Retrieving documents of interest

Document retrieval

- Currently reading article you like

Document retrieval

- Currently reading article you like
- **Goal**: Want to find similar article

Document retrieval

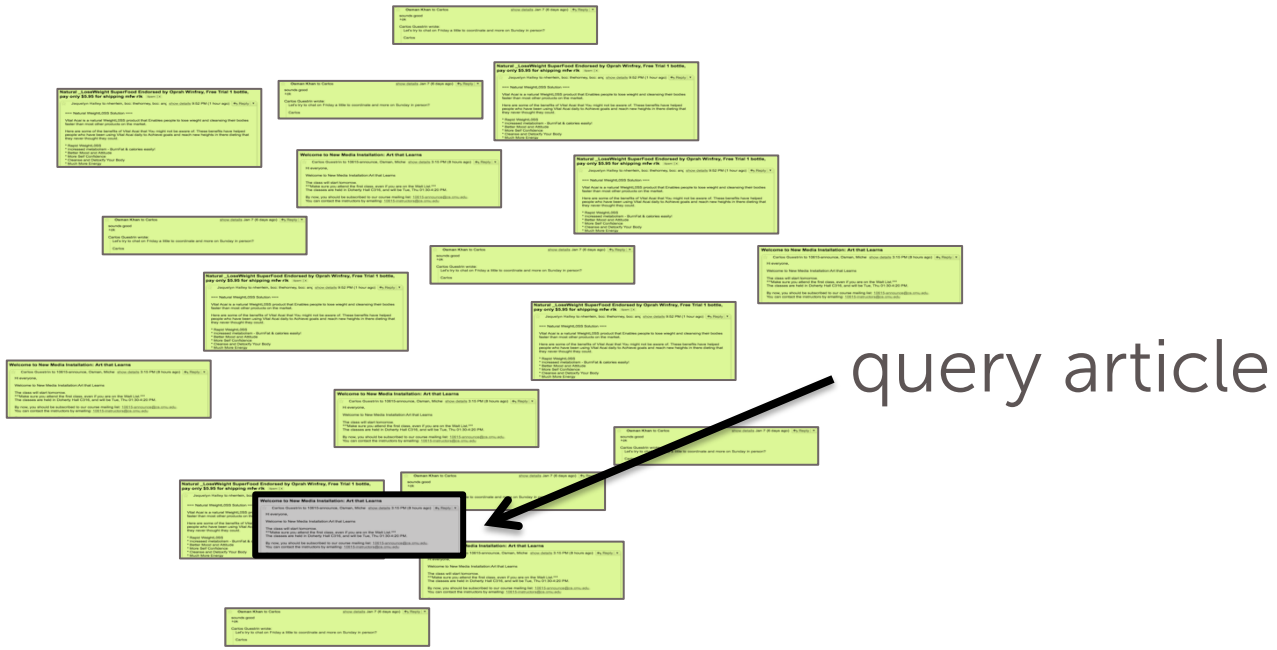
Challenges

- How do we measure similarity?
- How do we search over articles?

Retrieval as k-nearest neighbor search

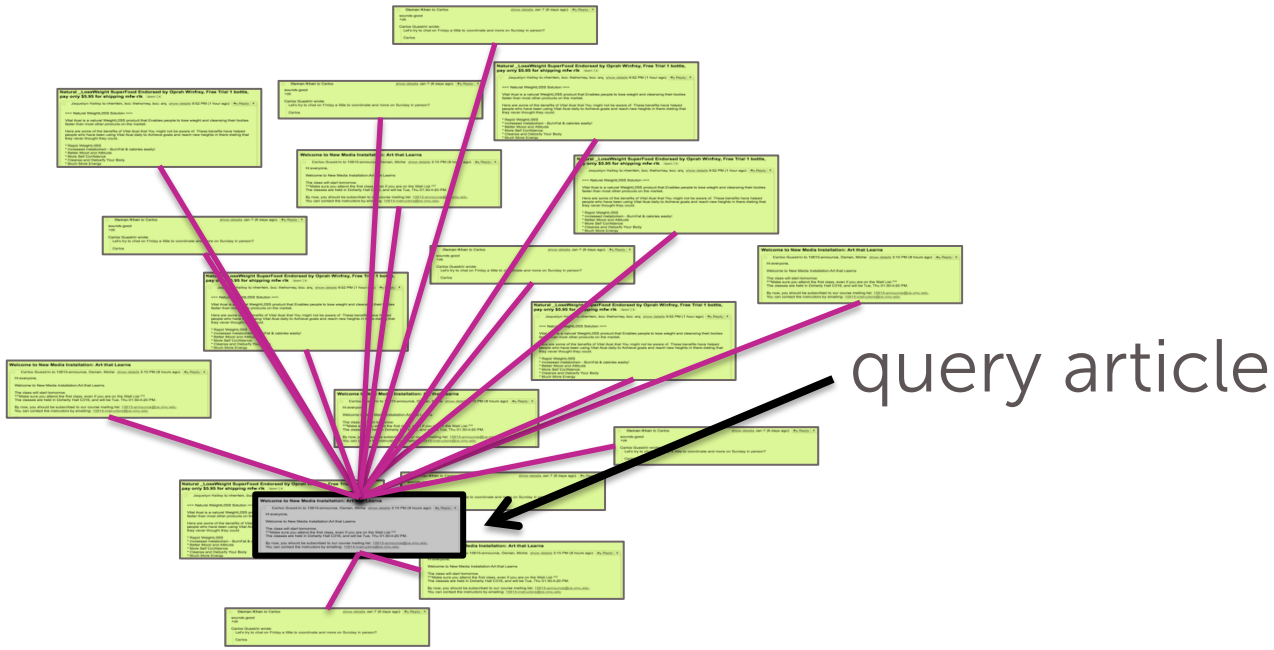
1-NN search for retrieval

Space of all articles,
organized by similarity of text



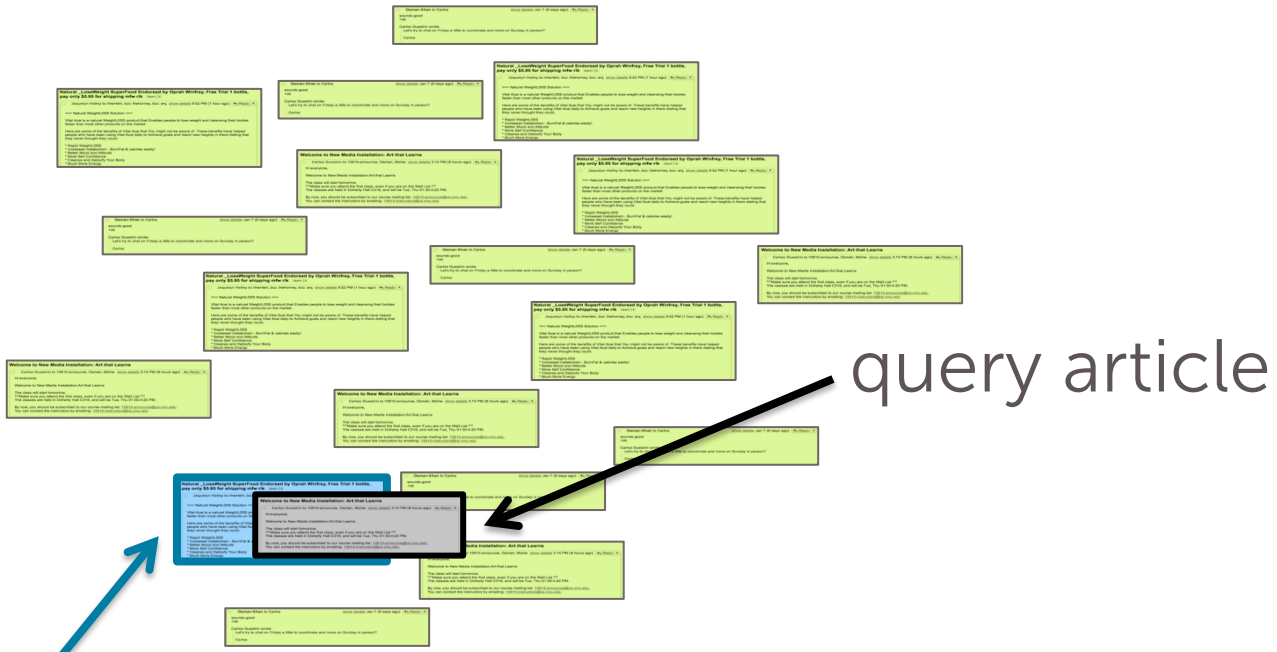
Compute distances to all docs

Space of all articles,
organized by similarity of text



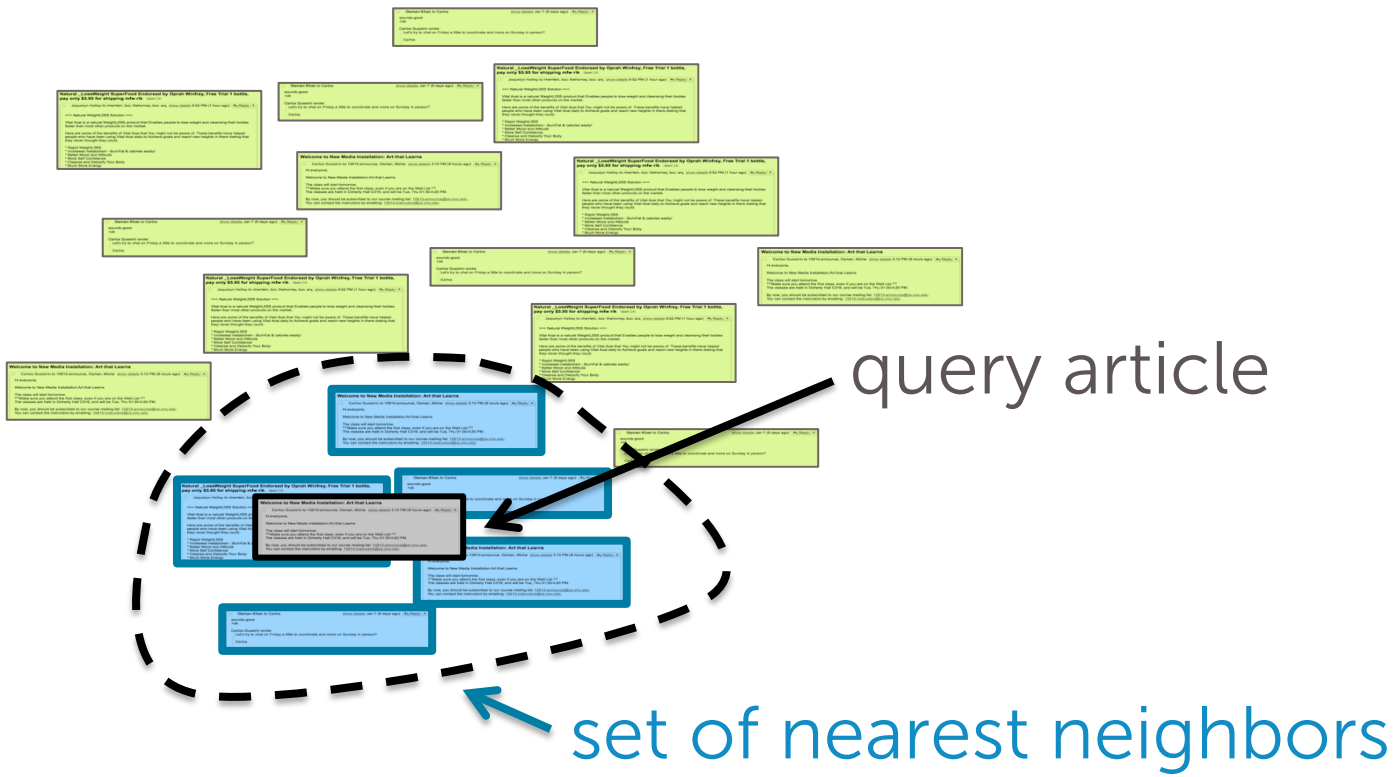
Retrieve “nearest neighbor”

Space of all articles,
organized by similarity of text



Or set of nearest neighbors

Space of all articles,
organized by similarity of text



1-NN algorithm

1 – Nearest neighbor

- **Input:** Query article  : $\mathbf{x}_q$

  Corpus of documents (N docs)

   : $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$

- **Output:** *Most* similar article  $\leftarrow x^{NN}$

Formally: $x^{NN} = \min_{x_i} \text{distance}(x_q, x_i)$

1-NN algorithm

closest document

Initialize **Dist2NN** = ∞,  = ∅

For i=1,2,…,N

query document x_q

x_i

Compute: δ = **distance(**  ,  **)**

document i from corpus

If δ < **Dist2NN**

x_i

set  = 

set **Dist2NN** = δ

Return most similar document  x^{NN}

closest document in corpus to query article

k-NN algorithm

k – Nearest neighbor

- **Input:** Query article  : $\mathbf{x}_q$

  Corpus of documents

   : $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$

- **Output:** *List of k* similar articles

Formally:

$X^{NN} = \{X^{NN_1}, \ldots, X^{NN_k}\}$

for all x_i not in X^{NN}, $\text{distance}(x_i, x_q) \geq \max_{X^{NN_j}, j=1..k} \text{distance}(X^{NN_j}, x_q)$

k-NN algorithm

sort first **k documents** by distance to query doc

Initialize **Dist2kNN** = sort($\delta_1,\ldots,\delta_k$) ← list of sorted distances

=  ,… )  ← list of sorted docs

For i=k+1,…,N

query doc

Compute: δ = **distance(**  ,  **)**

If δ < **Dist2kNN**[k] ← distance to k^{th} NN (furthest NN in set)

find j such that δ > **Dist2kNN**[j-1] but δ < **Dist2kNN**[j]

remove furthest house and shift queue:

 1:k] =  -1]

inserting new article

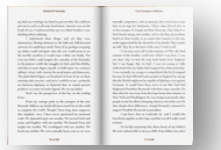
Dist2kNN[j+1:k] = **Dist2kNN**[j:k-1]

set **Dist2kNN**[j] = δ and  [j] = 

Return k most similar articles ← closest k docs to query doc

Critical elements of NN search

Item (e.g., doc) representation

$\mathbf{x}_q \leftarrow$ 

Measure of distance between items:

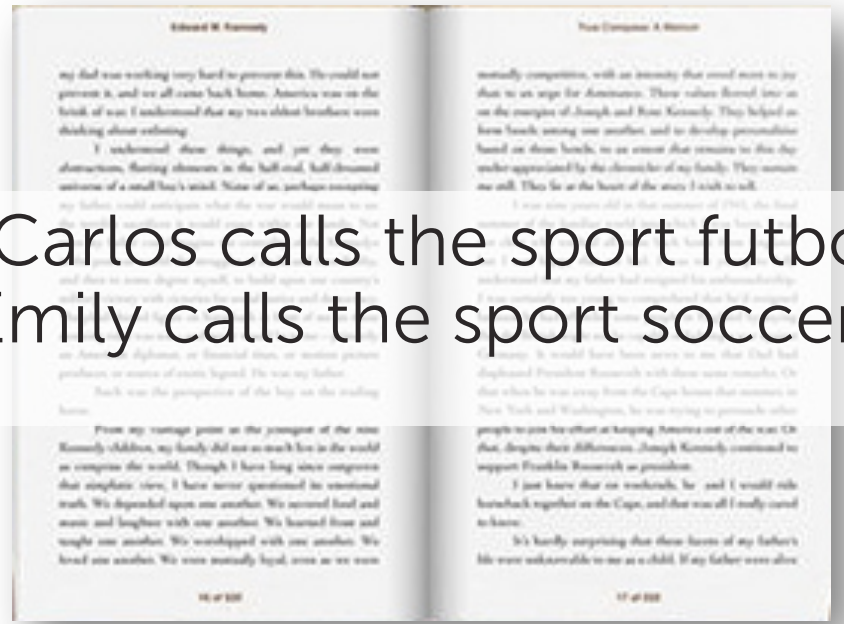
δ = distance($\mathbf{x}_i$, $\mathbf{x}_q$)

Document representation

Word count document representation

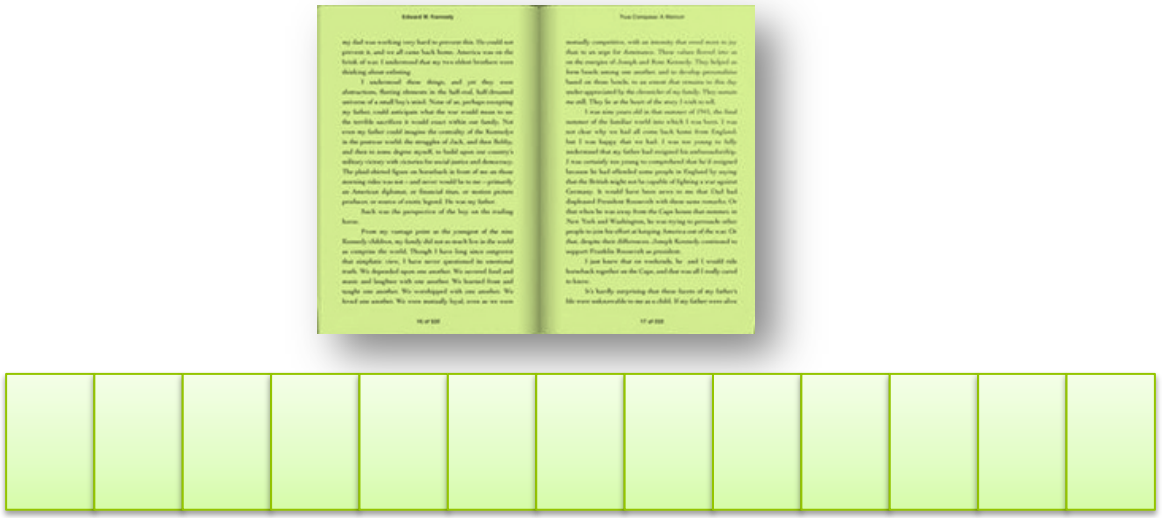
Bag of words model

- Ignore order of words
- Count # of instances of each word in vocabulary



"Carlos calls the sport futbol. Emily calls the sport soccer."

Issues with word counts – Rare words



Common words in doc: “the”, “player”, “field”, “goal”

Dominate rare words like: “futbol”, “Messi”

TF-IDF document representation

Emphasizes important words

- Appears frequently in document (common locally)
- Appears rarely in corpus (rare globally)

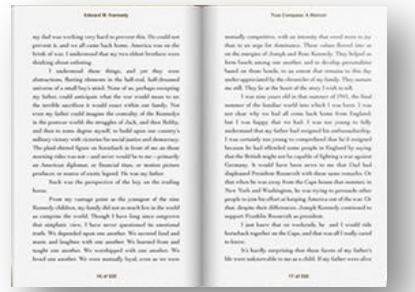
TF-IDF document representation

Emphasizes important words

- Appears frequently in document (common locally)

  Term frequency = word counts

- Appears rarely in corpus (rare globally)



TF-IDF document representation

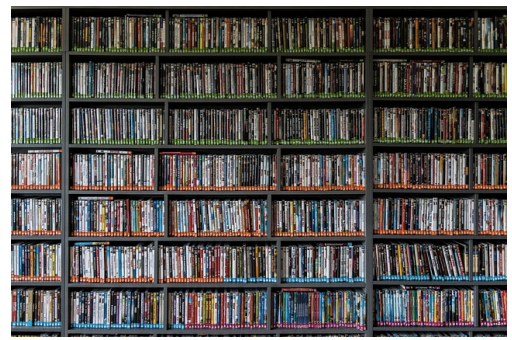
Emphasizes important words

- Appears frequently in document (common locally)

Term frequency = word counts

- Appears rarely in corpus (rare globally)

Inverse doc freq. = $\log \frac{\#\text{ docs}}{1 + \#\text{ docs using word}}$



TF-IDF document representation

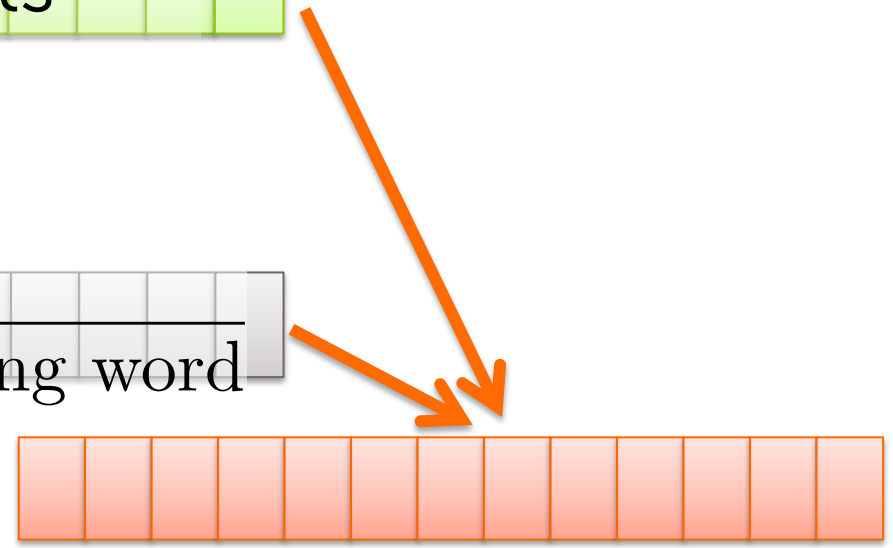
Emphasizes important words

- Appears frequently in document (common locally)

Term frequency = word counts

- Appears rarely in corpus (rare globally)

Inverse doc freq. = $\log \frac{\#\text{ docs}}{1 + \#\text{ docs using word}}$



Trade off: local frequency vs. global rarity

Distance metrics

Distance metrics: Defining notion of "closest"

In 1D, just Euclidean distance:

$$\text{distance}(x_i, x_q) = |x_i - x_q|$$

In multiple dimensions:

- can define many interesting distance functions
- most straightforwardly, might want to weight different dimensions differently

Weighting different features

Reasons:

– Some features are more relevant than others

 # bedrooms
 # bathrooms
sq.ft. living
sq.ft. lot
floors
year built
year renovated
waterfront

Weighting different features

Reasons:

– Some features are more relevant than others

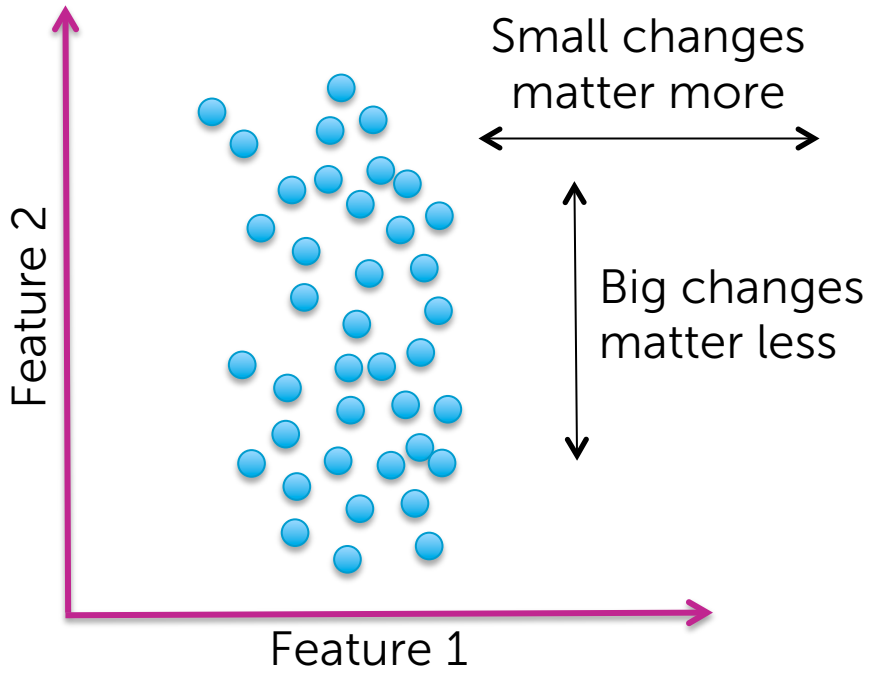
title
abstract
main body
conclusion

Weighting different features

Reasons:

– Some features are more relevant than others

– Some features vary more than others



Specify weights as a function of feature spread

For feature j:

$$\frac{1}{\max_i(\mathbf{x}_i[j]) - \min_i(\mathbf{x}_i[j])}$$

Scaled Euclidean distance

Formally, this is achieved via

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{a_1(\mathbf{x}_i[1]-\mathbf{x}_q[1])^2 + \ldots + a_d(\mathbf{x}_i[d]-\mathbf{x}_q[d])^2}$$

weight on each feature
(defining relative importance)

Effect of binary weights

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{a_1(\mathbf{x}_i[1]-\mathbf{x}_q[1])^2 + \ldots + a_d(\mathbf{x}_i[d]-\mathbf{x}_q[d])^2}$$

Setting weights as 0 or 1 is equivalent to **feature selection**

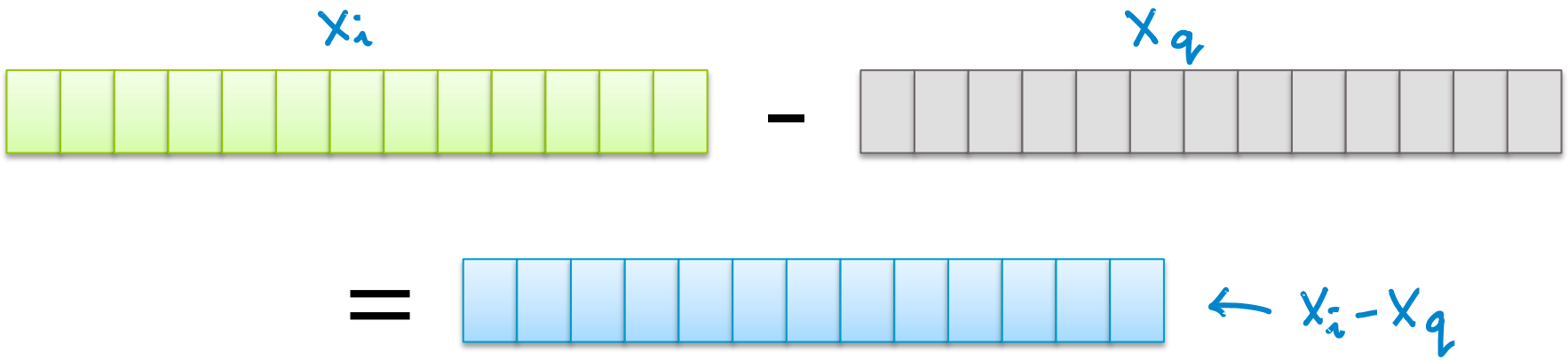
Feature engineering/ selection is **important**, but **hard**

(non-scaled) Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T(\mathbf{x}_i - \mathbf{x}_q)}$$

$$= \sqrt{(\mathbf{x}_i[1]-\mathbf{x}_q[1])^2 + \ldots + (\mathbf{x}_i[d]-\mathbf{x}_q[d])^2}$$



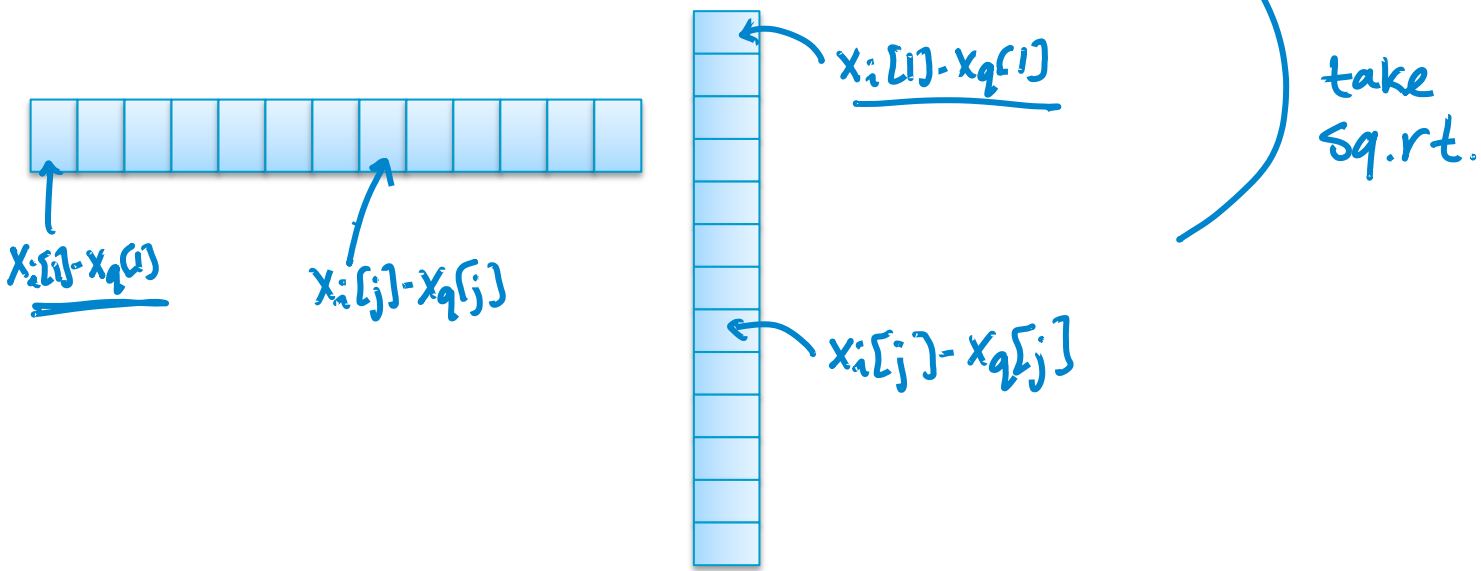
(non-scaled) Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T(\mathbf{x}_i - \mathbf{x}_q)}$$

$$= \sqrt{(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2 + \ldots + (\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$

distance^2 =



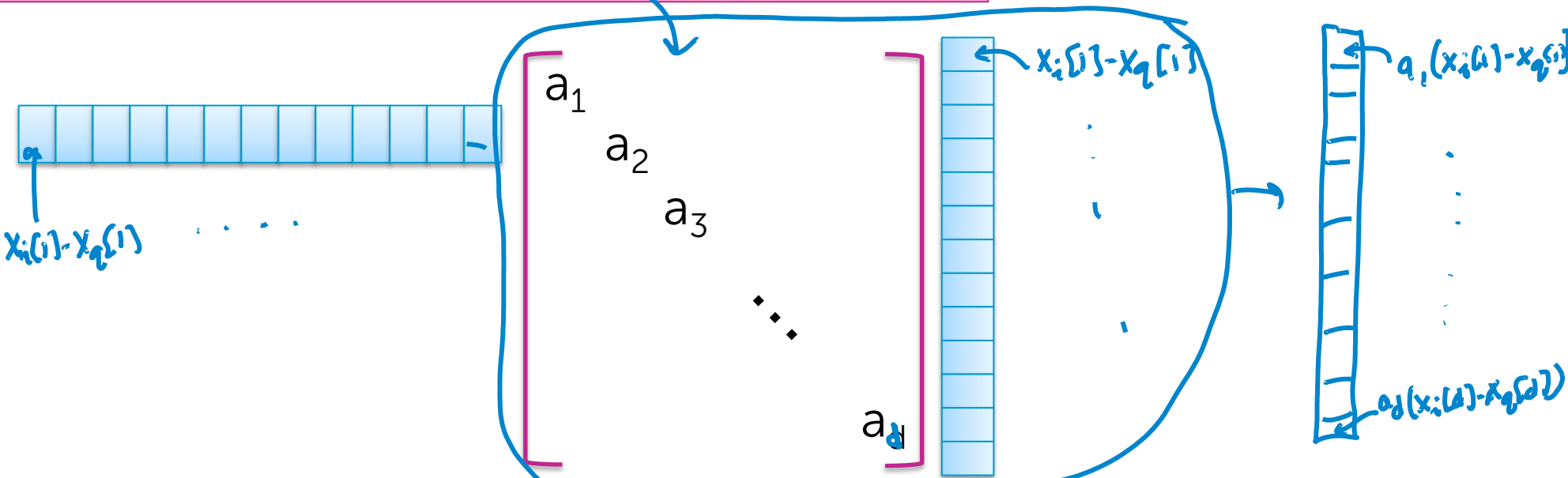
Scaled Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}_q)}$$

$$= \sqrt{a_1(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2 + \ldots + a_d(\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$

distance^2 =



Another natural inner product measure

x_q

1	0	0	0	5	3	0	0	1	0	0	0	0

Similarity

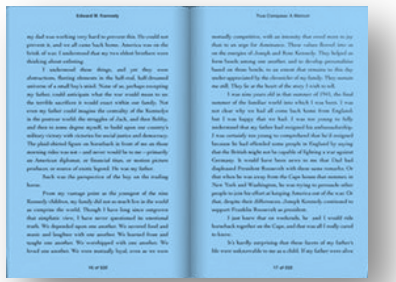
$$= \mathbf{x}_i^T \mathbf{x}_q$$

$$= \sum_{j=1}^{d} \mathbf{x}_i[j]\, \mathbf{x}_q[j]$$

3	0	0	0	2	0	0	1	0	1	0	0	0

x_i

= 13



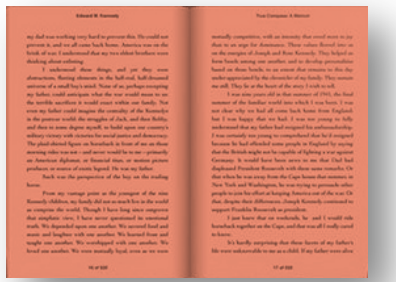
Another natural inner product measure

1	0	0	0	5	3	0	0	1	0	0	0	0

Similarity

= 0

0	0	1	0	0	0	9	0	0	6	0	4	0



Cosine similarity – normalize

Similarity = $\dfrac{\sum_{j=1}^{d} \mathbf{x}_i[j]\,\mathbf{x}_q[j]}{\sqrt{\sum_{j=1}^{d} (\mathbf{x}_i[j])^2}\sqrt{\sum_{j=1}^{d} (\mathbf{x}_q[j])^2}}$

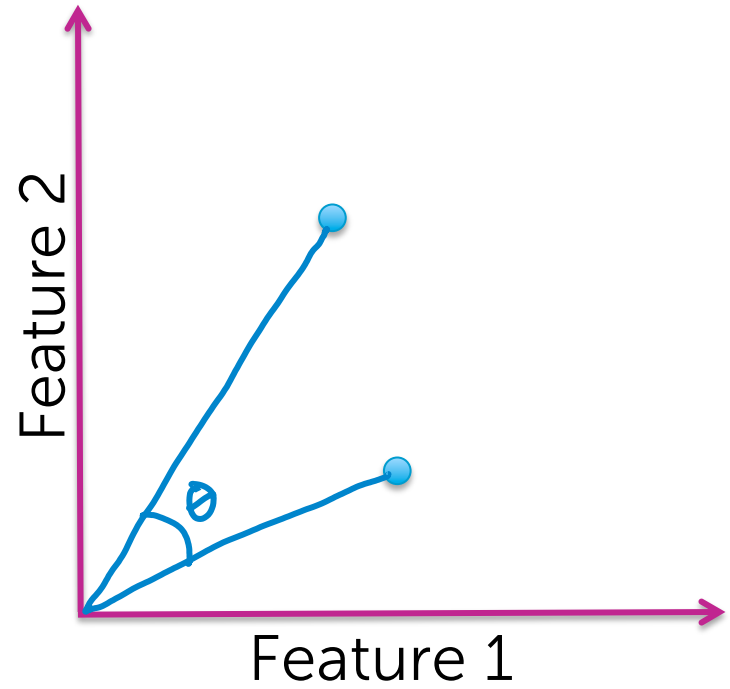
$a^T b = \|a\|\|b\|\cos(\theta)$

$$\mathbf{x}_i^T \mathbf{x}_q = \cos(\theta)$$

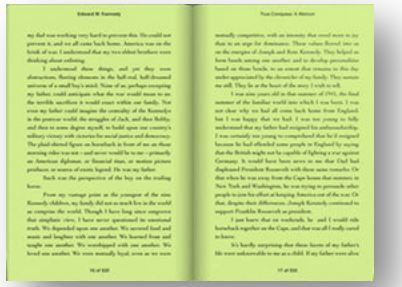
$$\|\mathbf{x}_i\|\,\|\mathbf{x}_q\|$$

$= \left(\frac{x_i}{\|x_i\|}\right)^T \left(\frac{x_q}{\|x_q\|}\right)$ first normalize

- Not a proper distance metric
- Efficient to compute for sparse vecs



Normalize



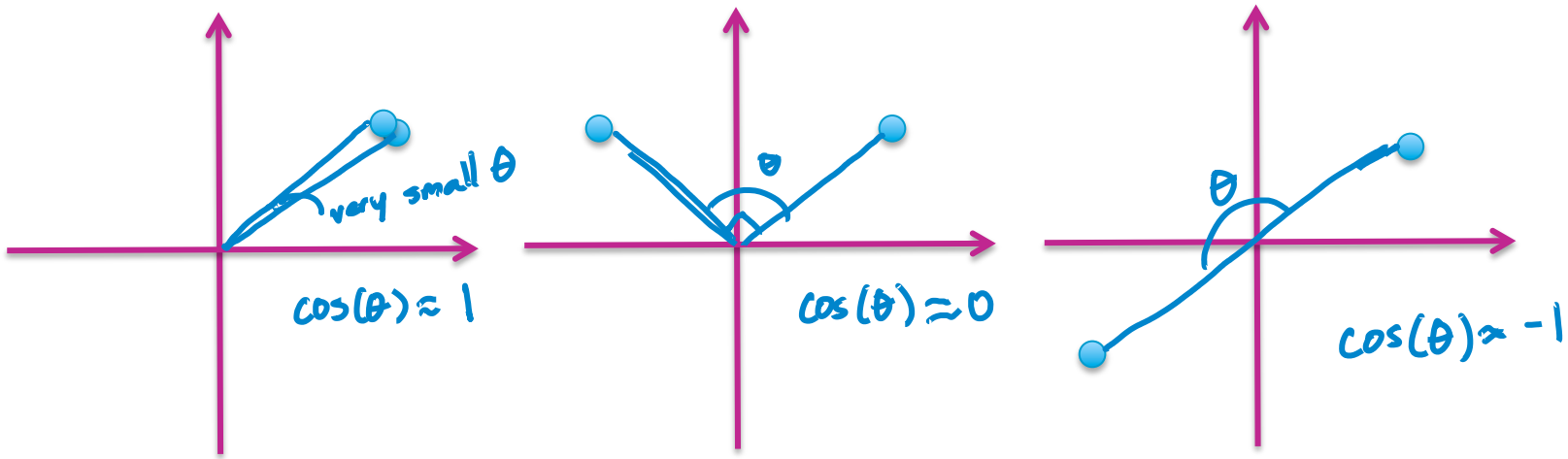
1	0	0	0	5	3	0	0	1	0	0	0	0

$\leftarrow x_i$

$$\sqrt{(1^2 + 5^2 + 3^2 + 1^2)} \leftarrow \|x_i\| = \sum_{j=1}^{d} x_i[j]^2$$

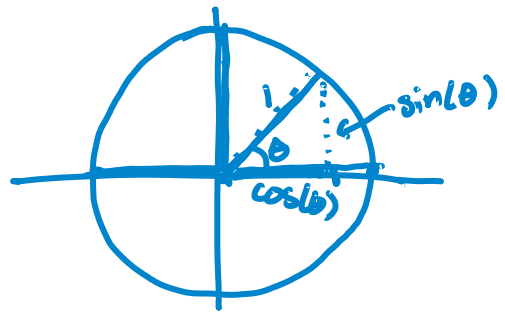
1/6	0	0	0	5/6	3/6	0	0	1/6	0	0	0	0

Cosine similarity



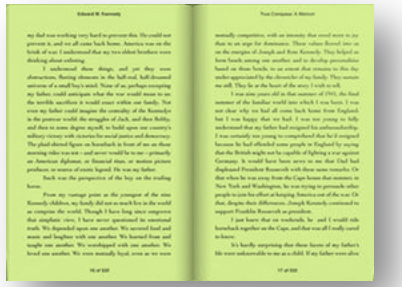
In general, $-1 <$ similarity < 1

For positive features (like tf-idf)
$0 <$ similarity < 1 } our focus



Define **distance = 1-similarity**

To normalize or not?



1	0	0	0	5	3	0	0	1	0	0	0	0

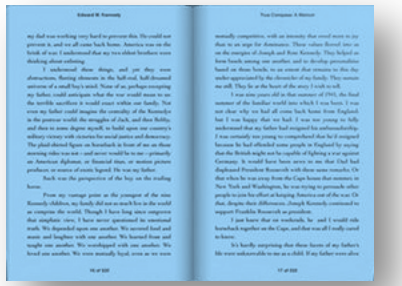
2	0	0	0	10	6	0	0	2	0	0	0	0

3	1	0	0	2	0	0	1	0	1	0	0	0

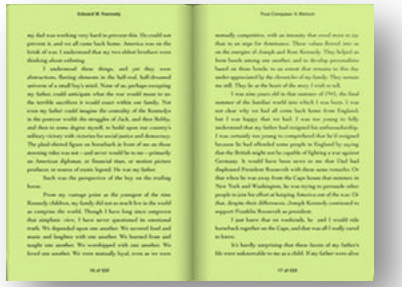
Similarity = 13

6	2	0	0	4	0	0	2	0	2	0	0	0

Similarity = 52



Normalize

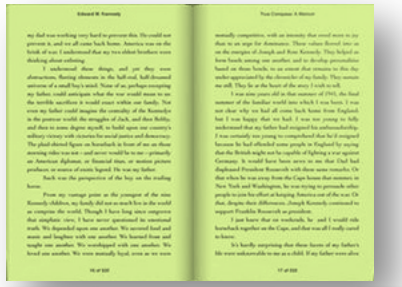


1	0	0	0	5	3	0	0	1	0	0	0	0

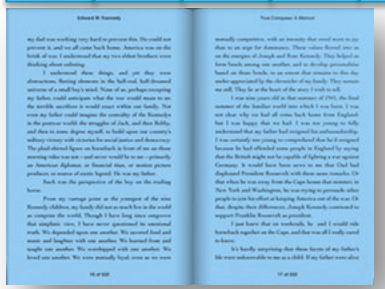
$$\sqrt{(1^2 + 5^2 + 3^2 + 1^2)}$$

1/6	0	0	0	5/6	3/6	0	0	1/6	0	0	0	0

In the normalized case

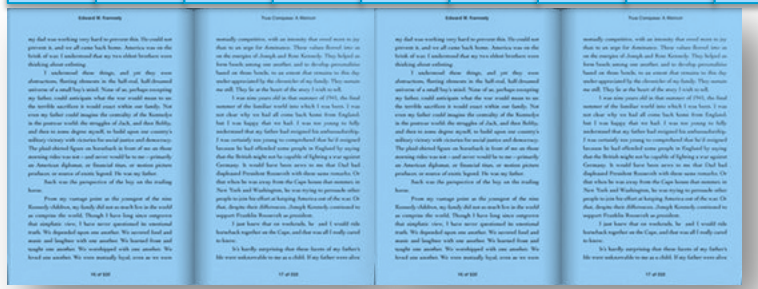


1/6	0	0	0	5/6	3/6	0	0	1/6	0	0	0	0
3/4	1/4	0	0	1/2	0	0	1/4	0	1/4	0	0	0



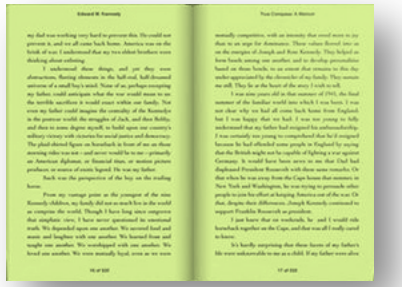
Similarity = 13/24

1/6	0	0	0	5/6	3/6	0	0	1/6	0	0	0	0
3/4	1/4	0	0	1/2	0	0	1/4	0	1/4	0	0	0



Similarity = 13/24

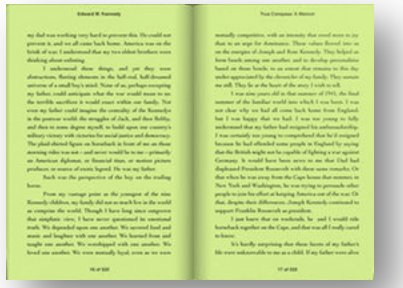
But not always desired…



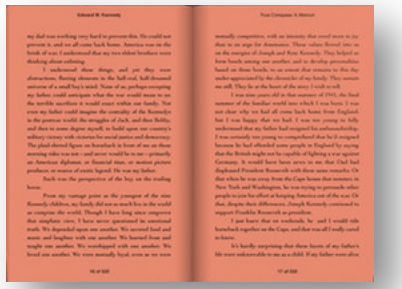
long document

short tweet

Normalizing can make dissimilar objects appear more similar



long document



long document

Common compromise:
Just cap maximum word counts

Other distance metrics

- Mahalanobis
- rank-based
- correlation-based
- Manhattan
- Jaccard
- Hamming
- …

Combining distance metrics

Example of document features:

1. Text of document
   - Distance metric: Cosine similarity
2. # of reads of doc
   - Distance metric: Euclidean distance

Add together with user-specified weights