67-300 SEARCH ENGINES

VECTOR SPACE MODEL

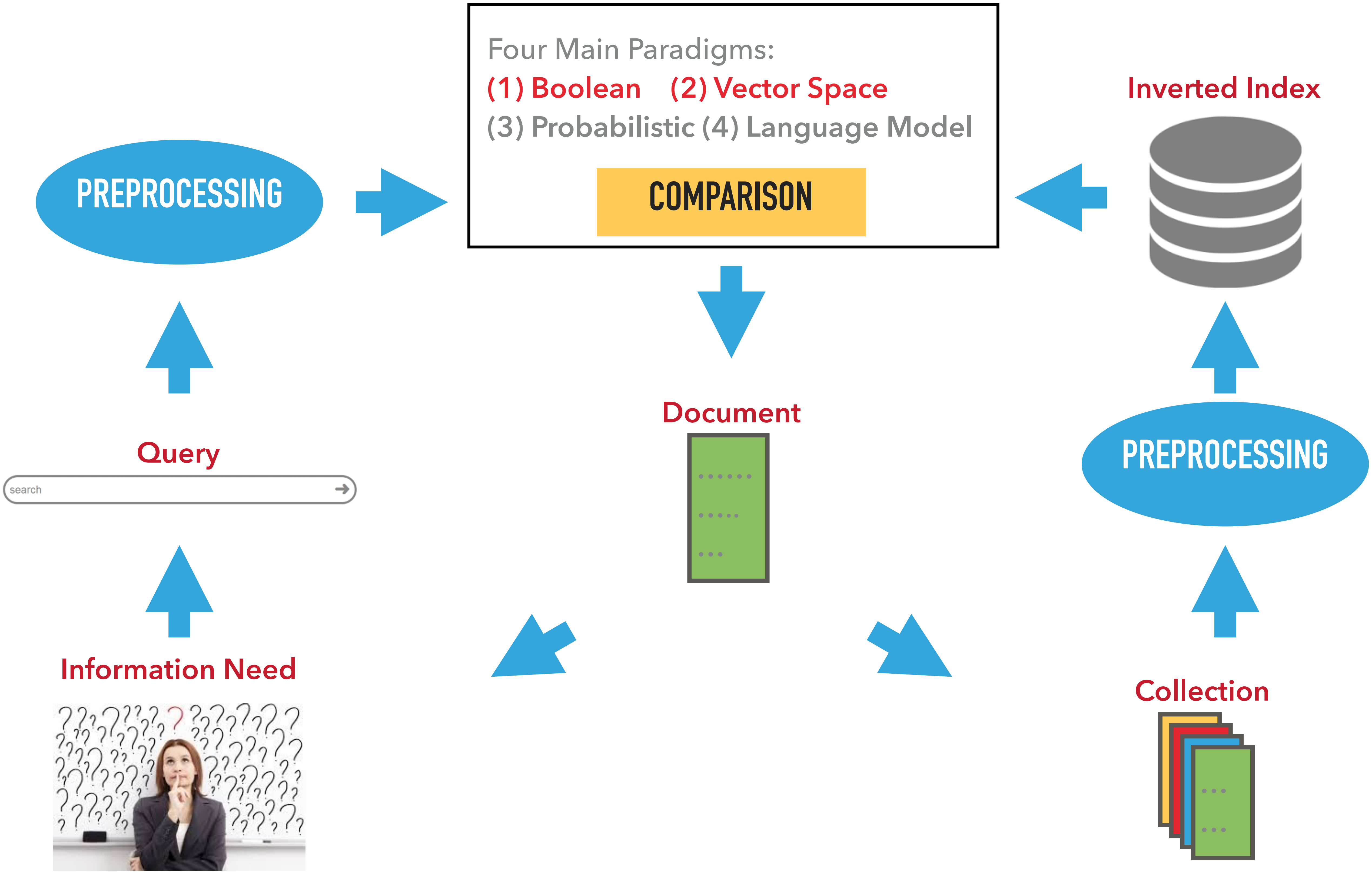
LECTURER: JOAO PALOTTI (JPALOTTI@ANDREW.CMU.EDU)

20TH MARCH 2016

LECTURE GOALS

- Drawbacks of Boolean Search Model
- Vector Space Model
- TD-IDF weighting

CLASS 3 - VECTOR SPACE MODEL

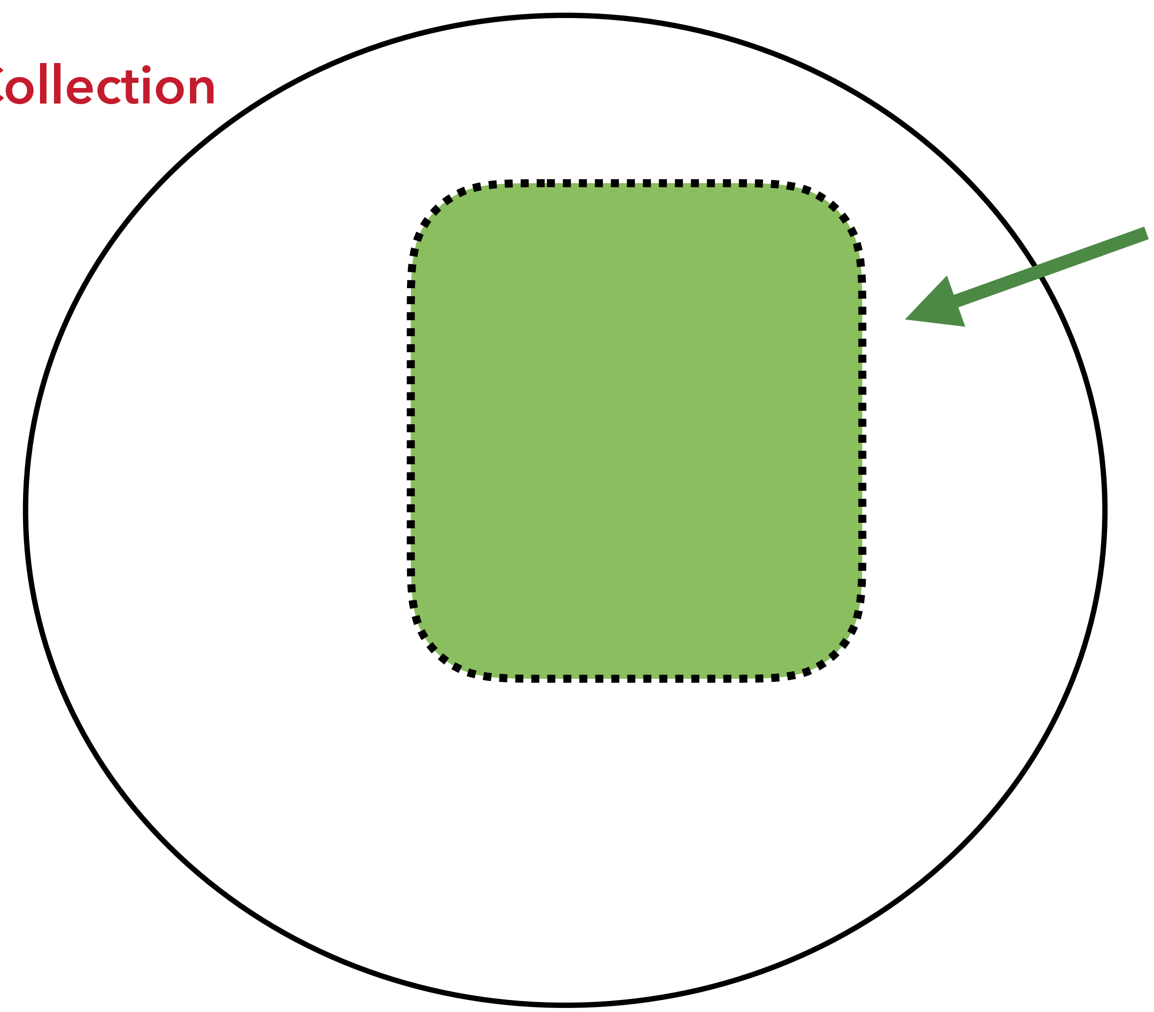


IR Book: Chapters 1 and 2

RECAP: BOOLEAN SEARCH MODEL

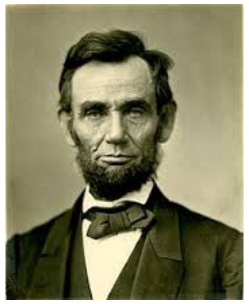
- Researching on US Republican Party
- Example: “lincoln”

Collection



Documents that contain token “lincoln”

LINCOLN



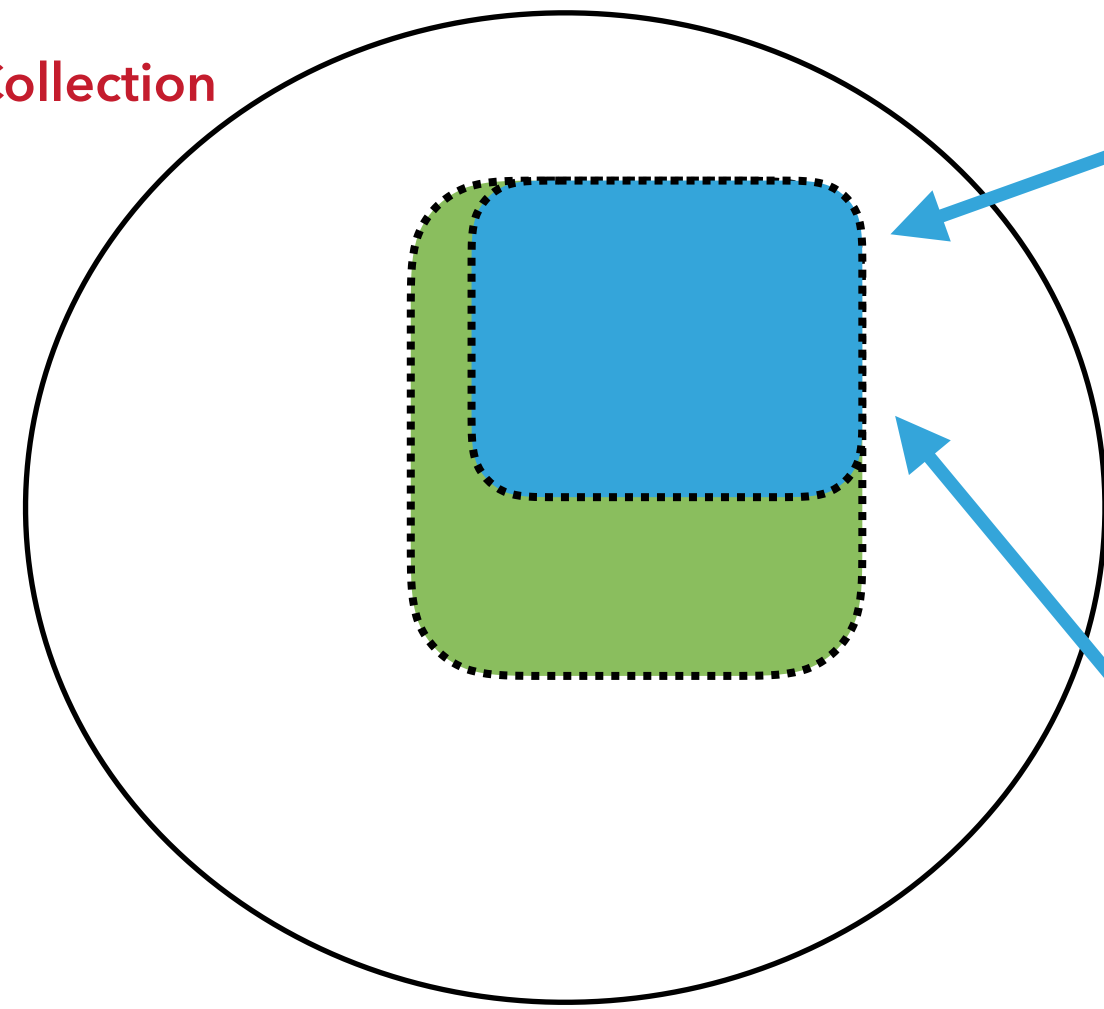
RECAP: BOOLEAN SEARCH MODEL

- Researching on US past presidents
- Example: “lincoln AND president”

RECAP: BOOLEAN SEARCH MODEL

- New Query: “lincoln AND president”

Collection



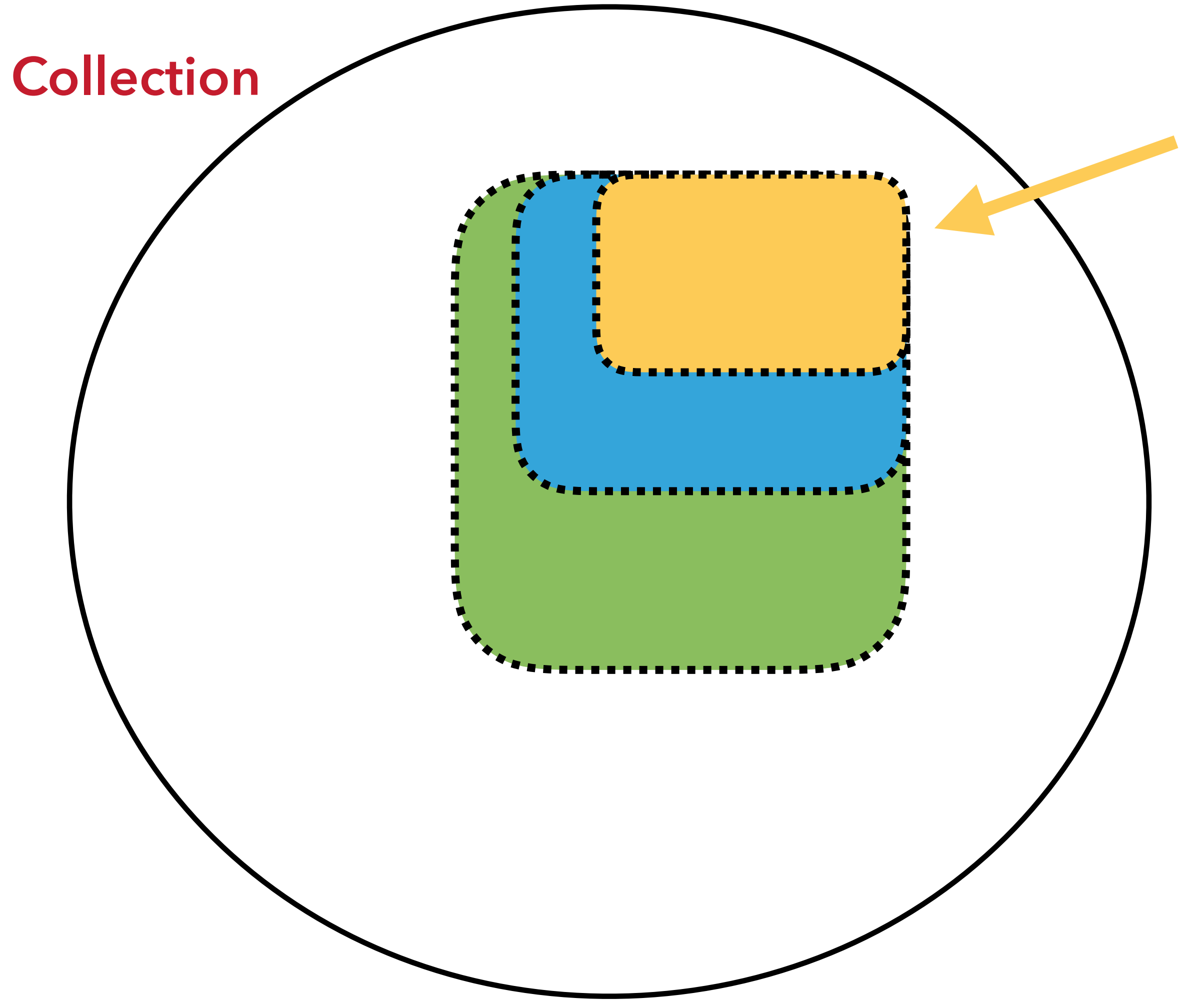
Documents that contain tokens “lincoln” and “president”

Ford Motor Company today announced that John Smith will succeed Frank K. as **president** of **Lincoln** Mercury

Still needs to refine query…

RECAP: BOOLEAN SEARCH MODEL

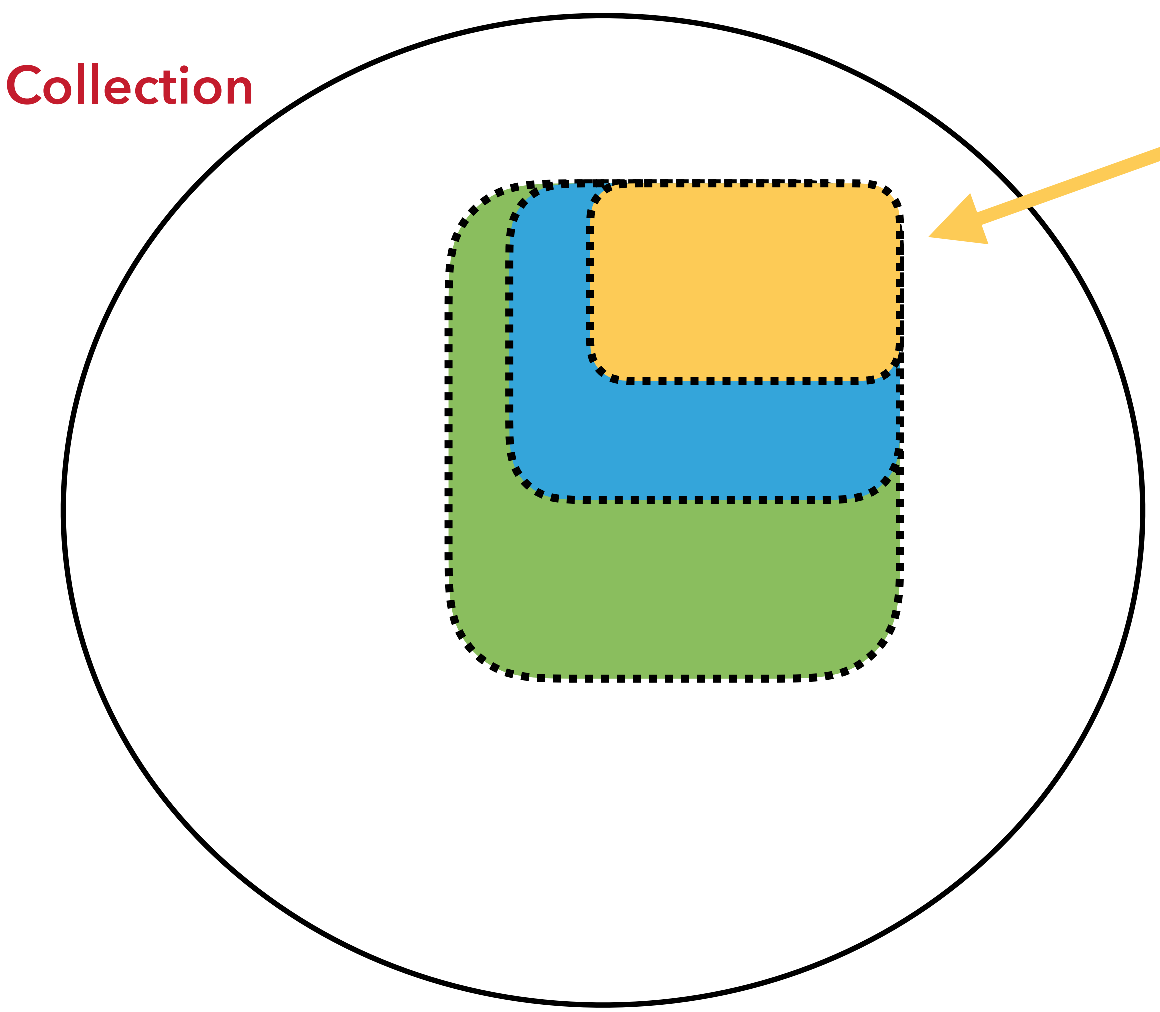
“lincoln AND president AND NOT(automobile OR car)”



Documents that contain tokens “lincoln” and “president” and do not contain “automobile” nor “car”

RECAP: BOOLEAN SEARCH MODEL

"lincoln AND president AND NOT(automobile OR car)"



Documents that contain tokens "lincoln" and "president" and do not contain "automobile" nor "car"

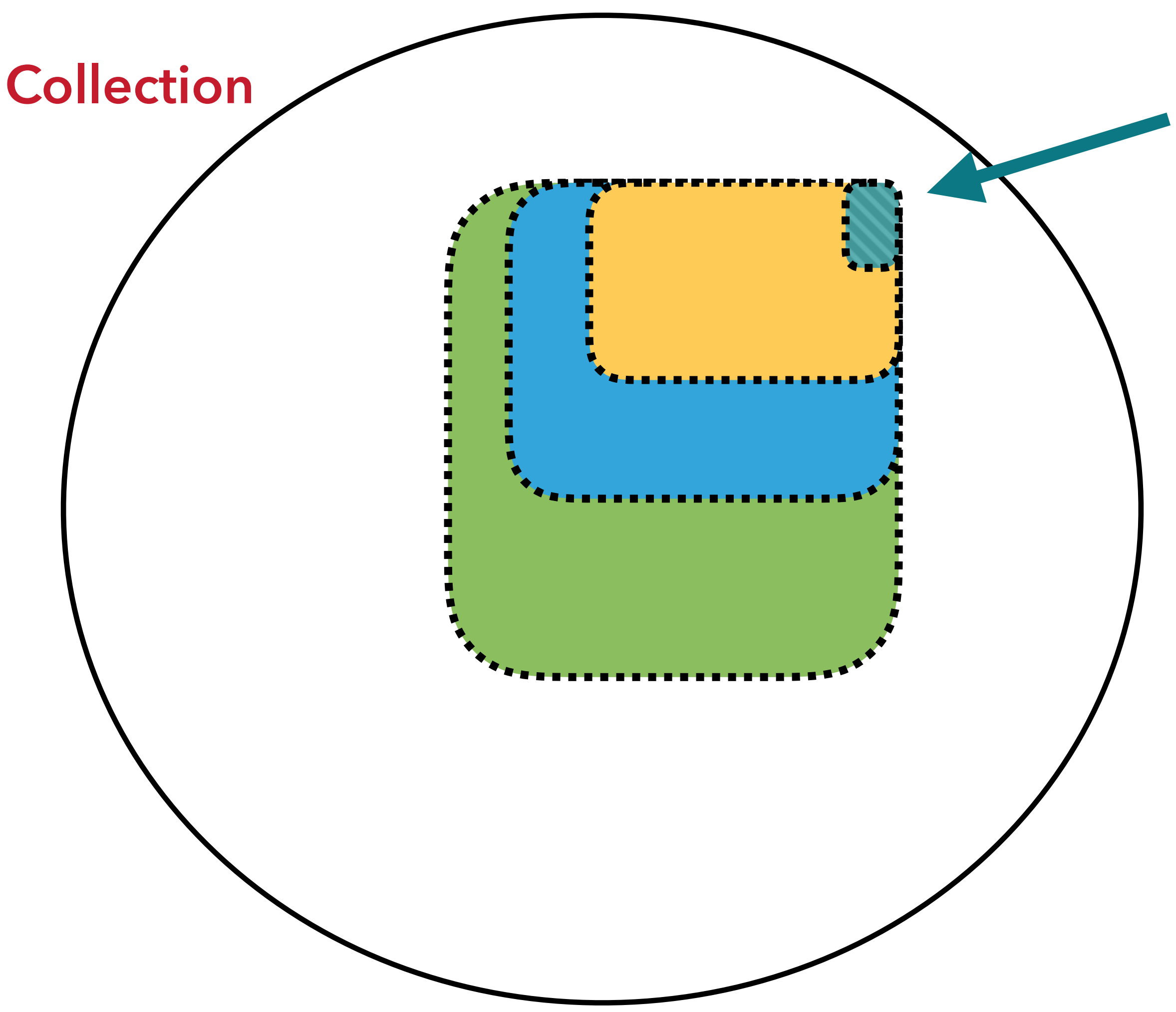
Lincoln's body departs Washington in a nine-**car** funeral train

- Some important documents are going to be missed.

- We still have a large number of documents to look at

RECAP: BOOLEAN SEARCH MODEL

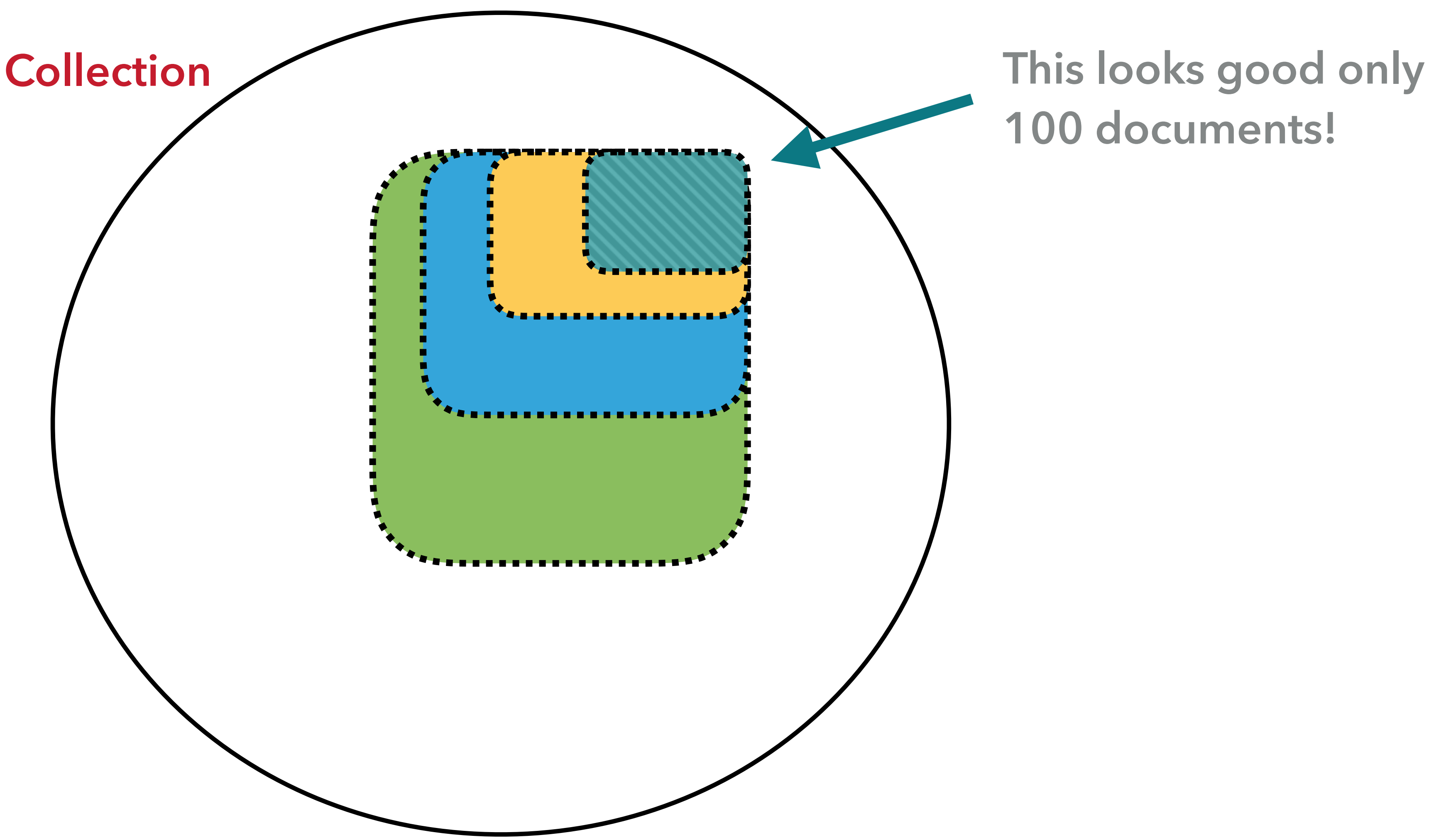
"lincoln AND president AND biography AND life AND birthplace AND gettysburg AND NOT(automobile OR car)"



Now we have only a very very small fraction of documents. We maybe were **too restricted!**

RECAP: BOOLEAN SEARCH MODEL

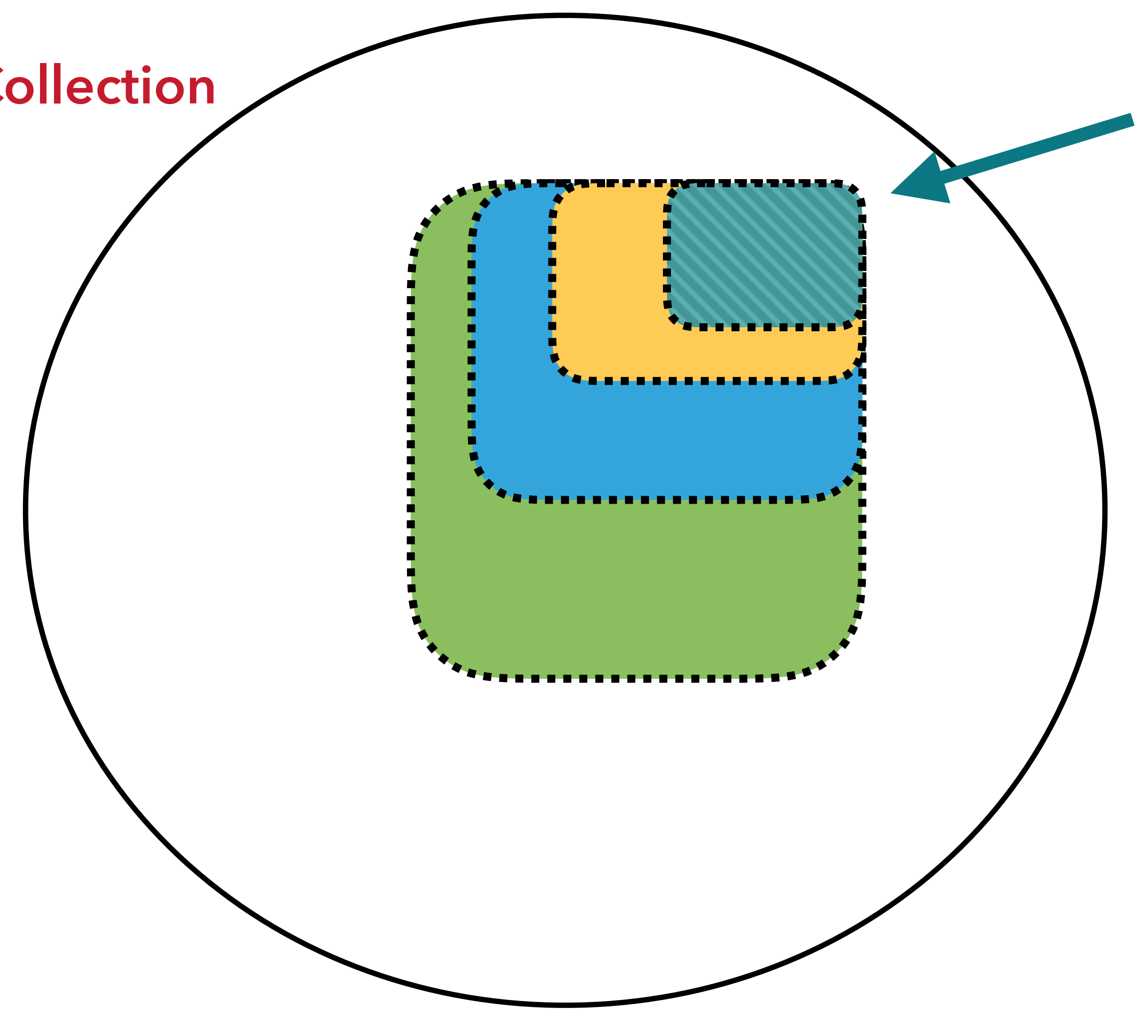
“lincoln AND president AND (biography OR life OR birthplace OR gettysburg) AND NOT(automobile OR car)”



RECAP: BOOLEAN SEARCH MODEL

“lincoln AND president AND (biography OR life OR birthplace OR gettysburg) AND NOT(automobile OR car)”

Collection



This looks good only 100 documents!

But…

President’s Day - Holiday activities - craft, mazes, word searches,… “The life of Washington”. Abraham Lincoln research institute open its doors this Tuesday.

could have been the first or the last document in the result set.

DRAWBACKS OF BOOLEAN SEARCH MODEL

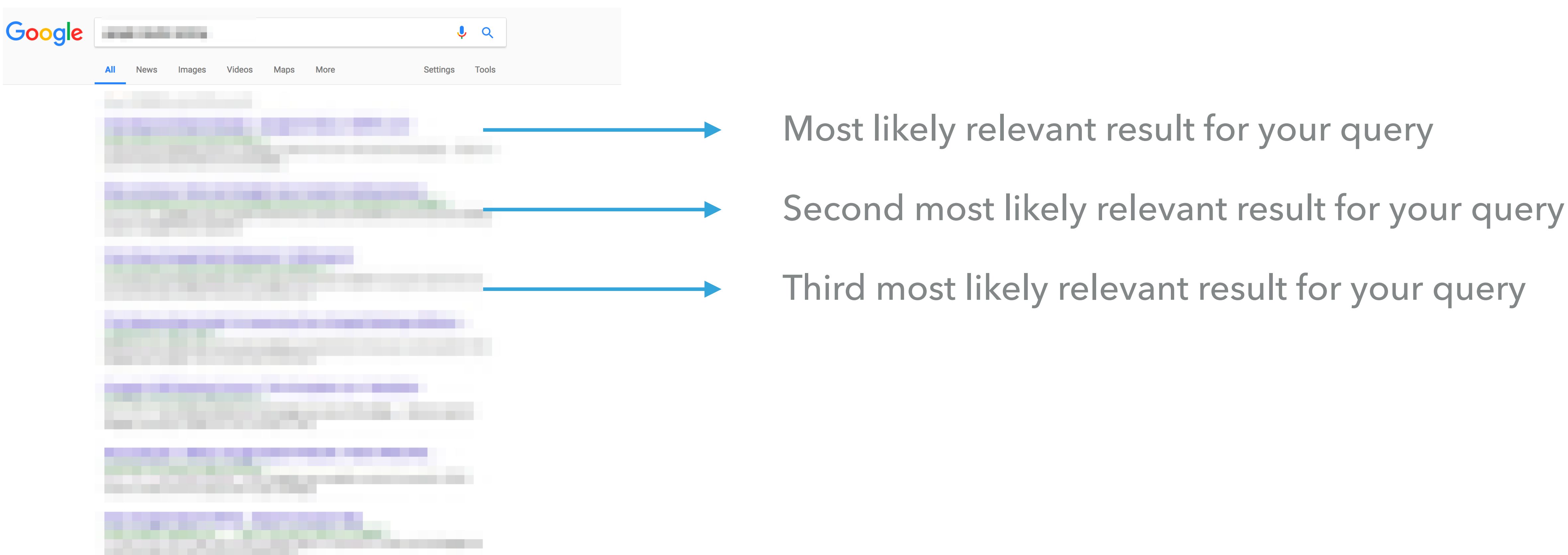
- Query is unlikely precise:
  - Over-constrained - too specific
  - Under-constrained - too broad
  - Very hard to balance these two extremes
- Even when balance is found not all documents are equally relevant!

GOAL OF A SEARCH ENGINE IS TO FIND ALL AND ONLY THE **RELEVANT DOCUMENTS** IN A COLLECTION GIVEN A PARTICULAR USER WITH A PARTICULAR INFORMATION NEED EXPRESSED IN A QUERY

IR Theorists

PROBABILITY RANKING PRINCIPLE

- A retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request.



VECTOR SPACE MODEL

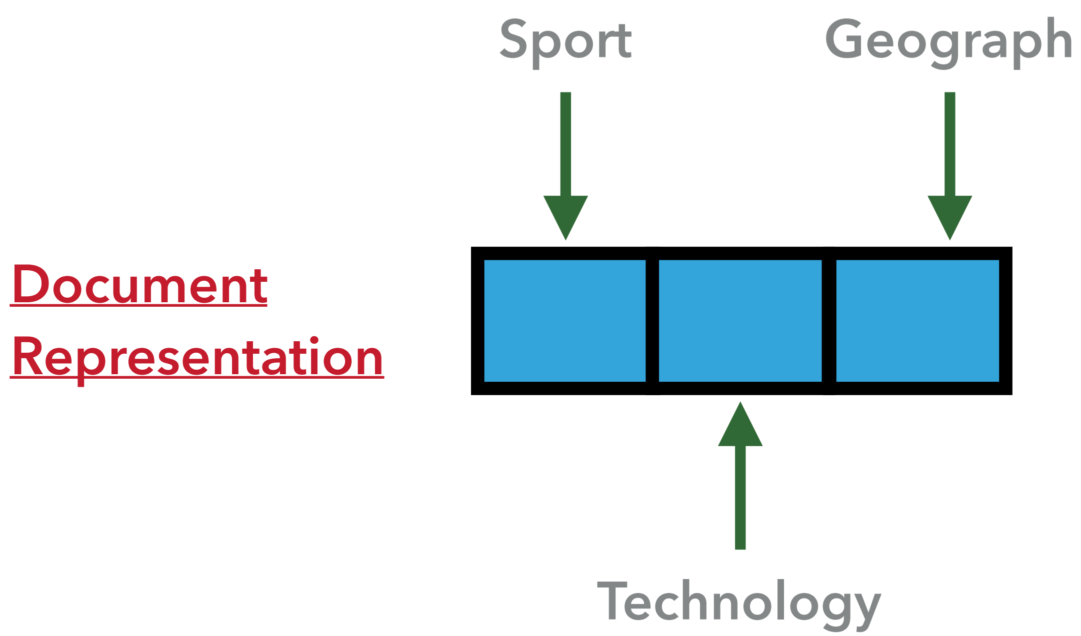
ASSUMPTIONS

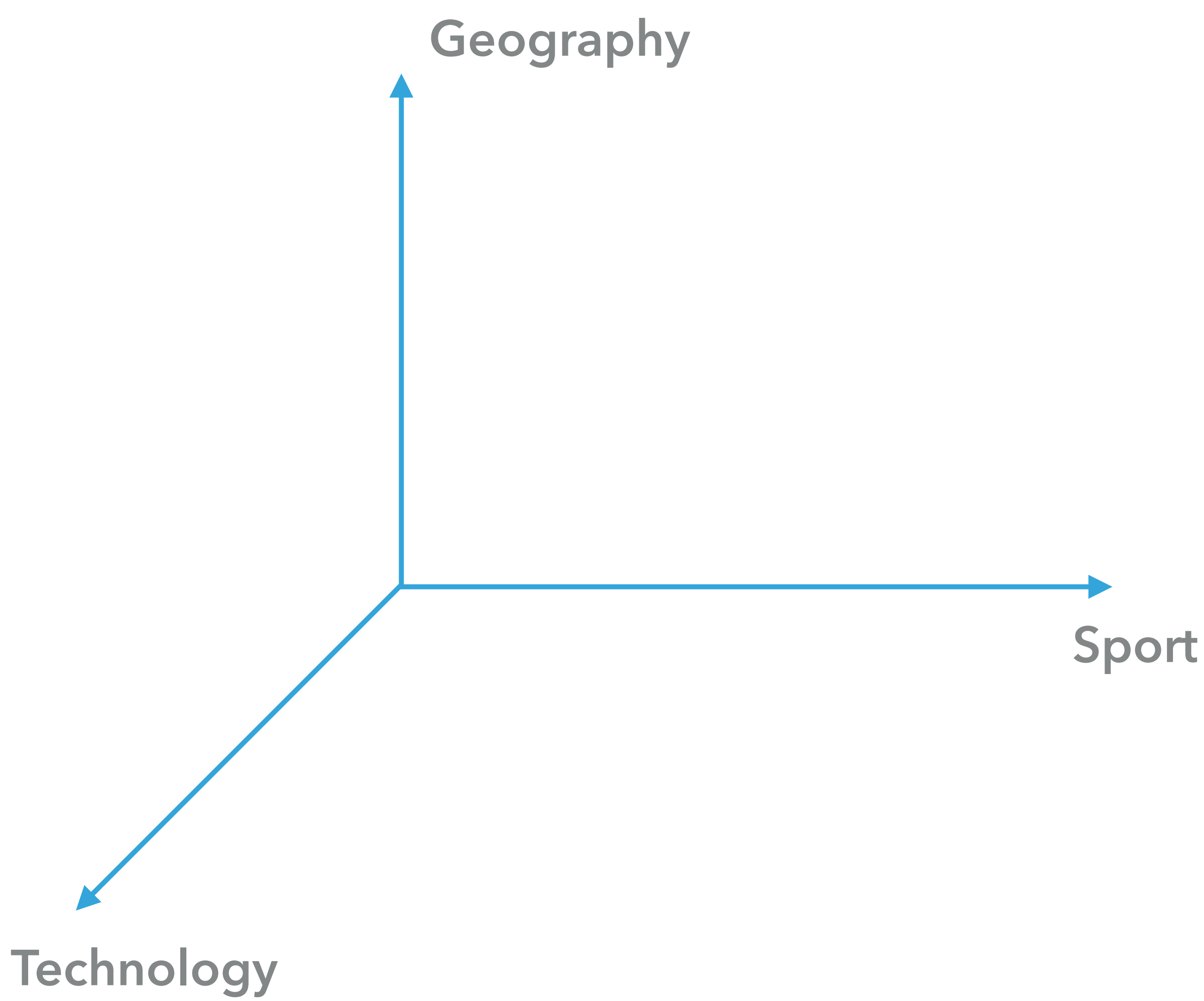
- Assume relevant documents are similar documents:
  - Relevance(d,q) == Similarity(d,q)

- Definition requires two components:
  1. Representation
  2. Similarity measure

REPRESENTATION

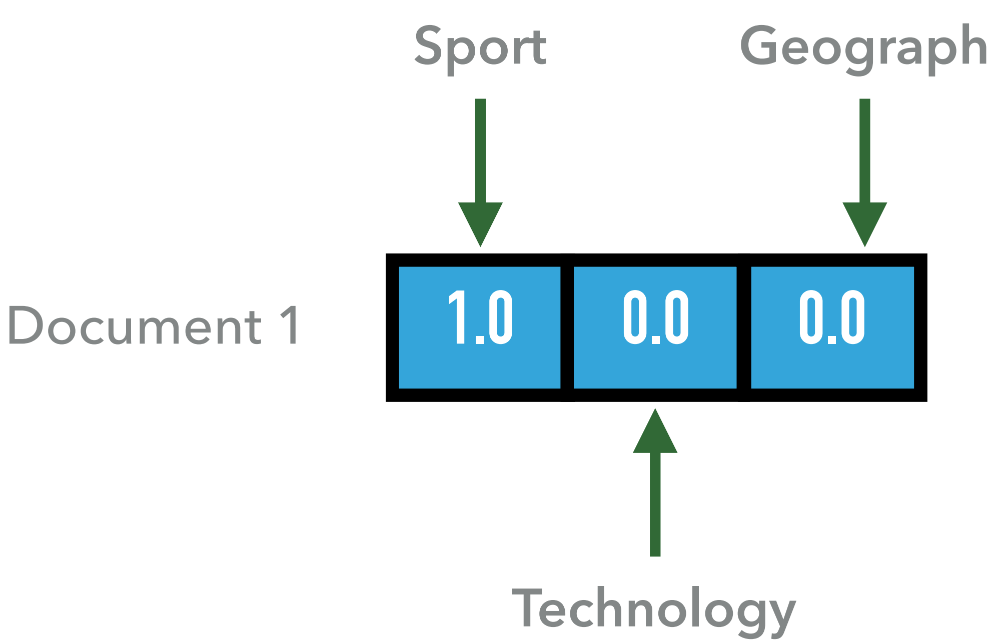
- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:

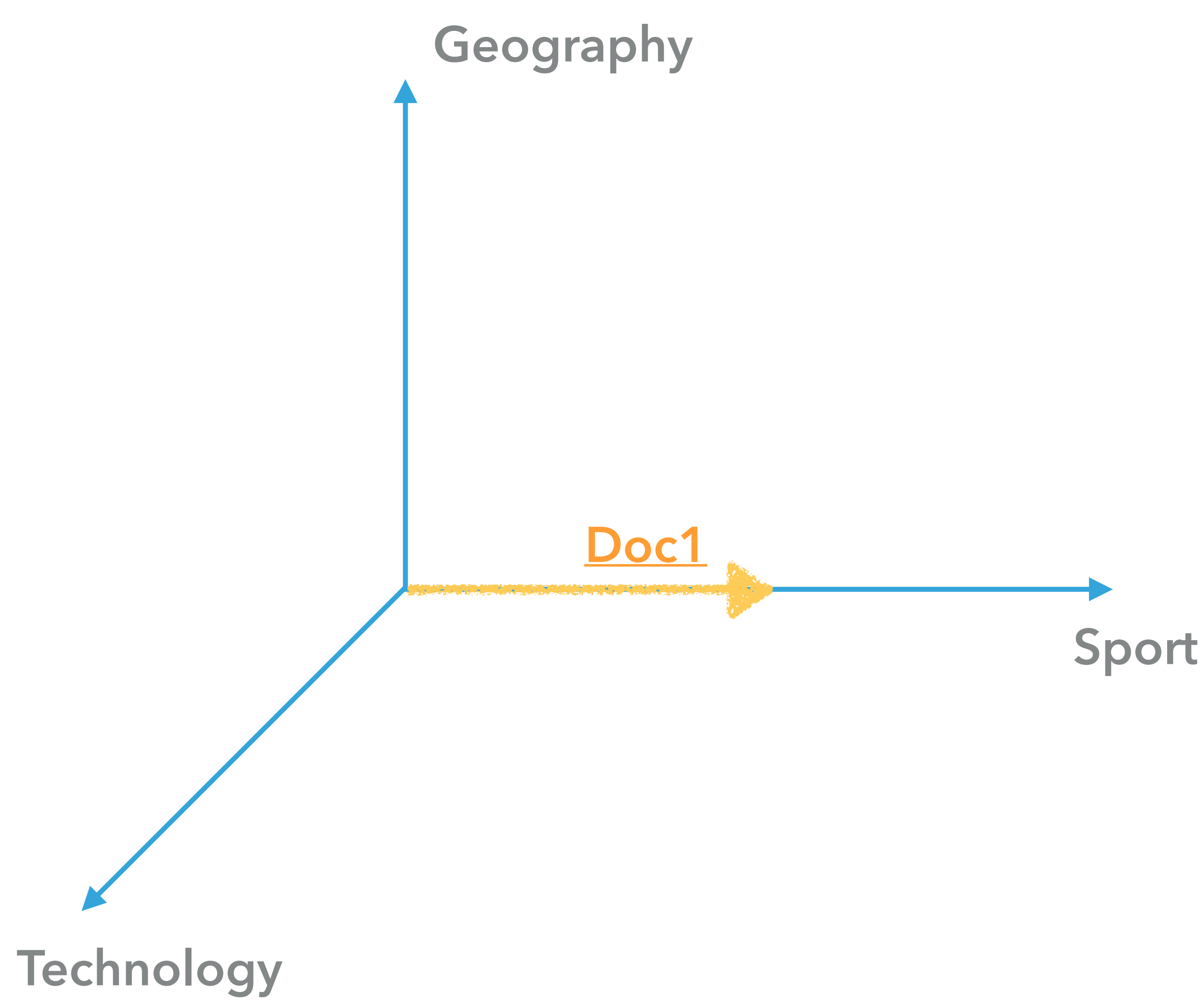




REPRESENTATION

- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:



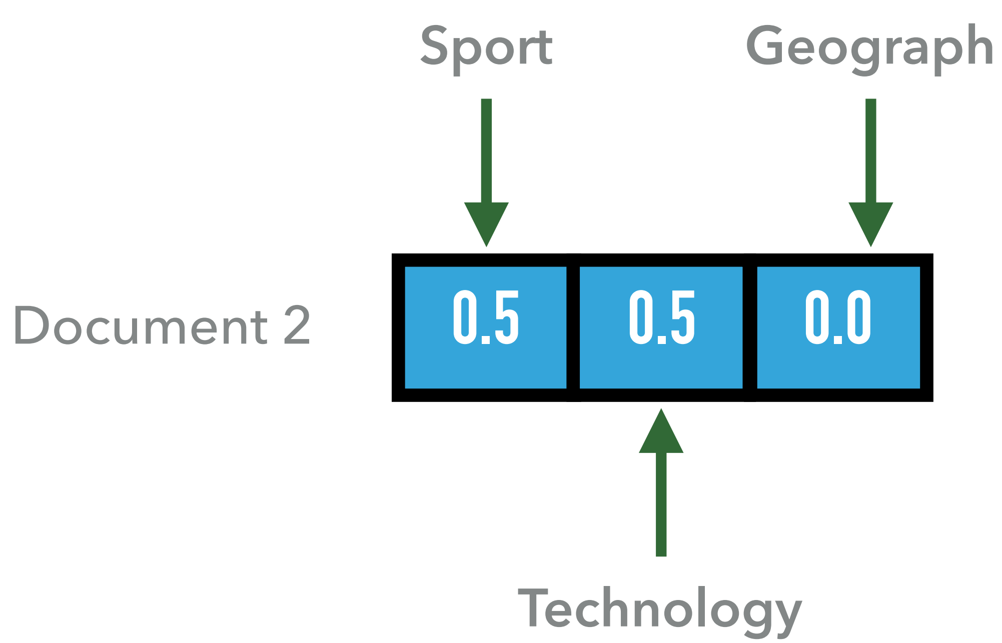


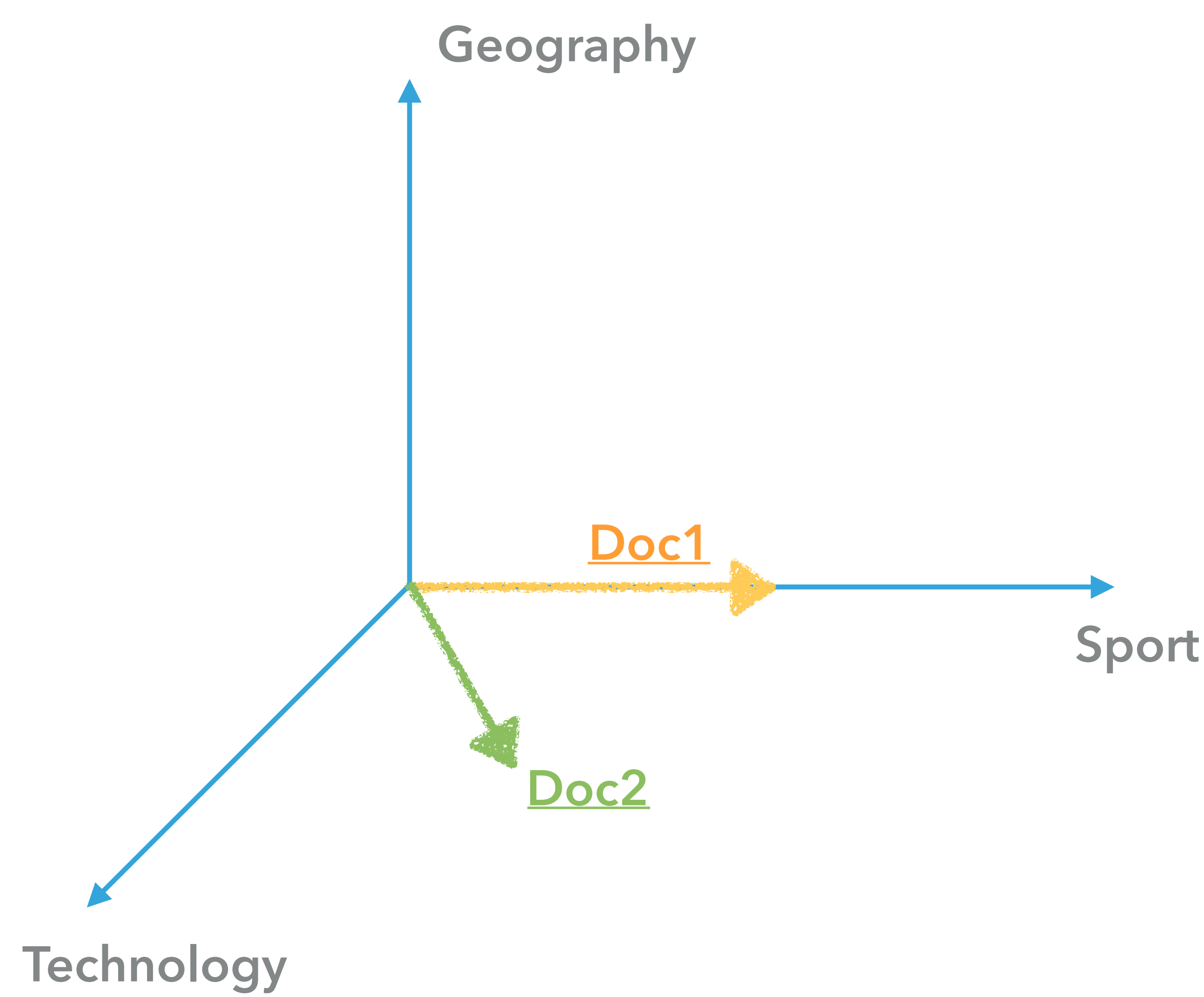
Document 1

"Running is a good exercise for your health"

REPRESENTATION

- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:



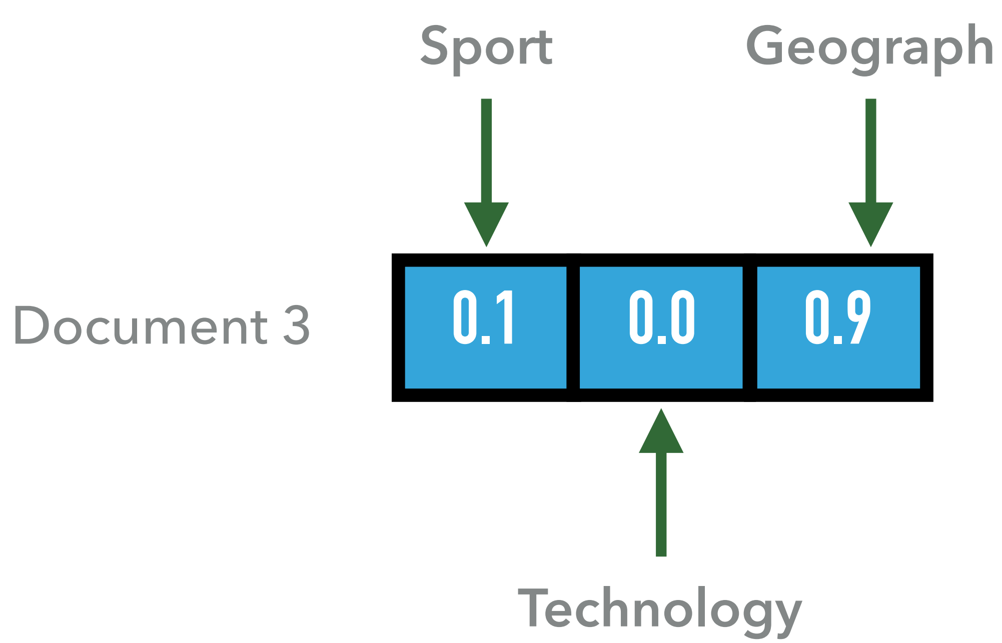


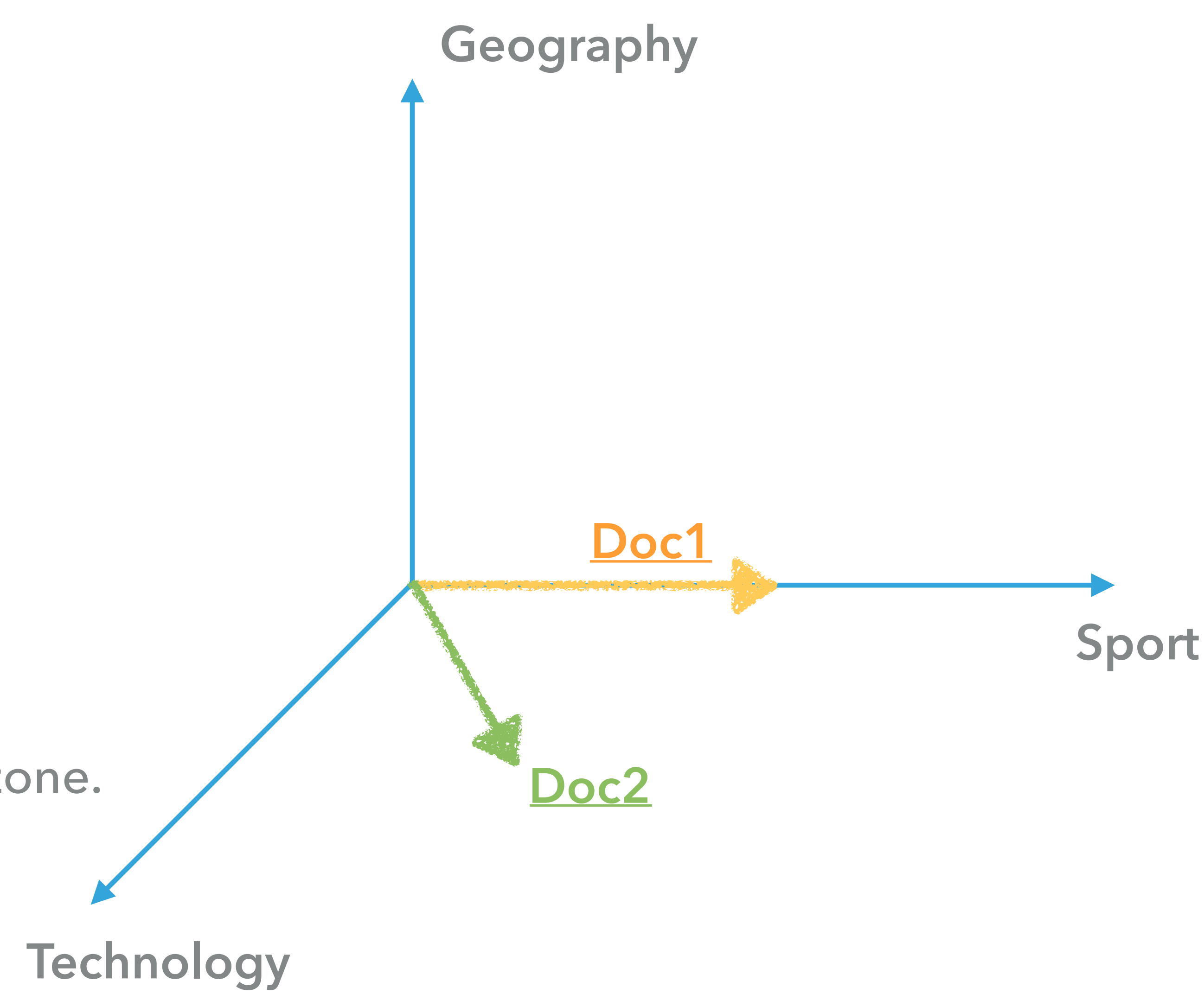
Document 2

"The use of sensors and big data
will change how football is played."

REPRESENTATION

- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:



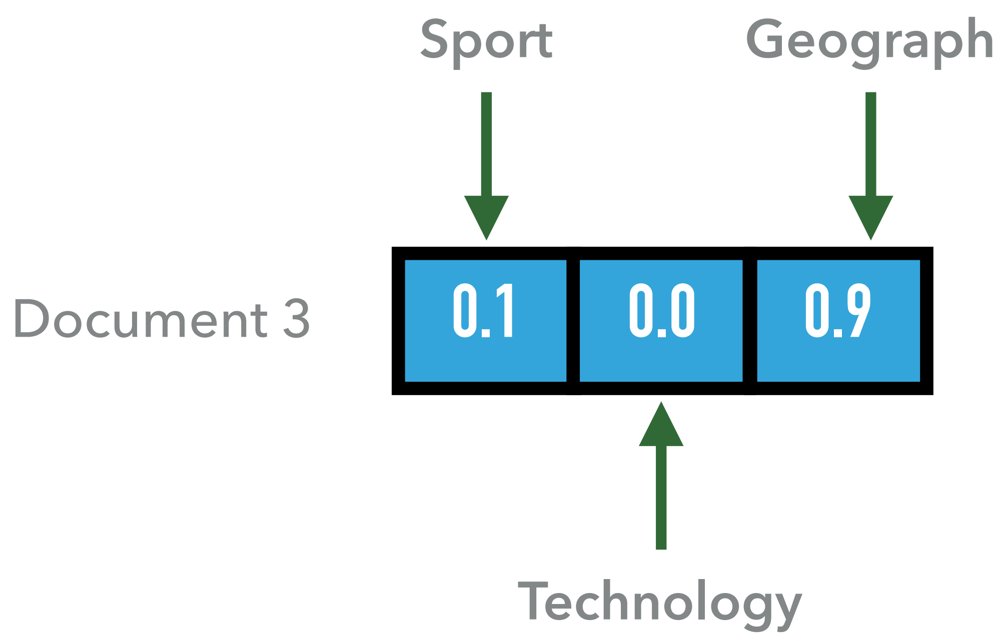


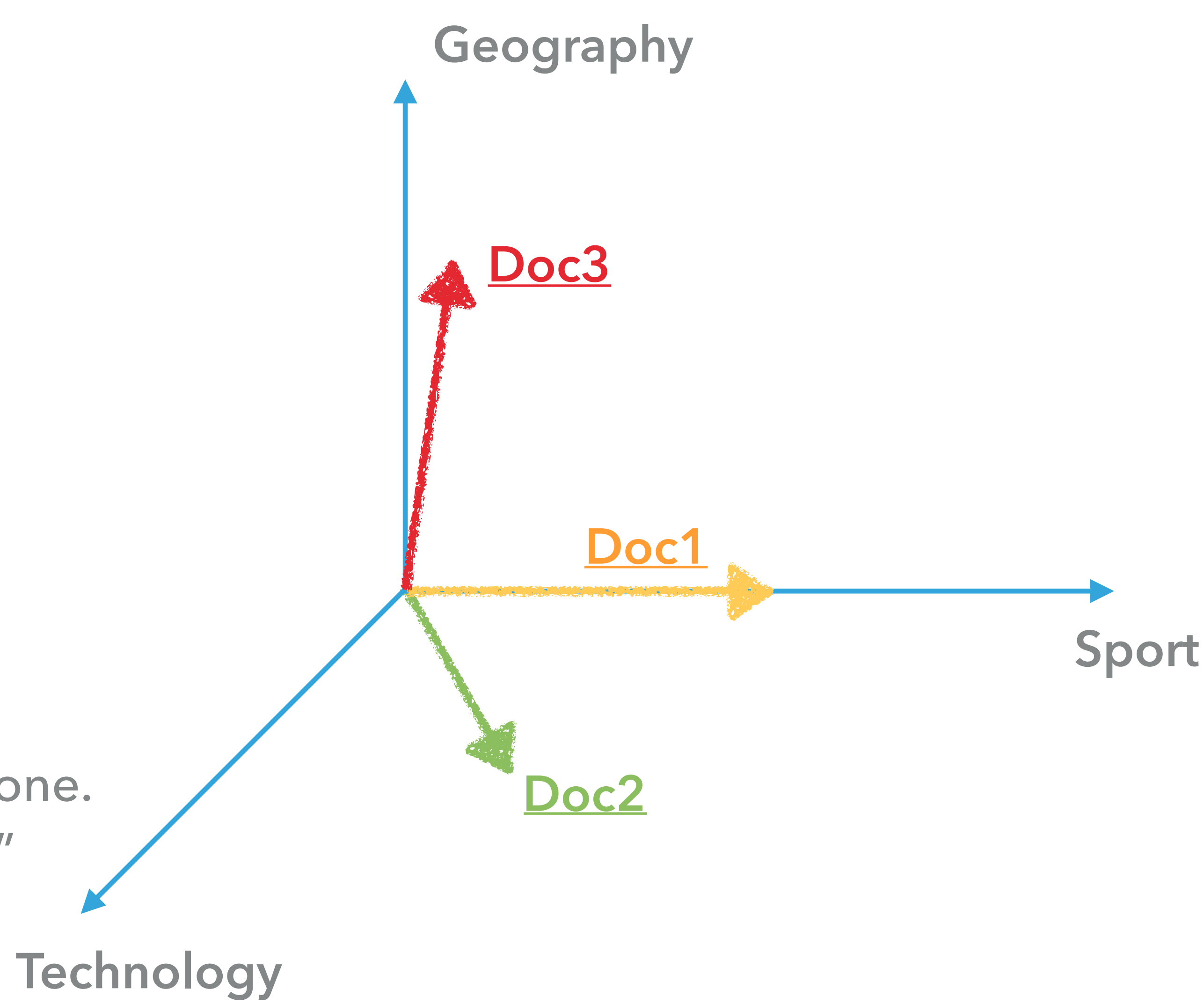
Document 3

"The golf championship will happen in a war zone. The war between Country A and Country B…"

REPRESENTATION

- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:



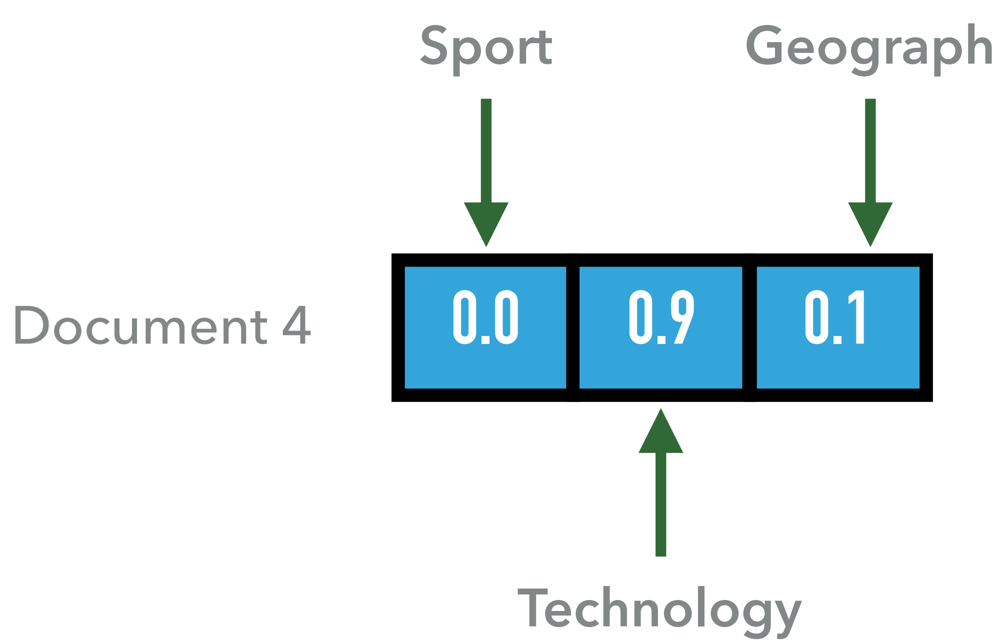


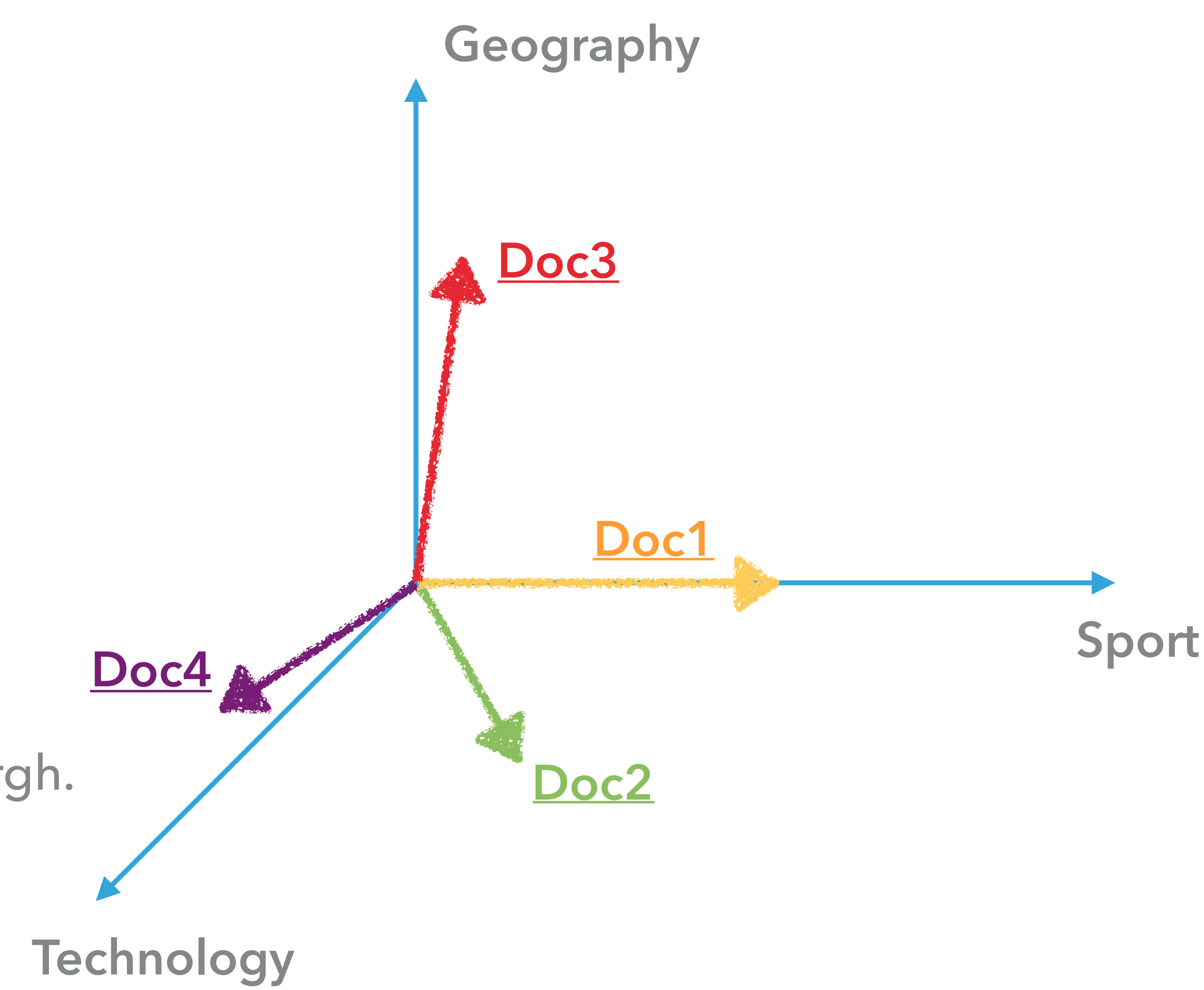
Document 3

"The golf championship will happen in a war zone. Both Country A and Country B are in way that.."

REPRESENTATION

- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:



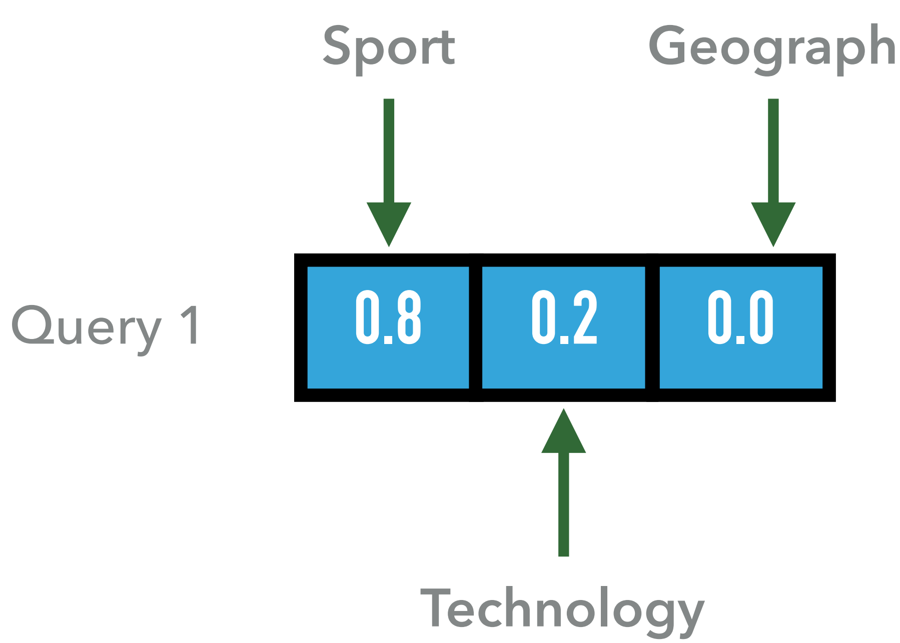


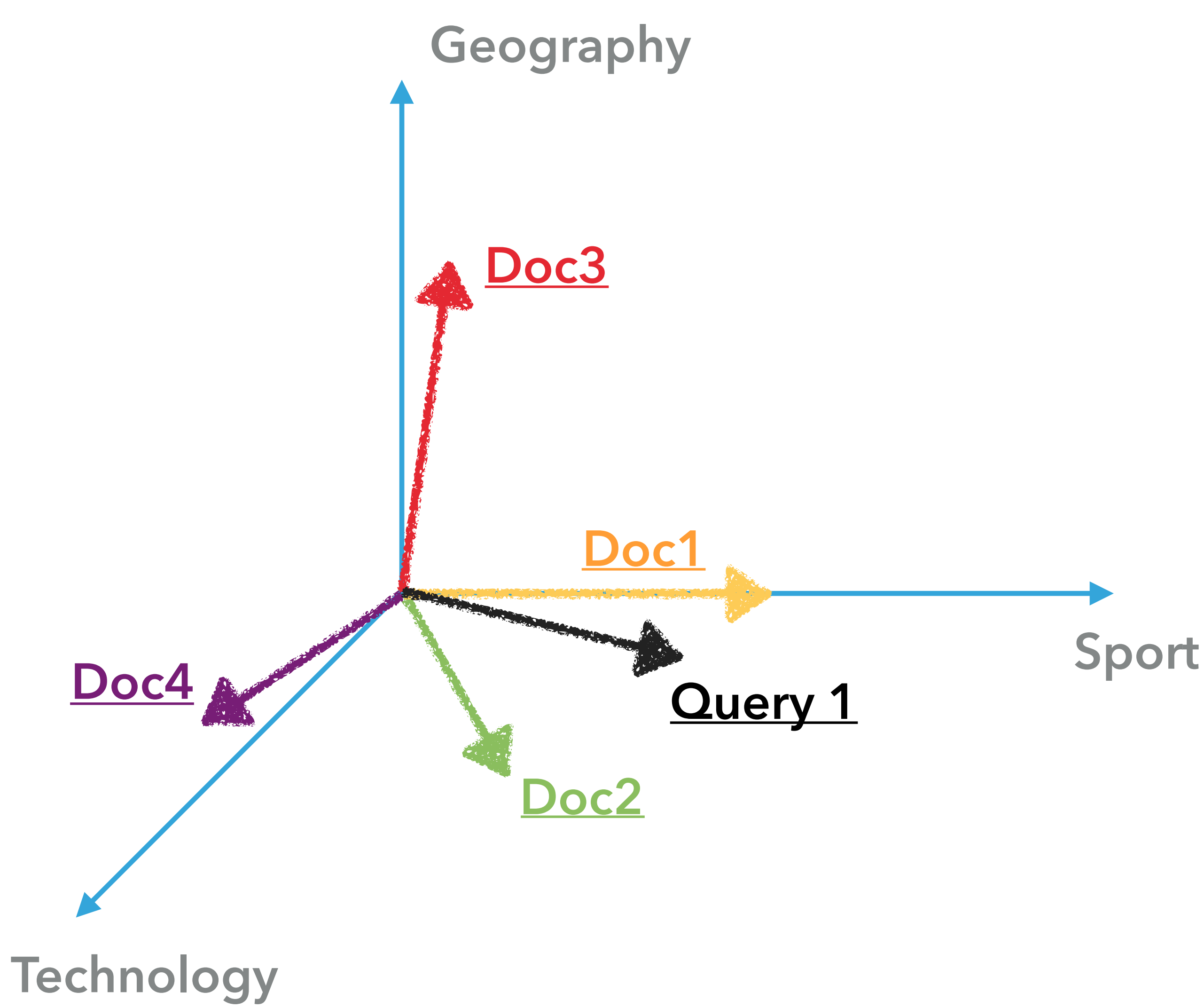
Document 4

"Uber driverless cars are being tested in Pittsburgh. This technology is based on the advances of…"

REPRESENTATION

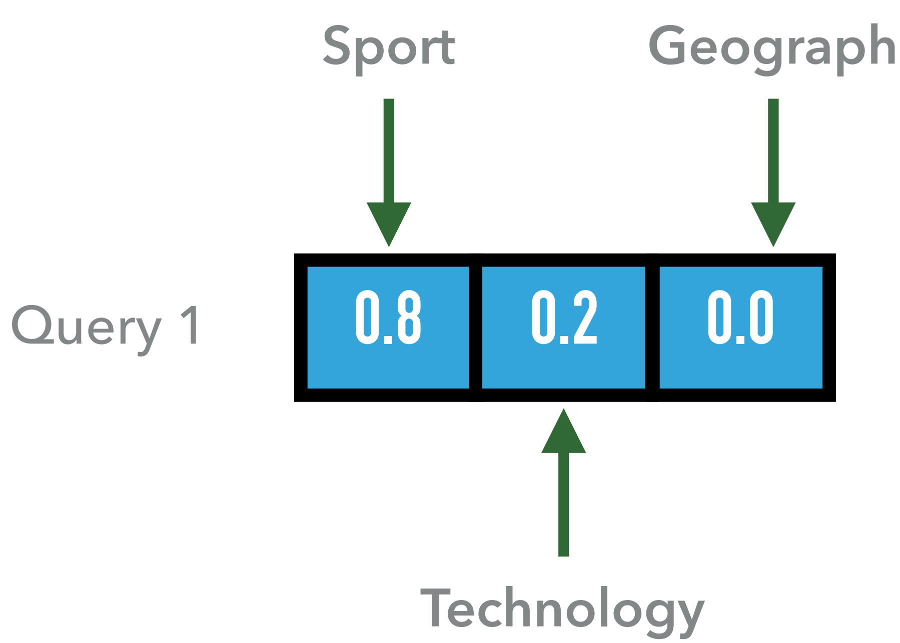
- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:

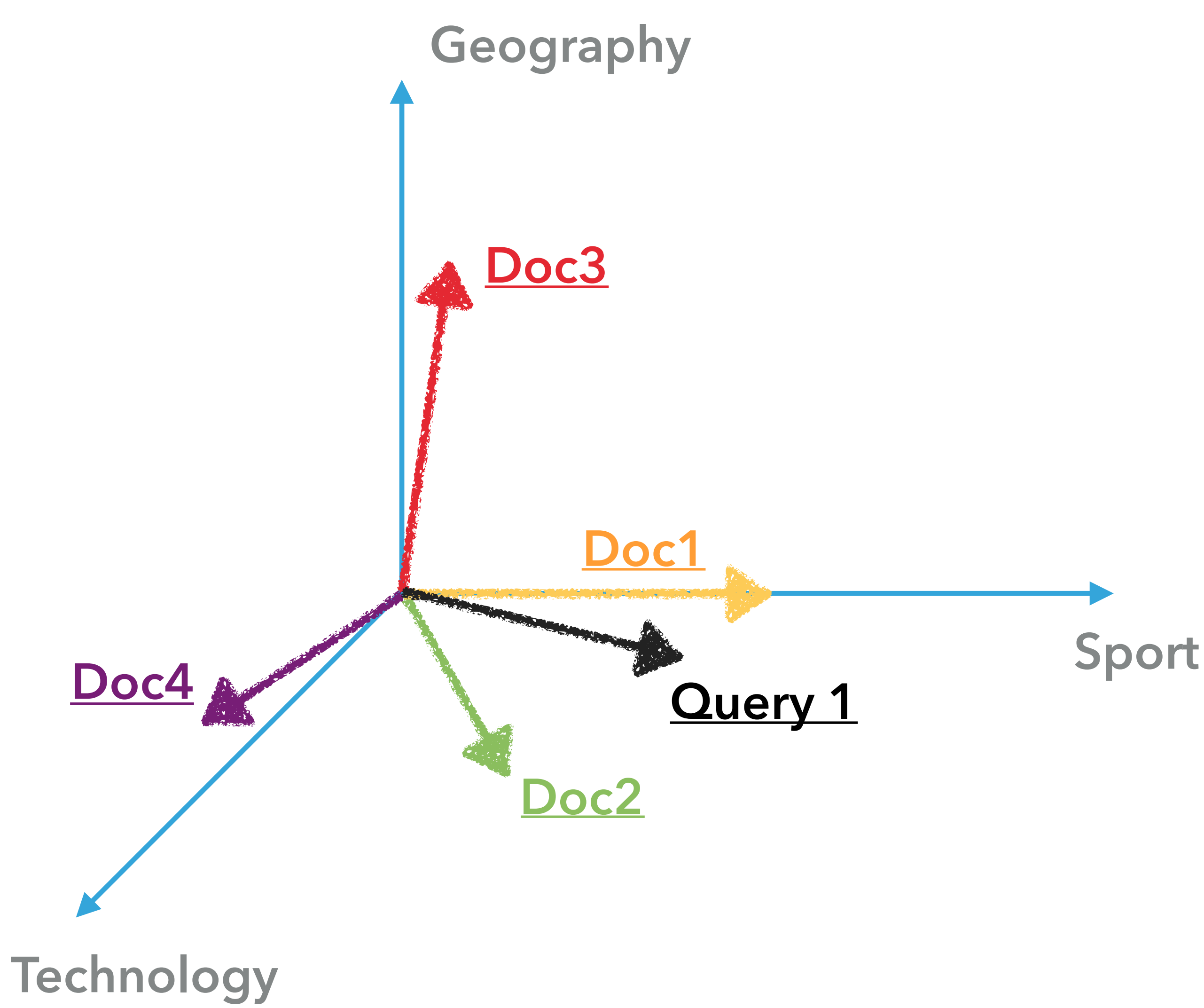




REPRESENTATION

- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:





How far are the documents from Query 1?

Which is the closest documents from Query 1?

Which is the furthest documents from Query 1?

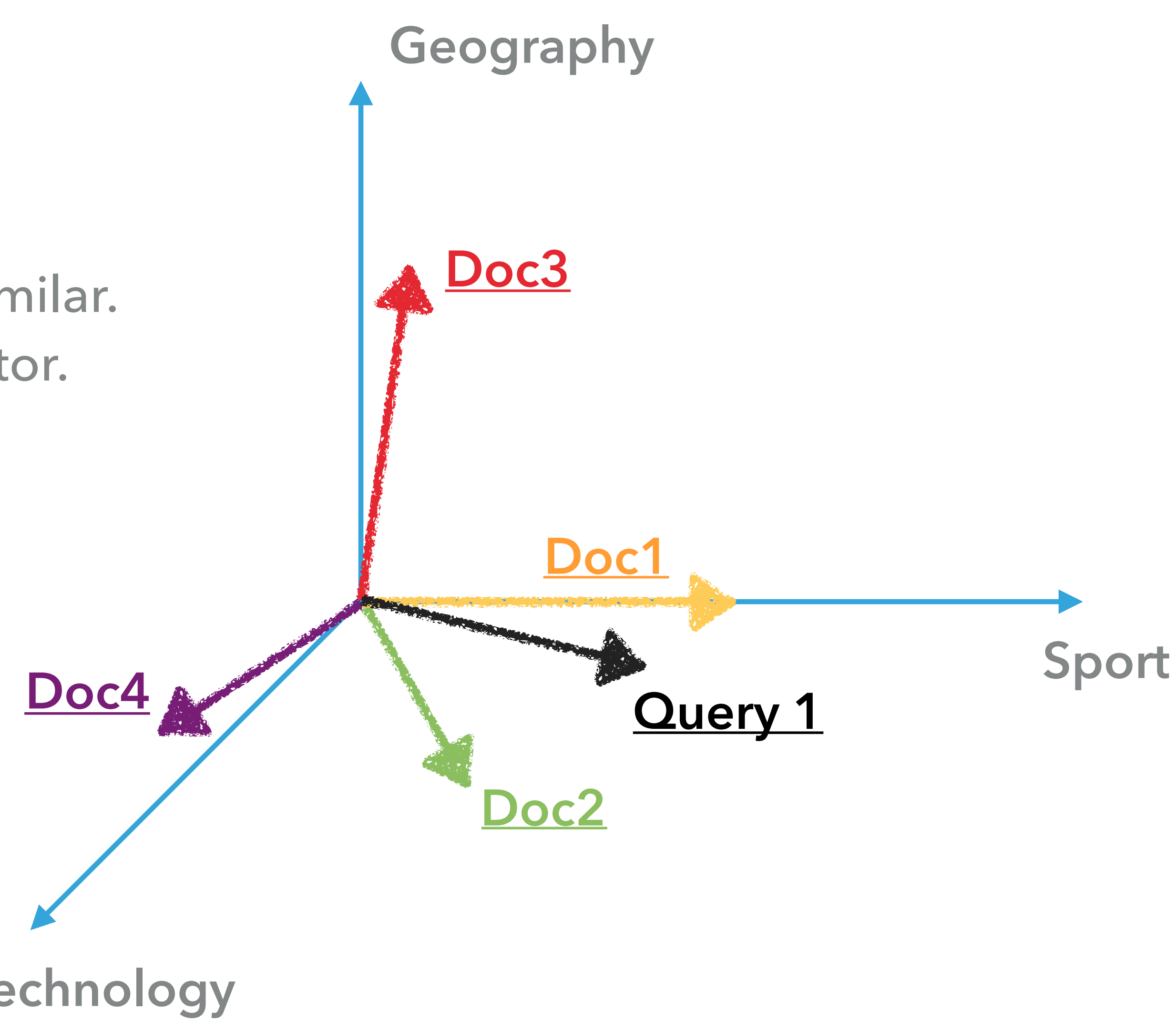
REPRESENTATION

- Documents and queries are represented as concept vectors. Each concept is a cell in a vector:

According to the Probability Ranking Principle:
the most relevant is ranked first.

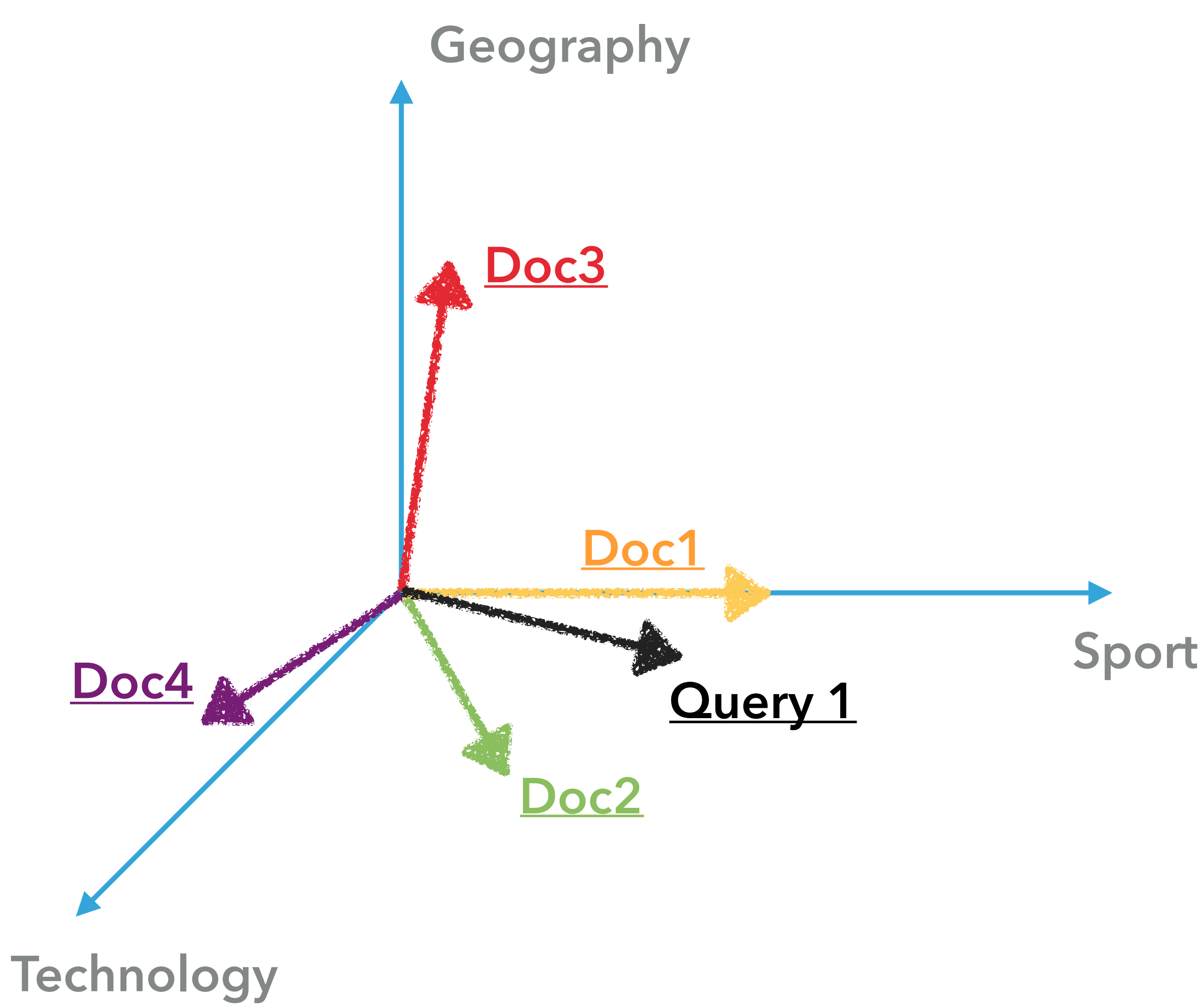
According to VSM the most relevant is the most similar.
We assume that the most similar is the closest vector.
Ranked by “distance”:

Doc1
Doc2
Doc4
Doc3



REPRESENTATION

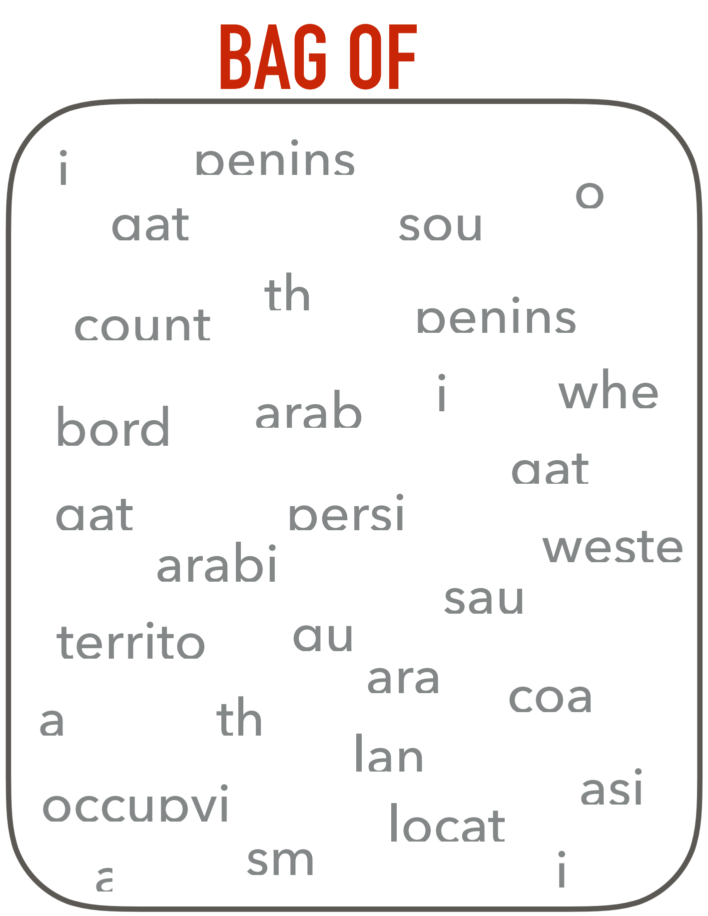
- Getting a semantic representation such as the concepts of “Geography”, “Sport”, “Technology” is hard (but not impossible).
- There is an easier way to obtain these vectors a representation that we have seen

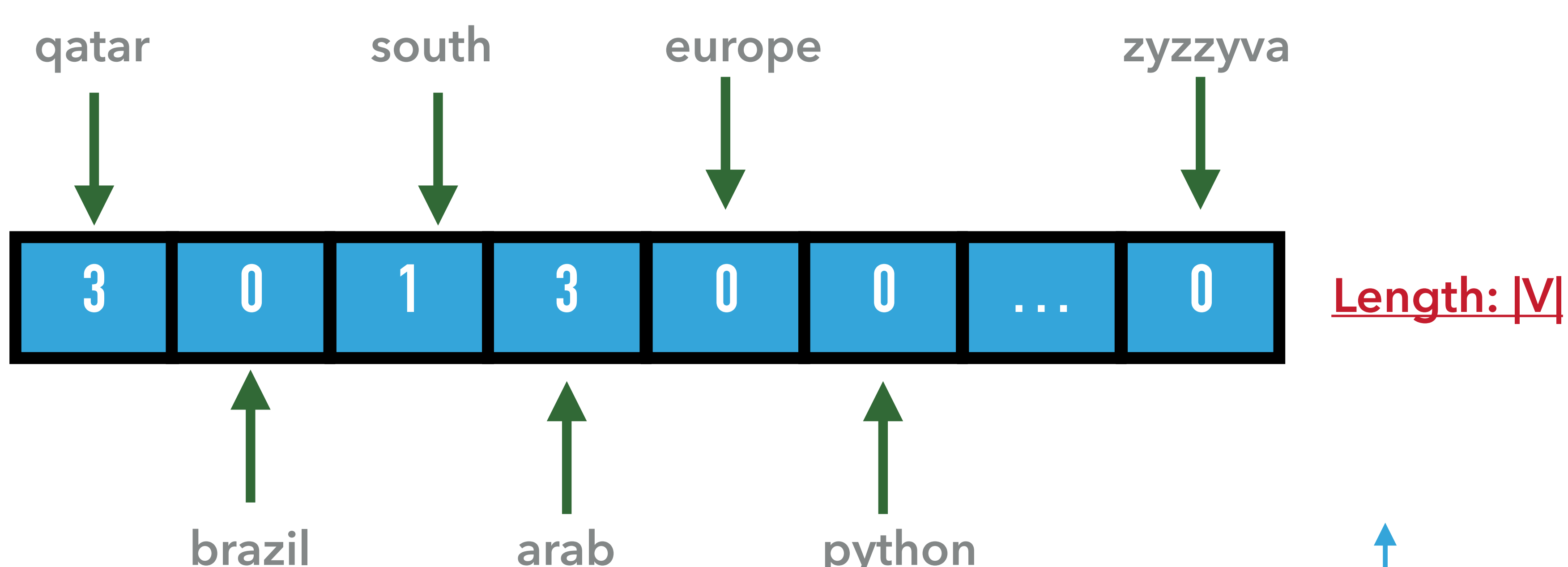


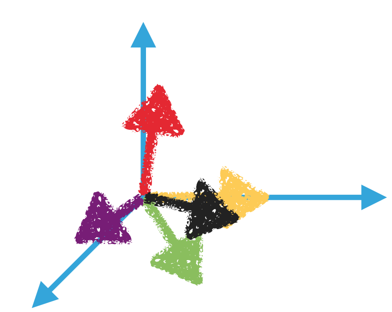
REPRESENTATION

- Getting a semantic representation such as the concepts of "Geography", "Sport", "Technology" is hard (but not impossible).
- There is an easier way to obtain these vectors a representation that we have seen

Bag of words with vocabulary size V can be represented as vectors in an V-dimensional space

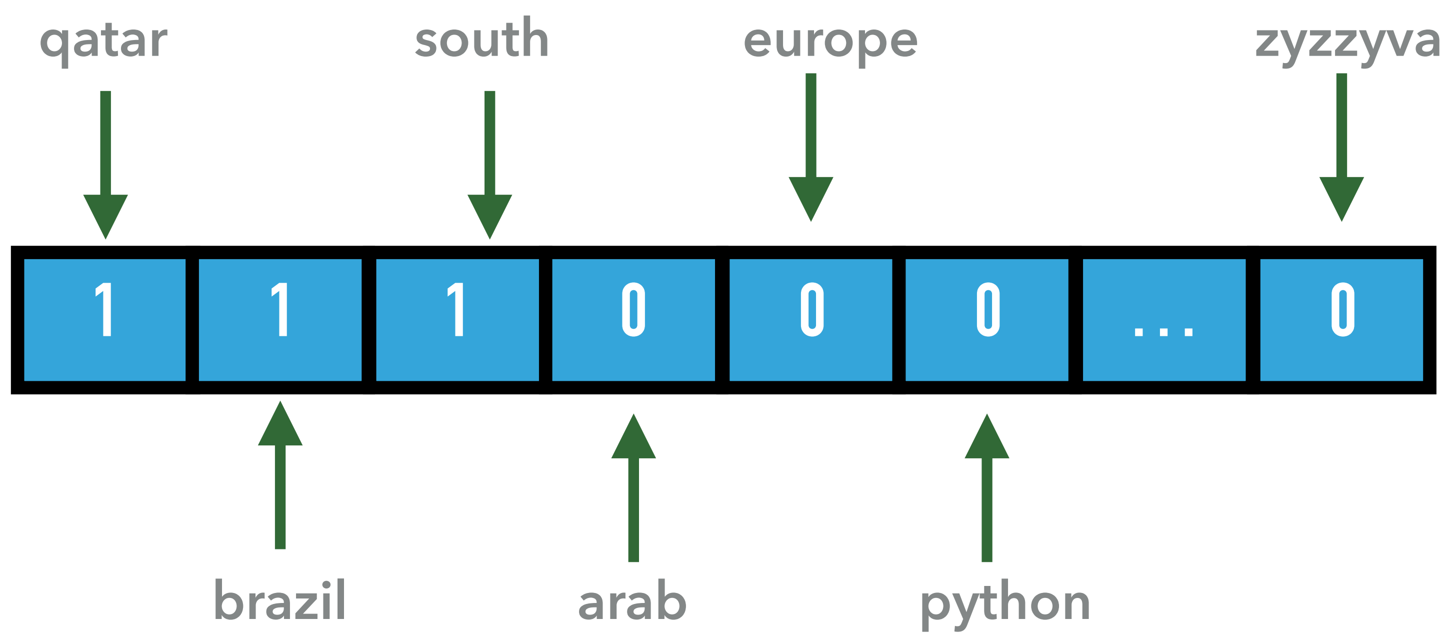




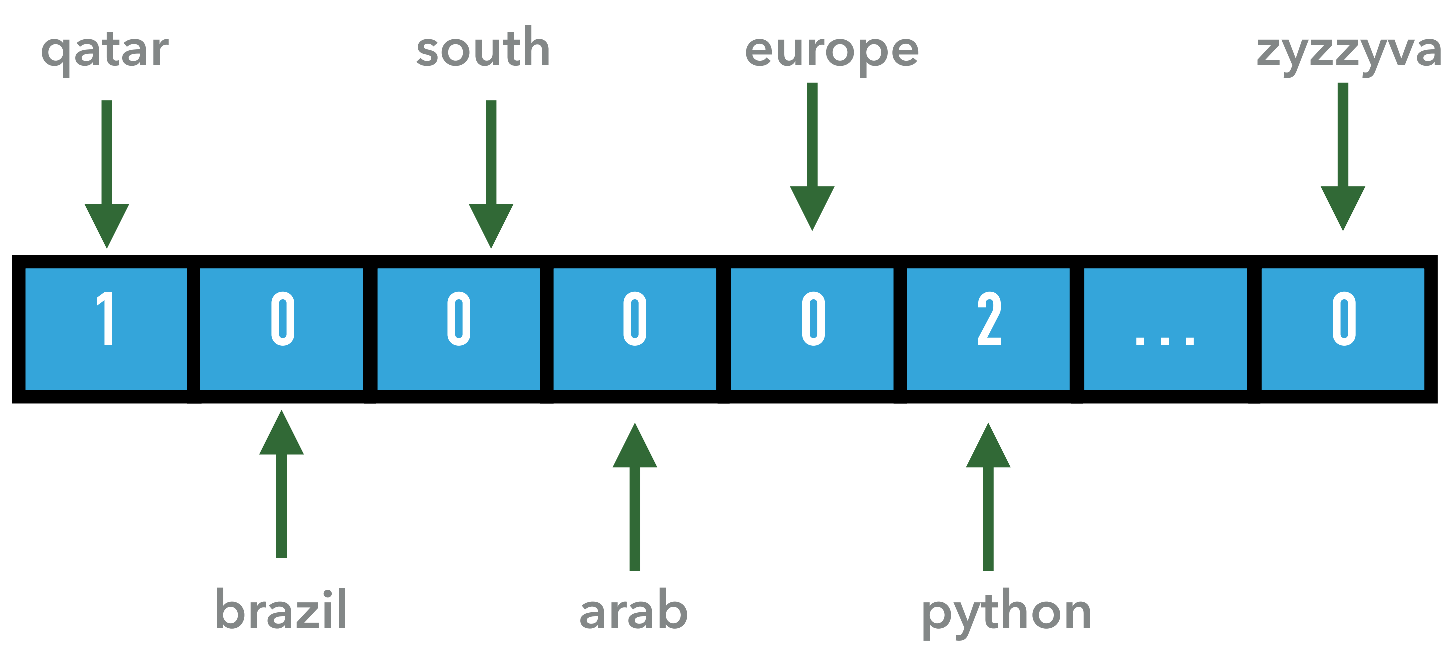


REPRESENTATION

qatar is in the north hemisphere and brazil is in the south

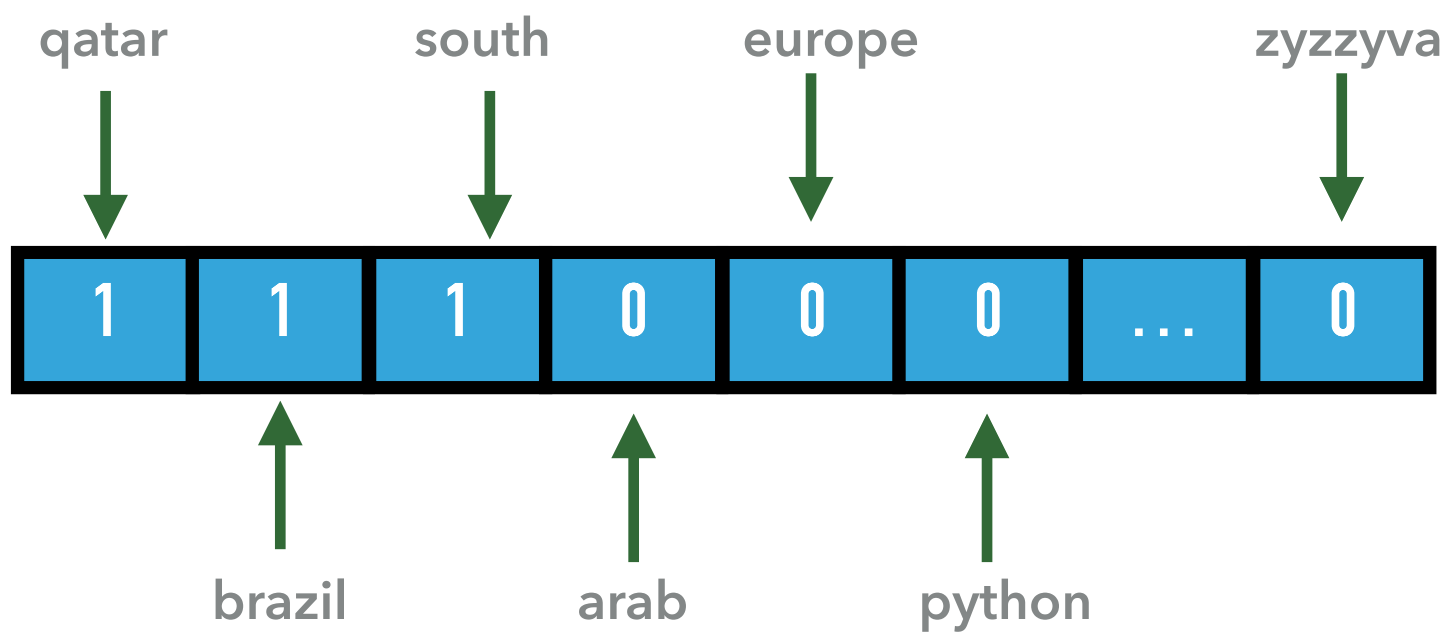


i love the python language but i am afraid i will find a real python in the desert in qatar

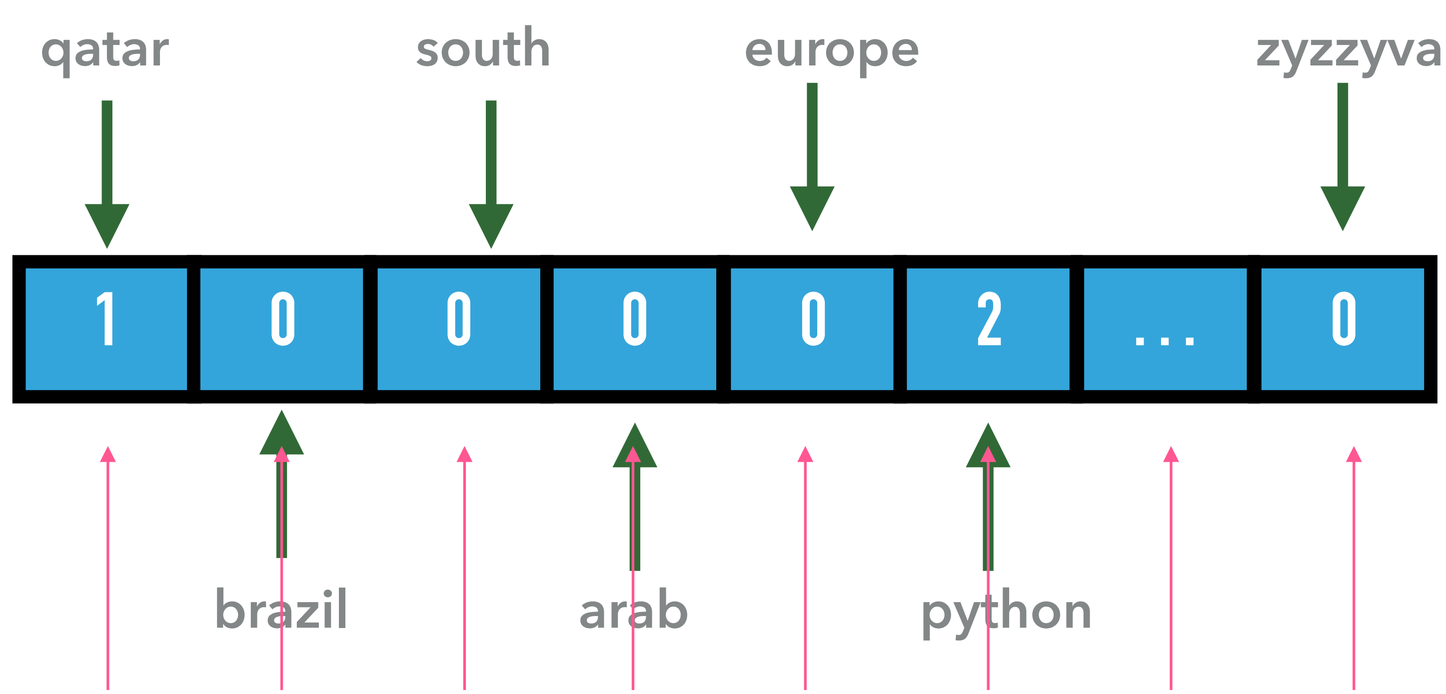


REPRESENTATION

qatar is in the north hemisphere and brazil is in the south



i love the python language but i am afraid i will find a real python in the desert in qatar



Every value is a weight for a word/token in the vector space model

HOW TO CALCULATE THESE WEIGHTS?

- **Empirical research** found two components to be highly relevant:
  - TF: Term frequency (in a document)
  - IDF: Inverse Document frequency (in the collection)

TF

TERM FREQUENCY

- A document is more likely to be more important if a query term occurs many times in it.
- Query: “beirut”

D1

The american band Beirut plays indie-rock music. Beirut started in 2006 playing in Santa Fe, New Mexico. The first concert of Beirut happened in New York…

D2

Beirut is the capital of Lebanon. Beirut has an estimated population of 300,000 inhabitants. Beirut…

D3

A Brazilian delegation is going to the Middle East to find business partners. The trip will first go to Cairo, then Beirut and finally Doha. Politicians expected that…

D4

Beirut is a drinking game that often appears in Hollywood movies. It is also known as beer pong. Beirut is played…

TERM FREQUENCY

- A document is more likely to be more important if a query term occurs many times in it.
- Query: “beirut”

D1

The american band Beirut plays indie-rock music. Beirut started in 2006 playing in Santa Fe, New Mexico. The first concert of Beirut happened in New York…

D2

Beirut is the capital of Lebanon. Beirut has an estimated population of 300,000 inhabitants. Beirut…

D3

A Brazilian delegation is going to the Middle East to find business partners. The trip will first go to Cairo, then Beirut and finally Doha. Politicians expected that…

D4

Beirut is a drinking game that often appears in Hollywood movies. It is also known as beer pong. Beirut is played…

Not very important document, although it happens to have the term beirut in it

TERM FREQUENCY

- A document is more likely to be more important if a query term occurs many times in it.
- Is it a linear relation?
- Doc1 has term *T* 15 times. Doc2 has term *T* only 5 times.
- **Is Doc1 three times more important than Doc2?**

TERM FREQUENCY

- A document is more likely to be more important if a query term occurs many times in it.
- Is it a linear relation?
- Doc1 has term *T* 15 times. Doc2 has term *T* only 5 times.
- **Is Doc1 three times more important than Doc2?**

Doha is the capital of Qatar.

Doha is the capital of Qatar. Doha is the capital of Qatar. Doha is the capital of Qatar.

3 times more important?

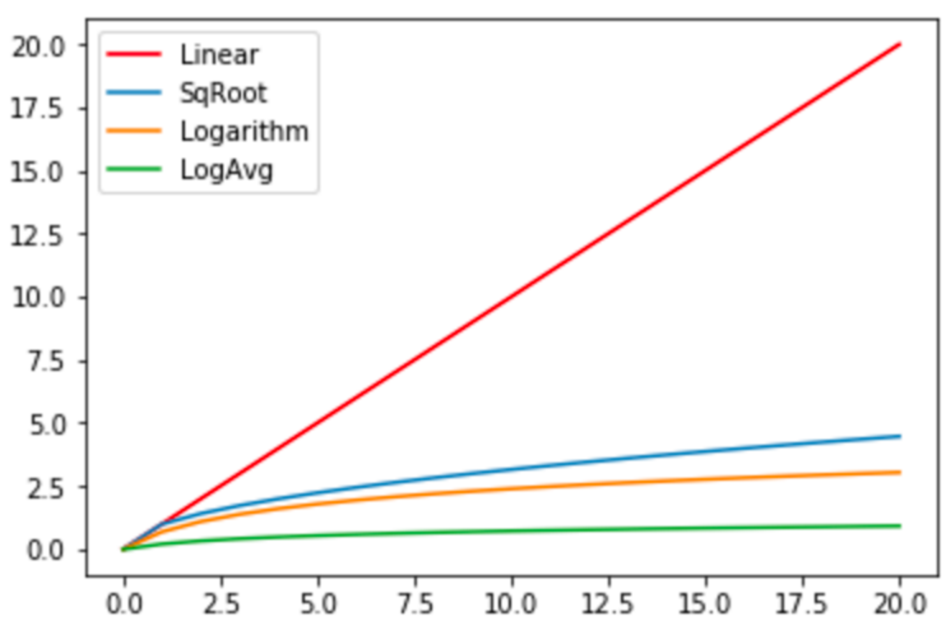
TERM FREQUENCY NORMALIZATION

- *Repeated occurrences* are less informative than *first occurrence*
- Relevance does not increase proportionally with number of term occurrence
- Again, empirical research found that using logarithm of TF is a very effective way to reduce the impact of these frequent terms.
- But it is not the only normalization that exists…

TERM FREQUENCY NORMALIZATION

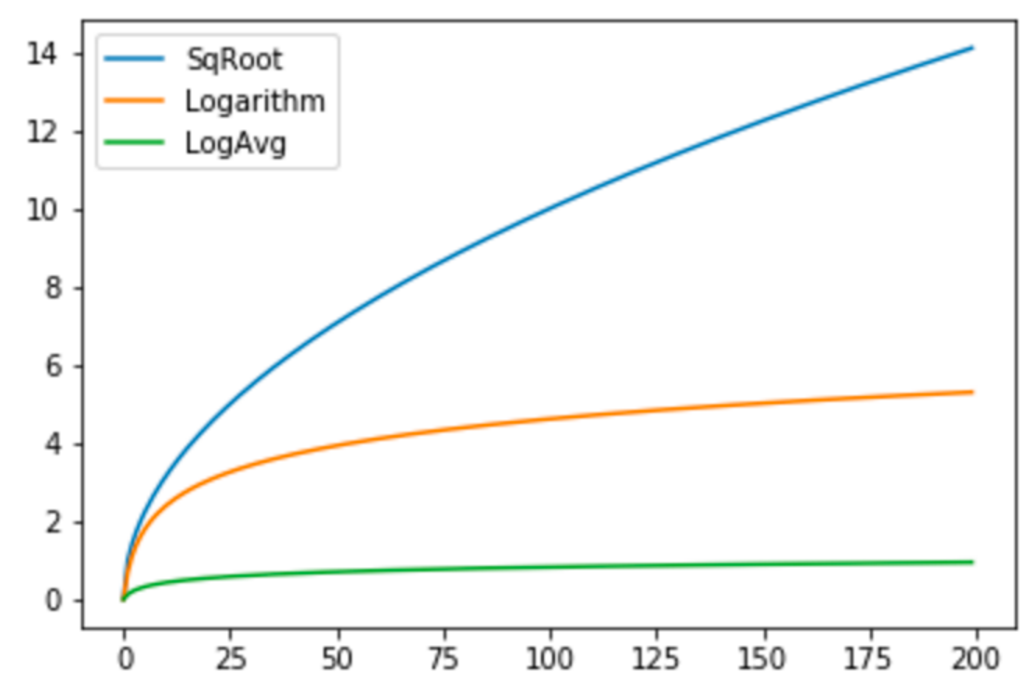
```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, 21) # [0,1,2,3,4....20]
linear = x
logarithm = np.log(x + 1.)
sqrt = np.sqrt(x)
logavg = np.log(1 + x) / ( 1 + np.log(np.average(x)))

plt.plot(x, linear, c="r")
plt.plot(x, sqrt)
plt.plot(x, logarithm)
plt.plot(x, logavg)

plt.legend(["Linear","SqRoot", "Logarithm", "LogAvg",])
```

<matplotlib.legend.Legend at 0x10b686d90>




```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, 200) # [0,1,2,3,4....20]

logarithm = np.log(x + 1.)
sqrt = np.sqrt(x)
logavg = np.log(1 + x) / ( 1 + np.log(np.average(x)))

plt.plot(x, sqrt)
plt.plot(x, logarithm)
plt.plot(x, logavg)

plt.legend(["SqRoot", "Logarithm", "LogAvg"])
```

<matplotlib.legend.Legend at 0x10ad15990>



TERM FREQUENCY NORMALIZATION

- *Not a linear relationship anymore:*

```
pairs = [(15,5), (150,50), (1500, 500), (15000, 5000)]

for (v1, v2) in pairs:
    print "Relationship between v1 (%d) and v2 (%d): %.3f. Logs: %.3f" %\
                        (v1,v2, v1 / v2, math.log(v1) / math.log(v2))
```

```
Relationship between v1 (15) and v2 (5): 3.000. Logs: 1.683
Relationship between v1 (150) and v2 (50): 3.000. Logs: 1.281
Relationship between v1 (1500) and v2 (500): 3.000. Logs: 1.177
Relationship between v1 (15000) and v2 (5000): 3.000. Logs: 1.129
```

IDF

DOCUMENT FREQUENCY

- Theory: a term is more discriminative if it occurs only in fewer documents
- We count the number of documents in which a term occurred.
- Query: "the bug zyzzyva"

It is a rare word, a document that contains this word might be important

It is a common word, a document that contains this word might not be the most important one

It will be in virtually every single document

DOCUMENT FREQUENCY

- Suppose the document frequency of “the”, “bug” and “zyzzyva” in our simple wikipedia collection is:
  - the -> 174.925
  - bug -> 414
  - zyzzyva -> 1
- Intuitively we want for query “the bug zyzzyva”
  - the – receives a small score
  - bug – receives a medium score
  - zyzzyva – receives a high score

inverse of document frequency

INVERSE DOCUMENT FREQUENCY

- Most used version of IDF:

$$idf_t = \log \frac{N}{df_t}$$

Total number of documents in collection

Total number of documents containing term t

Non-linear scaling

INVERSE DOCUMENT FREQUENCY

- Most used version of IDF:

$$idf_t = \log \frac{N}{df_t}$$

Total number of documents in collection

Total number of documents containing term t

Non-linear scaling

- Given N = 174.925, calculate IDF for:
  - the -> 174.925
  - bug -> 414
  - zyzzyva -> 1

INVERSE DOCUMENT FREQUENCY

- Most used version of IDF:

$$idf_t = \log \frac{N}{df_t}$$

Total number of documents in collection

Total number of documents containing term t

Non-linear scaling

- Given N = 174.925, calculate IDF for:
  - the -> 174.925
  - bug -> 414
  - zyzzyva -> 1

```
print "IDF for 'the' %.3f" % math.log(174925 / 174925)
print "IDF for 'bug' %.3f" %math.log(174925 / 414)
print "IDF for 'zyzzyva' %.3f" %math.log(174925 / 1)
```

```
IDF for 'the' 0.000
IDF for 'bug' 6.045
IDF for 'zyzzyva' 12.072
```

IMPORTANT NOTE

- There are plenty of formulations for TF and IDF:

Variants of TF weight

weighting scheme	TF weight
binary	$0, 1$
raw frequency	$f_{t,d}$
log normalization	$1 + \log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1-K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Variants of IDF weight

weighting scheme	IDF weight ($n_t = \lvert\{d \in D : t \in d\}\rvert$)
unary	1
inverse document frequency	$\log \frac{N}{n_t}$
inverse document frequency smooth	$\log\left(\frac{N}{1+n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N-n_t}{n_t}$

- What to keep in mind:
  - TF: term frequency **in a document** is important
  - IDF: rare terms **in a collection** are more important than common ones

KEEP IN MIND

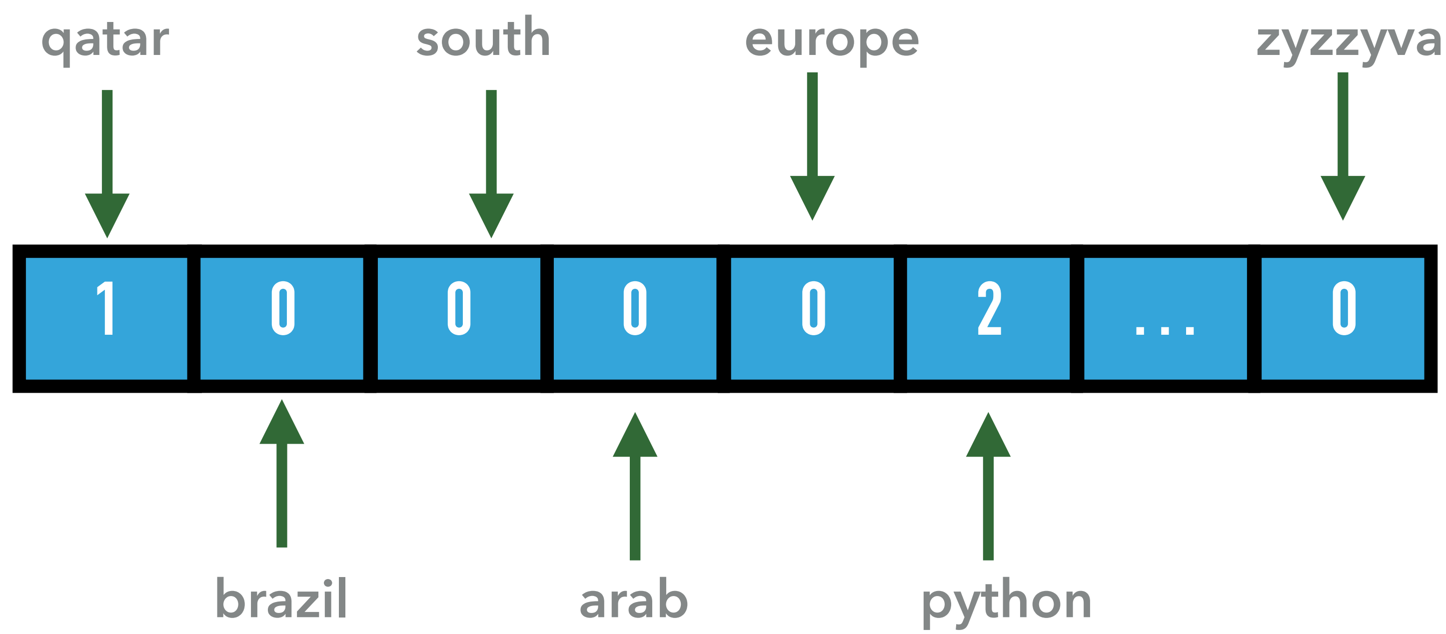
- TF-IDF
- TF/IDF
- TF*IDF
- TFIDF
- TF_IDF

$$w(t,d) = TF(t,d) \times IDF(t)$$

TFIDF AND VSM

TF-IDF WEIGHTS IN VSM

i love the python language but i am afraid i will find a real python in the desert in qatar



N = 100.000

QATAR:

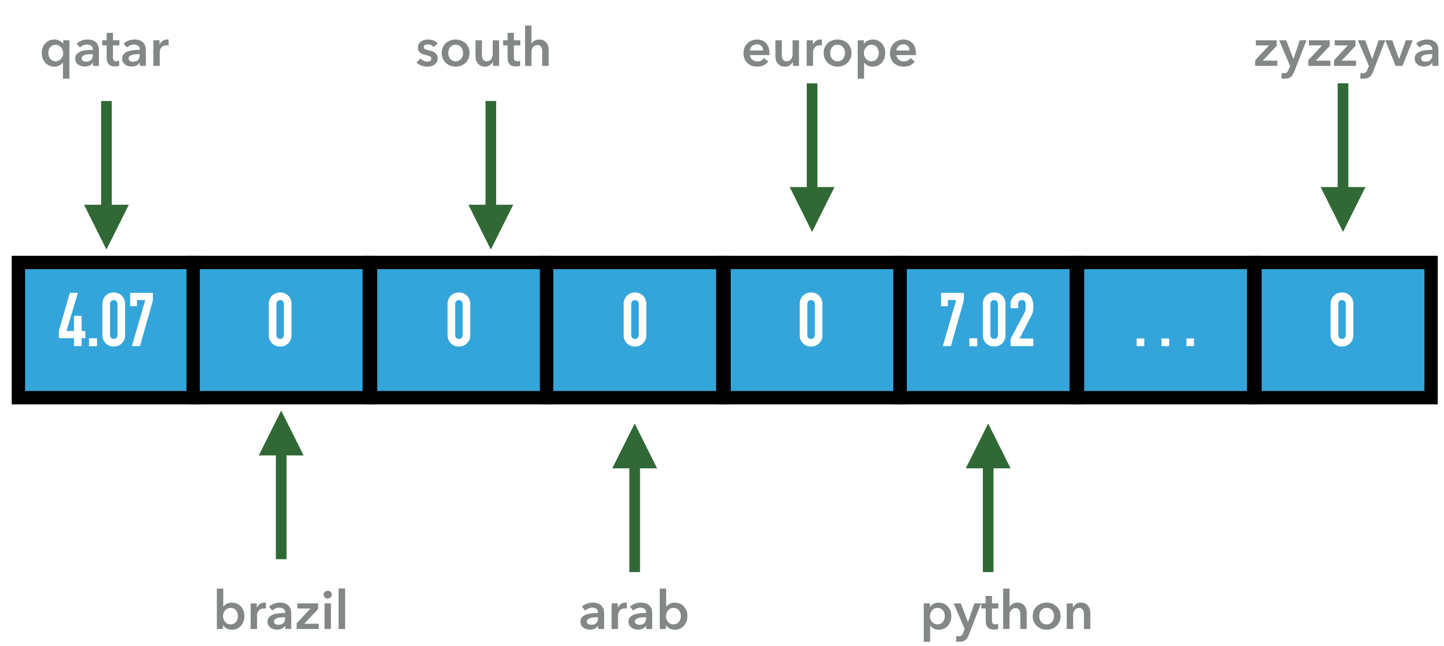
df_qatar = 280

idf_qatar = log(100K/280) = 5.87

PYTHON

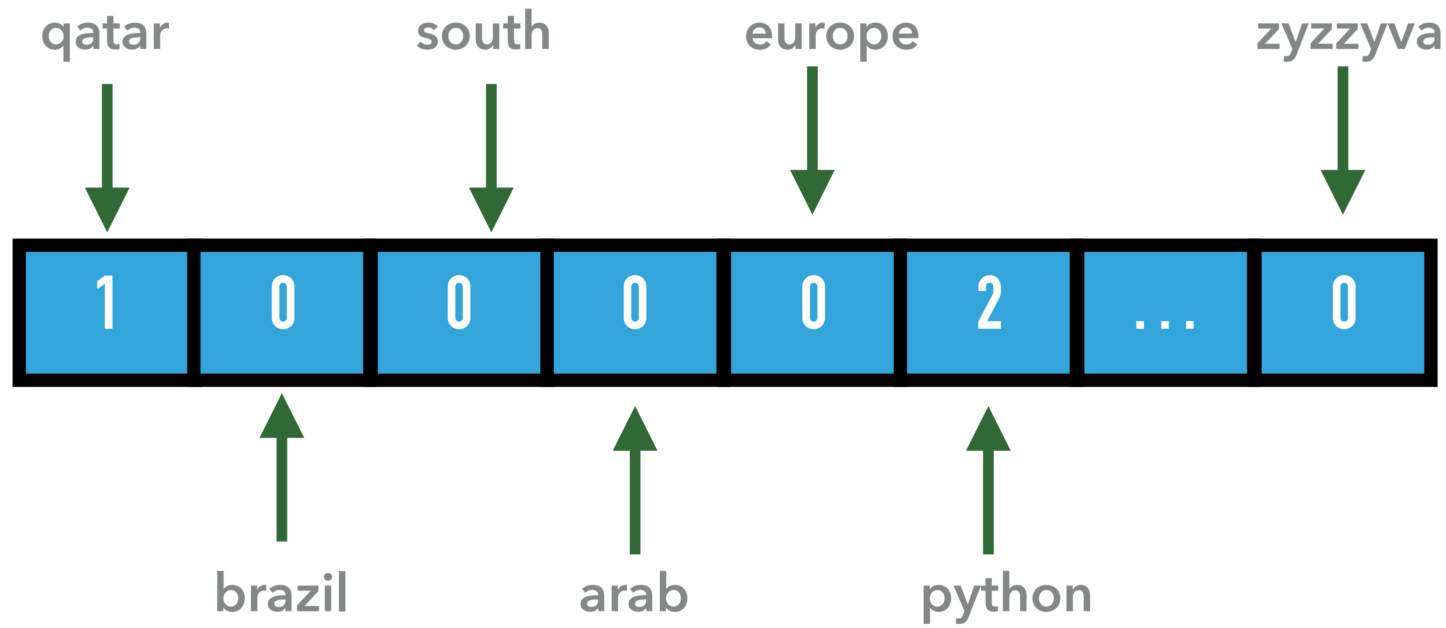
df_python = 160

idf_python = log(100K/160) = 6.44



TF-IDF WEIGHTS IN VSM

i love the python language but i am afraid i will find a real python in the desert in qatar



N = 100.000

QATAR:

df_qatar = 280

idf_qatar = log(100K/280) = 5.87

tf_qatar = log(1+1) = 0.69

tf_idf_qatar = 5.8 * 0.69 = 4.07

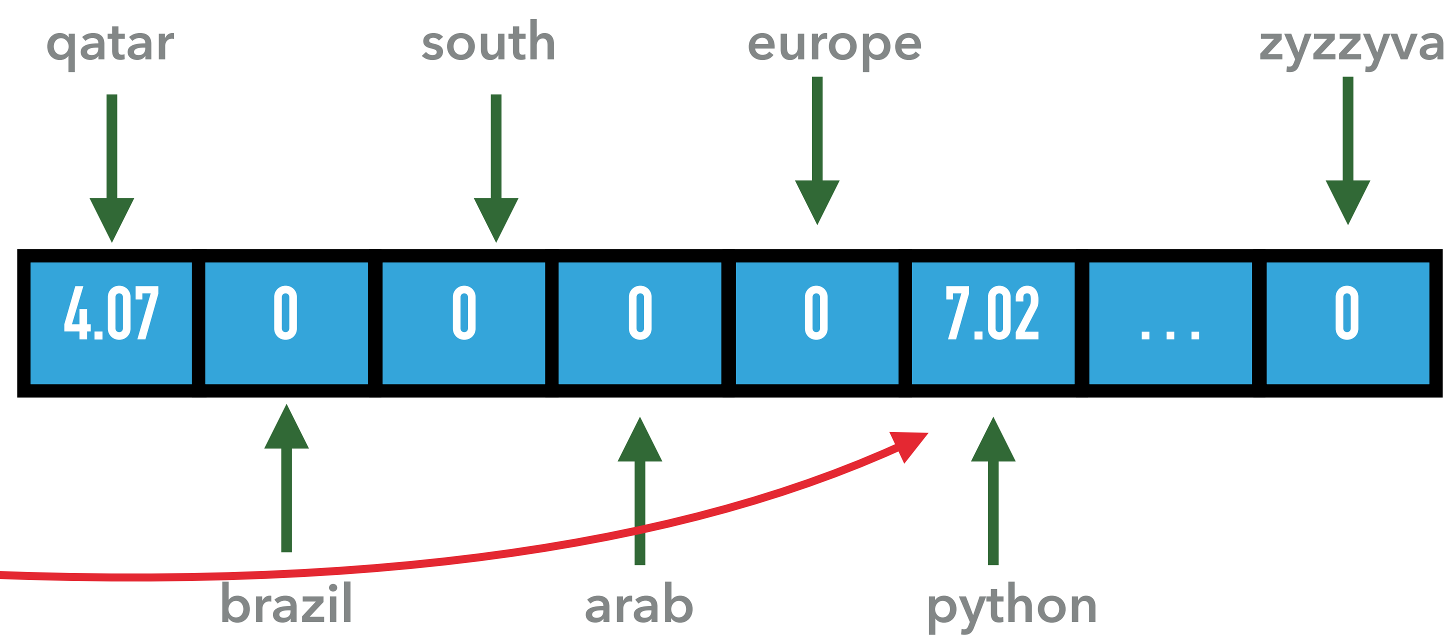
PYTHON

df_python = 160

idf_python = log(100K/160) = 6.44

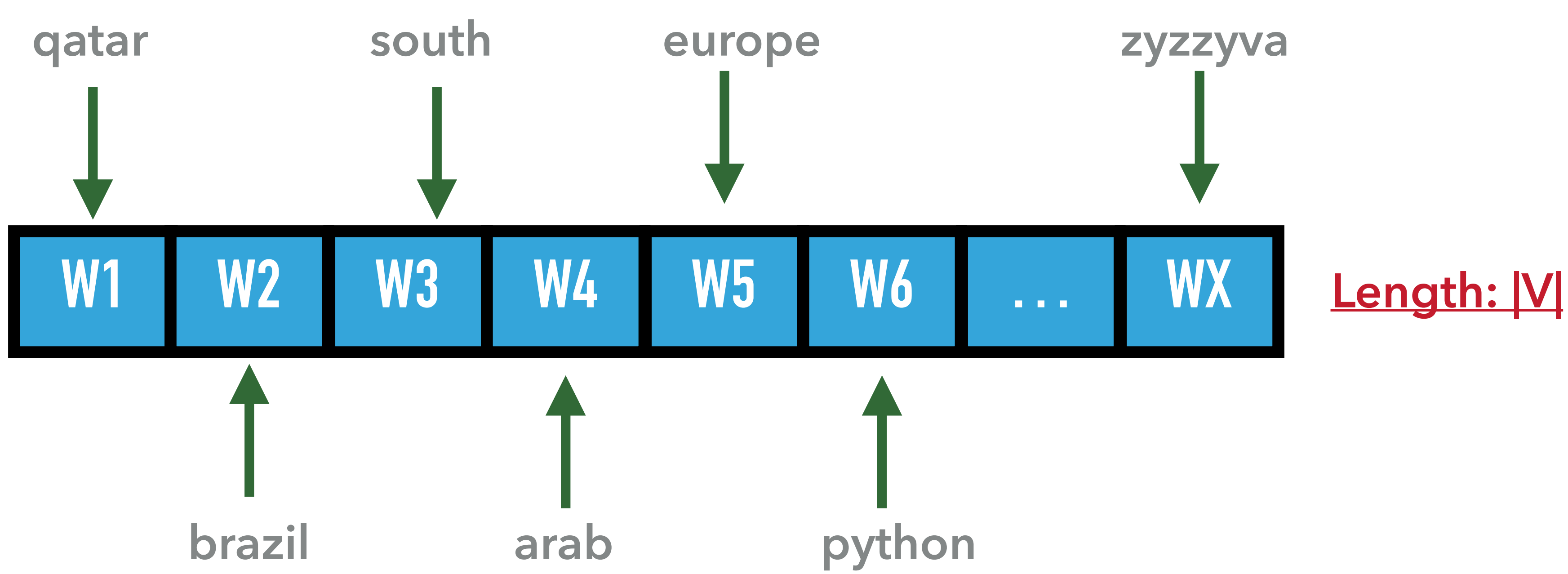
tf_python = log(1 + 2) = 1.09

tf_idf_python = 6.44 * 1.09 = 7.02



WHAT DO WE HAVE?

- We have a way to represent documents and queries



- We still need a similarity/distance measure!

SIMILARITY MEASURES

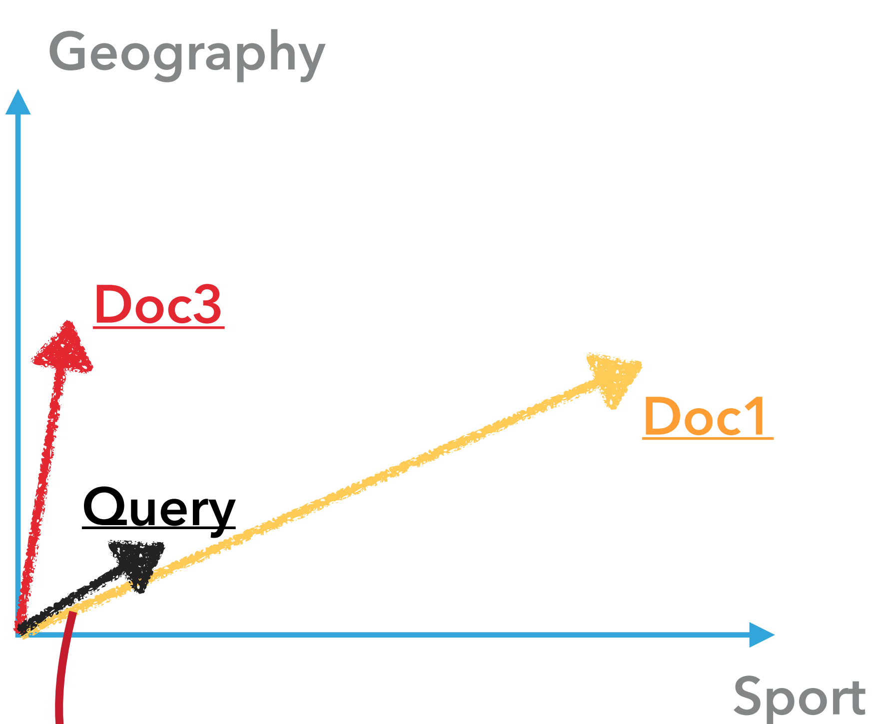
- There are many options for similarity measure:
  - Inner product
  - Dice / Jaccard similarity
  - Euclidian distance:

$$dist(q,d) = \sqrt{\sum_{t\in V}[\underbrace{tf(t,q)idf(t)}_{\text{query terms}} - \underbrace{tf(t,d)idf(t)}_{\text{document terms}}]^2}$$

  - Problem: longer documents will be strongly penalized by extra terms

FROM DISTANCE TO ANGLE

- It is more important to calculate how two vectors overlap

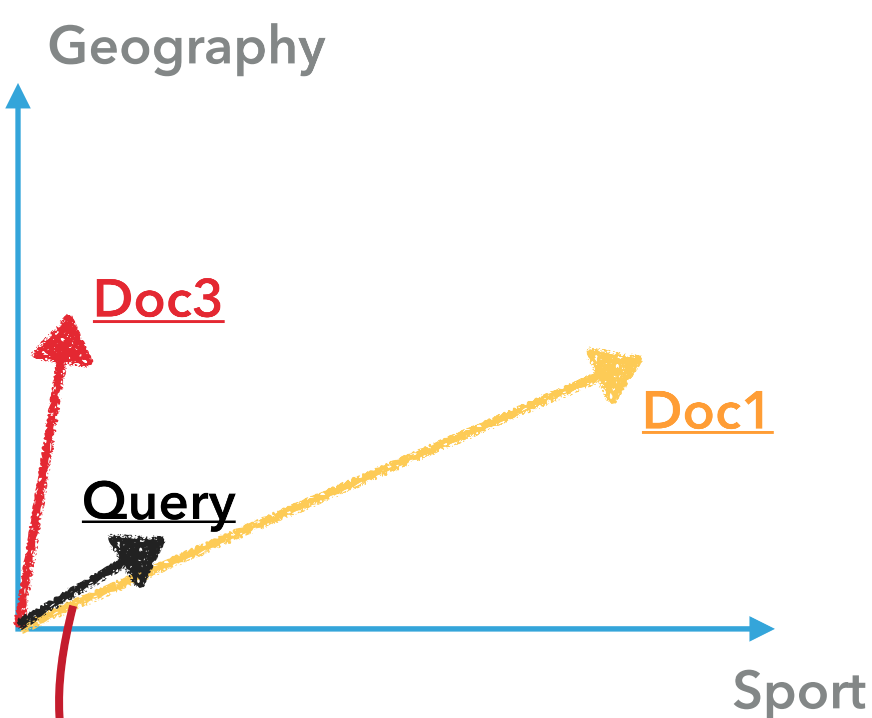


Euclidian(Doc3, Query) < Euclidian (Doc1, Query)

That is a problem! Doc 3 is not more relevant than Doc 1

FROM DISTANCE TO ANGLE

- It is more important to calculate how two vectors overlap



Solution: calculate **cosine similarity** instead!
Measure how close is the angle between two vectors,.

Euclidian(Doc3, Query) < Euclidian (Doc1, Query)
That is a problem! Doc 3 is not more relevant than Doc 1

COSINE SIMILARITY

- sim(d,q) = cosine(V_d, V_q) = cosine(A, B)

$$\cos(A,B) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

COSINE SIMILARITY - EXAMPLE

$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Vector: [coffee, tea, milk, sugar, cup]
- Doc 1: “Would you like a cup of coffee?
  - [1, 0, 0, 0, 1]
- Doc 2: “I take milk and sugar with coffee or tea”
  - [1, 1, 1, 1, 0]
- Doc 3: “The recipe uses a cup of milk and a cup of sugar”
  - [0, 0, 1, 1, 2]
- Query: “The coffee barista doesn’t drink coffee, but will drink milk”
  - [2, 0, 1, 0, 0]

COSINE SIMILARITY - EXAMPLE

$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Q: [2, 0, 1, 0, 0]
- D1: [1, 0, 0, 0, 1]
- D2: [1, 1, 1, 1, 0]
- D3: [0, 0, 1, 1, 2]

- ||Q|| = sqrt(2*2 + 0*0 + 1*1 + 0*0 + 0*0) = sqrt(5) = 2.24
- ||D1|| = sqrt(1*1 + 0*0 + 0*0 + 0*0 + 1*1) = sqrt(2) = 1.41
- ||D2|| = sqrt(1*1 + 1*1 + 1*1 + 1*1 + 0*0) = sqrt(4) = 2.00
- ||D3|| = sqrt(0*0 + 0*0 + 1*1 + 1*1 + 2*2) = sqrt(6) = 2.45

COSINE SIMILARITY - EXAMPLE

$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Q: [2, 0, 1, 0, 0]
- D1: [1, 0, 0, 0, 1]
- D2: [1, 1, 1, 1, 0]
- D3: [0, 0, 1, 1, 2]

- ||Q|| = sqrt(2*2 + 0*0 + 1*1 + 0*0 + 0*0) = sqrt(5) = 2.24
- ||D1|| = sqrt(1*1 + 0*0 + 0*0 + 0*0 + 1*1) = sqrt(2) = 1.41
- ||D2|| = sqrt(1*1 + 1*1 + 1*1 + 1*1 + 0*0) = sqrt(4) = 2.00
- ||D3|| = sqrt(0*0 + 0*0 + 1*1 + 1*1 + 2*2) = sqrt(6) = 2.45

- Q . D1 = 2*1 + 0*0 + 1*0 + 0*0 + 0*1 = 2
- Q . D2 = 2*1 + 0*1 + 1*1 + 0*1 + 0*0 = 3
- Q . D3 = 2*0 + 0*0 + 1*1 + 0*1 + 0*2 = 1

COSINE SIMILARITY - EXAMPLE

$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Q: [2, 0, 1, 0, 0]
- D1: [1, 0, 0, 0, 1]
- D2: [1, 1, 1, 1, 0]
- D3: [0, 0, 1, 1, 2]

- ||Q|| = sqrt(2*2 + 0*0 + 1*1 + 0*0 + 0*0) = sqrt(5) = 2.24
- ||D1|| = sqrt(1*1 + 0*0 + 0*0 + 0*0 + 1*1) = sqrt(2) = 1.41
- ||D2|| = sqrt(1*1 + 1*1 + 1*1 + 1*1 + 0*0) = sqrt(4) = 2.00
- ||D3|| = sqrt(0*0 + 0*0 + 1*1 + 1*1 + 2*2) = sqrt(6) = 2.45

- Q . D1 = 2*1 + 0*0 + 1*0 + 0*0 + 0*1 = 2
- Q . D2 = 2*1 + 0*1 + 1*1 + 0*1 + 0*0 = 3
- Q . D3 = 2*0 + 0*0 + 1*1 + 0*1 + 0*2 = 1

- Cos(Q, D1) = 2 / (2.24 * 1.41) = 0.63
- Cos(Q, D2) = 3 / (2.24 * 2.00) = 0.67
- Cos(Q, D3) = 1 / (2.24 * 2.45) = 0.18

COSINE SIMILARITY - EXAMPLE

$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Q: [2, 0, 1, 0, 0]
- D1: [1, 0, 0, 0, 1]
- D2: [1, 1, 1, 1, 0]
- D3: [0, 0, 1, 1, 2]

- ||Q|| = sqrt(2*2 + 0*0 + 1*1 + 0*0 + 0*0) = sqrt(5) = 2.24
- ||D1|| = sqrt(1*1 + 0*0 + 0*0 + 0*0 + 1*1) = sqrt(2) = 1.41
- ||D2|| = sqrt(1*1 + 1*1 + 1*1 + 1*1 + 0*0) = sqrt(4) = 2.00
- ||D3|| = sqrt(0*0 + 0*0 + 1*1 + 1*1 + 2*2) = sqrt(6) = 2.45

- Q . D1 = 2*1 + 0*0 + 1*0 + 0*0 + 0*1 = 2
- Q . D2 = 2*1 + 0*1 + 1*1 + 0*1 + 0*0 = 3
- Q . D3 = 2*0 + 0*0 + 1*1 + 0*1 + 0*2 = 1

- Cos(Q, D1) = 2 / (2.24 * 1.41) = 0.63 2nd
- Cos(Q, D2) = 3 / (2.24 * 2.00) = 0.67 1st
- Cos(Q, D3) = 1 / (2.24 * 2.45) = 0.18 3rd

CONCLUSION – VSM

- Empirically effective
- Intuitive
- Easy to implement

- Assume term independence
- Assume query and document to be the same
- Many arbitrary term weighting and similarity measures

TODAY'S LECTURE IN THE STANFORD IR BOOK

- Chapter 6.2 - Term frequency and weighting
- Chapter 6.3 - Vector space model
- Chapter 6.4 - TF-IDF variant functions