Indexing Vector Spaces Graphs Search Engines

Piero Molino
XY Lab

Relevance for New Publishing

Search Engines can be considered **publishers**

They select, filter, choose and sometimes create the content they give as result to queries

From the user point of view, the search process is a **blackbox**, like magic

Understanding it increases the **awareness of the risks** it has and could help bypassing or hacking it

Information Retrieval

Let a collection of **unstructured information** (set of texts)

Let an **information need** (query)

Objective: Find items of the collection that answers the information need

Representation

Simple/naive assumption

A text can be represented by the **words** it contains

Bag-of-words model

Bag of Words

"Me and John and Mary attend this lab"

Me:1

Bag of Words

"Me and John and Mary attend this lab"

Me:1 and:1

Bag of Words

"Me and John and Mary attend this lab"

Me:1 and:1 John:1

Bag of Words

"Me and John and Mary attend this lab"

Me:1 and:2 John:1

Bag of Words

"Me and John and Mary attend this lab"

Me:1 and:2 John:1 Mary:1

Bag of Words

"Me and John and Mary attend this lab"

Me:1 and:2 John:1 Mary:1 attend:1

Bag of Words

"Me and John and Mary attend this lab"

Me:1 and:2 John:1 Mary:1 attend:1 this:1

Bag of Words

"Me and John and Mary attend this lab"

Me:1 and:2 John:1 Mary:1 attend:1 this:1 lab:1

Inverted index

To access the documents, we build an **index**, just like humans do

Inverted Index: word-to-document

"Indice Analitico"

Example

doc1:"I like football"

doc2:"John likes football"

doc3:"John likes basketball"

I	→	doc1	
like	→	doc1	
football	→	doc1	doc2
John	→	doc2	doc3
likes	→	doc2	doc3
basketball	→	doc3	

Query

Information need represented with the same bag-of-words model

Who likes basketball? → Who:1 likes:1 basketball:1

Retrieval

Who:1 likes:1 basketball:1

I	→	doc1
like	→	doc1
football	→	doc1 doc2
John	→	doc2 doc3
likes	→	doc2 doc3
basketball	→	doc3

Result: [**doc2**, **doc3**]

(without any order... or no?)

Boolean Model

likes	→	doc2 doc3
basketball	→	doc3

doc3 contains both "likes" and "basketball", 2/3 of the input query terms, doc2 contains only 1/3

Boolean Model: represents only presence or absence of query words and ranks document according to how many query words they contain

Result: [**doc3**, **doc2**]
(with this precise order)

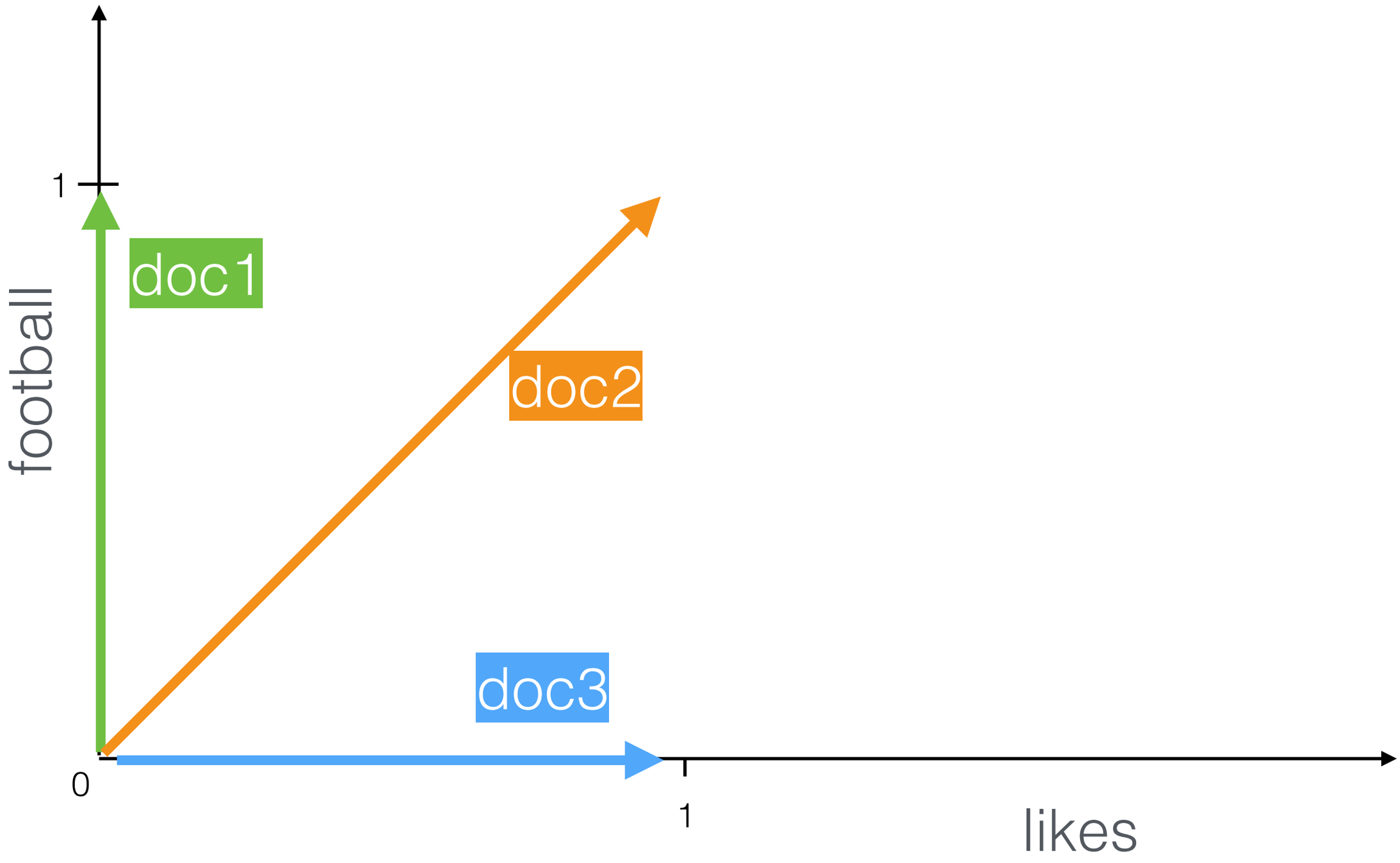
Vector Space Model

Represents documents and words as **vectors** or points in a (multidimensional) geometrical space

We build a **term x documents** matrix

	doc1	doc2	doc3
I	1	0	0
like	1	0	0
football	1	1	0
John	0	1	1
likes	0	1	1
basketball	0	0	1

Graphical Representation



Tf-idf

The cells of the matrix contain the frequency of a word in a document (term frequency tf)

This value can be **counterbalanced** by the number of documents that contain the word (inverse document frequency idf)

The final weight is called **tf-idf**

Tf-idf Example 1

"Romeo" is found 700 times in the document "Romeo and Juliet" and also in 7 other plays

The tf-idf of "Romeo" in the document "Romeo and Juliet" is 7000x(1/7)=100

Tf-idf Example 2

"and" is found 1000 times in the document "Romeo and Juliet" and also in 10000 other plays

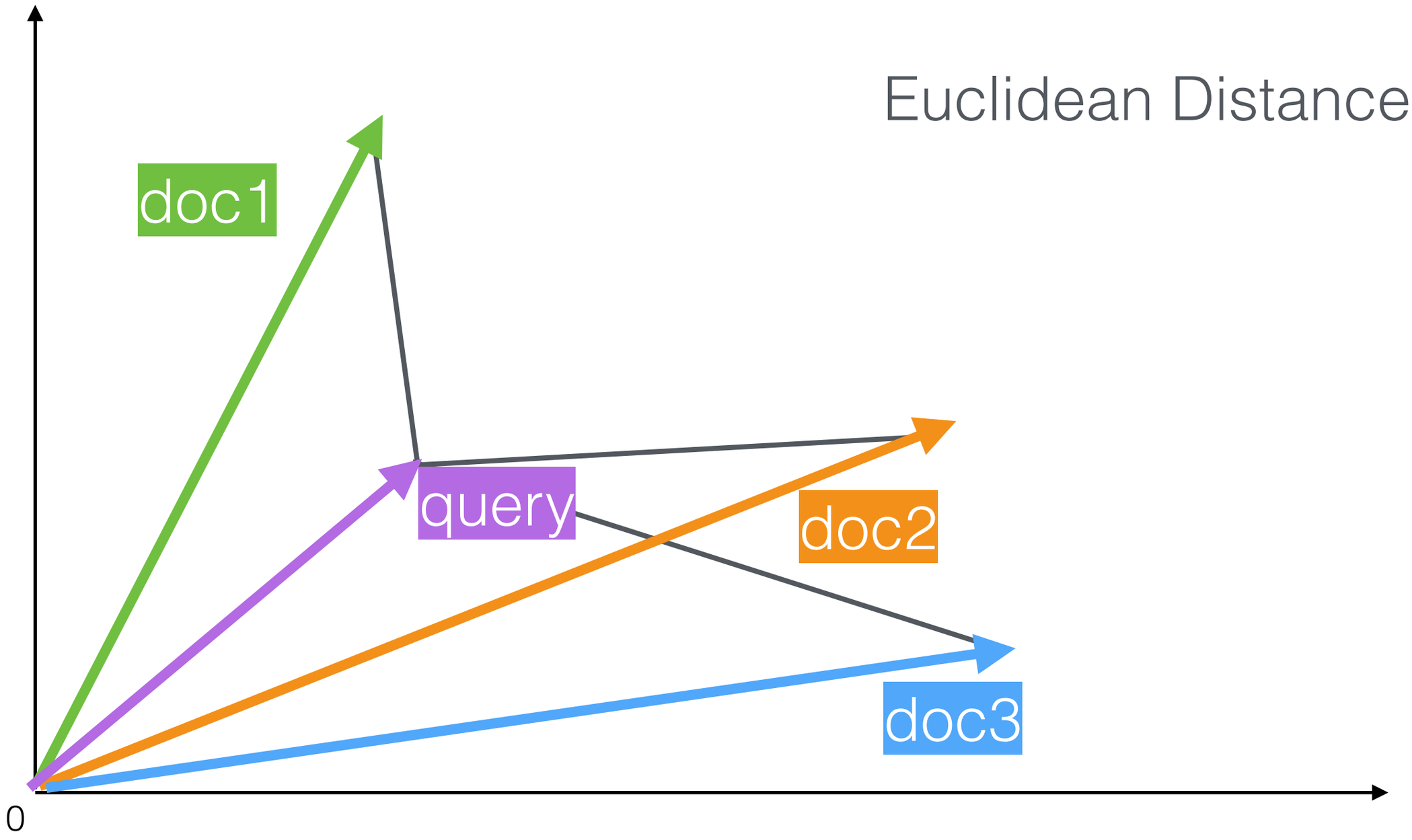
The tf-idf of "and" in the document "Romeo and Juliet" is 1000x(1/10000)=0.1

Similarity

Queries are like an additional column in matrix

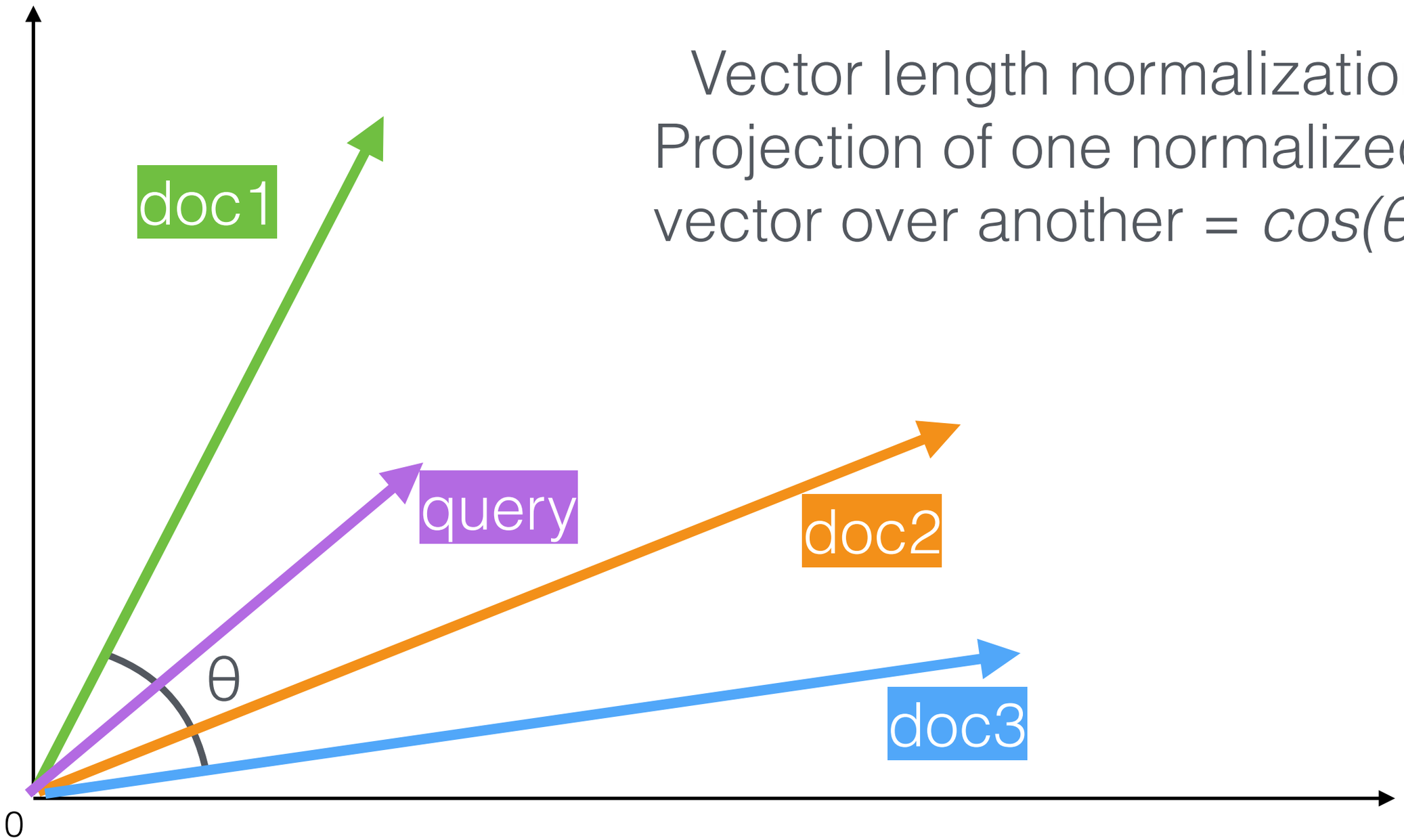
We can compute a **similarity** between document and queries

Similarity



Euclidean Distance

Similarity



Vector length normalization
Projection of one normalized vector over another = *cos(θ)*

Cosine Similarity

The **cosine** of the angle between two vectors is a measure of how similar two vectors are

As the vectors represents documents and queries, the cosine is a measure of similarity of how similar is a document with respect to the query

Ranking with Cosine Similarity

Shows how query and documents are **correlated**

1 = maximum correlation, 0 = no correlation, -1 = negative correlation

The documents obtained from the inverted index are **ranked** according to their **cosine similarity** wrt. the query

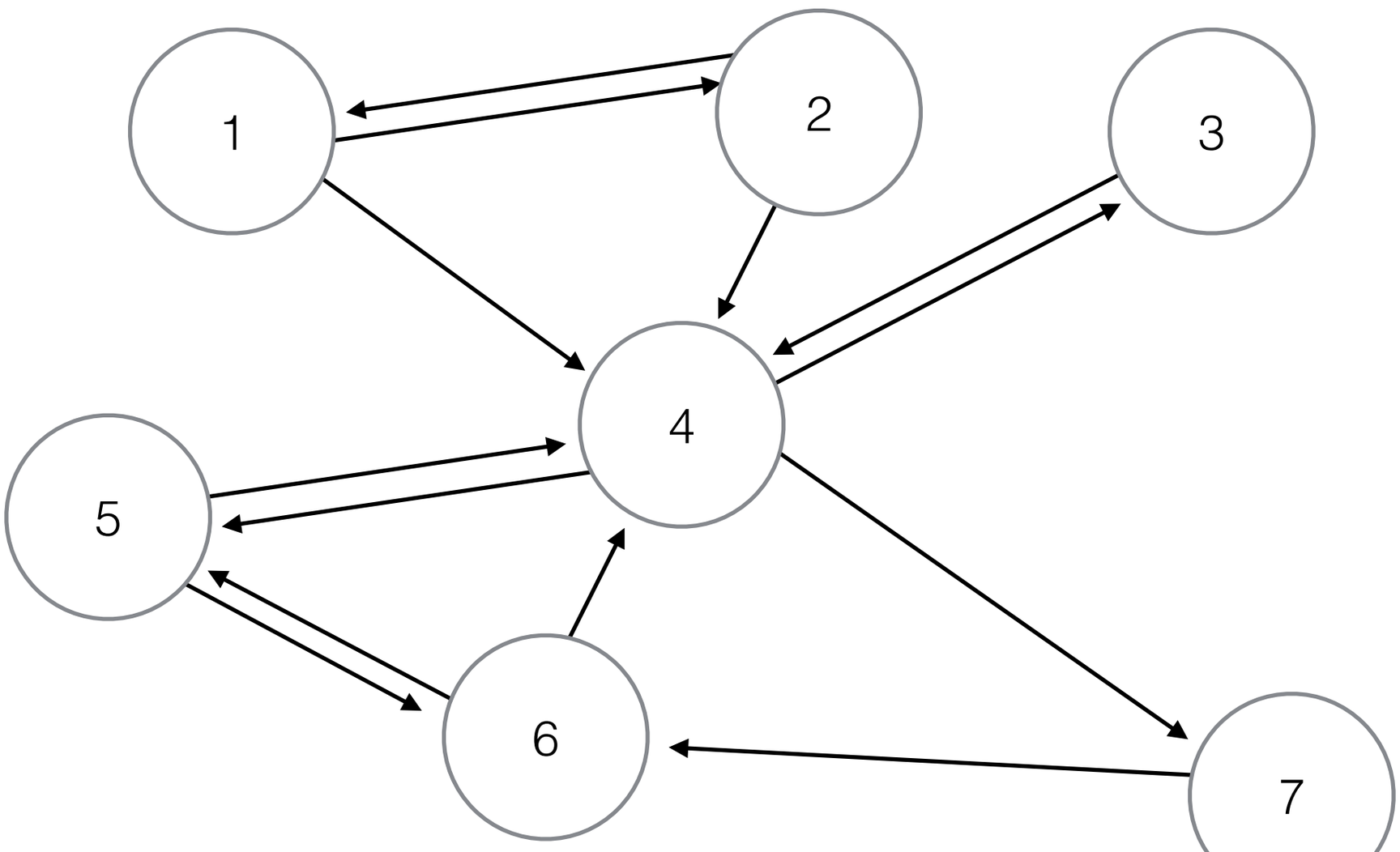
Ranking on the web

In the web documents (webpages) are **connected** to each other by hyperlinks

Is there a way to exploit this topology?

PageRank algorithm (and several others)

The web as a graph



Matrix Representation

	1	2	3	4	5	6	7
1		1/2		1/2			
2	1/2			1/2			
3				1			
4			1/3		1/3		1/3
5				1/2		1/2	
6				1/2	1/2		
7						1	

Simulating Random Walks

e = random vector, can be interpreted as **how important** is a node in the network

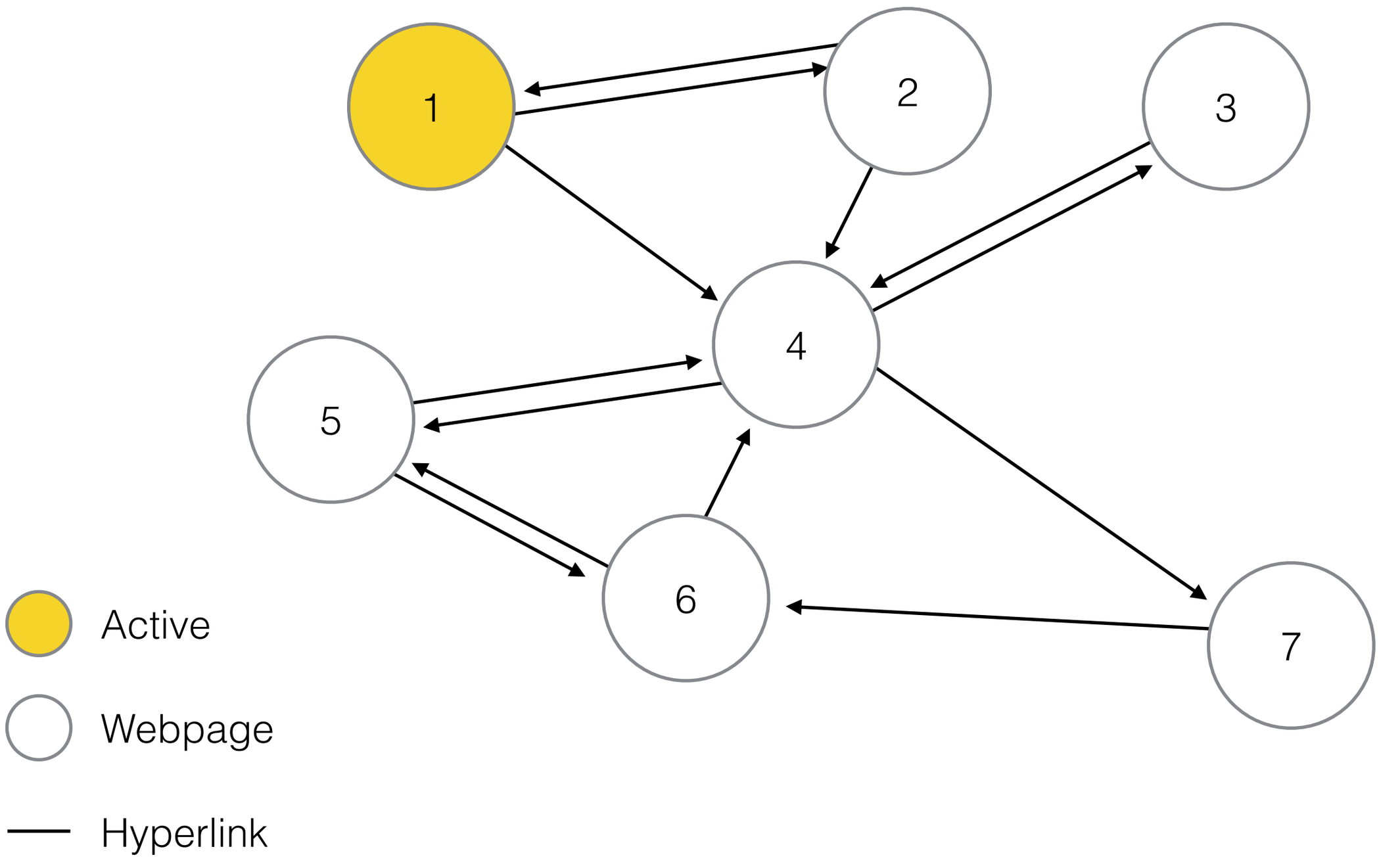
Es. ***e*** = 1:0.5, 2:0.5, 3:0.5, 4:0.5, ...

A = the matrix

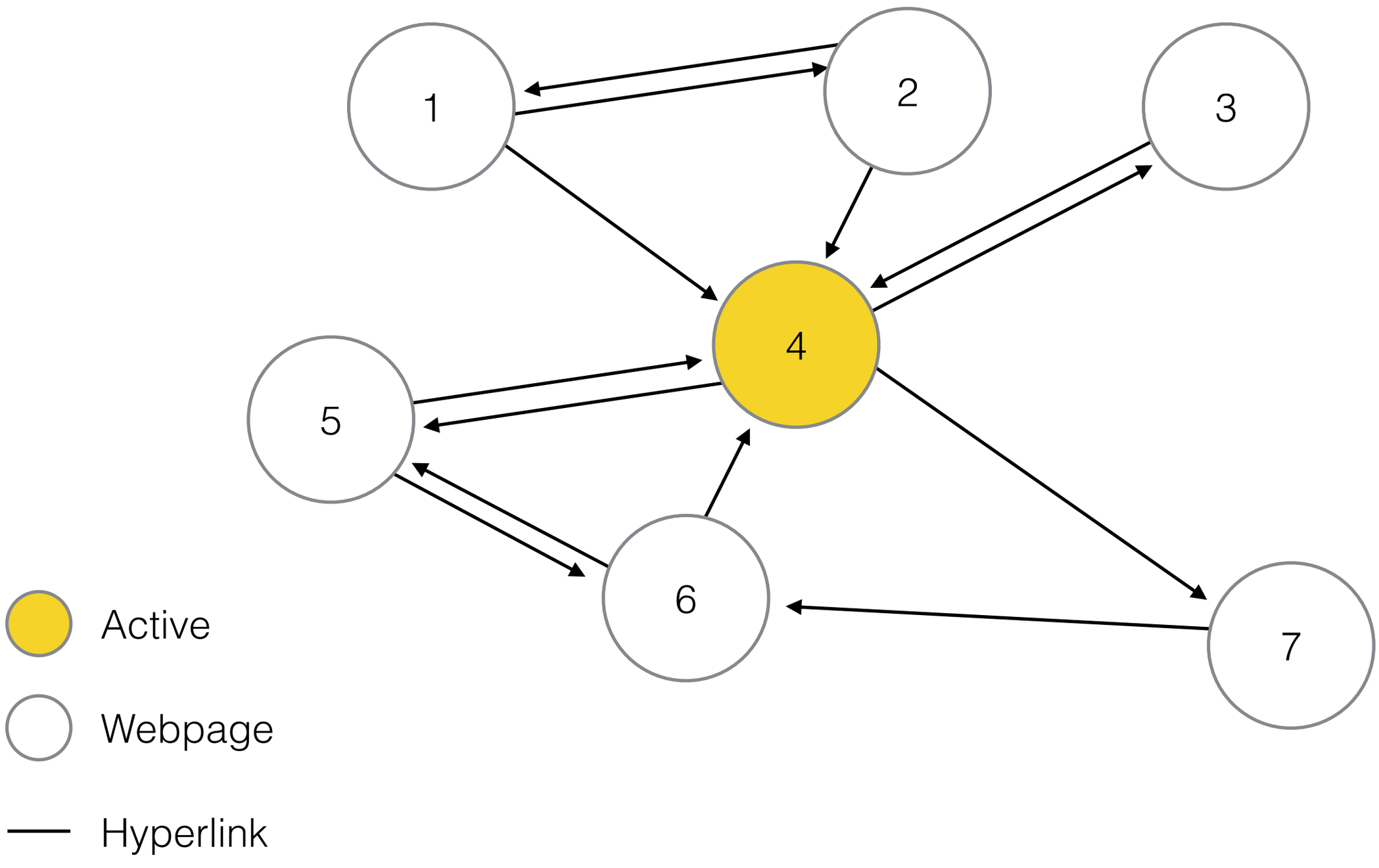
$\boldsymbol{e} = \boldsymbol{e} \cdot \boldsymbol{A}$, repeatedly is like simulating walking randomly on the graph

The process **converges** to a vector ***e*** where each value represents the **importance** in the network

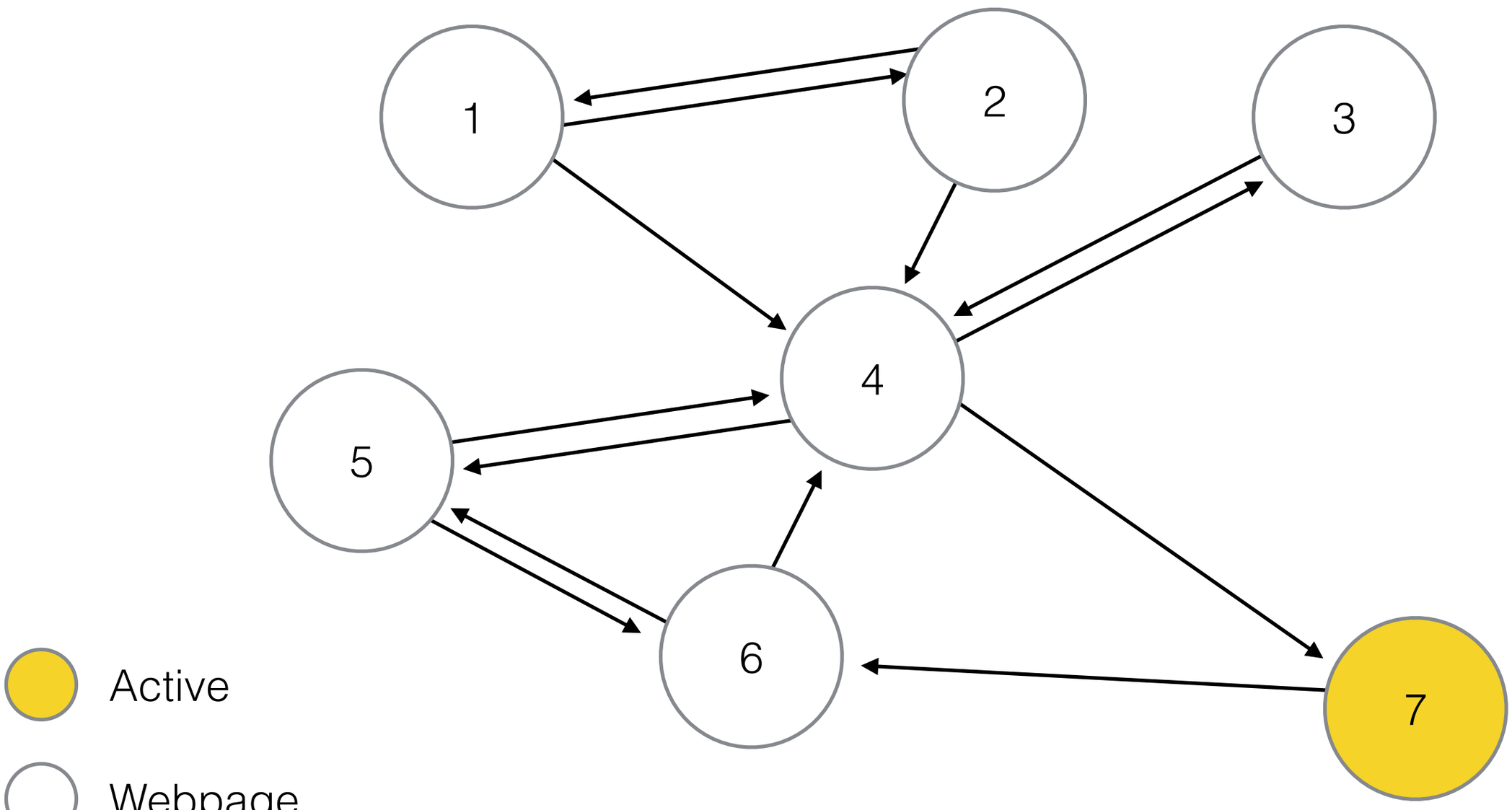
Random Walks



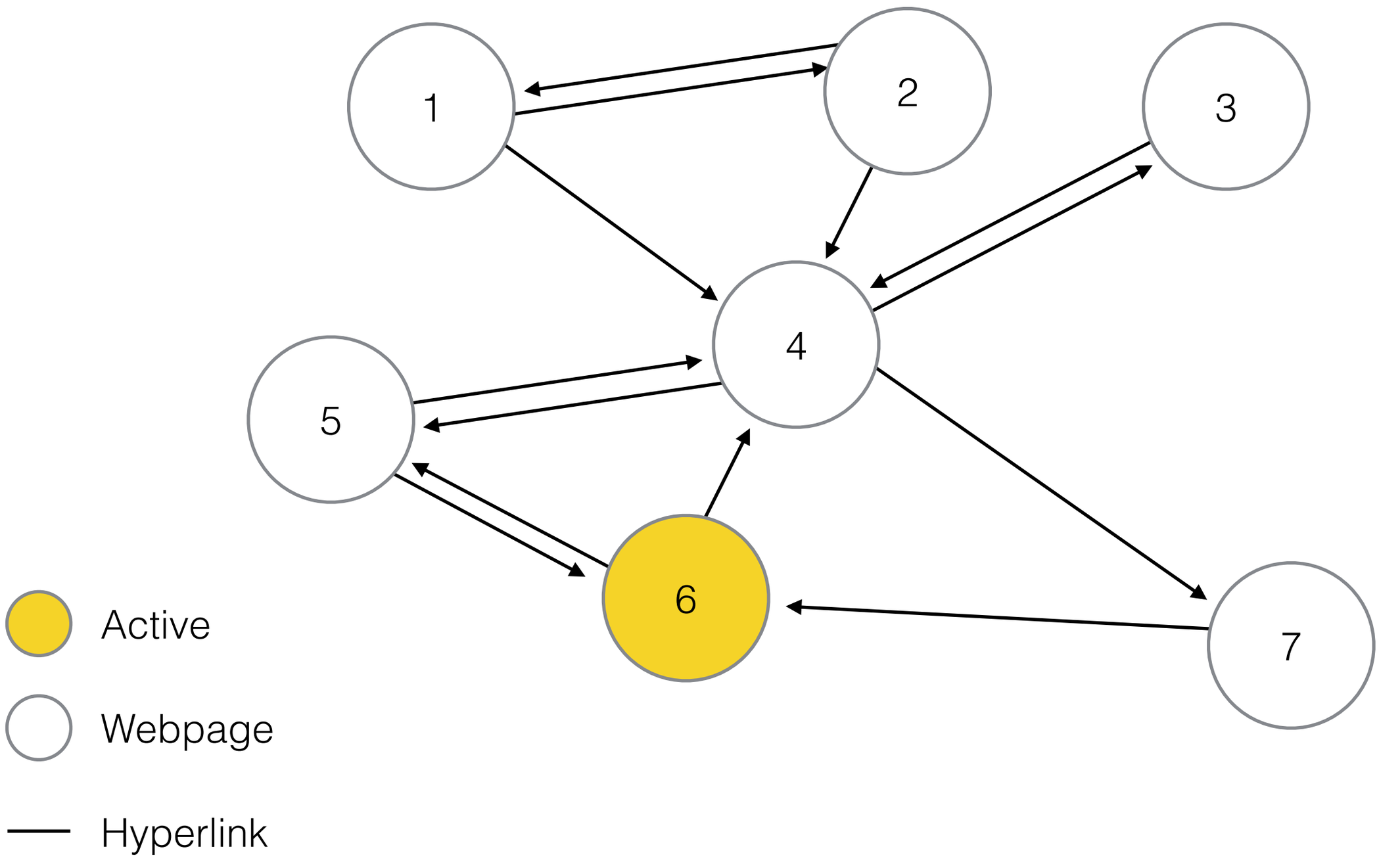
Random Walks



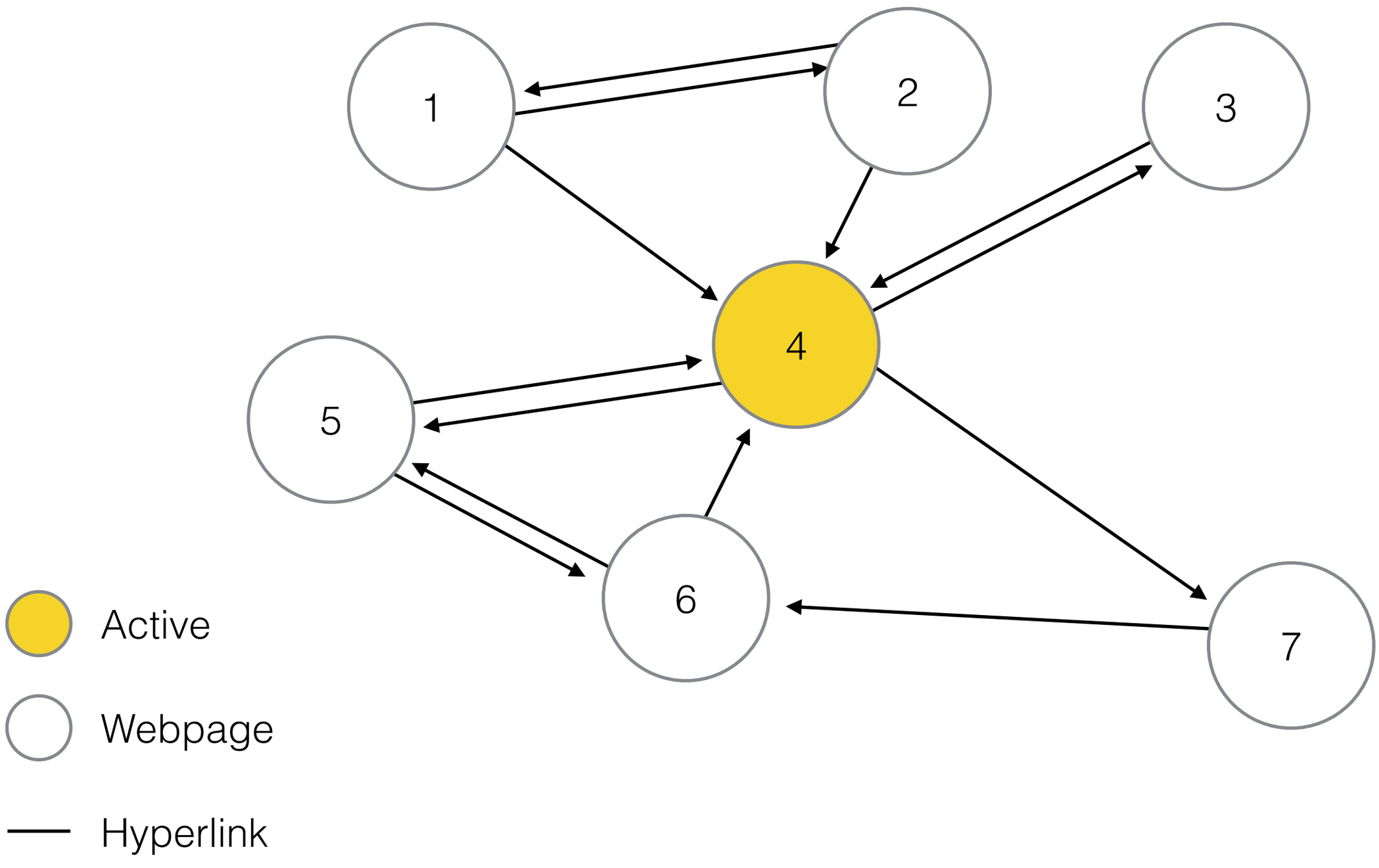
Random Walks



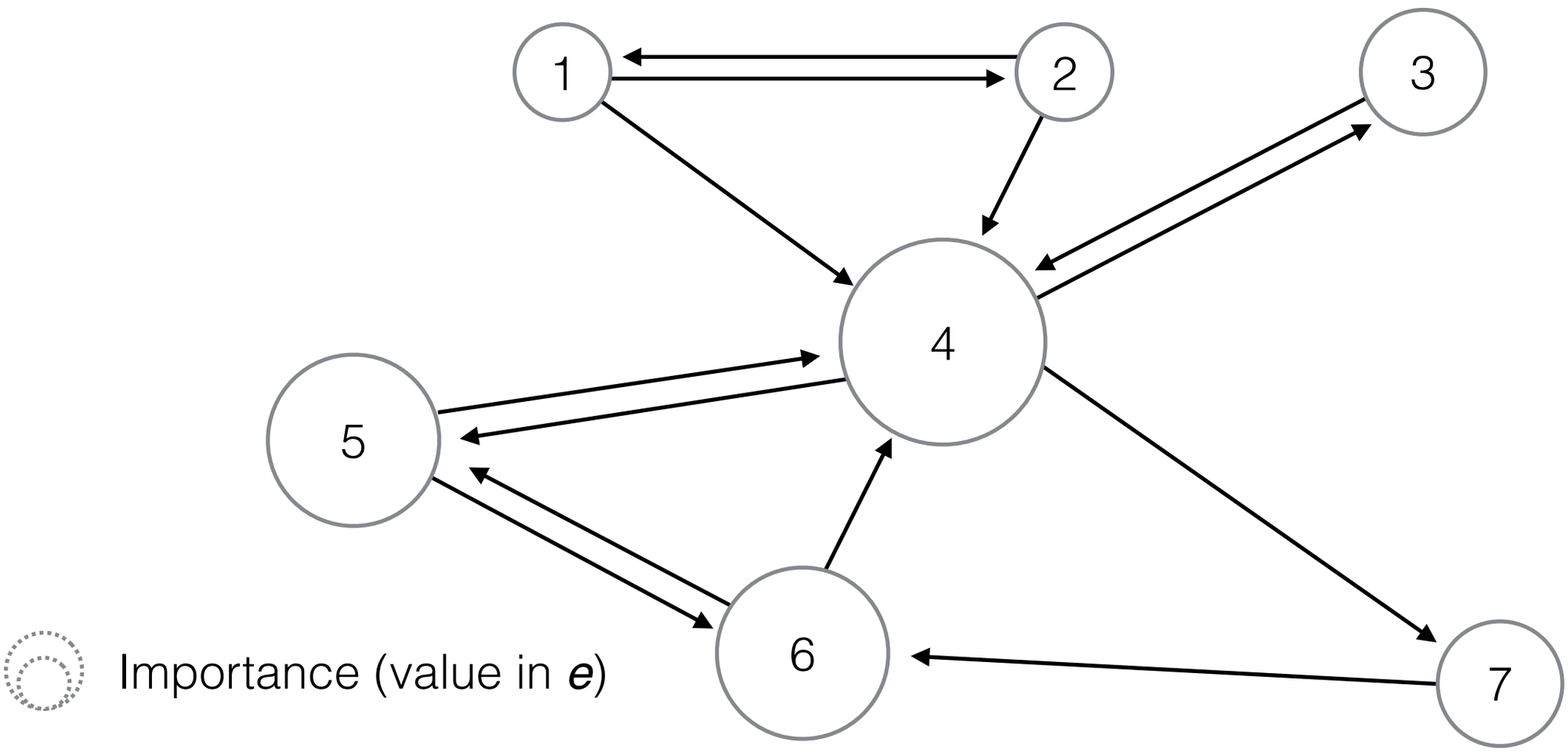
Random Walks



Random Walks



Importance - Authority



Ranking with PageRank

I	→	doc1
like	→	doc1
football	→	doc1 doc2
John	→	doc2 doc3
likes	→	doc2 doc3
basketball	→	doc3

e

doc1	→	0.3
doc2	→	0.1
doc3	→	0.5

The inverted index gives doc2 and doc3
Using the importance in ***e*** for ranking
Result: [**doc3**, **doc2**] (**ordered** list)

Real World Ranking

Real word search engines exploit Vector-Space-Model-like approaches, PageRank-like approaches and **several** others

They balance all the different factors observing what webpage you click on after issuing a query and using them as examples for a machine learning algorithm

Machine Learning

A machine learning algorithm works like a **baby**

You give him **examples** of what is a dog and what is a cat

He **learns** to look at the pointy ears, the nose and other aspects of the animals

When he sees a **new animal** he says "cat" or "dog" depending on his **experience**

Learning to Rank

The different **indicators** of how good a webpage for a query is (cosine similarity, PageRank, etc.) are like the nose, the tail and the ears of the animal

Instead of cat and dog, the algorithm classifies **relevant** or **non-relevant**

When you issue a query and click on a result it is marked as **relevant**, otherwise it is considered **non-relevant**

Issues

Among the different indicators there are a lot of **contextual** ones

Where you issue the query from, who you are, what you visited before, what result you clicked on before, ...

This makes the result for each person, in each place, in different moments in time, **different**

The filter bubble

A lot of diverse content is **filtered** from you

The search engine shows you what it thinks you will be interested on, based on all this contextual factors

Lack of transparency of the search process

A way out of the bubble? Maybe favoring **serendipity**

Thank you
for your attention