(query_8,M_decomposition,terms,3)
print(relevant_docs_8)
```

```
[[151, 0.4218138316235267], [111, 0.41488162259210043], [306, 0.31681114968467844]]
```

$$\rightarrow R - Precision_{q_8} = \frac{1}{3} = 0.33333$$

For the 8 queries, we obtain $ARP = 0.64247$

Table of Contents

Conclusion

- LSI attempts to overcome the common problems of search engines (Synonymy and Polysemy)
- LSI designed to uncover the latent semantic structure of a document space by building a semantic space.
- Documents and queries are represented in this semantic space and a similarity score is measured.
- **Drawbacks**:
  - SVD is expensive to compute
  - Requires re-computing SVD when new documents arrive

Greetings

Thank you for your attention ☺