# Information Retrieval – Search Engines

Dott. Lorenzo Vaiani
Dipartimento di Automatica e Informatica
Politecnico di Torino


Politecnico
di Torino

# Lecture goals

- Information Retrieval (IR)
  - Binary model
  - Inverted Index
  - Vector Space Model (VSM)
- Web search engines
  - HITS
  - PageRank
- ElasticSearch
  - Indexing
  - Scoring
  - APIs
  - Python library

# Lecture goals

- **Information Retrieval (IR)**
  - Binary model
  - Inverted Index
  - Vector Space Model (VSM)
- Web search engines
  - HITS
  - PageRank
- ElasticSearch
  - Indexing
  - Scoring
  - APIs
  - Python library

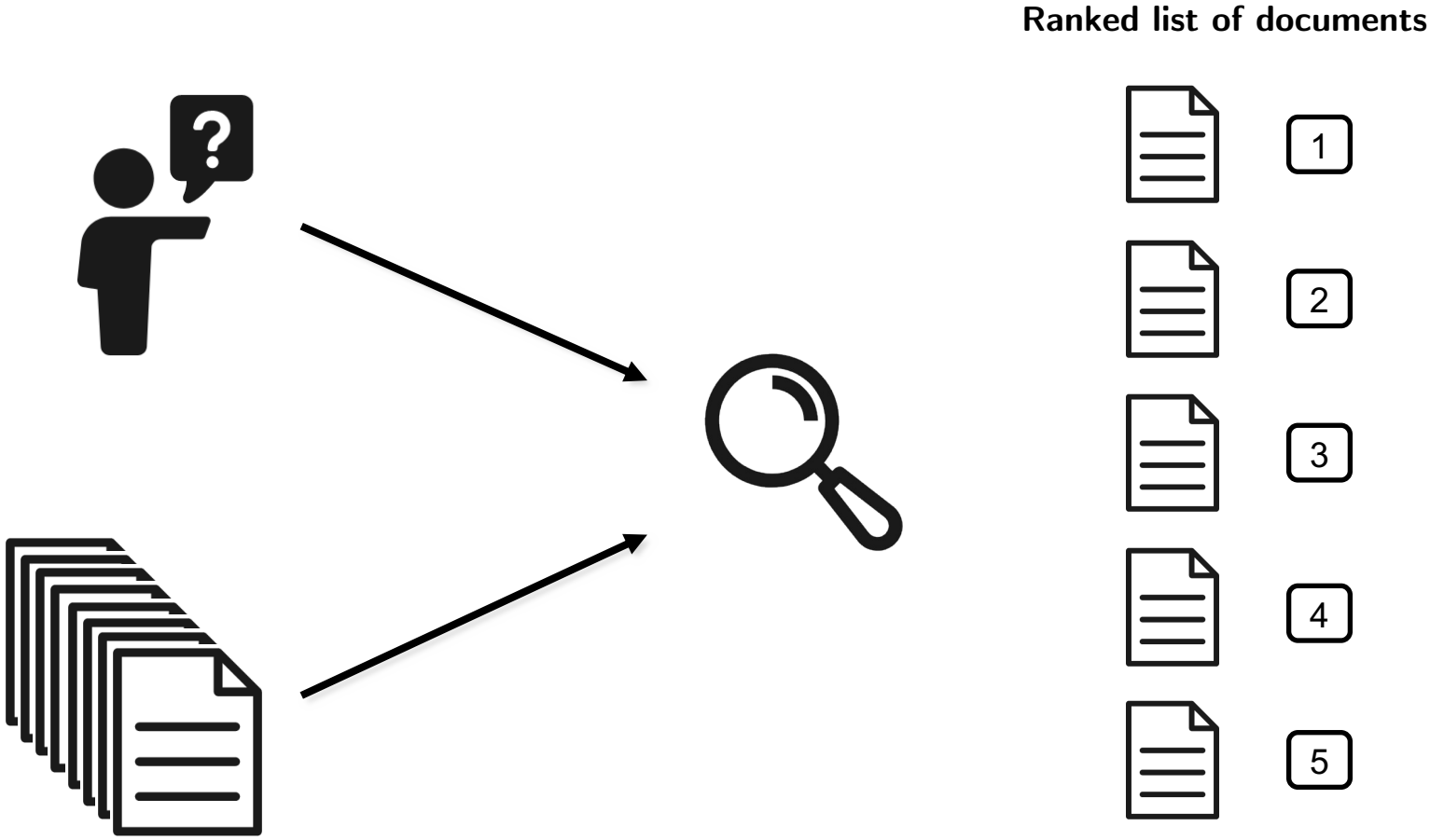
# **Information** Retrieval

Information retrieval (IR) is finding **material** of an **unstructured nature** that satisfies an information need from within **large collections.**

Schütze, H., Manning, C. D., & Raghavan, P. (2008). ***Introduction to information retrieval*** (Vol. 39, pp. 234-265). Cambridge: Cambridge University Press.

- **Documents**
- **Text in the majority of cases**
- **Stored in computers**

# Information **Retrieval**



# Binary model

- Simple model based on a set theory
- Queries as boolean expressions
- Adopted in traditional commercial systems

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ... |
|---|---|---|---|---|---|---|---|
| Antony | 1 | 1 | 0 | 0 | 0 | 1 | |
| Brutus | 1 | 1 | 0 | 1 | 0 | 0 | |
| Caesar | 1 | 1 | 0 | 1 | 1 | 1 | |
| Calpurnia | 0 | 1 | 0 | 0 | 0 | 0 | |
| Cleopatra | 1 | 0 | 0 | 0 | 0 | 0 | |
| mercy | 1 | 0 | 1 | 1 | 1 | 1 | |
| worser | 1 | 0 | 1 | 1 | 1 | 0 | |
| ... | | | | | | | |

► **Figure 1.1** A term-document incidence matrix. Matrix element $(t, d)$ is 1 if the play in column $d$ contains the word in row $t$, and is 0 otherwise.

Incidence matrix: https://nlp.stanford.edu/IR-book/information-retrieval-book.html

# Text processing

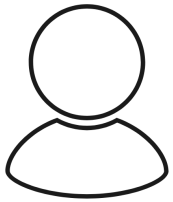
- Word tokenization
- Lowercasing
- Lemmatization/Stemming (root form)
- Stopwords removal

# Information need vs Query

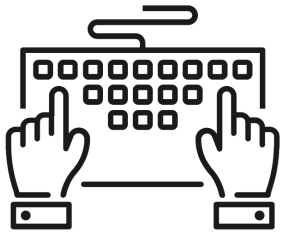
The **information need** is the topic about which the user desires to know more.
The **query** is what the user conveys to the computer in an attempt to communicate the information need

I would like to know if it is **sunny** tomorrow **morning** in Turin...

**Weather** tomorrow Turin



**Information need**



**Query**

# Indexing

To gain the speed benefits of indexing at retrieval time, we have to build the index in advance.

ICP-1

`python, nlp, tool, process, model, natural, language, deep, learning, framework, library`

ICP-2

`natural, language, vector, process, embeddings, word2vec, model, library, gensim, python`

ICP-3

`sentence, embeddings, natural, language, python, doc2vec, extend, word2vec, sent2vec, vector, gensim`

**Dictionary**

| Term | ID |
|---|---|
| python | 1 |
| nlp | 2 |
| tool | 3 |
| process | 4 |
| ... | ... |

# Indexing

To gain the speed benefits of indexing at retrieval time, we have to build the index in advance.

ICP-1 `1, 23, 35, 42, 56, 6, 7, 19, 27, 30, 11`

ICP-2 `6, 7, 14, 46, 45, 72, 88, 48, 55, 1`

ICP-3 `4, 45, 6, 7, 1, 21, 22, 72, 23, 14, 55`

**Dictionary**

| Term | ID |
|---|---|
| python | 1 |
| nlp | 2 |
| go | 3 |
| sentence | 4 |
| ... | ... |

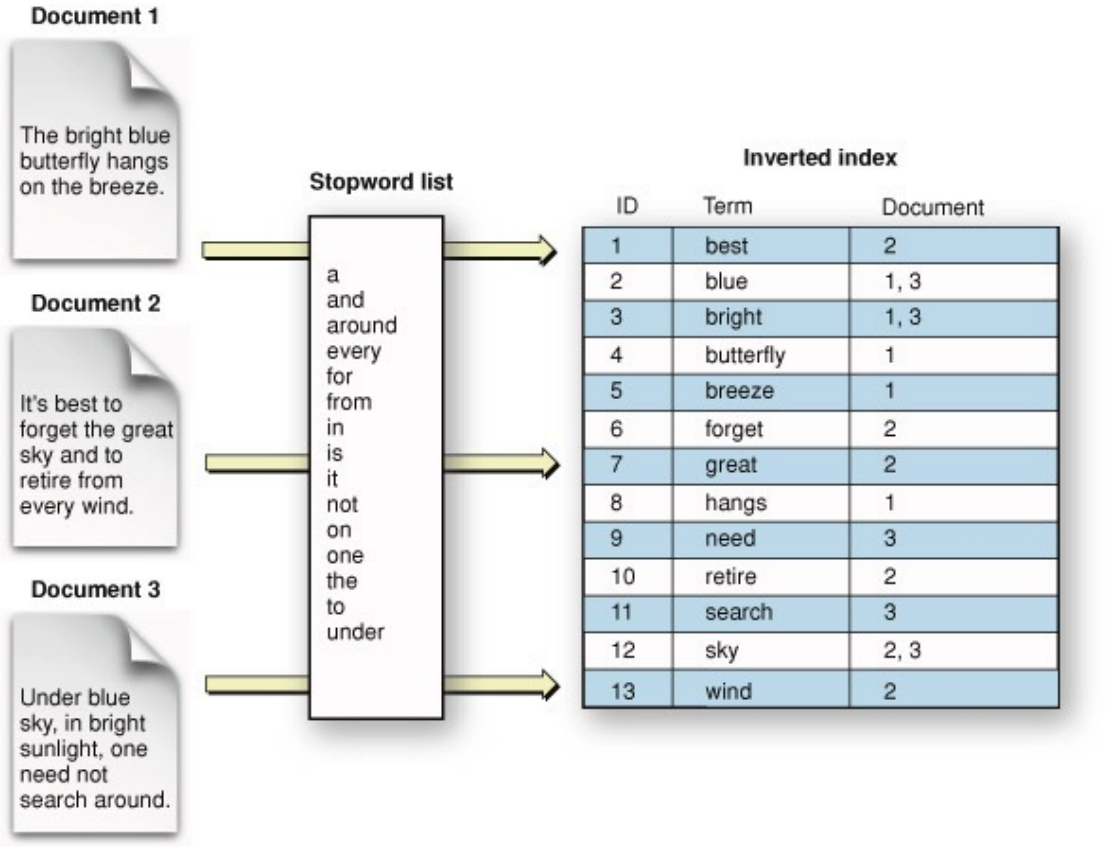
# Inverted index

The dictionary of terms is built by defining the set of unique words in the corpus.

Then for each term, there exist a list that records which documents the term occurs in.

Each item in the list is conventionally called a posting. The list is then called a **postings list**.

# Inverted index




https://community.hitachivantara.com/s/article/search-the-inverted-index

# Traditional Information Retrieval

- **Standard** IR models use the **syntax** of both the query and the document to compute the relevance score.

`Alice goes from her office to her home`

`Alice goes from her home to her office`

**Perfect Match using syntax**

**Semantically different!**

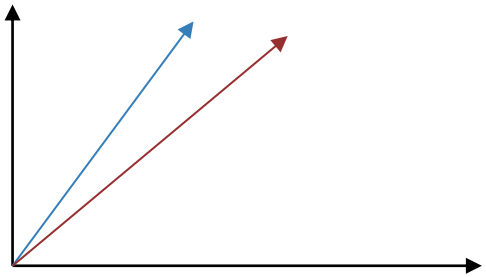
- **Semantic** information is **lost** and the order of the words is not taken into account.

# Neural Information Retrieval

- Neural IR models use the **semantics** of the query and the document to compute the relevance score.
- It leverages the representation given by modern neural architectures.

`Alice goes from her office to her home`

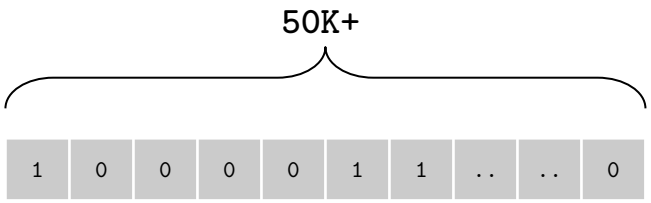
`Alice goes from her home to her office`



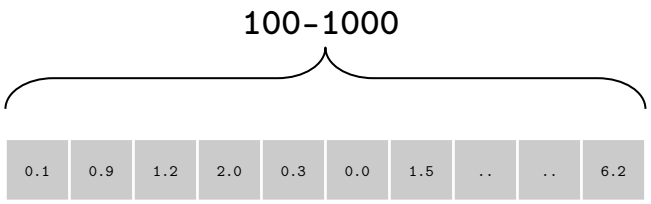
**Vector similarity is used to obtain textual similarity**

# Neural Information Retrieval

**Sparse (traditional) vector representation**




**Dense vector representation**

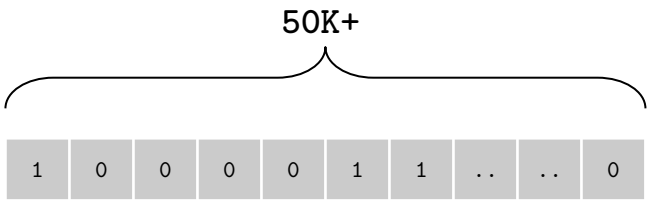



- **Word2Vec** can be used to compute the semantic vectors
- Vector representations can be computed for the query and the document computing the average of the word vectors.

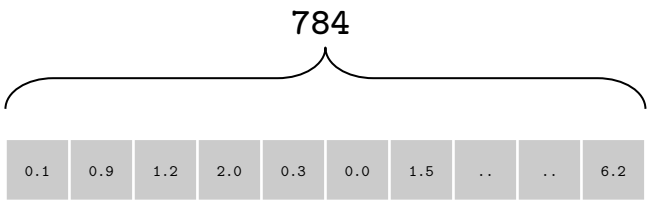
`Not optimal`

# Neural Information Retrieval

**Sparse (traditional) vector representation**



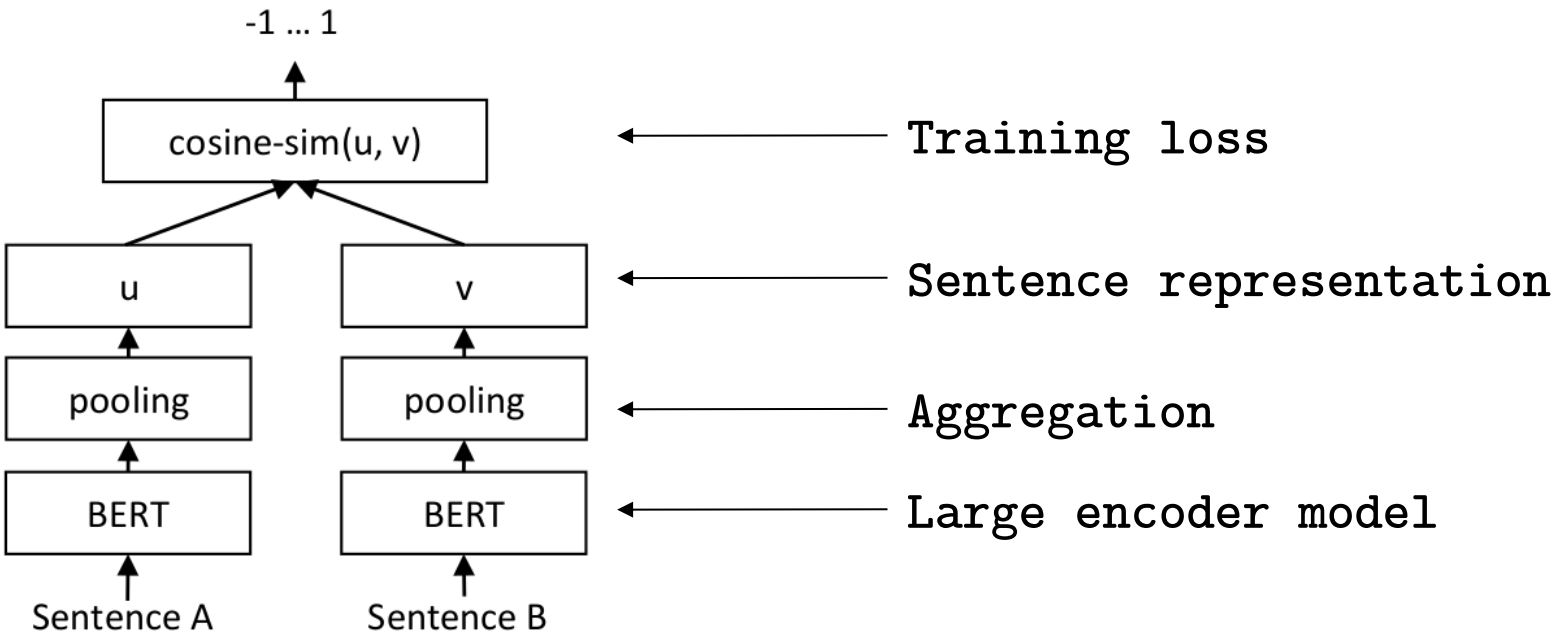

**Dense vector representation**




- **Transformers models** are used to compute the semantic vectors that represent the entire sentence.
- The semantic similarity is computed by computing the cosine similarity between the vectors of the sentences.

`Better`

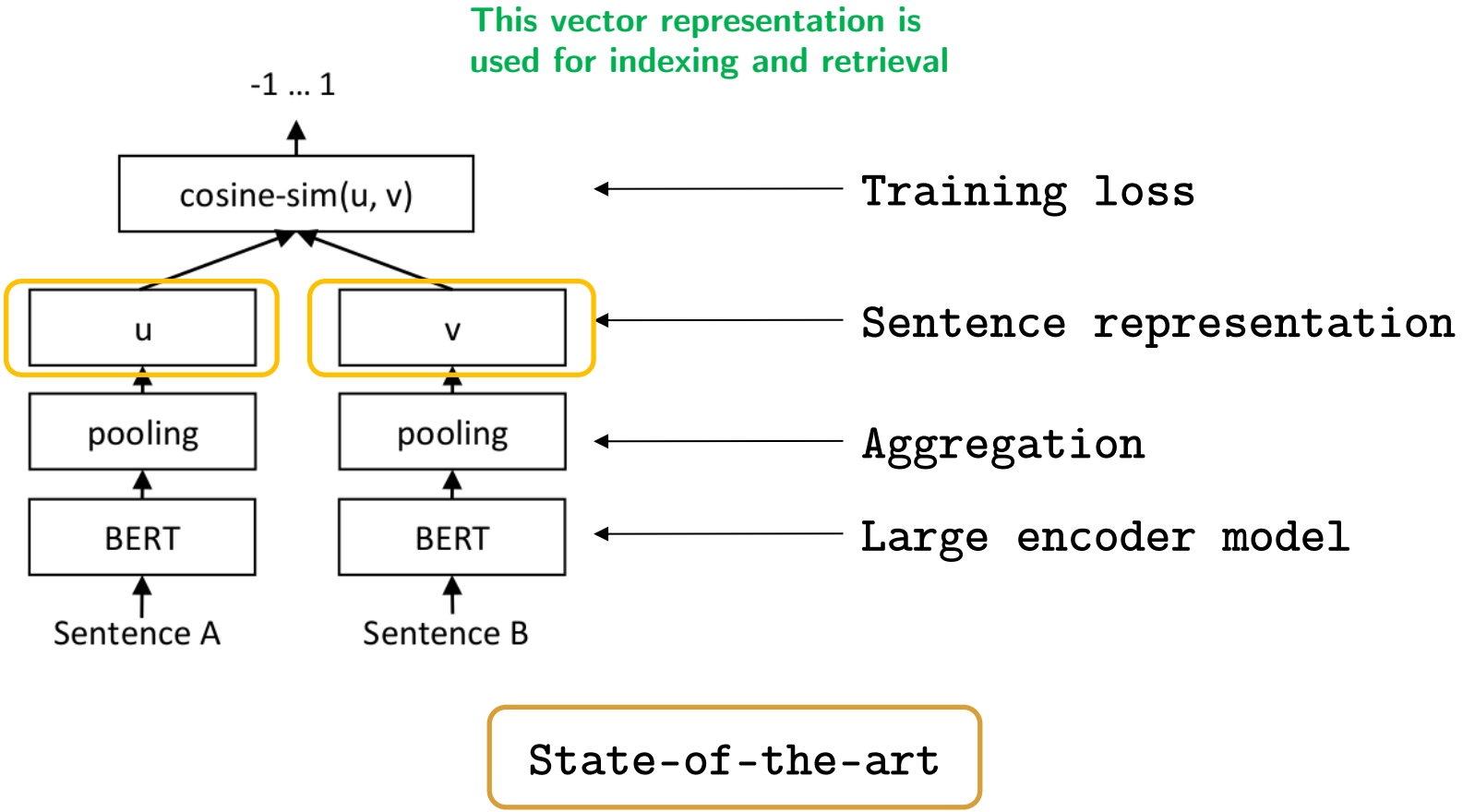
# Neural Information Retrieval




State-of-the-art

https://sbert.net/docs/training/overview.html#network-architecture

# Neural Information Retrieval




https://sbert.net/docs/training/overview.html#network-architecture

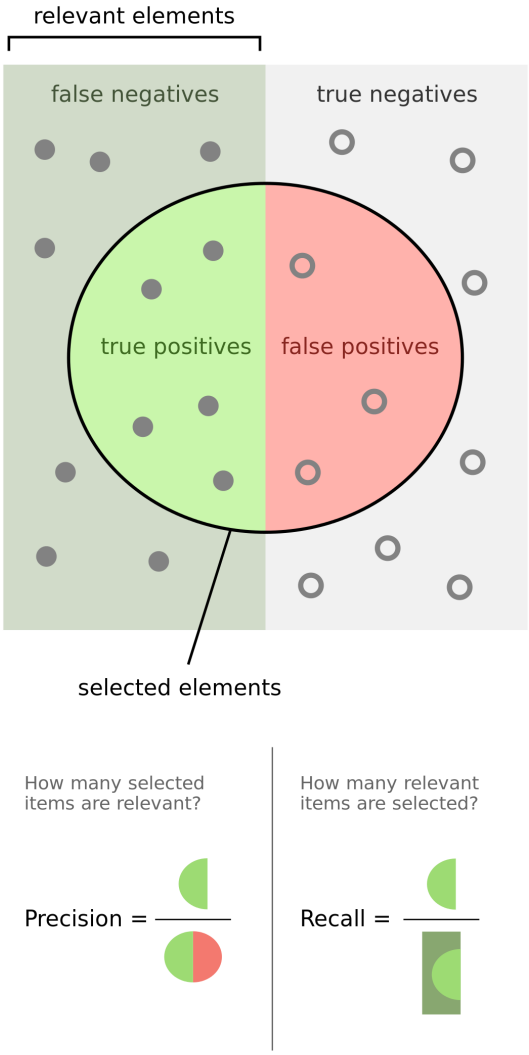
# IR evaluation

- Once we have a model or a technique that is able to compute the relevance score between a query and a document, we need to evaluate it.
- The evaluation is typically done using a collection of **queries and documents** that are not used for training.
- Each query is associated with a set of relevant documents that answer the query.
- The evaluation is done by computing the **precision and recall** of the model.

# IR evaluation

**Precision**: What fraction of the returned results are relevant to the information need?

**Recall:** What fraction of the relevant documents in the collection were turned by the system?



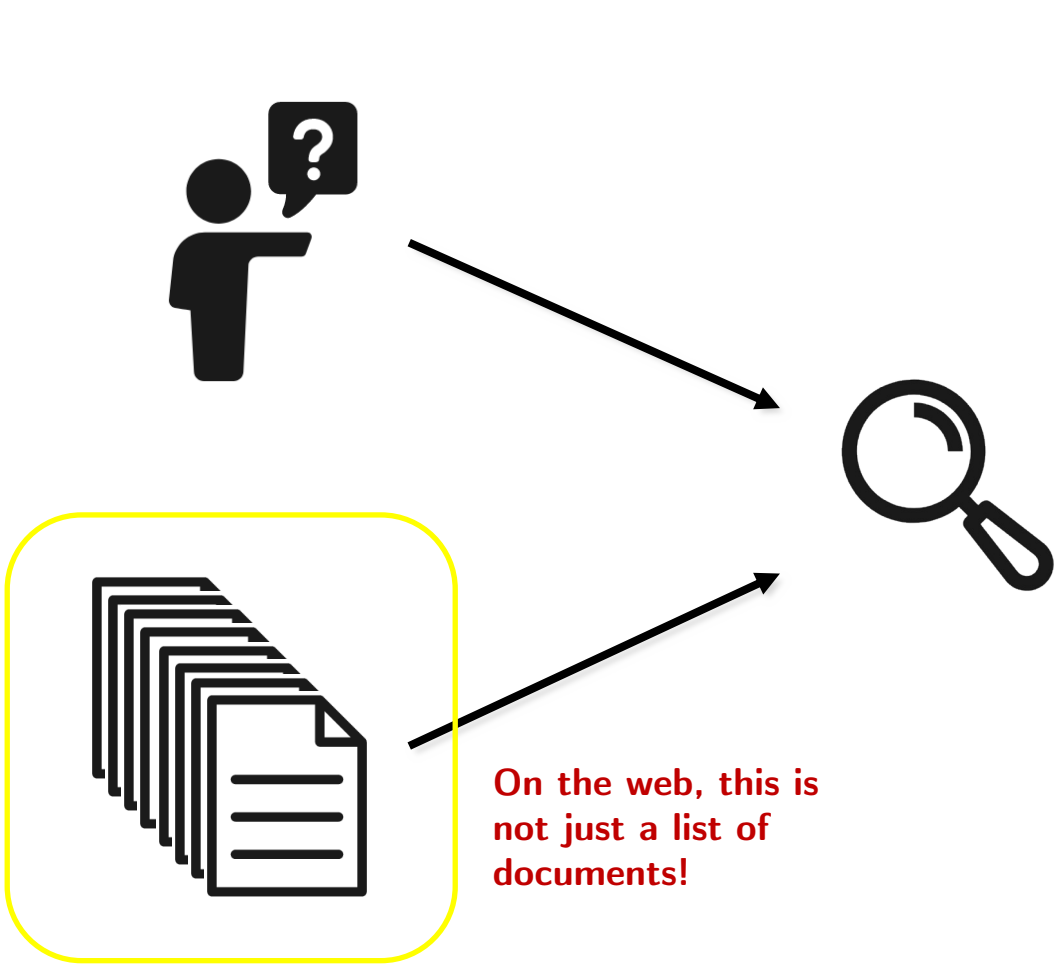

 

# Lecture goals

- Information Retrieval (IR)
  - Binary model
  - Inverted Index
  - Vector Space Model (VSM)
- **Web search engines**
  - HITS
  - PageRank
- ElasticSearch
  - Indexing
  - Scoring
  - APIs
  - Python library

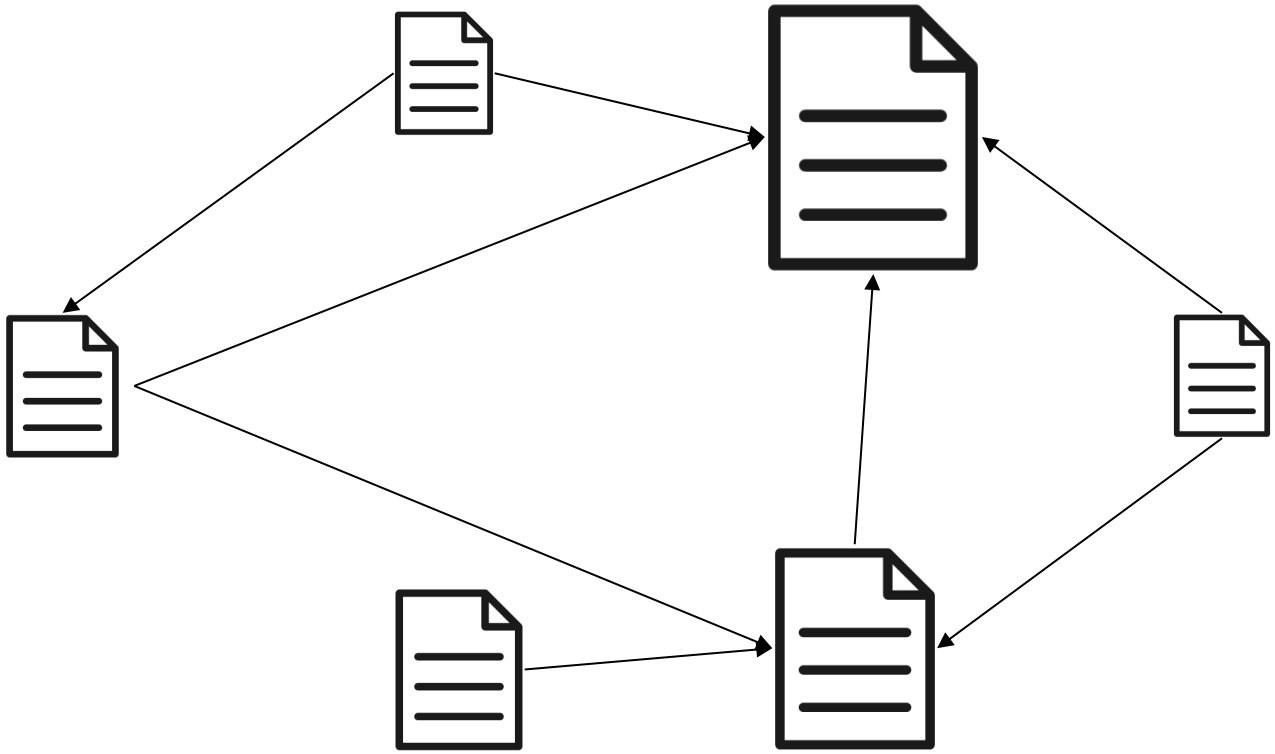
 

# Graph modelling



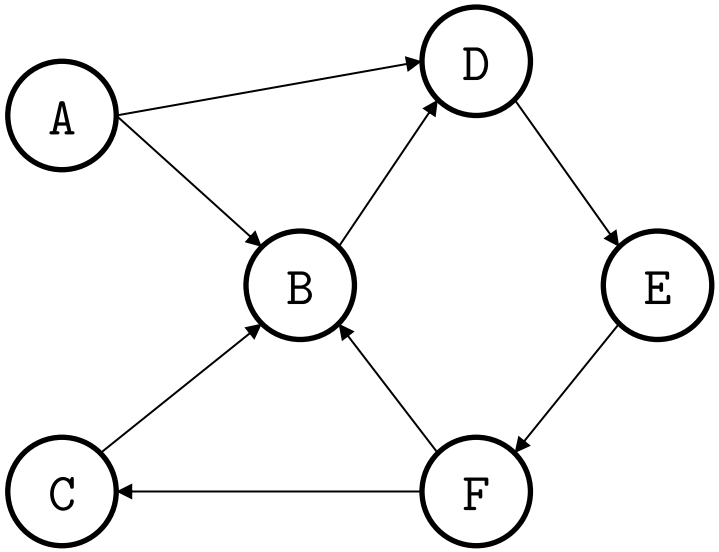
# Graph modelling

**Hyperlinks**: create a link **from** a webpage **to** another



# Link analysis

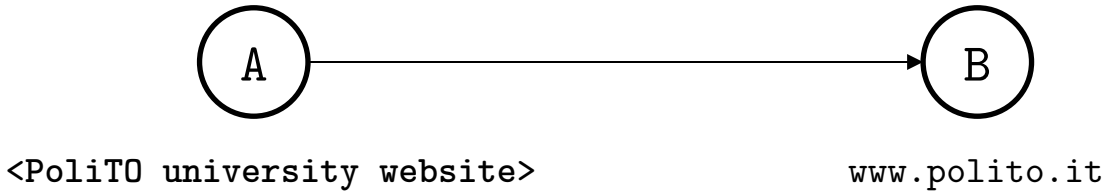
The analysis of hyperlinks and the graph structure of the Web has been instrumental in the development of web search.




It is one of many factors considered by web search engines in computing a composite score for a web page on any given query.

# Link analysis

The hyperlink from A to B represents an endorsement of page B, by the creator of page A.

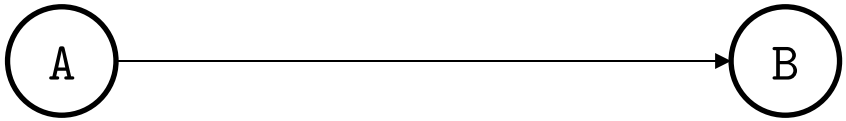


# Link analysis

The hyperlink from A to B represents an endorsement of page B, by the creator of page A.

**What website is more important? A or B?**




<PoliTO university website> www.polito.it

# HITS algorithm

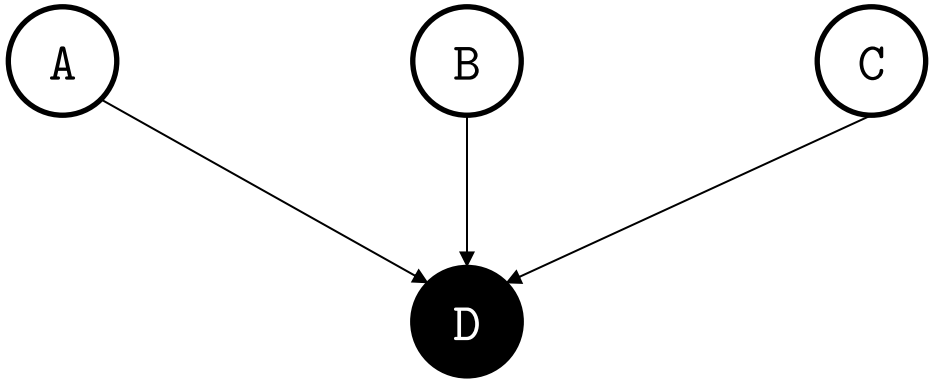
It is a is a link analysis algorithm that rates Web pages, developed by **Jon Kleinberg**.

**Intuition:** certain web pages, known as hubs, served as large directories that were not actually authoritative in the information that they held, but were used as compilations of a broad catalog of information that led users direct to other authoritative pages.

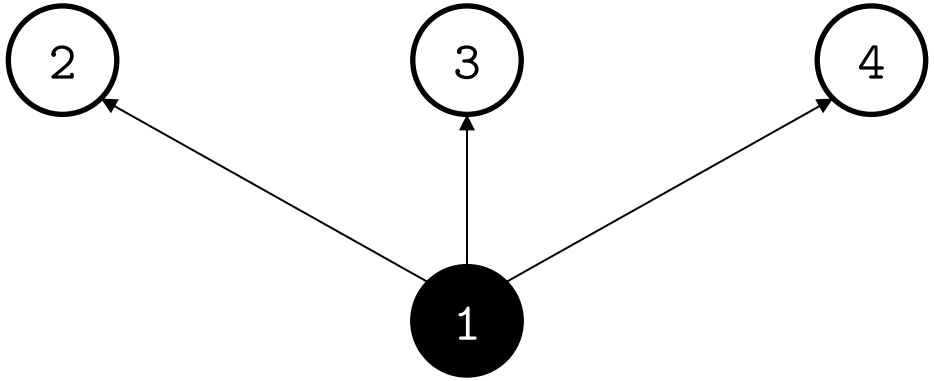
The algorithm assigns two scores for each page:
- **Authority**: estimates the value of the content of the page
- **Hub**: estimates the value of its links to other pages.

# Hubs & Authorities




**D** has an high «authority score» because **it is referenced by** many pages.




**1** has an high «hub score» because **it references** many pages.

# HITS algorithm

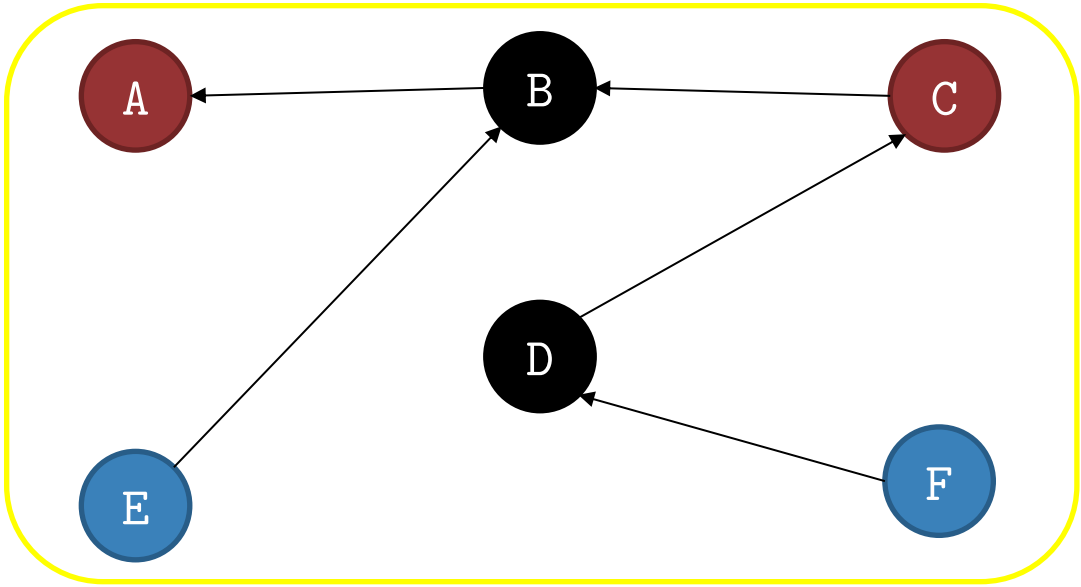
1. Obtain the **root set**: the set of pages that answer to the user query – it uses item-based similarity (e.g., text similarity).

# HITS algorithm

1. Obtain the **root set**: the set of pages that answer to the user query – it uses item-based similarity (e.g., text similarity).
2. Augment the root set to obtain the **base set:** generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it.

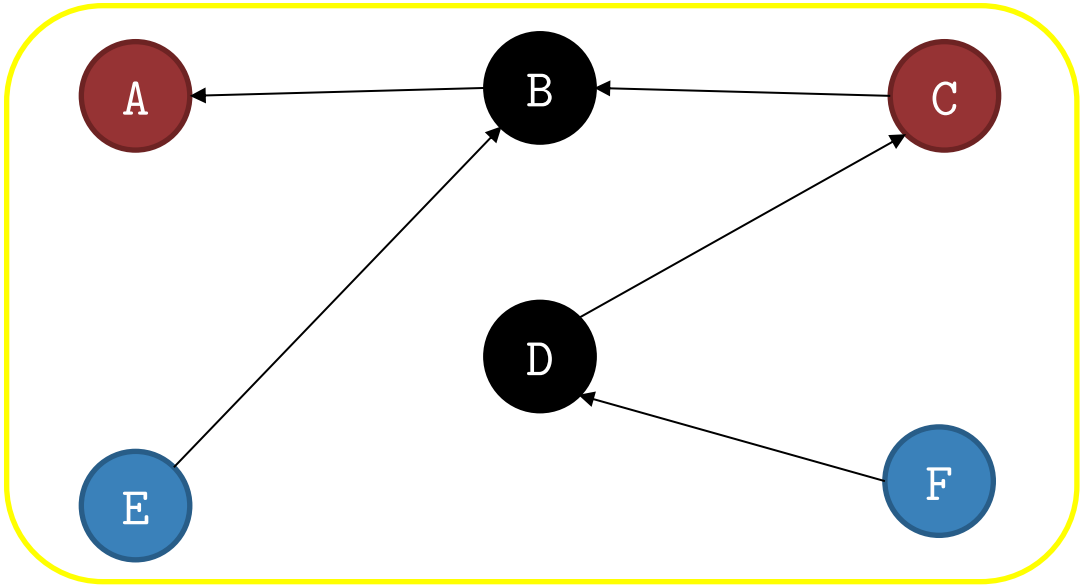
**Focused Subgraph**



# HITS algorithm

- HITS computation is performed only on this focused subgraph
- The reason for constructing a base set is to ensure that most (or many) of the strongest authorities are included

**Focused Subgraph**



# HITS algorithm

Iterate over two basic steps:

**Authority update**: Each node's authority score is set to be equal to the sum of the hub scores of each node that points to it. A node is given a high authority score by being linked from pages that are recognized as Hubs for information.

**Hub update**: Update each node's hub score to be equal to the sum of the authority scores of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

# HITS algorithm

Considerations:

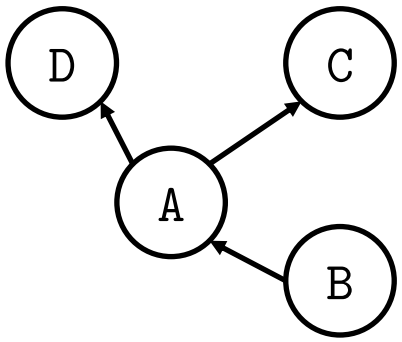
**Initialization:** all nodes have both hub and authority scores set to 1.

**Normalization:** after each update normalize the scores according to the sum of all scores.

$$AUTH_i = \frac{AUTH_i}{\sum_{j=1}^{N} AUTH_j} \qquad HUB_i = \frac{HUB_i}{\sum_{j=1}^{N} HUB_j}$$

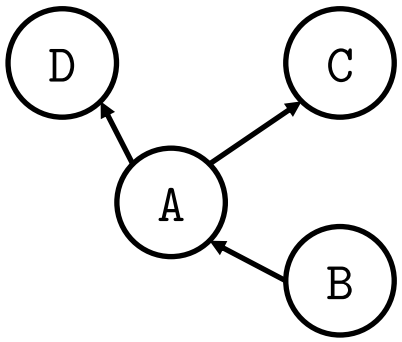
# HITS - authority update




**Iteration 0**

- $H_A = 1, A_A = 1$
- $H_B = 1, A_B = 1$
- $H_C = 1, A_C = 1$
- $H_D = 1, A_D = 1$

# HITS - authority update




**Iteration 0**

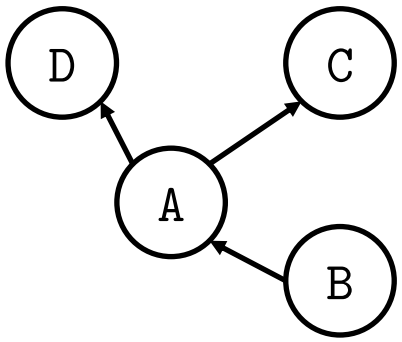
- $H_A = 1, A_A = 1$
- $H_B = 1, A_B = 1$
- $H_C = 1, A_C = 1$
- $H_D = 1, A_D = 1$

**Iteration 1a**

- $H_A = 1, A_A = ?$
- $H_B = 1, A_B = ?$
- $H_C = 1, A_C = ?$
- $H_D = 1, A_D = ?$

**Authority update**: Each node's authority score is set to be equal to the sum of the hub scores of each node that points to it.

# HITS - authority update




**Iteration 0**

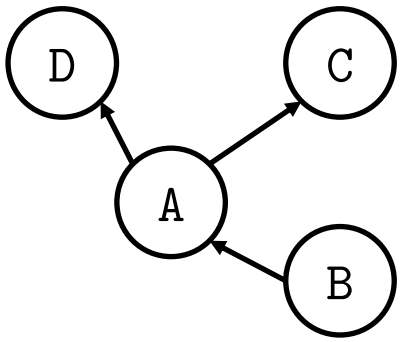
- $H_A = 1, A_A = 1$
- $H_B = 1, A_B = 1$
- $H_C = 1, A_C = 1$
- $H_D = 1, A_D = 1$

**Iteration 1a**

- $H_A = 1, A_A = 1$
- $H_B = 1, A_B = 0$
- $H_C = 1, A_C = 1$
- $H_D = 1, A_D = 1$

**Authority update**: Each node's authority score is set to be equal to the sum of the hub scores of each node that points to it.

# HITS - hub update




**Iteration 0**

- $H_A = 1, A_A = 1$
- $H_B = 1, A_B = 1$
- $H_C = 1, A_C = 1$
- $H_D = 1, A_D = 1$

**Iteration 1a**

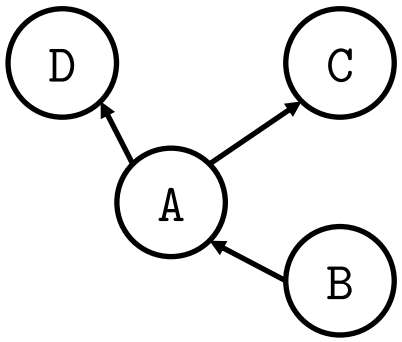
- $H_A = 1, A_A = 1$
- $H_B = 1, A_B = 0$
- $H_C = 1, A_C = 1$
- $H_D = 1, A_D = 1$

**Iteration 1h**

- $H_A = 2, A_A = 1$
- $H_B = 1, A_B = 0$
- $H_C = 0, A_C = 1$
- $H_D = 0, A_D = 1$

**Hub update**: Update each node's hub score to be equal to the sum of the authority scores of each node that it points to.

# HITS - hub update




**Iteration 0**

- $H_A = 1, A_A = 1$
- $H_B = 1, A_B = 1$
- $H_C = 1, A_C = 1$
- $H_D = 1, A_D = 1$

**Iteration 1a**

- $H_A = 1, A_A = 1$
- $H_B = 1, A_B = 0$
- $H_C = 1, A_C = 1$
- $H_D = 1, A_D = 1$

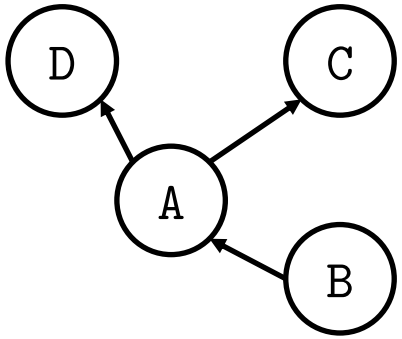
**Iteration 1h**

- $H_A = 2, A_A = 1$
- $H_B = 1, A_B = 0$
- $H_C = 0, A_C = 1$
- $H_D = 0, A_D = 1$

**Iteration 1n**

- $H_A = 2/3, A_A = 1/3$
- $H_B = 1/3, A_B = 0$
- $H_C = 0, A_C = 1/3$
- $H_D = 0, A_D = 1/3$

# HITS – with networkx

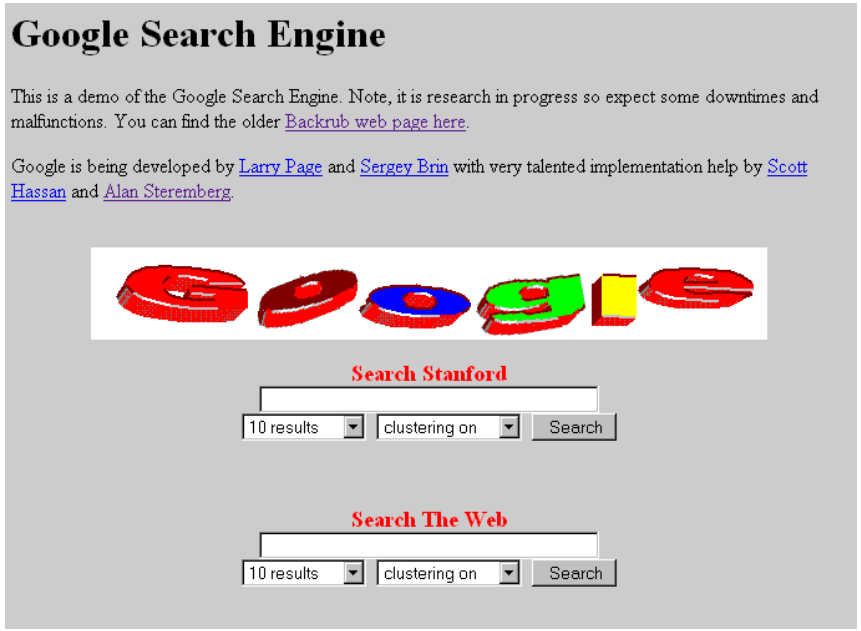



```python
import networkx as nx
from networkx.algorithms.link_analysis.hits_alg import hits

# directed graph
G = nx.DiGraph()
G.add_nodes_from(["A", "B", "C", "D"])
G.add_edges_from([("A", "D"), ("A", "C"), ("B", "A")])

# running hits
h, a = nx.hits(G, max_iter=100)
print ("HUBS scores:", h)
print ("AUTH scores:", a)
```

# PageRank – history

Larry Page and Sergey Brin developed PageRank at Stanford University in 1996 as part of a research project about a new kind of search engine.

This is the original paper:
Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank Citation Ranking : Bringing Order to the Web. WWW 1999.



# HITS vs Pagerank

Pagerank, like HITS, is an iterative algorithm based on the linkage of the documents on the web.

HITS is **query dependent**. The scores resulting from the link analysis are influenced by the search terms;

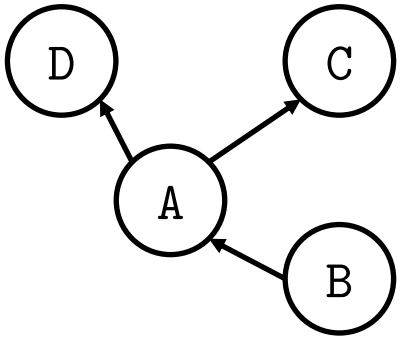
It is executed at **query time**, with the associated hit on performance.

HITS computes **two scores** per document, as opposed to a single score (Pagerank)

It is processed on a **small subset of relevant documents** (base set), not all documents (PageRank).

# Pagerank

**Intuition:** PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page.




**Rule 1.** Links from a page to itself are ignored.

**Rule 2.** Multiple outbound links from one page to another page are treated as a single link.

**Rule 3.** It is initialized to the same value for all pages.
(1/4 = 0.25 in the example)

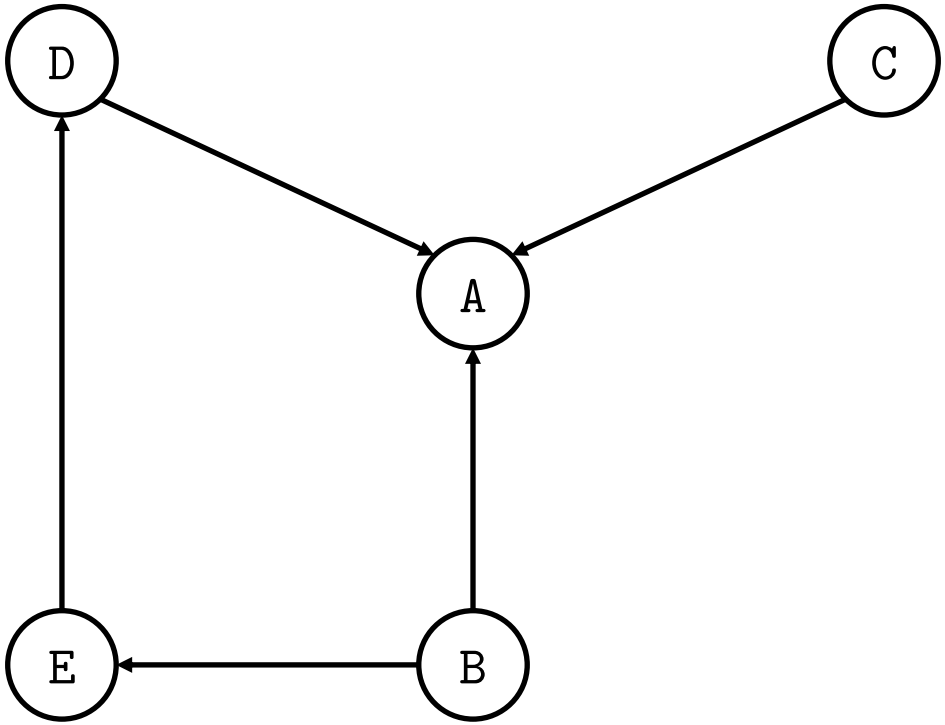
# Pagerank formula

$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$

- λ is an hyperparameter (damping factor)
  - Probability λ to **follow a link** into the page
  - Probability 1-λ to **teleport** to another random page in the graph
  - Typical value: 0.85
- N is total number of pages
- Pagerank equally split among out links

# Pagerank

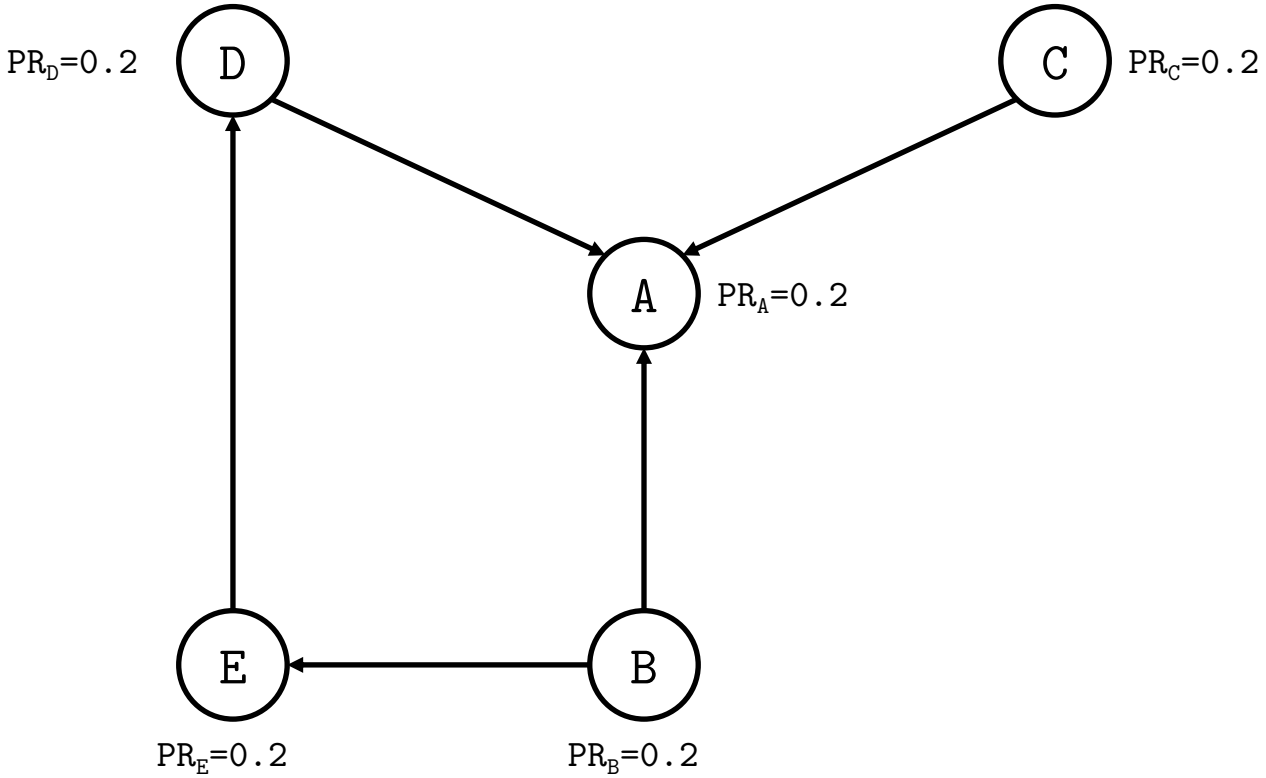
$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$



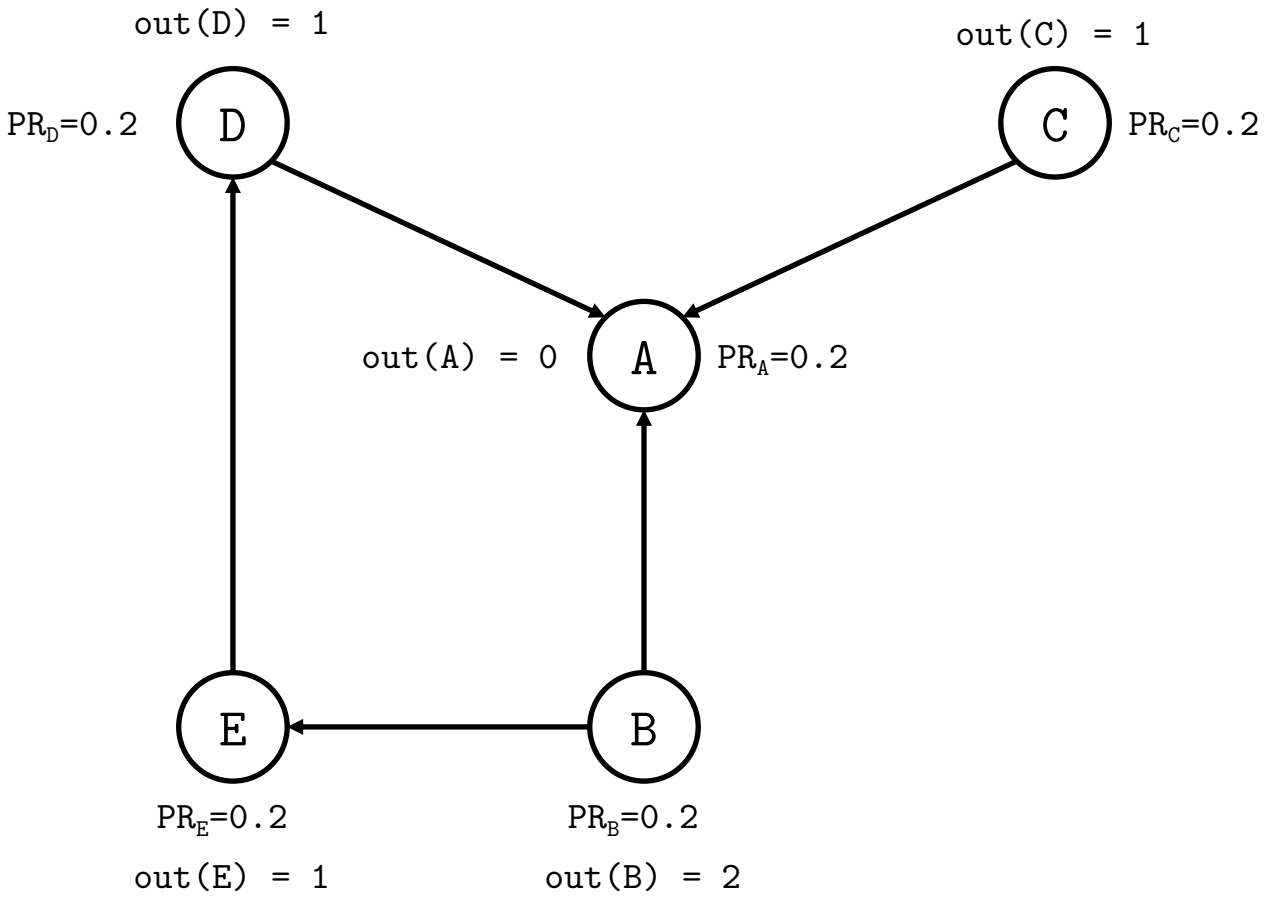
# Pagerank

$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$



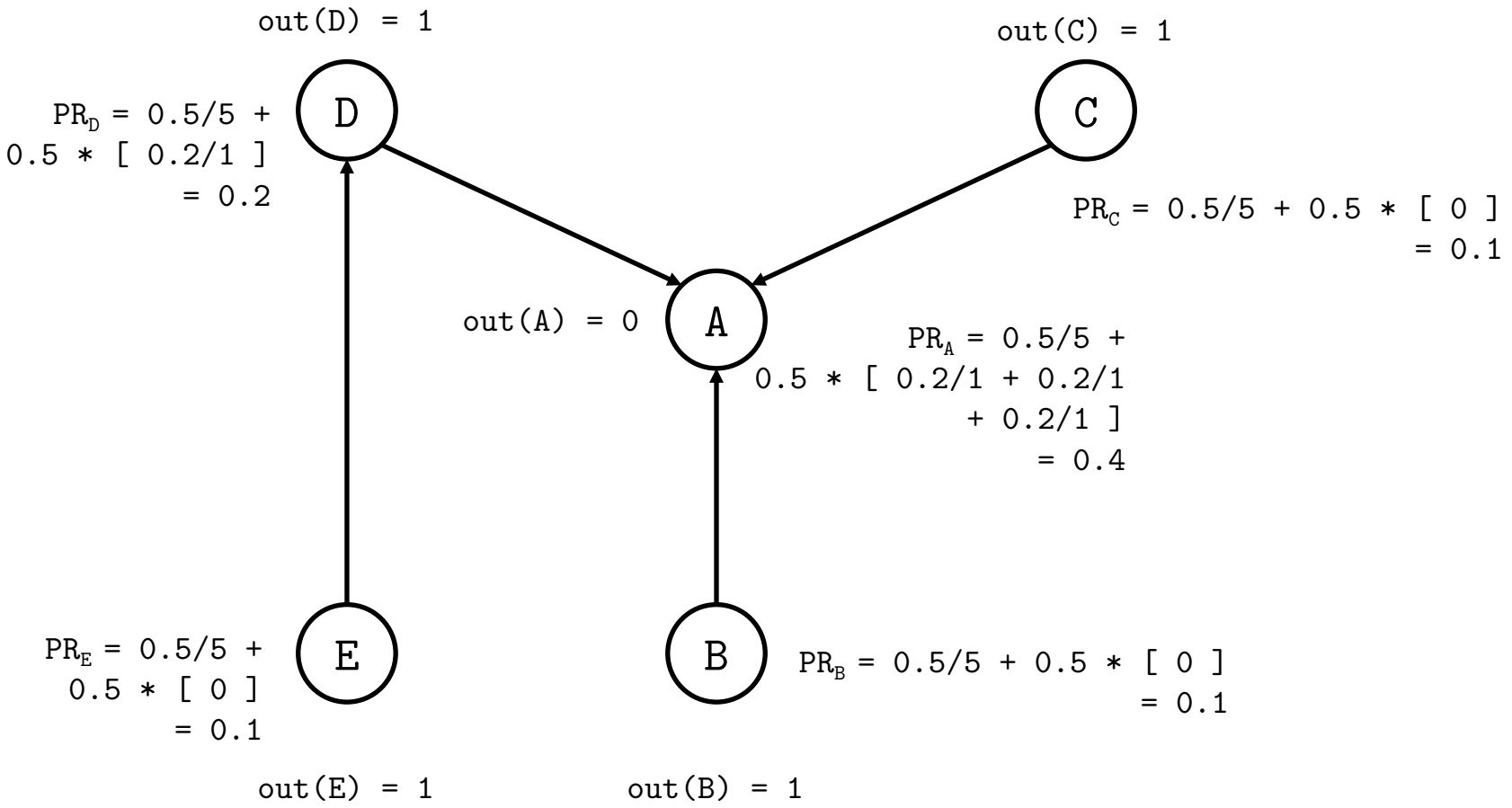
# Pagerank

$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$



# Pagerank

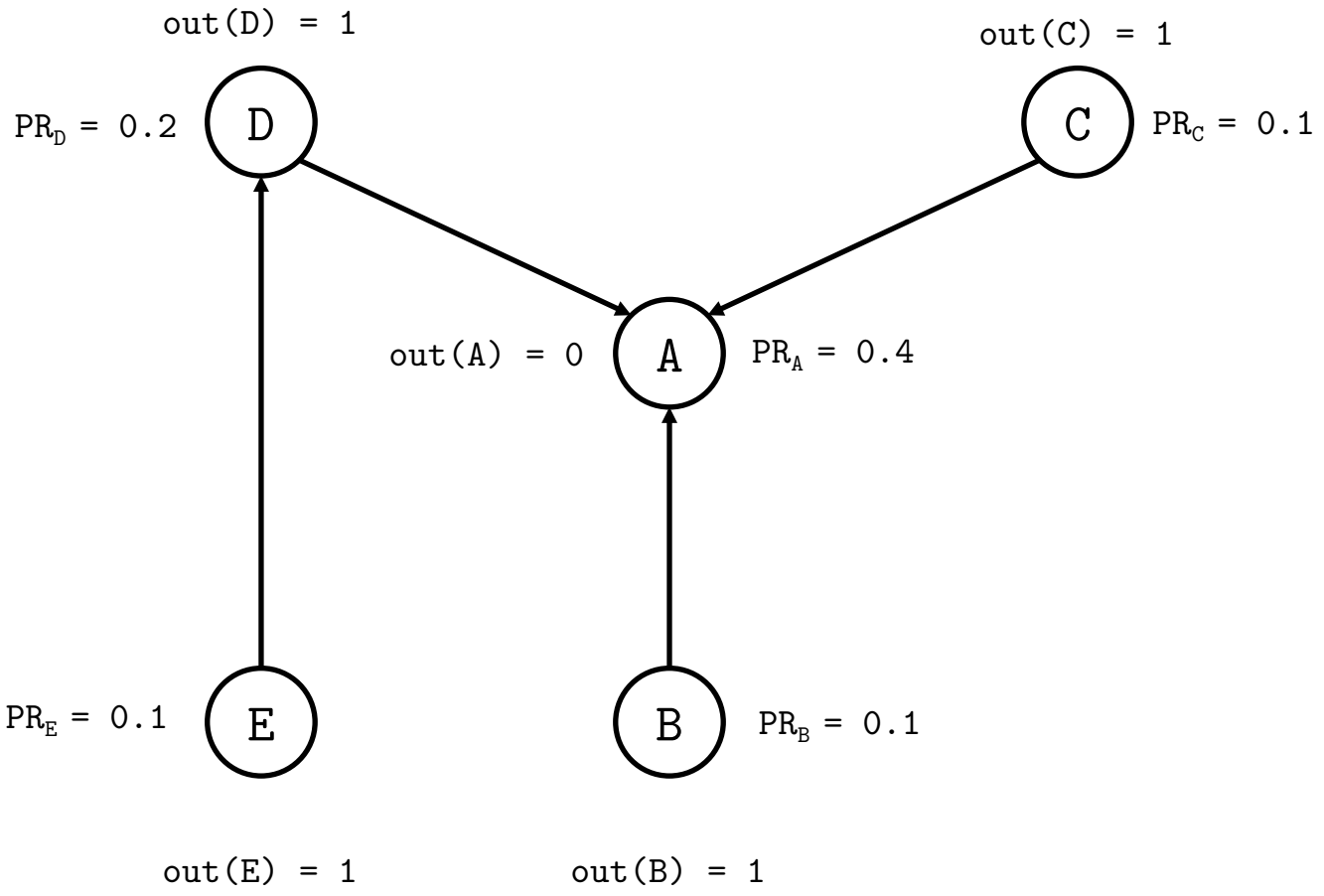
$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$



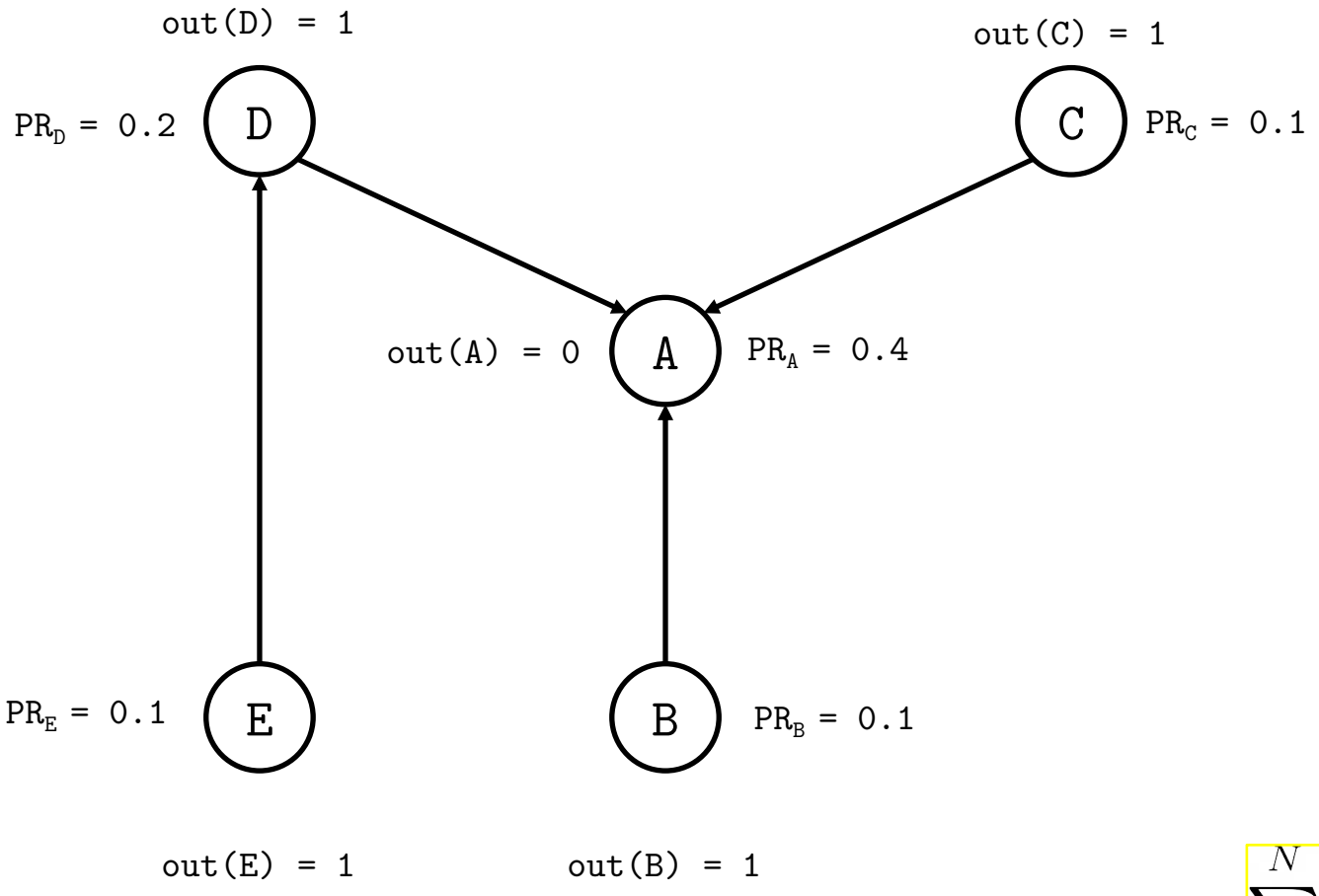
# Pagerank

$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$



# Pagerank

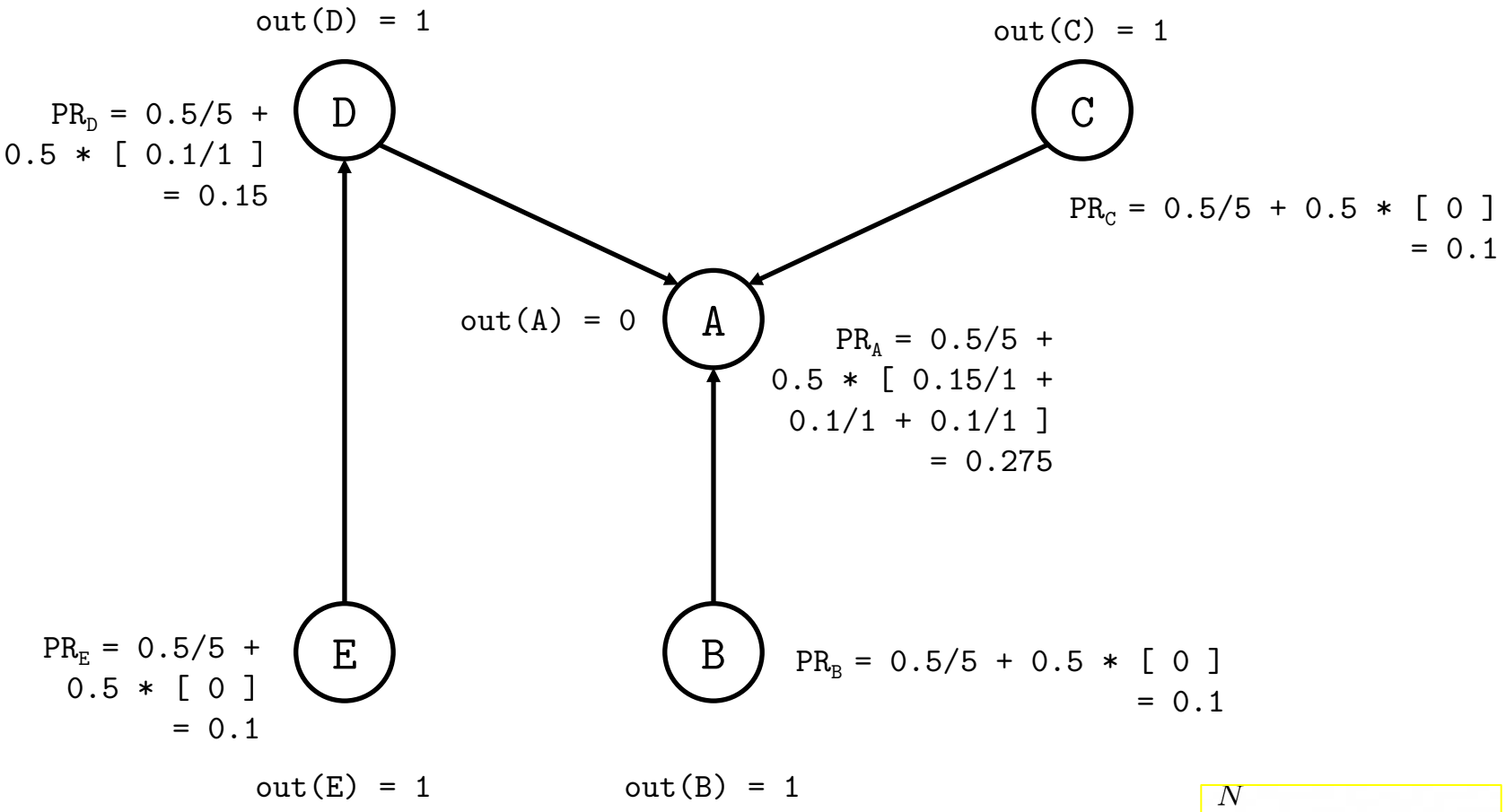
$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$




$$\sum_{j=1}^{N} PR_j = 0.9$$

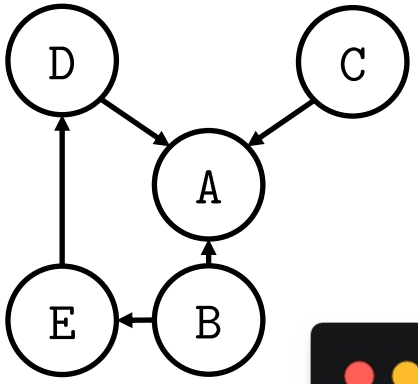
# Pagerank

$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$




$$\sum_{j=1}^{N} PR_j = 0.725$$

# Pagerank – with networkx




```python
import networkx as nx
from networkx.algorithms.link_analysis.pagerank_alg import pagerank
# directed graph
G = nx.DiGraph()
G.add_nodes_from(["A", "B", "C", "D", "E"])
G.add_edges_from([("D", "A"), ("C", "A"), ("B", "A"), ("E", "D")])

# running pagerank
p = nx.pagerank(G, max_iter=100)
print ("PR scores:", p)
```

# Lecture goals

- Information Retrieval (IR)
  - Binary model
  - Inverted Index
  - Vector Space Model (VSM)
- Web search engines
  - HITS
  - PageRank
- **ElasticSearch**
  - Indexing
  - Scoring
  - APIs
  - Python library

# ElasticSearch


- It is a framework to build search engines
  - Real-time
  - Distributed
  - Scalable and efficient
- Popular users:
  - GitHub
  - Wikipedia
  - StackOverflow

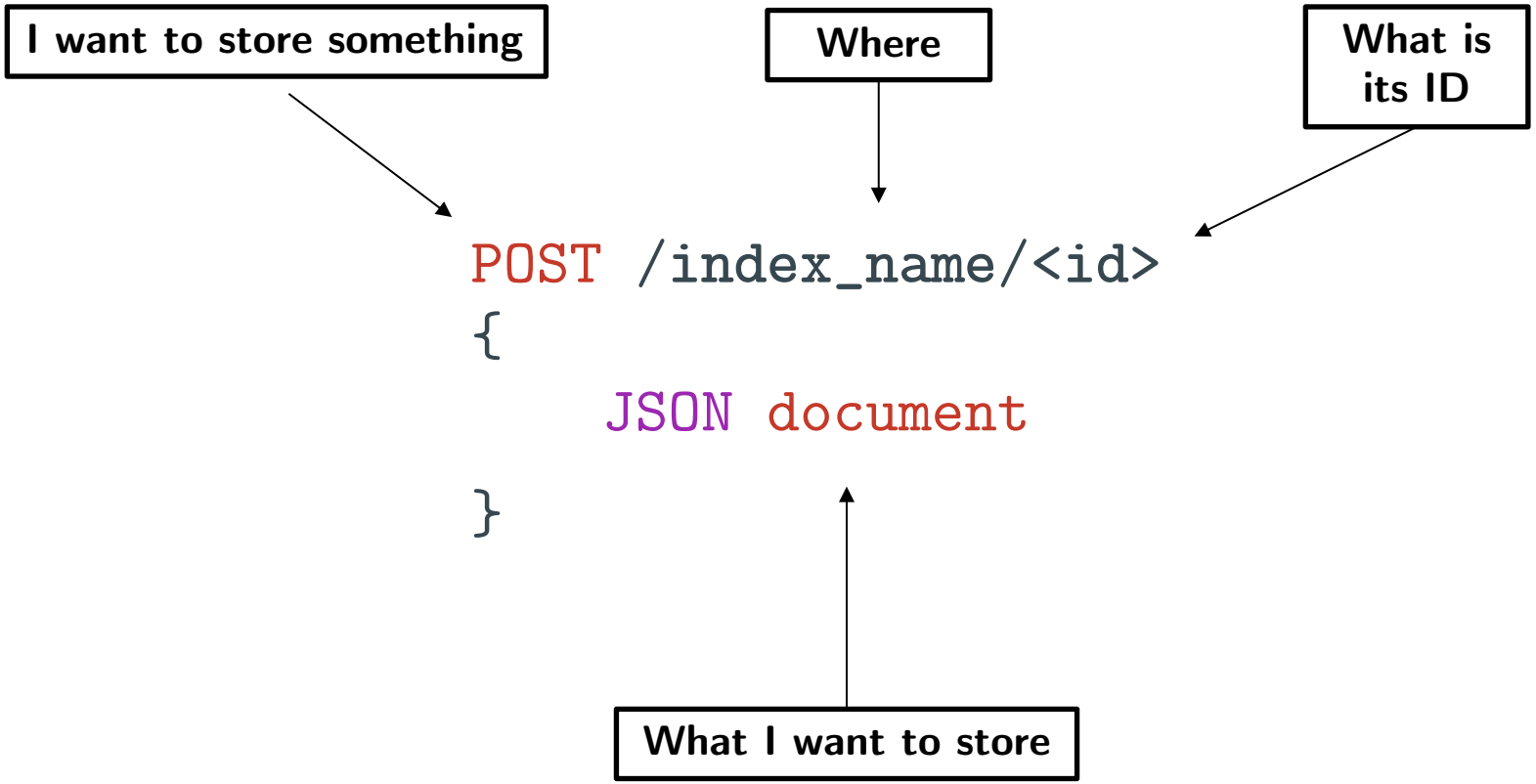
 

# ElasticSearch


- Document-oriented database
  - Indexes complex documents that may contain heterogeneous fields
- It uses Lucene library
  - To generate indexes
- **RESTful API** allows interactions with any programming language
- Each document is stored in a specific index (similar to a specific DB)

# Indexing documents



# Update & Delete

```
PUT index_name/123/_update
{
  "color" : "red",
}
```

```
DELETE index_name/id
```

- Documents are immutable
  - The **update** consists in re-indexing the document itself
- Document **removal** is not immediate

# Document Scoring

- The **score** is the fulcrum of search engine
  - floating-point number
  - Identified by the field `_score`
- Standard scoring function is TF-IDF (or BM25)
  - It is fast to compute
  - Semantic? No!
  - It can be modified
- higher `_score` values correspond to more relevant documents (descending order)

# Custom Scoring

```
GET /_search
{
  "query": {
    "function_score": {
      "query": {
        "match": { "message": "Deep NLP" }
      },
      "script_score": {
        "script": {
          "source": "JS-like formula"
        }
      }
    }
  }
}
```

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-function-score-query.html#function-script-score

# Dense vector field

```
PUT my-index-000001
{
  "mappings": {
    "properties": {
      "my_vector": {
        "type": "dense_vector",
        "dims": 768
      },
      "my_text" : {
        "type" : "keyword"
      }
    }
  }
}

PUT my-index-000001/_doc/1
{
  "my_text" : "text1",
  "my_vector" : [78.5, 45, ...]
}

PUT my-index-000001/_doc/2
{
  "my_text" : "text2",
  "my_vector" : [-0.5, 10, ...]
}
```

```
{
  "script_score": {
    "query": {"match_all": {}},
    "script": {
      "source": "cosineSimilarity(params.query_vector, 'my_vector') + 1.0",
      "params": {"query_vector": query_vector}
    }
  }
}
```

**Encode input at query time**

**Encode documents while indexing**

# elasticsearch-py

- Official low-level client for Elasticsearch.
  - documentation
- It was designed as very thin wrapper around Elasticsearch's REST API
- Very easy to use interface
  - Once you know how ES works

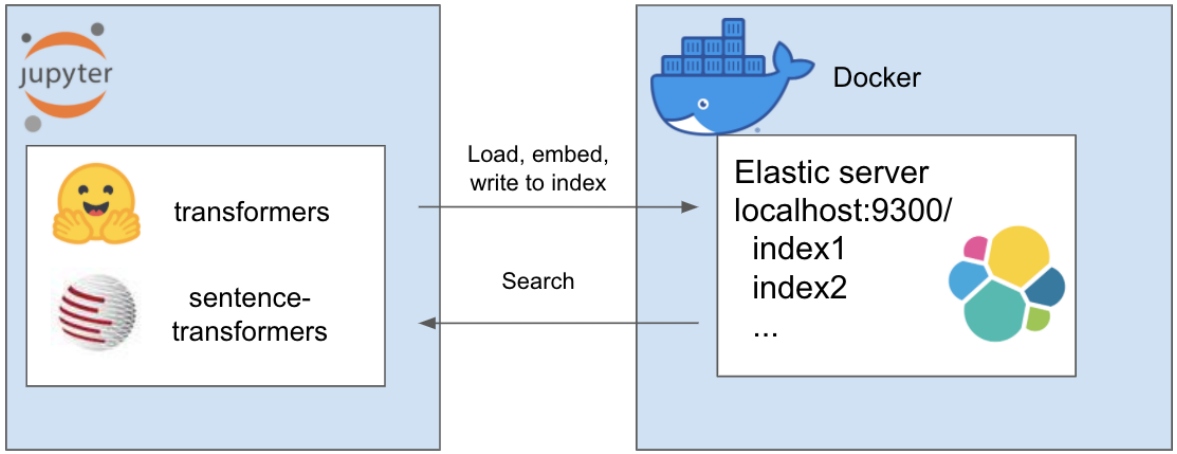
 

# elasticsearch-py

```python
# after pip install elasticsearch
from elasticsearch import Elasticsearch

# create connection
es = Elasticsearch('127.0.0.1', port=9200)

# create index
create_index = es.indices.create(index='DNLP', ignore=400)

personal_doc = {
    "name": "Moreno",
    "surname": "La Quatra",
    "role": "Teacher Assistant",
    "bio": "PhD Student focusing his research in NLP."
}
res = es.index(index="DNLP", body=personal_doc)
```

# ElasticTransformers

**ElasticTransformers** is used to manage ElasticSearch input/output operations. It leverages the sentence-transformers library that can compute sentence embeddings representations based on deep transformer-based models.



# References


**1. [Book]** Schütze, H., Manning, C. D., & Raghavan, P. (2008**). Introduction to information retrieval** (Vol. 39, pp. 234-265). Cambridge: Cambridge University Press.
https://nlp.stanford.edu/IR-book/information-retrieval-book.html

**2. [Paper]** Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). **The PageRank citation ranking: Bringing order to the web**. Stanford InfoLab.
https://www.semanticscholar.org/paper/The-PageRank-Citation-Ranking-%3A-Bringing-Order-to-Page-Brin/eb82d3035849cd23578096462ba419b53198a556

**3. [Paper]** Kleinberg, J. M. (1998, January). **Authoritative sources in a hyperlinked environment**. In SODA (Vol. 98, pp. 668-677).
http://www.cs.cornell.edu/home/kleinber/auth.pdf

**4. [Book]** Gormley, Clinton, and Zachary Tong. **Elasticsearch: the definitive guide: a distributed real-time search and analytics engine.** " O'Reilly Media, Inc.", 2015.
https://www.oreilly.com/library/view/elasticsearch-the-definitive/9781449358532/

**5. Pagerank @ Brown University**
http://cs.brown.edu/courses/cs016/static/files/assignments/projects/GraphHelpSession.pdf

# Exercises

What of the following statement holds **False** for IR systems?

A. In the binary model queries are defined using boolean expressions.

B. The dictionary of terms for the inverted index is built by defining the set of unique words in thecorpus.

C. Recall is not used for evaluating IR systems.

D. Vector Space Models represent both document and queries as vector of weights.

E. TF-IDF can be used to obtain document-query matching score.

# Exercises

What of the following statement holds **False** for IR systems?

A. In the binary model queries are defined using boolean expressions.

B. The dictionary of terms for the inverted index is built by defining the set of unique words in thecorpus.

C. Recall is not used for evaluating IR systems.

**It is one of the metric used for the evaluation**

D. Vector Space Models represent both document and queries as vector of weights.

E. TF-IDF can be used to obtain document-query matching score.

# Exercises

Considering the HITS algorithm, during the HUB update step:

A. Each node relevance score is normalized by the number N of nodes.

B. Each link relevance score is normalized by the number N of nodes.

C. For each node, the hub score is the sum of the authority scores of each node that it points to.

D. For each node, the hub score is the sum of the hub scores of each node that points to it.

E. None of the above.

# Exercises

Considering the HITS algorithm, during the HUB update step:

A. Each node relevance score is normalized by the number N of nodes.

B. Each link relevance score is normalized by the number N of nodes.

C. For each node, the hub score is the sum of the authority scores of each node that it points to.

D. For each node, the hub score is the sum of the hub scores of each node that points to it.

E. None of the above.

# Thank you!

TA contact: lorenzo.vaiani@polito.it