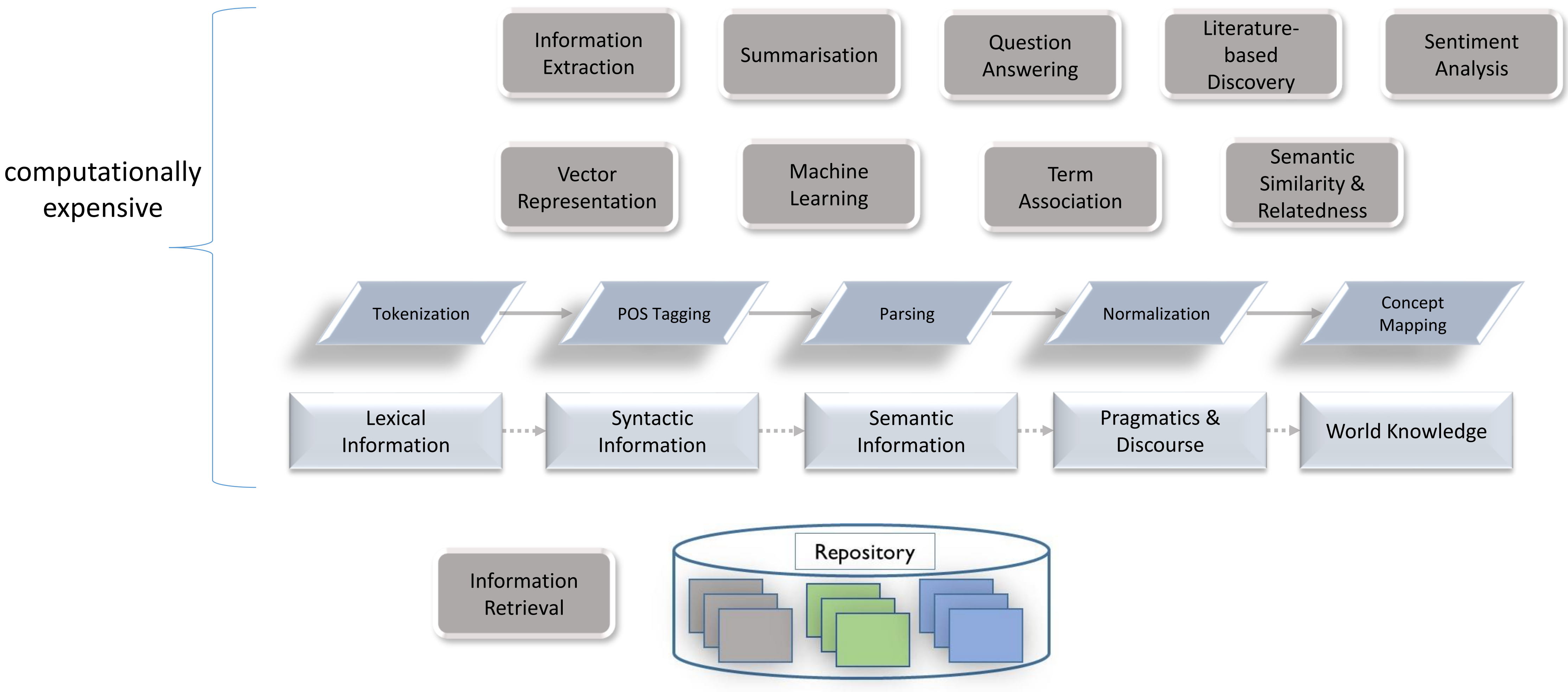
Information Retrieval (IR)

please hit the record button, Bridget

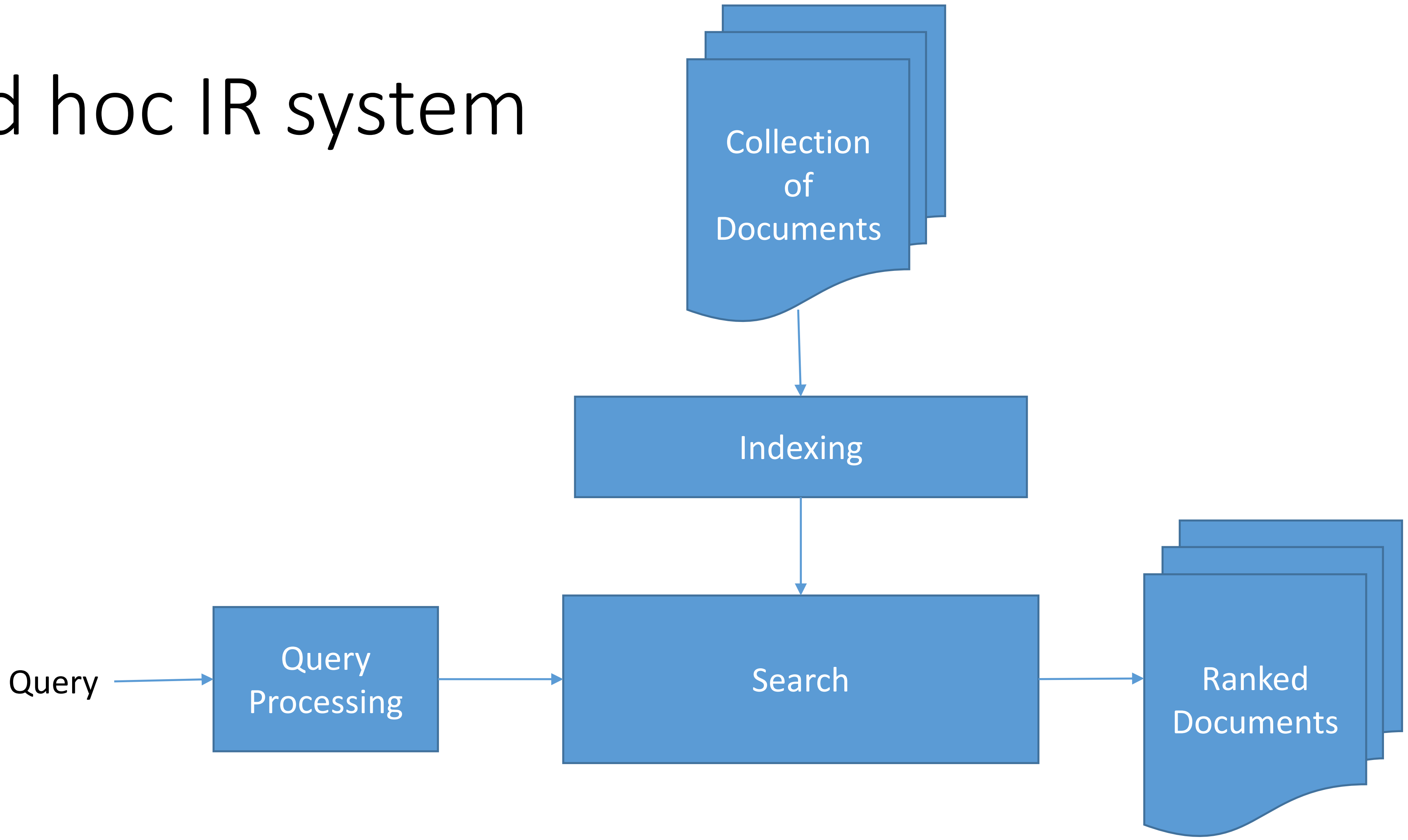
Natural language processing stack



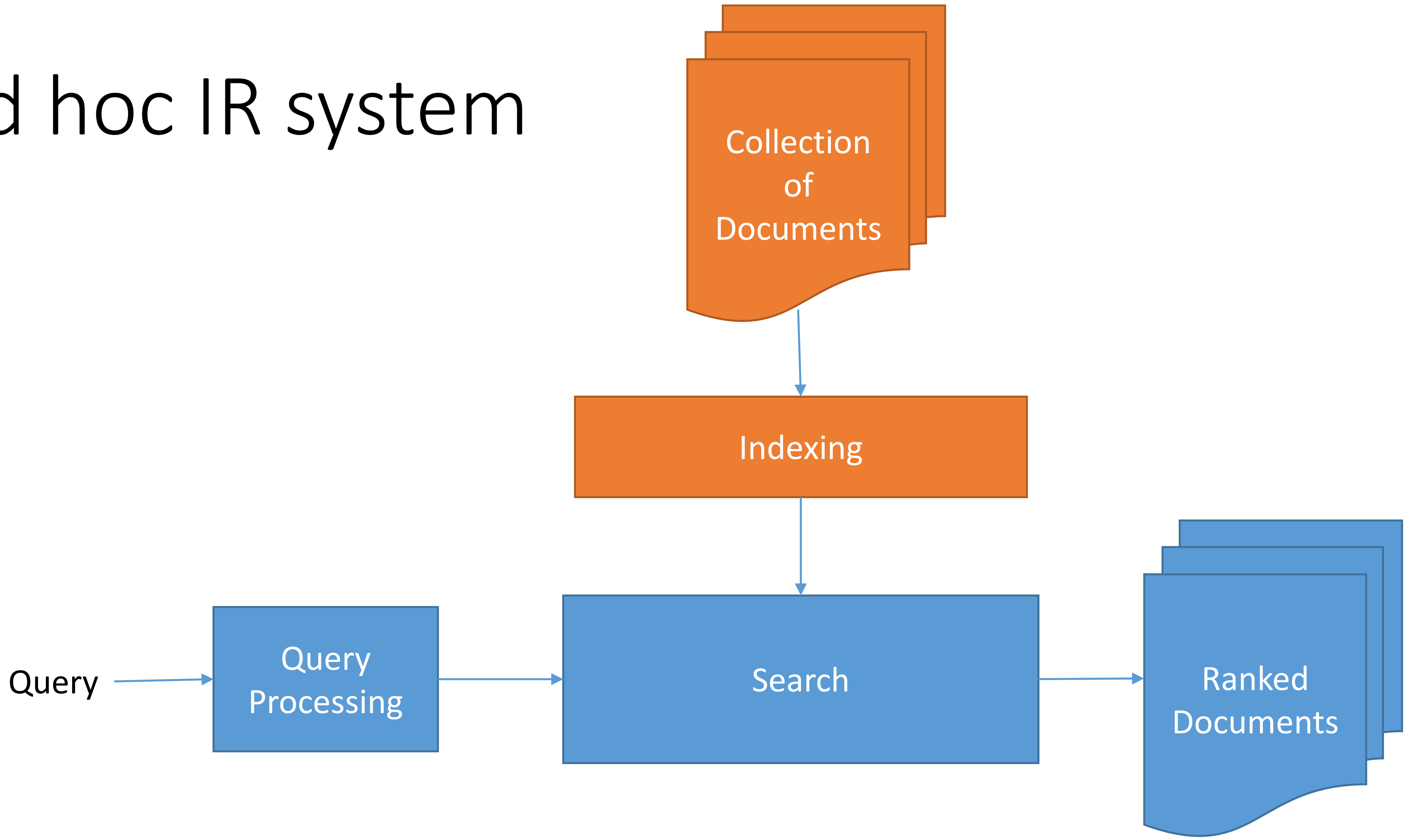
Information Retrieval

- Focus:
    - Storage of text documents
    - Retrieval of documents based on users’ query

Ad hoc IR system



Ad hoc IR system



How do we go about representing a document?

Two approaches

- Inverted Index
- Vector Space Model

Approach: Inverted Index

Inverted index

Inverted index consists of:

- a lexicon of terms
- a posting list for each term that records which document the term occurs in

Doc 1 one fish, two fish

Doc 2 red fish, blue fish

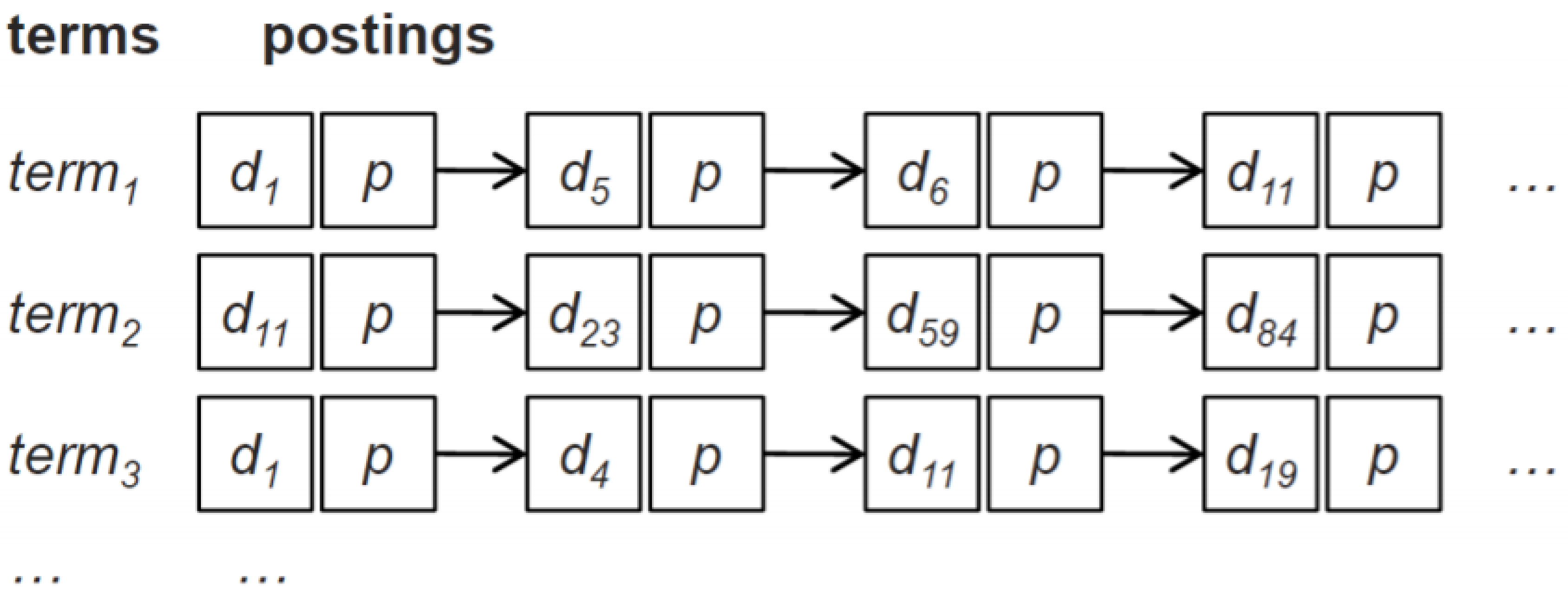
Doc 3 cat in the hat

Doc 4 green eggs and ham

	1	2	3	4
blue		1		
cat			1	
egg				1
fish	1	1		
green				1
ham				1
hat			1	
one	1			
red		1		
two	1			

blue	→ 2
cat	→ 3
egg	→ 4
fish	→ 1 → 2
green	→ 4
ham	→ 4
hat	→ 3
one	→ 1
red	→ 2
two	→ 1

Posting Lists

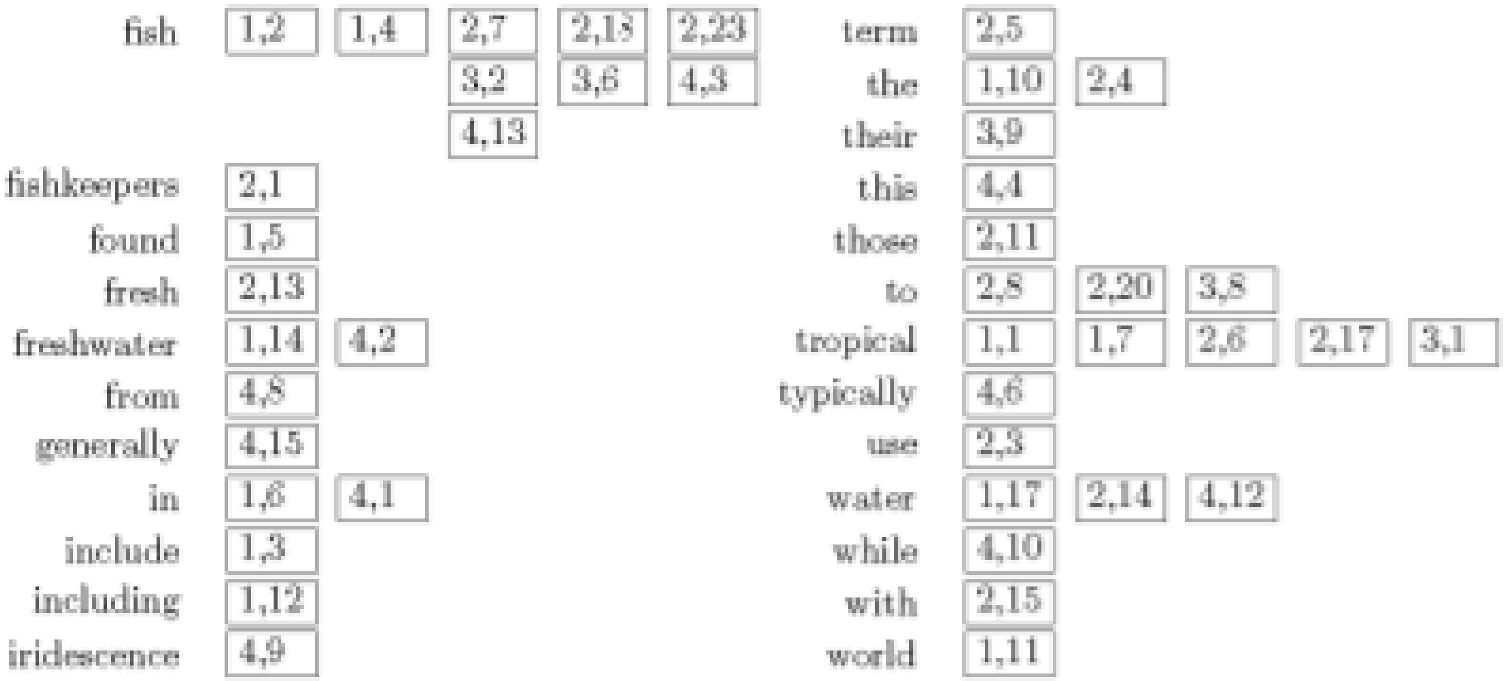


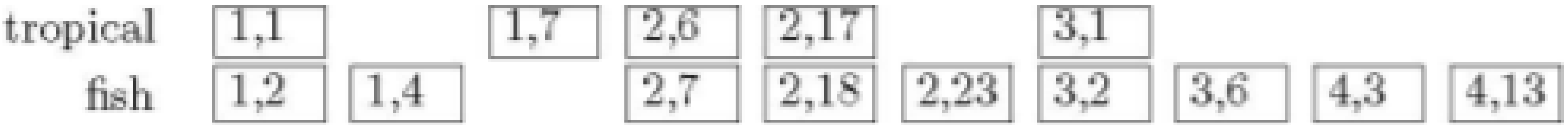
Posting list is comprised of individual postings for each term which consists of:

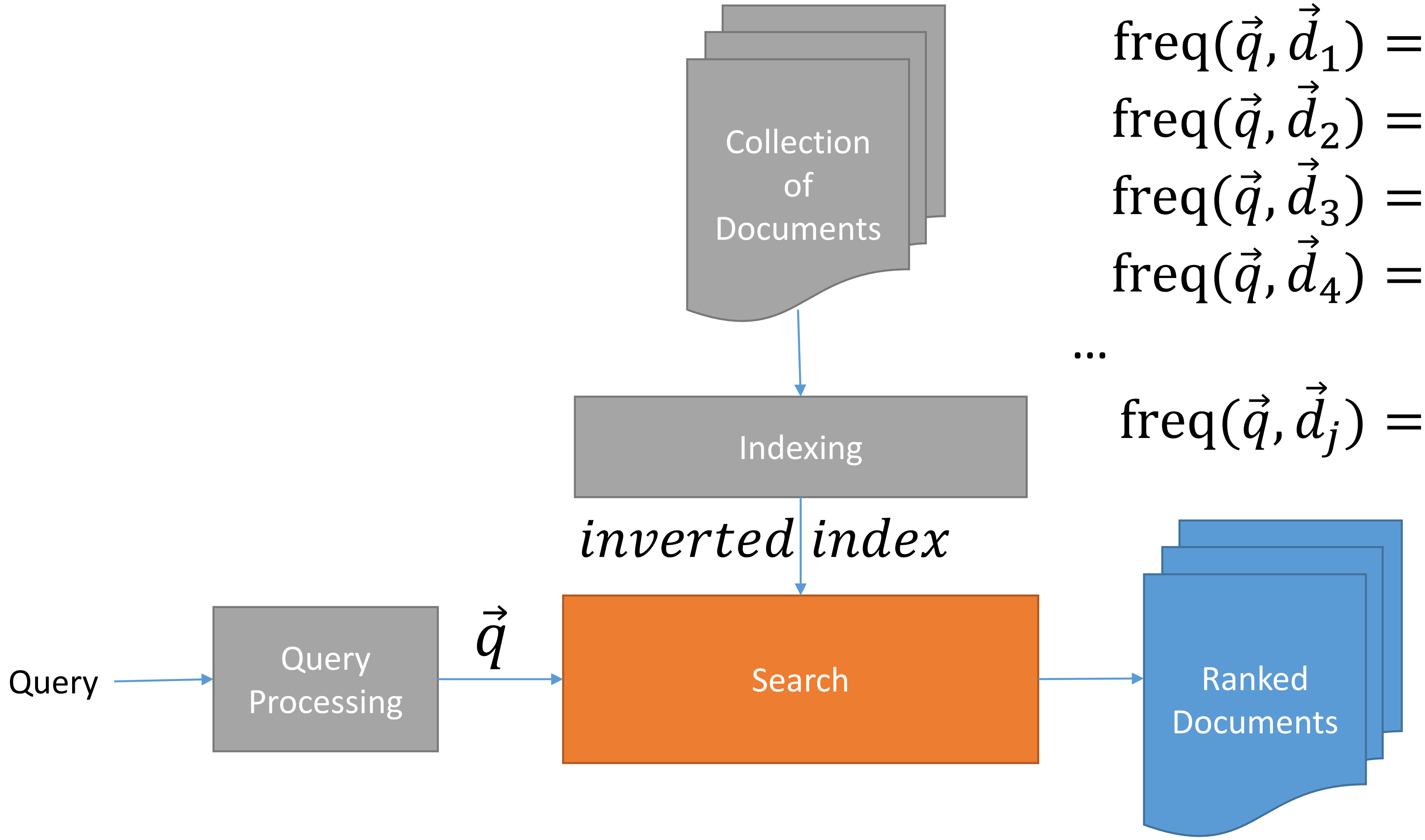
- a document id
- metadata about the term within the document, such as:
  - frequency
  - location

Matches

- Match phrases within a window:
    - find “tropical fish”
    - find “tropical” within 5 words of “fish”
- Word positions in inverted lists make these queries efficient







$$\mathrm{freq}(\vec{q}, \vec{d}_1) = rank_1$$
$$\mathrm{freq}(\vec{q}, \vec{d}_2) = rank_2$$
$$\mathrm{freq}(\vec{q}, \vec{d}_3) = rank_3$$
$$\mathrm{freq}(\vec{q}, \vec{d}_4) = rank_4$$

…

$$\mathrm{freq}(\vec{q}, \vec{d}_j) = rank_j$$

Lucene

https://lucene.apache.org/

Lucene Core is a Java library providing powerful indexing and search features, as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities. The PyLucene sub project provides Python bindings for Lucene Core.

Lucene

https://lucene.apache.org/

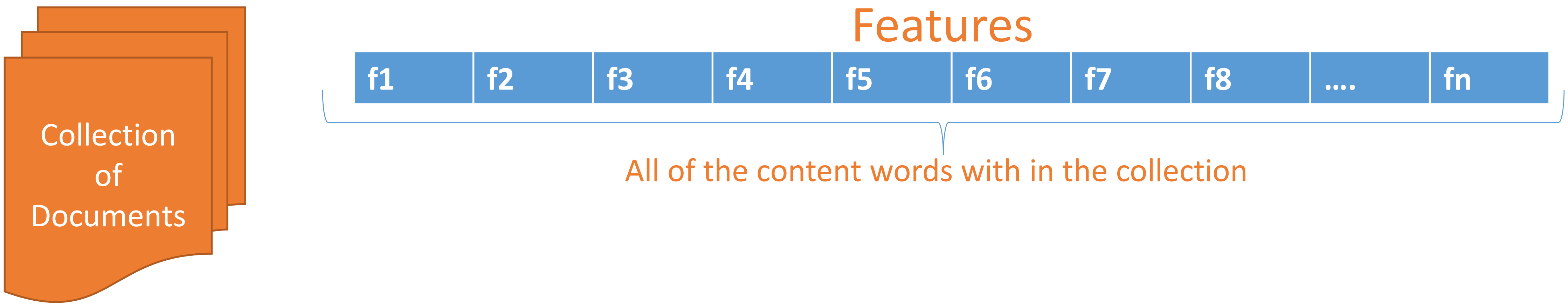
Lucene Core is a Java library providing powerful indexing and search features, as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities. The PyLucene sub project provides Python bindings for Lucene Core.

Solr™ is a high performance search server built using Lucene Core. Solr is highly scalable, providing fully fault tolerant distributed indexing, search and analytics. It exposes Lucene's features through easy to use JSON/HTTP interfaces or native clients for Java and other languages.

Approach: Vector Space Model

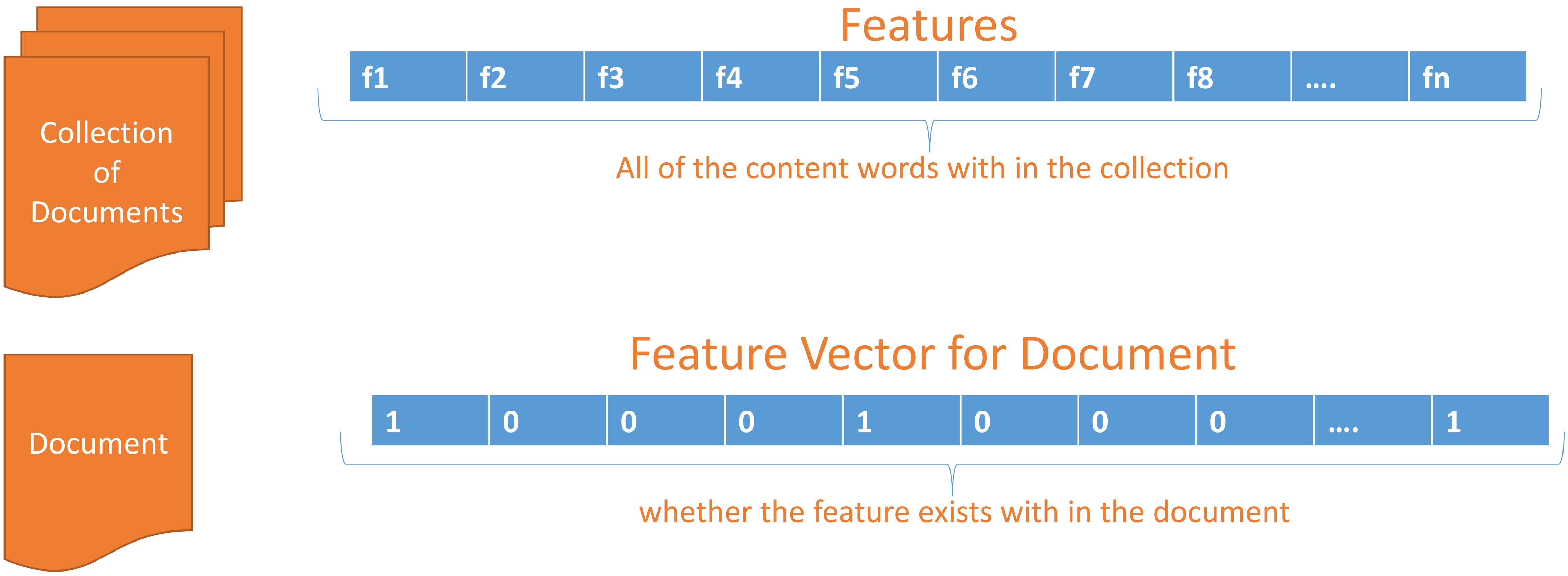
Vector Space Model

- Documents are represented as a vector of features representing terms (words) that occur with in the collection



Vector Space Model

- Documents and queries are represented as a vector of features representing terms (words) that occur with in the collection



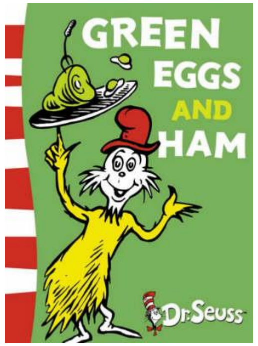
Vector Space Model

- Documents and queries are represented as a vector of features representing terms (words) that occur with in the collection

Features

cat	hat	green	eggs	ham	sam	grinch	stole	….	tree

All of the content words with in the collection

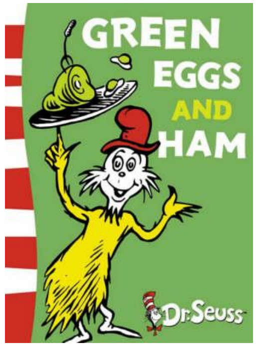


Feature Vector for Document

0	0	1	1	1	1	0	0	….	0

whether the feature exists with in the document

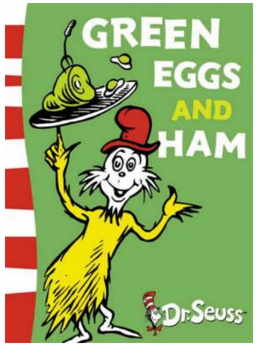
Mathematically



0	0	1	1	1	1	0	0	….	0

$$\vec{d}_j = (0,0,1,1,1,1,0,0, \dots, 0)$$

Term weighting



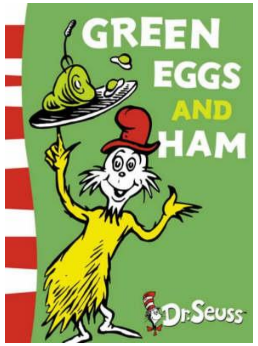
Binary Vector

0	0	1	1	1	1	0	0	….	0

Frequency Vector

0	0	5	10	6	50	0	0	….	0

Term weighting



Binary Vector

0	0	1	1	1	1	0	0	….	0

Frequency Vector

0	0	5	10	6	50	0	0	….	0

IDF Vector

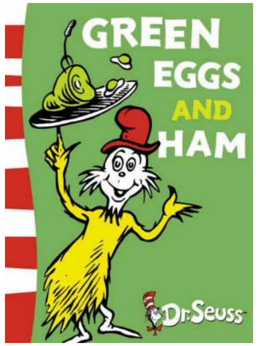
0	0	2.9	2.3	2.8	.69	0	0	….	0

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with

- N: total number of documents in the corpus
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t, d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

Term weighting



Binary Vector

0	0	1	1	1	1	0	0	....	0

Frequency Vector

0	0	5	10	6	50	0	0	....	0

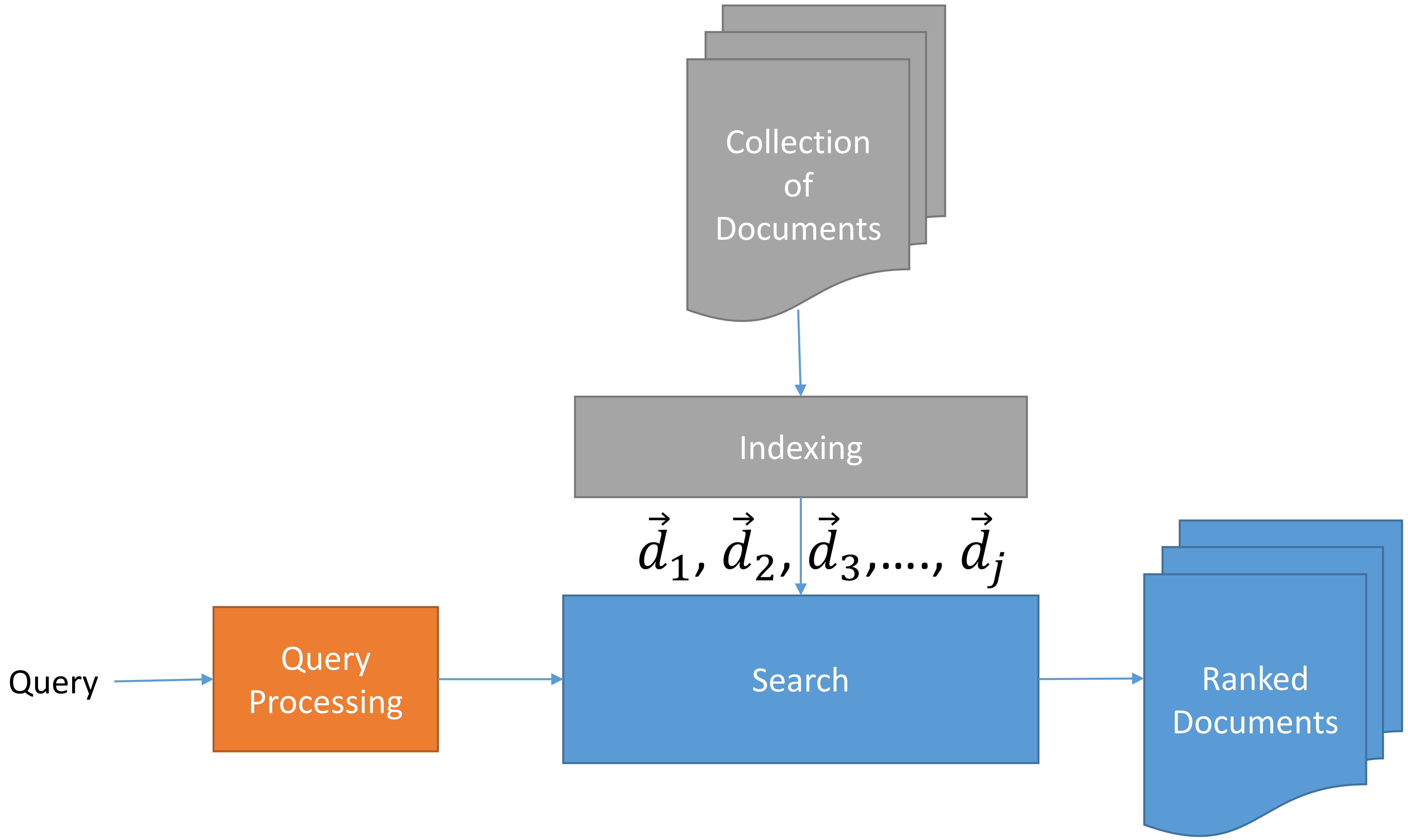
IDF Vector

0	0	2.9	2.3	2.8	.69	0	0	....	0

TF-IDF Vector

0	0	14.5	23	16.8	16.8	0	0	....	0

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$$


Collection of Documents
Indexing
$\vec{d}_1, \vec{d}_2, \vec{d}_3, \ldots., \vec{d}_j$
Query
Query Processing
Search
Ranked Documents

Collection of Documents

Features

f1	f2	f3	f4	f5	f6	f7	f8	….	fn

Document

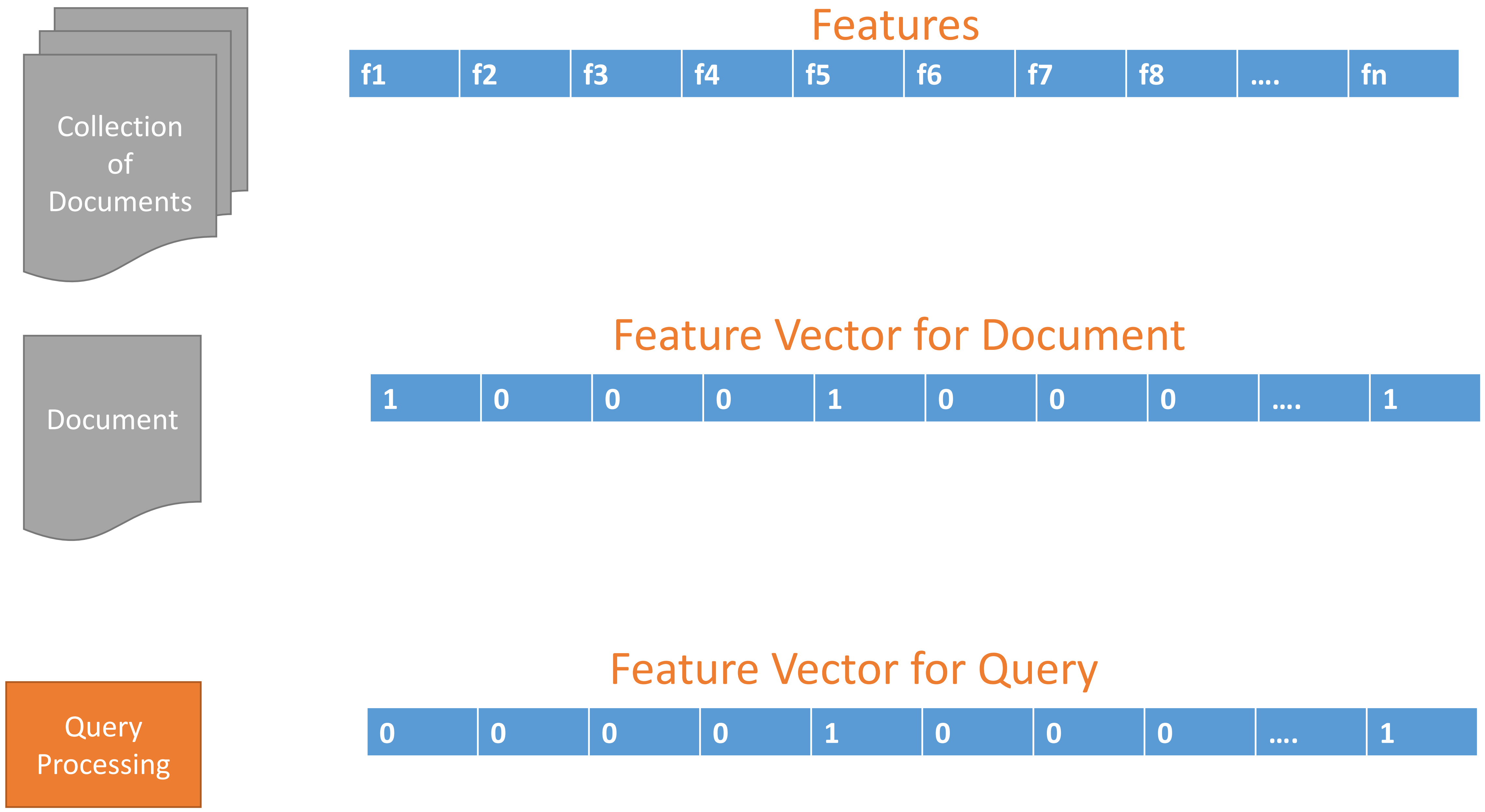
Feature Vector for Document

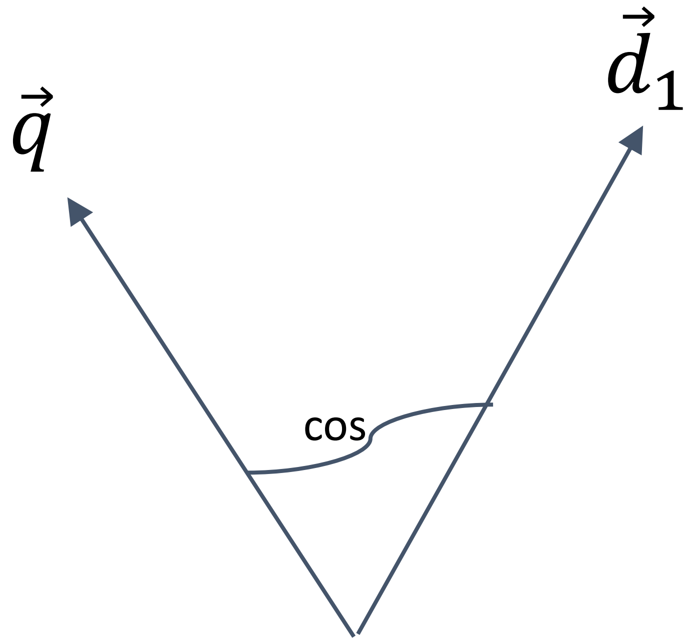
1	0	0	0	1	0	0	0	….	1

Query Processing

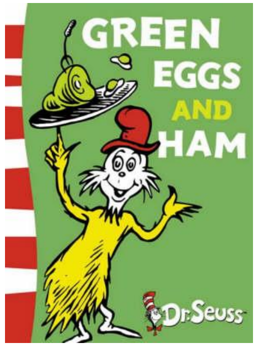
Feature Vector for Query

0	0	0	0	1	0	0	0	….	1


Collection of Documents
Features
f1 f2 f3 f4 f5 f6 f7 f8 …. fn
Document
Feature Vector for Document
1 0 0 0 1 0 0 0 …. 1
Query Processing
Feature Vector for Query
0 0 0 0 1 0 0 0 …. 1



$$Cosine(\vec{x}\,\vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|}$$



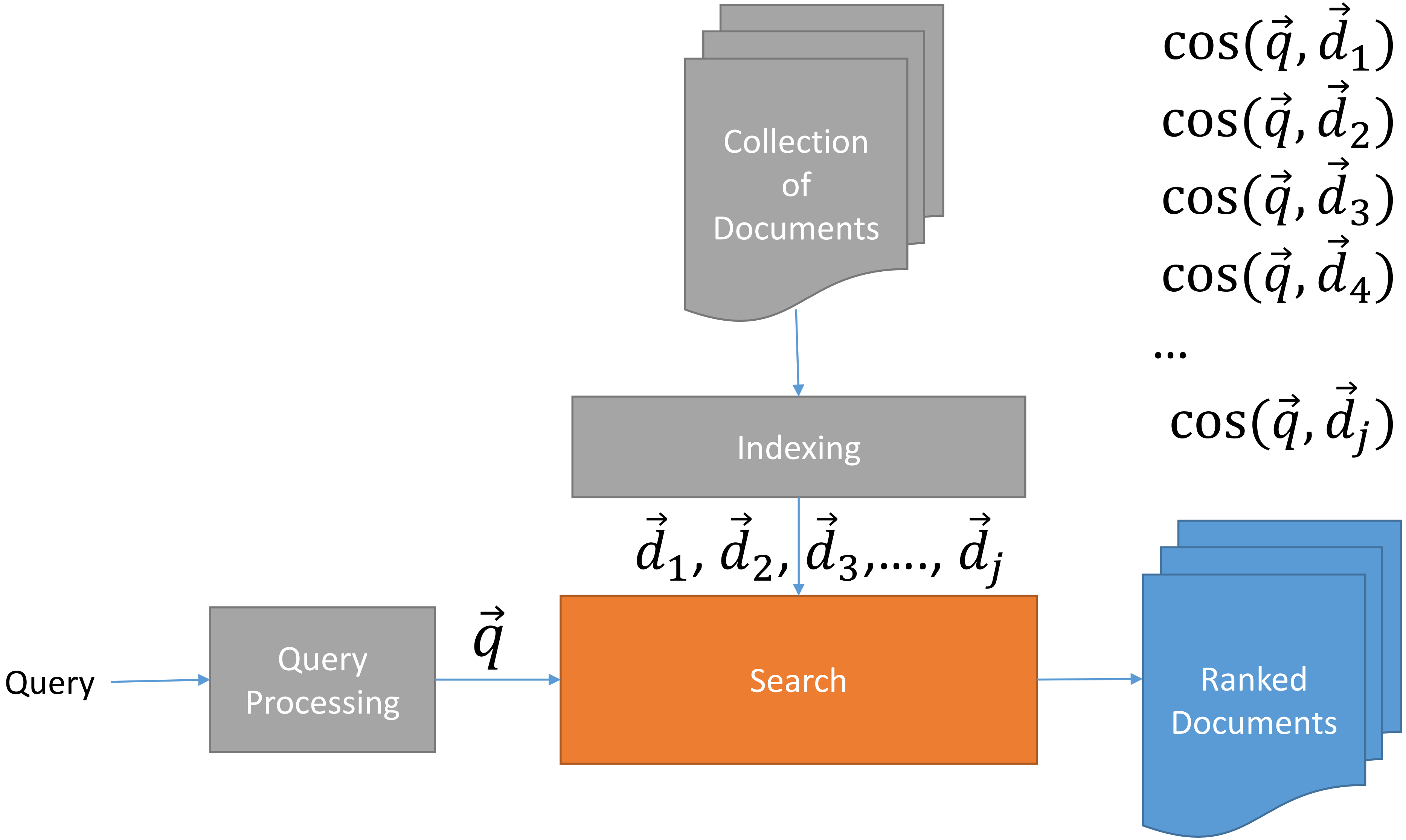
sam I am

Feature Vector for Document

0	0	1	1	1	1	0	0	....	0

Feature Vector for Query

0	0	0	0	0	1	0	0	....	0



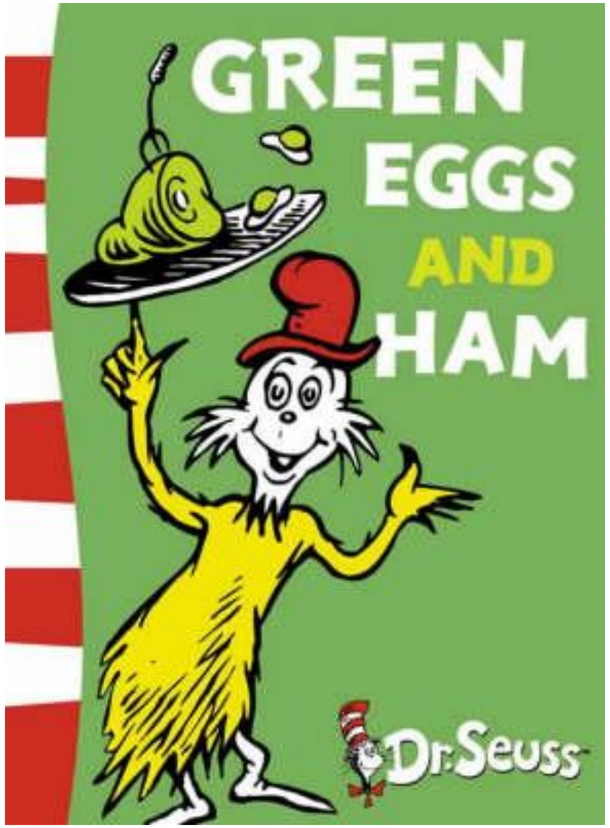
$$\cos(\vec{q}, \vec{d}_1) = rank_1$$
$$\cos(\vec{q}, \vec{d}_2) = rank_2$$
$$\cos(\vec{q}, \vec{d}_3) = rank_3$$
$$\cos(\vec{q}, \vec{d}_4) = rank_4$$
…
$$\cos(\vec{q}, \vec{d}_j) = rank_j$$

Preprocessing

Features are words in the documents

Preprocessing

Features are words in the documents



I am sam. Sam I am. That Sam I am. That Sam I am. I do not like that Sam I am.

Do you like green eggs and ham?

I do not like them Sam I am. I do not l like green eggs and ham.

Do you like them in box?
Would you like them with a fox?

- stoplist
- punctuation
- lemmatization
- stemming

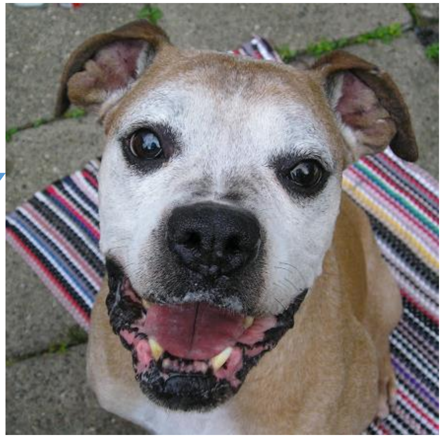
Disadvantages

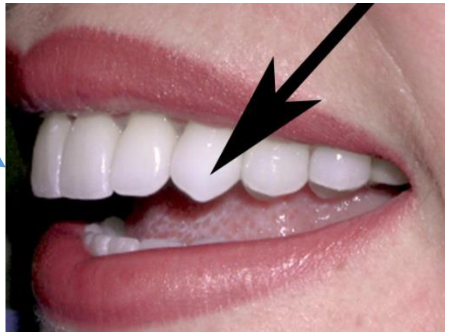
Disadvantages

north american
canine breeders

Query
Processing

Search

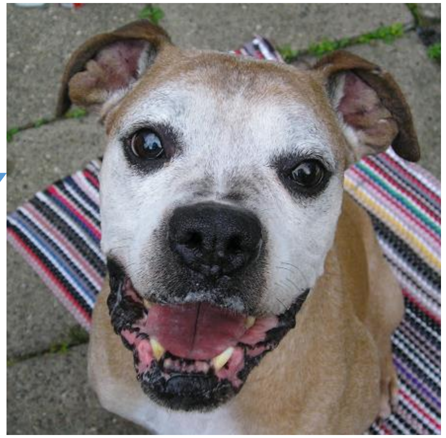


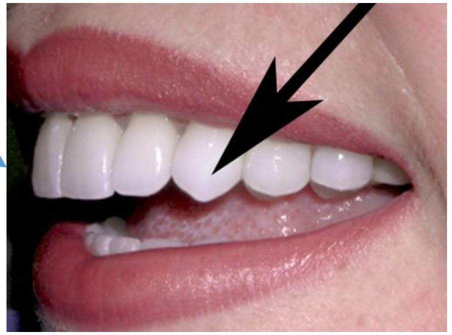


VSM = Vector Space Model

Disadvantages

north american canine breeders → Query Processing → Search





What NLP system could help us with this? WSD

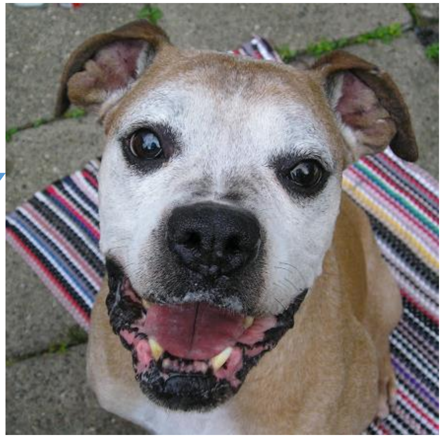
VSM = Vector Space Model

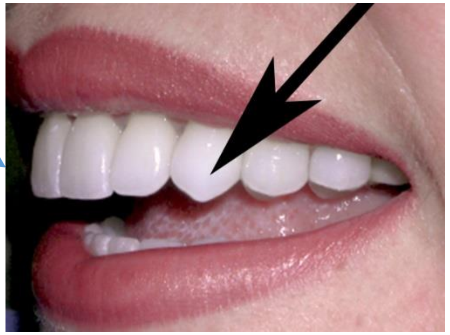
Disadvantages

north american
canine breeders

Query
Processing

Search



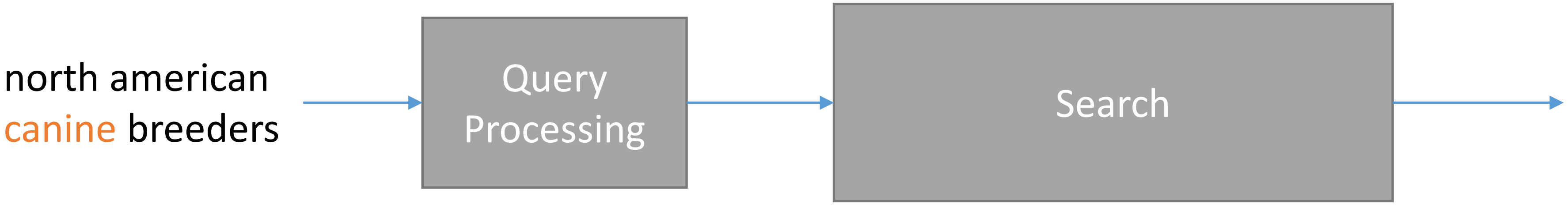


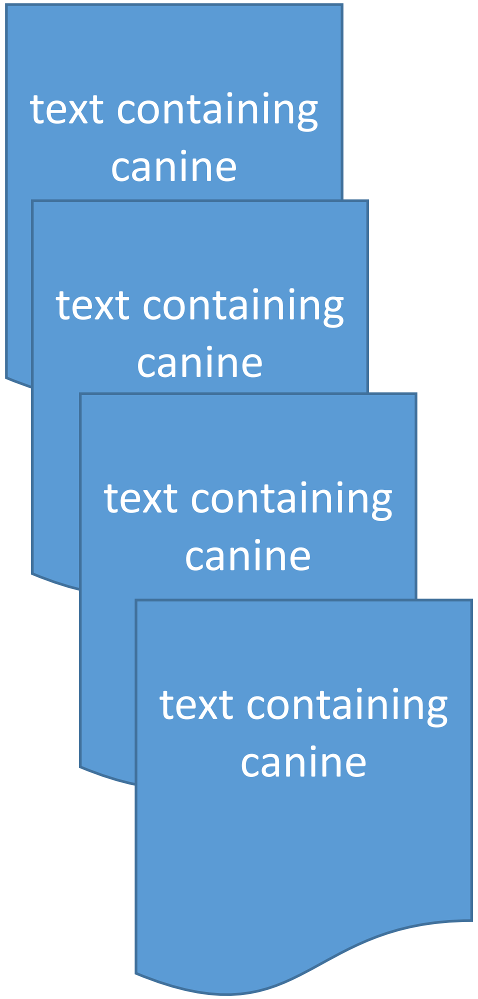
What NLP system could help us with this? WSD

Cost benefit analysis:

- What is the rate of ambiguity in the domain?
- Is the WSD system accurate enough to aid in the disambiguation?

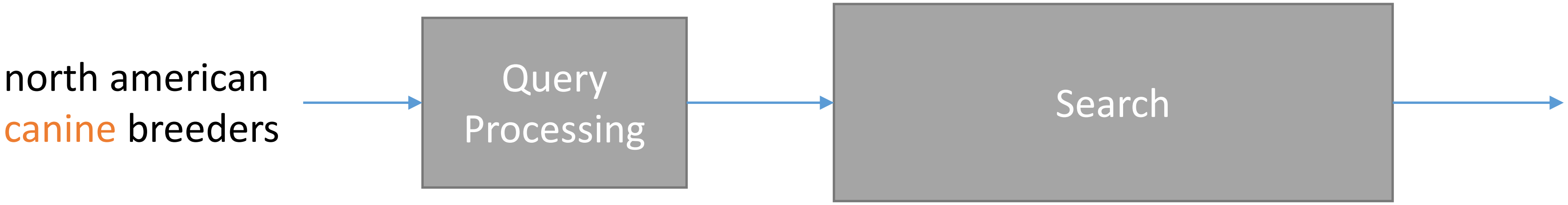
Disadvantages

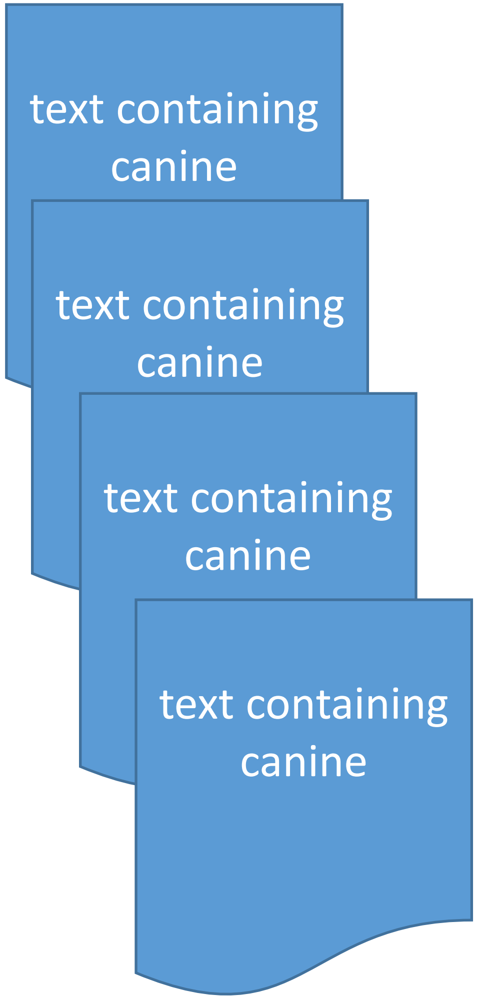


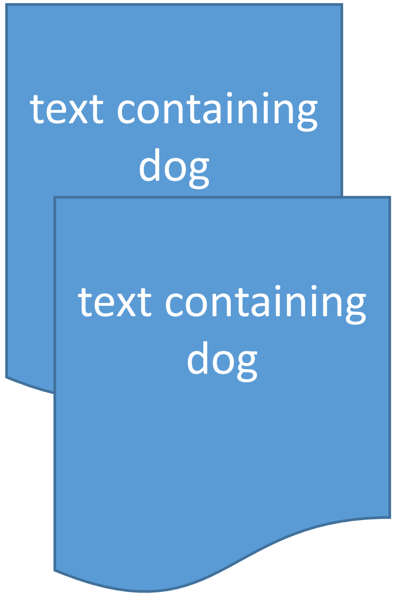


But what will it be missing?

Disadvantages

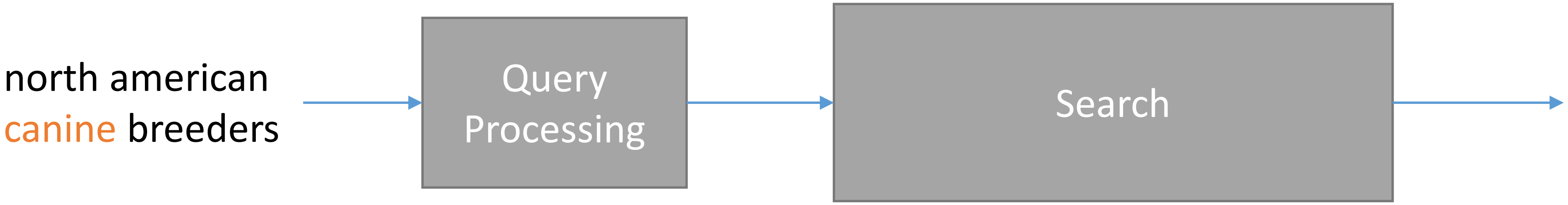


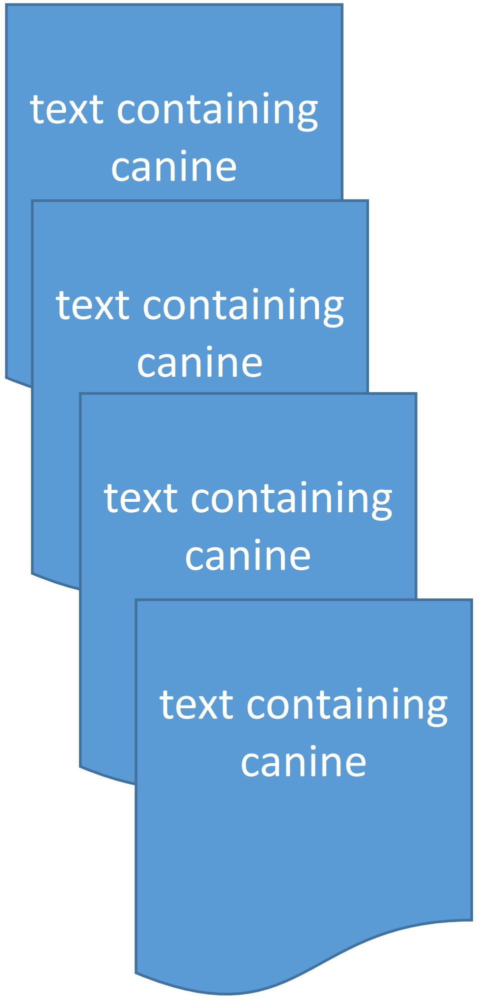




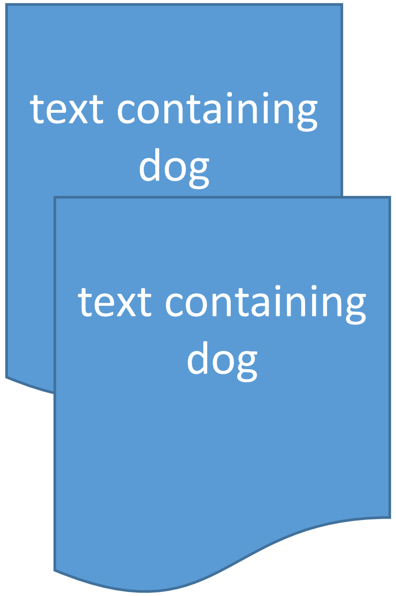
But what will it be missing?

Disadvantages





But what will it be missing?



What NLP system can help us with this?

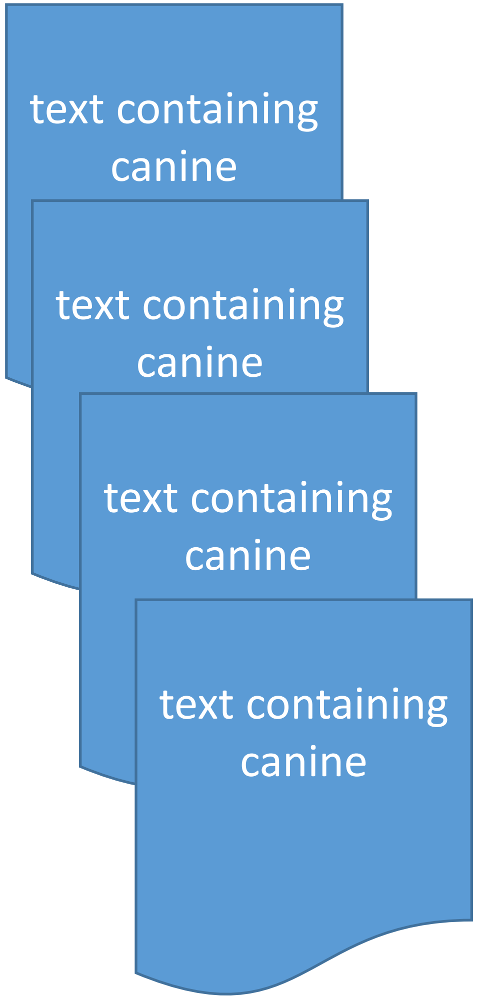
semantic similarity and relatedness

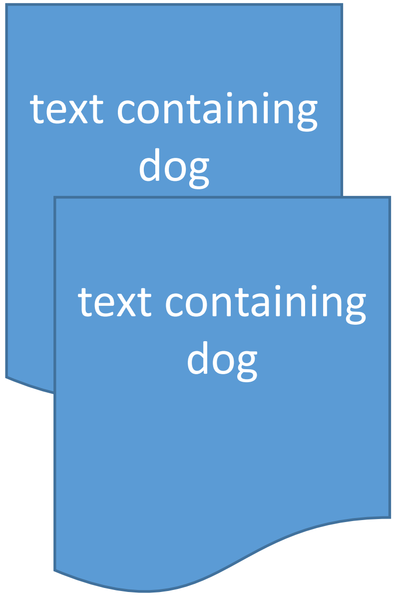
Disadvantages

What NLP system can help us with this?

semantic similarity and relatedness

north american canine breeders





Cost benefit analysis:

- computational time?
- what semantic similarity or relatedness threshold should we use?
- what measure should we use?

Query Expansion

Query Expansion

The user's original query is expanded by addition of terms that are synonymous with or related to the original terms

what doggy breeds are good with kids

what doggy breeds are good with kids

- doggy:
  - dog
  - canine
  - mutt
  - pooch
- good:
  - excellent
  - safe
  - tolerant
- breeds:
  - types

what doggy breeds are good with kids

- doggy:
  - dog
  - canine
  - mutt
  - pooch
- good:
  - excellent
  - safe
  - tolerant
- breeds:
  - types
- do we remove: what, are and with

what doggy breeds are good with kids

- doggy:
  - dog
  - canine
  - mutt
  - pooch
- good:
  - excellent
  - safe
  - tolerant

- breeds:
  - types
- do we remove: what, are and with

Evaluation of Information Retrieval

- Precision: how many documents returned are relevant
- Recall: how many of the documents did you miss

$$Precision = \frac{|R|}{|T|} \qquad Recall = \frac{|R|}{|U|}$$

R = relevant documents returned by the system

T = total documents returned by the system

U = documents in the collection that are relevant

Evaluation

Problem with these two metrics for IR

- Does not incorporate any rank information.

System 1 ranking: $\underbrace{\vec{d}_1\ \vec{d}_2\ \vec{d}_3\ \vec{d}_4\ \vec{d}_5}_{\text{Not relevant}}\ \underbrace{\vec{d}_6\vec{d}_7\vec{d}_8\ \vec{d}_9\ \vec{d}_{10}}_{\text{Relevant}}$

System 2 ranking: $\underbrace{\vec{d}_1\ \vec{d}_2\ \vec{d}_3\ \vec{d}_4\ \vec{d}_5}_{\text{Relevant}}\ \underbrace{\vec{d}_6\vec{d}_7\vec{d}_8\ \vec{d}_9\ \vec{d}_{10}}_{\text{Not relevant}}$

Precision and Recall for both systems is the same but which is the better system?

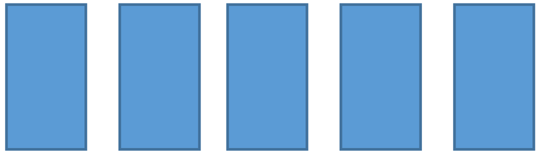
Mean Average Precision (MAP)

- In this approach, descend through the ranked list of terms and note the precision only at those points where a relevant item has been encountered

so we are weighting the precision on the ranking

$$MAP = \frac{1}{R_r} \sum_{d \in R_r} Precision_r(d)$$

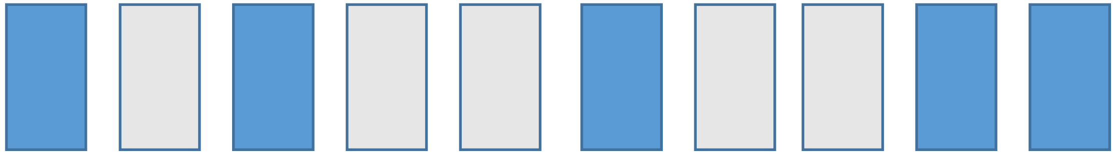
Mean Average Precision (MAP) Example



= relevant documents for query

$$MAP = \frac{1}{R_r} \sum_{d \in R_r} Precision_r(d)$$

Ranking #1



Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5

$$MAP = \frac{(1.0 + 0.67 + 0.5 + 0.44 + 0.5)}{5} = 0.62$$

Why bother doing Information Retrieval?

Why not just processing the entire document set?

- Time consuming

Why bother doing Information Retrieval?

Why not just processing the entire document set?

- Time consuming
- Errors

Why bother doing Information Retrieval?

Why not just processing the entire document set?

- Time consuming
- Errors
- Reduces ambiguity

Natural language processing stack