Information retrieval

Flexible querying methods

Clovis Galiez

Laboratoire Jean Kuntzmann, Statistiques pour les sciences du Vivant et de l'Homme

September 21, 2022

Today's outline

- Short summary of last lecture
- Embeddings
- Ranking

What to remember from last time?

Remember...

What are the main points you remember from last lectures?

What to remember from last time?

Remember...

What are the main points you remember from last lectures?

- Web IR is split in distinct steps:
  - Gathering and indexing data from the web (**crawling**)
  - Retrieving documents relevant to a query
  - Ranking the valid answers according to relevance
- The involved data is **big**

  Need efficient representation and algorithms
- Boolean querying are not flexible
- Easy to integrate information of tokens in the model **at no cost**
- tf-idf does not solve the synonymy issue

The vector space model and the latent semantics

Representing documents as vectors in $\mathbb{R}^T$

From binary presence/absence...

	tok 1	tok 2	tok 3	tok 4	tok 5	...
	election	president	crazy	united	United States	...
doc 1	1	1	0	0	1	...
doc 2	0	1	1	0	1	...
doc 3	1	1	1	0	1	...
...	...	...	...	...	...	...

Representing documents as vectors in $\mathbb{R}^T$

...to real vector space.

	tok 1	tok 2	tok 3	tok 4	tok 5	...
	election	president	crazy	united	United States	...
doc 1	0.01	0.02	0	0	0.006	...
doc 2	0	0.013	0.001	0	0.001	...
doc 3	0.0031	0.008	0.0043	0	0.0021	...
...	...	...	...	...	...	...

Vector representation of a document

A document can be represented by a vector of the fraction information associated to each of its token:

$$D_t = \frac{\#\text{ t in D}}{\#\text{ tokens in D}} \times [-\log_2(\frac{\#\text{doc including token } t}{\#\text{docs}})]$$

Vector representation of a document

A document can be represented by a vector of the fraction information associated to each of its token:

$$D_t = \frac{\text{\# t in D}}{\text{\# tokens in D}} \times [-\log_2(\frac{\text{\#doc including token } t}{\text{\#docs}})]$$

$||\vec{D}||_1$ carries the total information carried by a document:

Vector representation of a document

A document can be represented by a vector of the fraction information associated to each of its token:

$$D_t = \frac{\text{\# t in D}}{\text{\# tokens in D}} \times [-\log_2(\frac{\text{\#doc including token } t}{\text{\#docs}})]$$

$||\vec{D}||_1$ carries the total information carried by a document:

- low if the document contains only common tokens
- average if the document contains few exceptional tokens
- high if the document contains only exceptional items

Querying a set of vector

Represent the query the same way:

$$Q_t = \frac{\text{\# t in Q}}{\text{\# tokens in Q}} \times I(t)$$

How to retrieve documents related to the query?

Querying a set of vector

Represent the query the same way:

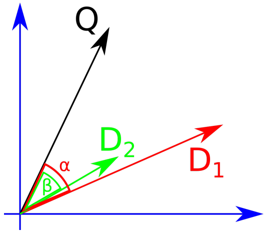
$$Q_t = \frac{\text{\# t in Q}}{\text{\# tokens in Q}} \times I(t)$$

How to retrieve documents related to the query?
Dot product:

$$\vec{D} \cdot \vec{Q} = \sum_t D_t . Q_t$$

The higher the dot product, the more informative tokens $\vec{Q}$ and $\vec{D}$ share...
and the more relevant should be the D with respect to the query Q.

The cosine similarity



Consine similarity

$$\text{cosim}(\vec{D}, \vec{Q}) = \frac{\vec{D} \cdot \vec{Q}}{||\vec{D}||_2 . ||\vec{Q}||_2}$$

A flexible querying system?

With the vector space model, information of the tokens are now automatically taken into account.
Does it solve the synonymous problem?

Example

Query: `result elections United States`
Doc title: "White House election: live results!"

A flexible querying system?

With the vector space model, information of the tokens are now automatically taken into account.
Does it solve the synonymous problem?

Example

Query: `result elections United States`
Doc title: "White House election: live results!"

Can we work directly from the data?

Embeddings

From TF-IDF to Embeddings

TF-IDF allows to have a vector representation of documents in the "space" of tokens.

From TF-IDF to Embeddings

TF-IDF allows to have a vector representation of documents in the "space" of tokens.

Embeddings

Embeddings aim at reducing space of tokens to less dimension in an useful way: a token will live in a small dimensional space ($D_E = 300$) such that semantically similar token lie close to each other in space.

Embedding from data: introductory example

After the next *house*, you turn right.

Embedding from data: introductory example

After the next *house*, you turn right.
After the next *building*, you turn right.

Embedding from data: introductory example

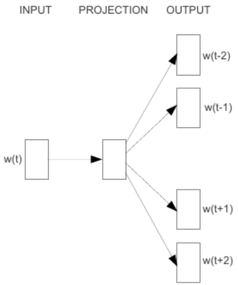
After the next *house*, you turn right.
After the next *building*, you turn right.

Hypothesis

The context of a word is giving its meaning. So similar context $\approx$ similar meaning.

Word2vec: predict the context of a token

The core idea of word2vec is to learn a vector representation allows to predict the context of the token. Thereby, tokens appearing in similar context will be encoded closely in the vector space.



Skip-gram

[Mikolov, Tomas; et al. (2013)]

word2vec's semantic relations

The word2vec embeddings have interesting semantic features[1].

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[1]Note that `GloVe` is better at this

What algorithms for querying with embeddings?

Optimized search algorithm based on reverse sparse index does not work anymore.

Documents are now represented as a dense matrix.

What algorithms for querying with embeddings?

⚠ Optimized search algorithm based on reverse sparse index does not work anymore.

Documents are now represented as a dense matrix.

How to query this matrix?

What algorithms for querying with embeddings?

Optimized search algorithm based on reverse sparse index does not work anymore.

Documents are now represented as a dense matrix.

How to query this matrix?

Since semantically similar tokens are close in space, we need to find nearest neighbors in the D_E-dimensional space.

What algorithms for querying with embeddings?

⚠ Optimized search algorithm based on reverse sparse index does not work anymore.

Documents are now represented as a dense matrix.

How to query this matrix?

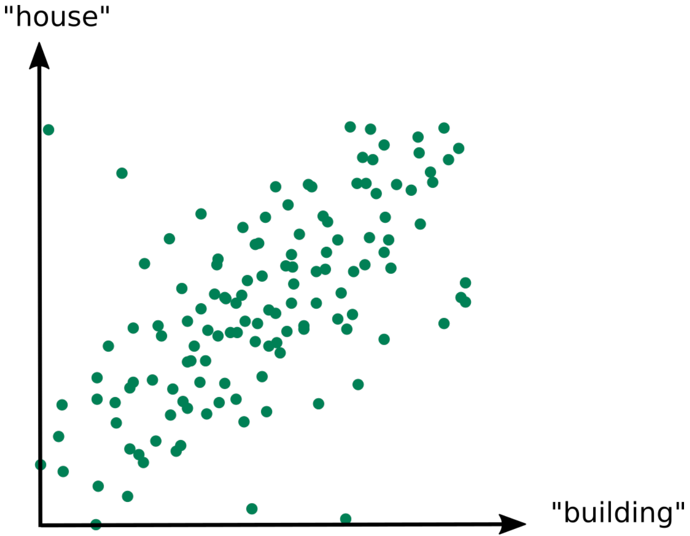
Since semantically similar tokens are close in space, we need to find nearest neighbors in the D_E-dimensional space.

Locally sensitive hashing (LSH)

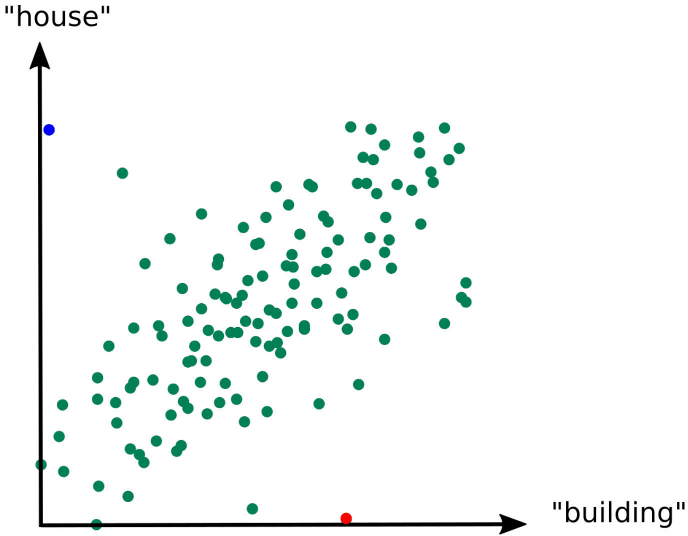
- Hash the query vector with k (discrete) hash functions: $\mathcal{O}(1)$
- Look for documents sharing same hash codes:: $\mathcal{O}(1)$
- Can compute exact scalar product against all retrieved documents.

Latent semantics: understand dimensionality reduction

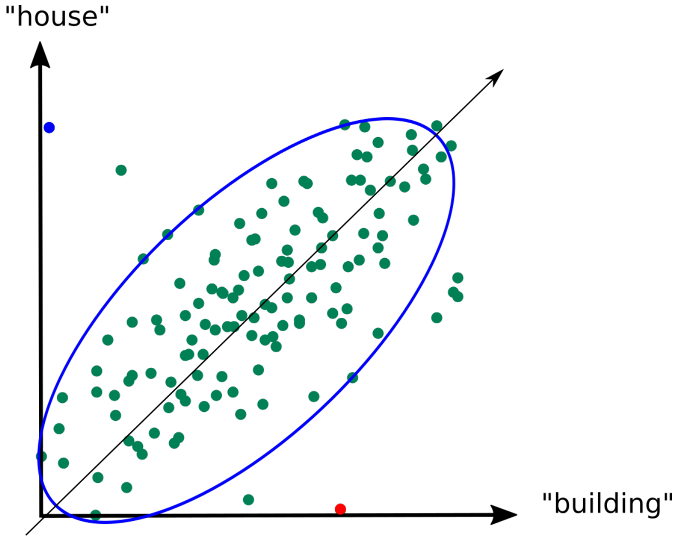
Linear version of embeddings



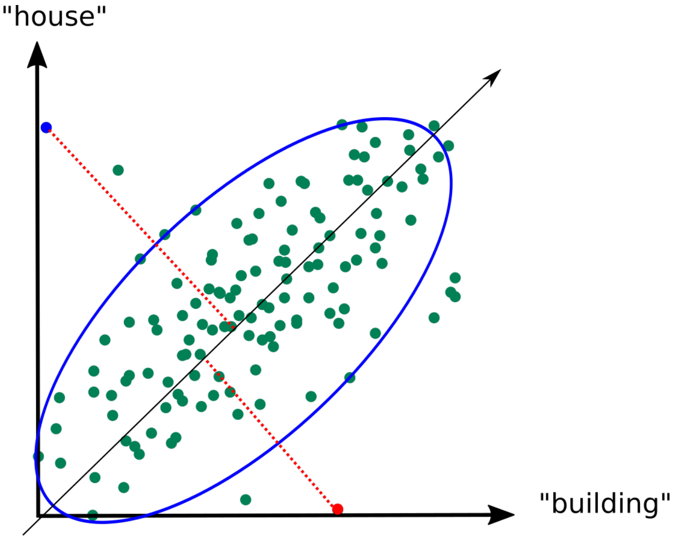
Linear version of embeddings



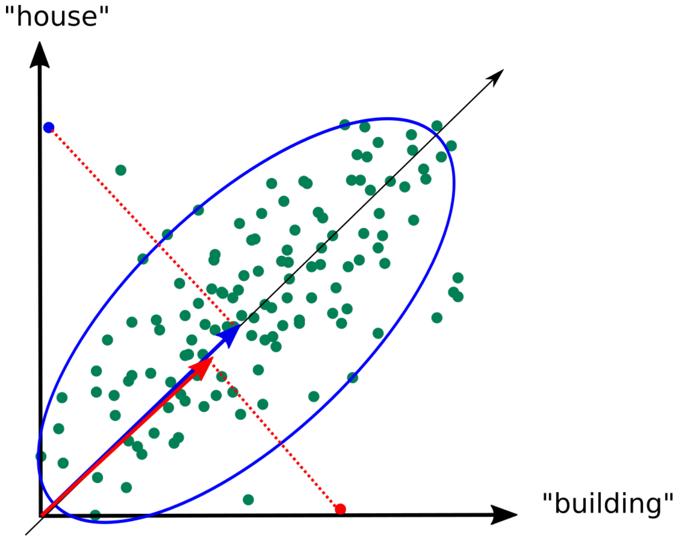
Linear version of embeddings



Linear version of embeddings



Linear version of embeddings



Special structure of the data: correlations

In practice a tf matrix looks like:

Interlude

Video

Special structure of the data: correlations

In practice a tf matrix looks like:

Interlude

Video

We observe...

A block structure.

Special structure of the data: correlations

In practice a tf matrix looks like:

Interlude

Video

We observe...

A block structure.

PCA

PCA "discover" automatically those blocks

Data compresssion approach: Low rank approximation

Theorem (Eckart–Young–Mirsky)

The best[a] r-rank approximation $\hat{M}$ of M is given by the projection on the subspace formed by the eigenvectors of $M^\top M$ corresponding to the r biggest eigen values.

[a] In the sense minimizing $||M - \hat{M}||_F = \sum_{i,j}(m_{i,j} - \hat{m}_{i,j})^2$

The projection to the low rank space (columns of $V^\top$ in SVD decomposition $M = U\Sigma V^\top$) collapse similar (i.e. *correlated*) tokens to the same component. This space is called the **Latent semantic space**.

Code example for word2vec

```
>>> import gensim.downloader
>>> word2vec_vectors = gensim.downloader.load('word2vec-google-news-300')
[==================] 100.0% 1662.8/1662.8MB downloaded
>>> word2vec_vectors.get_vector("house")
array([ 1.57226562e-01, -7.08007812e-02,  5.39550781e-02, -1.89208984e-02,
    9.17968750e-02,  2.55126953e-02,  7.37304688e-02, -5.68847656e-02,
    ...
>>> word2vec_vectors.similarity("house","building")
0.4378754
>>> word2vec_vectors.similarity("house","dog")
0.25689757
>>> word2vec_vectors.similarity("house","happy")
0.11390656
```

Examples of biases in word2vec

```
>>> word2vec_vectors.similarity("man","father")
0.4201101
>>> word2vec_vectors.similarity("woman","mother")
0.60763067
>>> word2vec_vectors.similarity("man","smart")
0.09229658
>>> word2vec_vectors.similarity("woman","smart")
0.050040156
>>> word2vec_vectors.similarity("man","robber")
0.5585119
>>> word2vec_vectors.similarity("woman","robber")
0.45501366
>>> word2vec_vectors.similarity("mexican","thief")
0.12186743
>>> word2vec_vectors.similarity("american","thief")
0.036840104
```

Some open-source libraries

Some libs and features:

- Information Retrieval: Gensim (implements word2vec, fasttext and querying)
  - - : no transformer model
- NLP: Stanza (tokenization, named entity recognition)
- Embeddings: spaCy (in particular BERT)
- NLP+embeddings: Flair
  - - : few languages supported

Dealing with the *truth*

How to deal with the truth?

It is almost impossible to deal with truth judgment only from the document data.

However, we can assume that we trust information coming from *authorities* (well-known newspaper, official website, etc.).

How to deal with the truth?

It is almost impossible to deal with truth judgment only from the document data.

However, we can assume that we trust information coming from *authorities* (well-known newspaper, official website, etc.).

Idea

Rank the results of the querying system according to their authority.

How do we know who is the authority ?

How to deal with the truth?

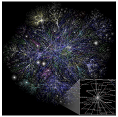
It is almost impossible to deal with truth judgment only from the document data.

However, we can assume that we trust information coming from *authorities* (well-known newspaper, official website, etc.).

Idea

Rank the results of the querying system according to their authority.

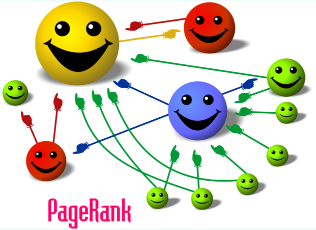
How do we know who is the authority ?

→ We extract it from the web structure  !

Authority and web structure

Who is the authority?

If you only represent the web by a graph where each node is a web page and each directed edge is an HTML link.

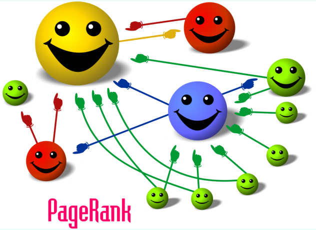


How would you recognize an authority?

Authority and web structure

Who is the authority?

If you only represent the web by a graph where each node is a web page and each directed edge is an HTML link.

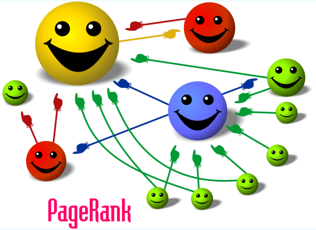


How would you recognize an authority?
The authority is higher when a node is pointed at (by other authorities).

Authority and web structure

Who is the authority?

If you only represent the web by a graph where each node is a web page and each directed edge is an HTML link.



How would you recognize an authority?
The authority is higher when a node is pointed at (by other authorities).

Imagine an algorithm able to detect/rank authorities.

PageRank formalization (simple version)

Random surfer model

Imagine a user having the following behavior clicking on random links on the Internet.

The more links leading to a page, the more chance (and the more times) the user visits the page.

PageRank formalization (simple version)

Random surfer model

Imagine a user having the following behavior clicking on random links on the Internet.

The more links leading to a page, the more chance (and the more times) the user visits the page.
After a loooong time, we measure the average number of times the user visited a given page P, we denote R_P.

Definition of the rank according to PageRank

We define the authority/ranking of a page by the R_P value.

PageRank algorithm (simple version)

Data: $A :=$ graph of the WWW $\quad A_{ij} \quad = \quad \begin{cases} \frac{1}{N_j} & \text{if link from } j \text{ to } i \\ 0 & \text{else} \end{cases}$

Result: Ranking of web pages

$R^{(0)} := S$;

repeat

$\quad R^{(i+1)} \leftarrow AR^{(i)}$

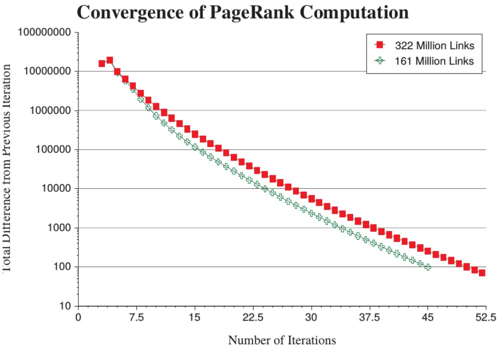
$\quad \delta \leftarrow ||R^{(i)} - R^{(i+1)}||_1$

until $\delta \leq \epsilon$;

Algorithm 1: simplified PageRank

Milestone of Google (algo designed by L. Page, Google co-founder), and drove the initial success of Google.

PageRank convergence



[L. Page, 98]

PageRank without sink effect

Sink effect

What if a page does not have any outgoing connection?

It will "trap" the user and have an artificially high rank.

PageRank without sink effect

Sink effect

What if a page does not have any outgoing connection?

It will "trap" the user and have an artificially high rank.

The random eager surfer

Imagine the user having now the following behavior[a]

- click on a random link on the current web page with probability $p(t)$
- or jump to a random web page on the Internet with probability $1 - p(t)$

[a] In the original paper by Page, the balance between the two events is given by its trap feeling: the more trapped it gets, the more likely the user will jump somewhere else.

Full PageRank

To avoid a *sink* effect, we introduce random jumps to a set of pages encoded in E.

Data: Graph of the WWW

Result: Ranking of web pages

$R^{(0)} := S$;

repeat

$R^{(i+1)} \leftarrow AR^{(i)}$

$d \leftarrow ||R^{(i+1)}||_1 - ||R^{(i+1)}||_1$

$R^{(i+1)} \leftarrow R^{(i+1)} + d.E$

$\delta \leftarrow ||R^{(i)} - R^{(i+1)}||_1$

until $\delta \leq \epsilon$;

Algorithm 2: PageRank

Full PageRank

Note that the vector E encodes the distribution of pages where the user is willing to jump to.

Full PageRank

Note that the vector E encodes the distribution of pages where the user is willing to jump to.

6 Personalized PageRank

An important component of the PageRank calculation is E – a vector over the Web pages which is used as a source of rank to make up for the rank sinks such as cycles with no outedges (see Section 2.4). However, aside from solving the problem of rank sinks, E turns out to be a powerful parameter to adjust the page ranks. Intuitively the E vector corresponds to the distribution of web pages that a random surfer periodically jumps to. As we see below, it can be used to give broad general views of the Web or views which are focussed and personalized to a particular individual.

We have performed most experiments with an E vector that is uniform over all web pages with

...

Such personalized page ranks may have a number of applications, including personal search engines. These search engines could save users a great deal of trouble by efficiently guessing a large part of their interests given simple input such as their bookmarks or home page. We show an example of this in Appendix A with the "Mitchell" query. In this example, we demonstrate that while there are many people on the web named Mitchell, the number one result is the home page of a colleague of John McCarthy named John Mitchell.

[L. Page, 98]

Summary

- Tf-Idf vector representation of a document
- Flexible vector queries (cosine similarity)
- Latent semantics (lower rank projection of the tf matrix)
- PageRank

Next lectures: can we make it?

- TP (Implemenation and experiments around IR systems)
  - Tokenizer
  - Tf-Idf matrix construction
  - Page Rank implementation
  - Mini-search engine
- (more) machine learning in IR

Another interpretation

Exercise

If M is a tf matrix and Q a binary vector over tokens, what does MQ represent?

Another interpretation

Exercise

If M is a tf matrix and Q a binary vector over tokens, what does MQ represent?

The fraction of occurrences of tokens of Q in each document.

Another interpretation

Exercise

If M is a tf matrix and Q a binary vector over tokens, what does MQ represent?

The fraction of occurrences of tokens of Q *in each document.*

If D is a binary vector over documents, what does $M^\top D$ represent?

Another interpretation

Exercise

If M is a tf matrix and Q a binary vector over tokens, what does MQ represent?

The fraction of occurrences of tokens of Q in each document.

If D is a binary vector over documents, what does $M^\top D$ represent?

The cumulated frequencies of each token in the corpus D.

Another interpretation

Exercise

If M is a tf matrix and Q a binary vector over tokens, what does MQ represent?

The fraction of occurrences of tokens of Q in each document.

If D is a binary vector over documents, what does $M^\top D$ represent?

The cumulated frequencies of each token in the corpus D.

If Q a binary vector over tokens, what does $M^\top MQ$ represent?

Another interpretation

Exercise

If M is a tf matrix and Q a binary vector over tokens, what does MQ represent?

The fraction of occurrences of tokens of Q in each document.

If D is a binary vector over documents, what does $M^\top D$ represent?

The cumulated frequencies of each token in the corpus D.

If Q a binary vector over tokens, what does $M^\top MQ$ represent?

The cumulated frequencies of tokens in the (virtual) corpus matching Q.

Another interpretation

Exercise

If M is a tf matrix and Q a binary vector over tokens, what does MQ represent?
The fraction of occurrences of tokens of Q in each document.
If D is a binary vector over documents, what does $M^\top D$ represent?
The cumulated frequencies of each token in the corpus D.
If Q a binary vector over tokens, what does $M^\top MQ$ represent?
The cumulated frequencies of tokens in the (virtual) corpus matching Q.

What does it mean that $M^\top MQ = \lambda.Q$?

Another interpretation

Exercise

If M is a tf matrix and Q a binary vector over tokens, what does MQ represent?
The fraction of occurrences of tokens of Q in each document.
If D is a binary vector over documents, what does $M^\top D$ represent?
The cumulated frequencies of each token in the corpus D.
If Q a binary vector over tokens, what does $M^\top MQ$ represent?
The cumulated frequencies of tokens in the (virtual) corpus matching Q.

What does it mean that $M^\top MQ = \lambda.Q$? What if λ is small? big?

Algebra theorem

Eigenvectors of $M^\top M$, $\vec{C}_i$ are orthogonal and form a basis of the token space.
We can define a new scalar product:

$$\vec{D'} = \sum \alpha_i \vec{C}_i$$
$$\vec{Q'} = \sum \beta_i \vec{C}_i$$

We can compare search documents matching query Q using $\vec{D'}.\vec{Q'} = \sum \alpha_i.\beta_i$ or cosim$(\vec{D'}, \vec{Q'})$:)