Advanced Human Language Technologies

Similarity Models

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Outline

Similarity Models

- Similarity models measure how alike are two objects (products, patients, molecules, words, sentences, . . .).
- Objects (words, sentences, documents...) are represented as feature-vectors, feature-sets, distribution-vectors, ...
- *Similarity* may also be interpreted as *proximity* or *affinity*
- *Similarity* may also be seen as the opposite of *distance*, *difference*, or *divergence*.
- Different uses and applications in AI.

Applications of Similarity Models

- **Recommendation systems**. E.g. finding similar patients to propose similar treatments, finding similar products to offer them as potentially interesting, find similar news items to recommend, etc.
- **Prediction systems**. (Example-based Learning, EBL). E.g. predict possible diagnoses based on similar patients, predict product sales based on similar products, classify news items based on similar texts, etc.
- **Clustering systems**. E.g.: Group data in clusters to discover new patterns, offer aggregated views to the user, speed up searches, etc.

Applications of Similarity Models to HLT

- **Text similarity tasks**: Plagiarism detection, news items tracking, related readings recommendation, question answering, FAQ management, ...
- **Text analysis tasks**: Tasks such as PoS Tagging, parsing, NERC, etc can be approached using EBL.
- **Text Classification tasks**: (EBL, again). E.g.: news items routing, sentiment analysis, spam detection, ...
- **Evaluation of NL generation tasks**: Evaluate machine translation, automatic summarization, or report generation comparing the system output with reference texts.
- **Alias detection**: (Useful for coreference detection) find different mentions of the same entity (e.g. *Stanford President John Hennessy*, *Stanford University President Hennessy*, *President John Hennessy*, *Stanford Provost John Hindirck*).

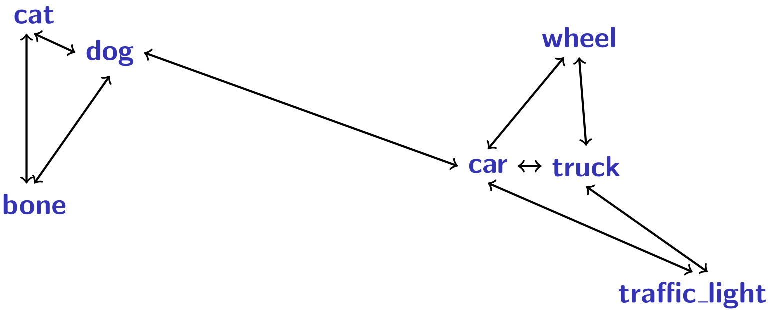
Distance, Similarity, & Relatedness

- We talk about *distance* when metric properties hold:
  - $d(x, x) = 0$
  - $d(x, y) > 0$ when $x \neq y$
  - $d(x, y) = d(y, x)$ (simmetry)
  - $d(x, z) \leqslant d(x, y) + d(y, z)$ (triangular inequation)
- We use *similarity* in the general case
  - Function: $sim : A \times B \rightarrow S$ (where S is often $[0, 1]$)
  - Homogeneous: $sim : A \times A \rightarrow S$ (e.g. word-to-word)
  - Heterogeneous: $sim : A \times B \rightarrow S$ (e.g. word-to-document)
  - Not necessarily symmetric, or holding triangular inequation.
- We can compute one from the other:

$$sim(A, B) = \frac{1}{1 + d(A, B)}; \quad d(A, B) = \frac{1}{sim(A, B)} - 1$$

- *Similarity* is often interpreted as a measure of *relatedness*.

Distance, Similarity, & Relatedness



$d(car, wheel) > d(car, truck)$; $sim(car, wheel) < sim(car, truck)$;

$d(car, dog) >> d(car, truck)$; $sim(car, dog) << sim(car, truck)$;

$d(cat, bone) > d(dog, bone)$; $sim(cat, bone) < sim(dog, bone)$;

Information used to compute similarity

The utility/meaning of a similarity/distance measure depends on how compared objects are represented.

- Information internal to compared units
  - Words: char n-grams, word form, lemma, morphology, PoS, sense, domain, ...
  - Sentences/Documents: bag of words, parse tree, syntactic roles, collocations, word n-grams, Named Entities, ...
- Information external to compared units (context)
  - Words: bag-of-words in context, parse tree, collocations, word n-grams, Named Entities, ...
  - Sentences/Documents: Words in nearby sentences, document meta-information, ...

Approaches to Similarity Computation

- **String/Sequence edit-distance approaches.**
  Can only be applied to sequences of elements (characters, words, proteins...)
- **Vector/Set based approaches.**
  General approach, can be applied to any kind of object once we represent it as a [feature] vector or set.
  - Vector similarities/distances
  - Set similarities/distances
- **Knowledge-based approaches.**
  Require some (graph-like) knowledge representation.
  - WordNet distances
- **Corpus-based approaches (distributional semantics).**
  Describe meaning based on occurrence contexts.
  - Sparse representations (term-term/term-document matrix)
  - Dense representations (LSI, Word Embeddigns)

Outline

String/Sequence edit-distance approaches

Sequences of any kind

- word : sequence of characters
- sentence : sequence of words (or characters too)
- DNA: sequence of bases A,T,C,G
- Health Record : sequence of clinical events
- ...

Some Edit Distances

- LCS (Longest Common Subsequence): ED allowing deletion and insertion.
- Levenhstein: ED allowing deletion, insertion and substitution.
- Damerau-Levenhstein: ED allowing insertion, deletion, substitution, and transposition of two adjacent elements.

Edit distances can be efficiently computed using dynammic programming.

Example: Levenhstein

```
def Levenshtein(s, t):

   n = len(s)
   m = len(t)
   d = [ [ 0 for j in range(0,m+1) ] for i in range(0,n+1) ]

   # source prefixes can be transformed into empty string by
   # dropping all characters
   for i in range(1,n+1): d[i][0] = i

   # target prefixes can be reached from empty source prefix
   # by inserting every character
   for j in range(1,m+1): d[0][j] = j

   for i in range(1,n+1):
      for j in range(1,m+1):

         subst = 0 if s[i-1] == t[j-1] else 1   # substitution cost

         d[i][j] = min(d[i-1][j] + 1,           # deletion
                       d[i][j-1] + 1,           # insertion
                       d[i-1][j-1] + subst)     # substitution

   return d[n][m]
```

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ									
S									
U									
N									
D									
A									
Y									

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1								
U	2								
N	3								
D	4								
A	5								
Y	6								

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1	0							
U	2								
N	3								
D	4								
A	5								
Y	6								

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1	0	1						
U	2	1	1						
N	3								
D	4								
A	5								
Y	6								

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1	0	1	2					
U	2	1	1	2					
N	3	2	2	2					
D	4								
A	5								
Y	6								

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3				
U	2	1	1	2	2				
N	3	2	2	2	3				
D	4	3	3	3	3				
A	5								
Y	6								

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4			
U	2	1	1	2	2	3			
N	3	2	2	2	3	3			
D	4	3	3	3	3				
A	5								
Y	6								

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5		
U	2	1	1	2	2	3	4		
N	3	2	2	2	3	3	4		
D	4	3	3	3	3	4	3		
A	5	4	3	4	4	4			
Y	6								

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	
U	2	1	1	2	2	3	4	5	
N	3	2	2	2	3	3	4	5	
D	4	3	3	3	3	4	3	4	
A	5	4	3	4	4	4	4	3	
Y	6	5	4	4					

Example: Levenhstein

	λ	S	A	T	U	R	D	A	Y
λ	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
U	2	1	1	2	2	3	4	5	6
N	3	2	2	2	3	3	4	5	6
D	4	3	3	3	3	4	3	4	5
A	5	4	3	4	4	4	4	3	4
Y	6	5	4	4	5	5	5	4	**3**

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ										
Spokesman										
confirms										
senior										
government										
advisor										
was										
shot										

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1									
confirms	2									
senior	3									
government	4									
advisor	5									
was	6									
shot	7									

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1								
confirms	2									
senior	3									
government	4									
advisor	5									
was	6									
shot	7									

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1	2							
confirms	2	2	2							
senior	3									
government	4									
advisor	5									
was	6									
shot	7									

Example: Levenhstein

	λ	The	spokesman	said	the	senior	advisor	was	shot	dead
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1	2	3						
confirms	2	2	2	3						
senior	3	3	3	3						
government	4									
advisor	5									
was	6									
shot	7									

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1	2	3	4					
confirms	2	2	2	3	4					
senior	3	3	3	3	4					
government	4	4	4	4	4					
advisor	5									
was	6									
shot	7									

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1	2	3	4	5				
confirms	2	2	2	3	4	5				
senior	3	3	3	3	4	4				
government	4	4	4	4	4	5				
advisor	5	5	5	5	5	5				
was	6									
shot	7									

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1	2	3	4	5	6			
confirms	2	2	2	3	4	5	6			
senior	3	3	3	3	4	4	5			
government	4	4	4	4	4	5	5			
advisor	5	5	5	5	5	5	5			
was	6									
shot	7									

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1	2	3	4	5	6	7		
confirms	2	2	2	3	4	5	6	7		
senior	3	3	3	3	4	4	5	6		
government	4	4	4	4	4	5	5	6		
advisor	5	5	5	5	5	5	5	6		
was	6	6	6	6	6	6	6	5		
shot	7									

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1	2	3	4	5	6	7	8	
confirms	2	2	2	3	4	5	6	7	8	
senior	3	3	3	3	4	4	5	6	7	
government	4	4	4	4	4	5	5	6	7	
advisor	5	5	5	5	5	5	5	6	7	
was	6	6	6	6	6	6	6	5	6	
shot	7	7	7	7	7	7	7	6	5	

Example: Levenhstein

	λ	*The*	*spokesman*	*said*	*the*	*senior*	*advisor*	*was*	*shot*	*dead*
λ	0	1	2	3	4	5	6	7	8	9
Spokesman	1	1	2	3	4	5	6	7	8	9
confirms	2	2	2	3	4	5	6	7	8	9
senior	3	3	3	3	4	4	5	6	7	8
government	4	4	4	4	4	5	5	6	7	8
advisor	5	5	5	5	5	5	5	6	7	8
was	6	6	6	6	6	6	6	5	6	7
shot	7	7	7	7	7	7	7	6	5	**6**

Outline

Outline

Vector similarities/distances

When objects are represented as [feature] vectors, we can use vector-space distances.

- Manhattan distance
- Euclidean distance
- Chebychev distance
- Camberra distance
- Cosine *similarity*
- Dot Product *similarity*
- ...

Vector similarities/distances

- Commonly used norms belong to the family of Minkowsky distances:

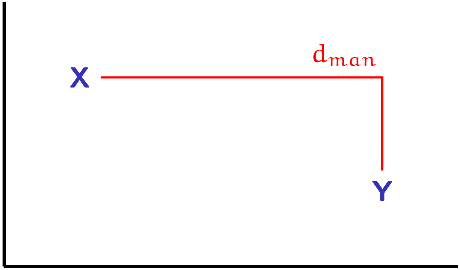
$$d_{min}(\vec{x}, \vec{y}) = L_r(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{N} |x_i - y_i|^r \right)^{\frac{1}{r}}$$

- L_1 and L_2 norms are particular cases of orders 1 and 2
- Chebychev distance is the limit L_∞.

Vector similarities/distances

- L_1 norm, a.k.a. Manhattan distance, taxi-cab distance, city-block distance:

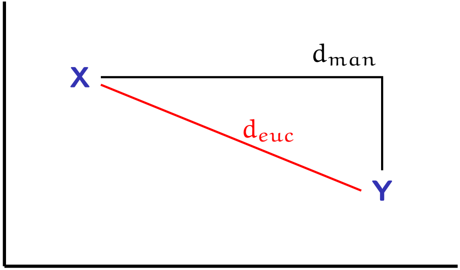
$$d_{man}(\vec{x}, \vec{y}) = L_1(\vec{x}, \vec{y}) = \sum_{i=1}^{N} |x_i - y_i|$$



Vector similarities/distances

- L_2 norm, a.k.a. Euclidean distance:

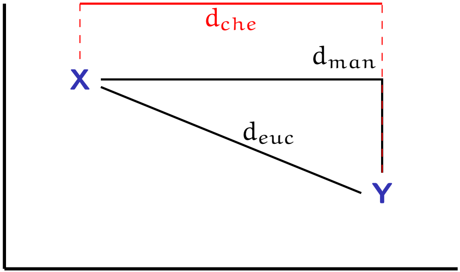
$$d_{euc}(\vec{x}, \vec{y}) = L_2(\vec{x}, \vec{y}) = |\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$



Vector similarities/distances

- The limit of Minkowsky distance is Chebychev distance:

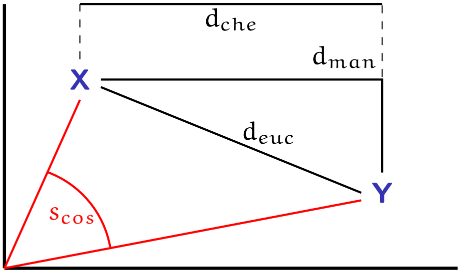
$$d_{che}(\vec{x}, \vec{y}) = L_\infty = \lim_{r \to \infty} L_r(\vec{x}, \vec{y}) = \max_i |x_i - y_i|$$



Vector similarities/distances

- Cosine is a similarity, not a distance:

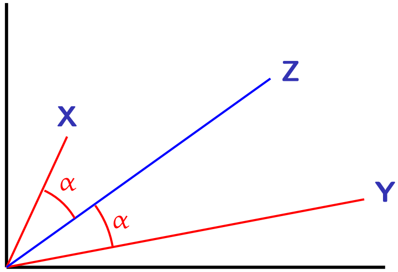
$$\mathrm{sim}_{cos}(\vec{x},\vec{y}) = \cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}|\cdot|\vec{y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\cdot\sqrt{\sum_i y_i^2}}$$



Vector similarities/distances

- Dot product (or scalar product) is also similarity, that takes into account not only the angle but also the norm of the vectors:

$$\mathrm{sim}_{\mathrm{dot}}(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} = \sum_i x_i y_i$$



$$\begin{aligned}\mathrm{sim}_{\mathrm{cos}}(X, Z) &= \mathrm{sim}_{\mathrm{cos}}(Y, Z)\\ &= \cos\alpha \approx 0.84\end{aligned}$$

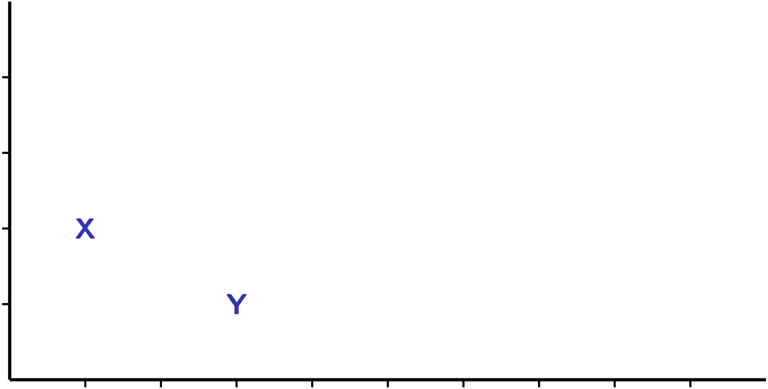
$$\mathrm{sim}_{\mathrm{dot}}(X, Z) = X \cdot Z \approx 8.2$$

$$\mathrm{sim}_{\mathrm{dot}}(Y, Z) = Y \cdot Z \approx 21.3$$

Vector similarities/distances

- Camberra distance is similar to L_1 but relative to the distance to origin:

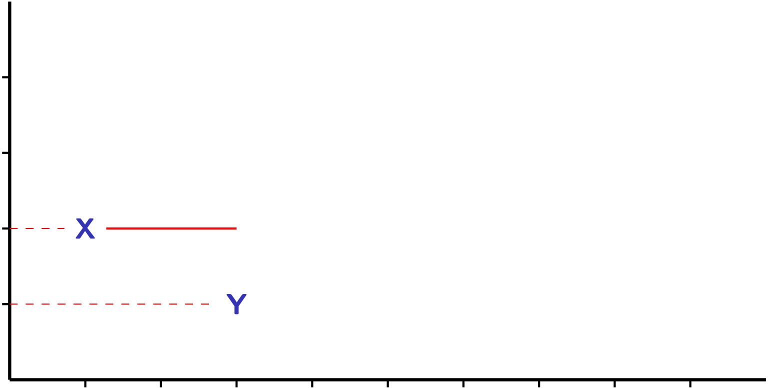
$$d_{cam}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$



Vector similarities/distances

- Camberra distance is similar to L_1 but relative to the distance to origin:

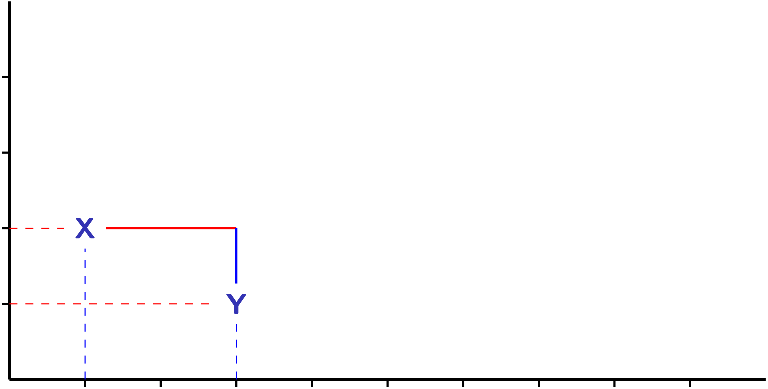
$$d_{cam}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$



Vector similarities/distances

- Camberra distance is similar to L_1 but relative to the distance to origin:

$$d_{cam}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$

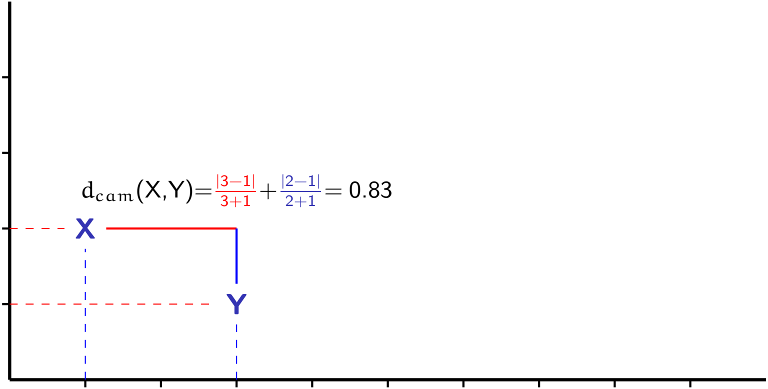


Vector similarities/distances

- Camberra distance is similar to L_1 but relative to the distance to origin:

$$d_{cam}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$

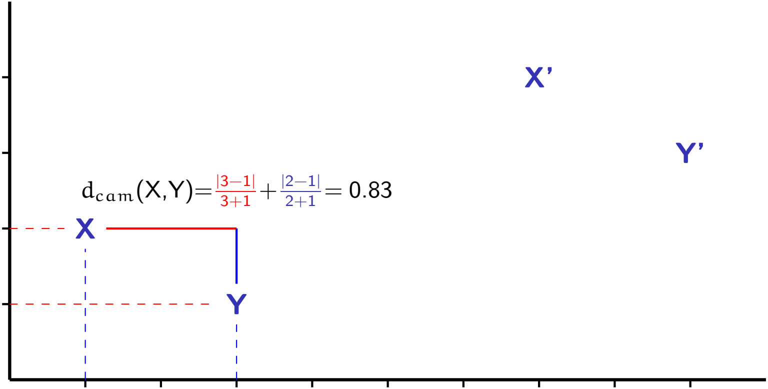
$d_{cam}(X,Y) = \frac{|3-1|}{3+1} + \frac{|2-1|}{2+1} = 0.83$



Vector similarities/distances

- Camberra distance is similar to L_1 but relative to the distance to origin:

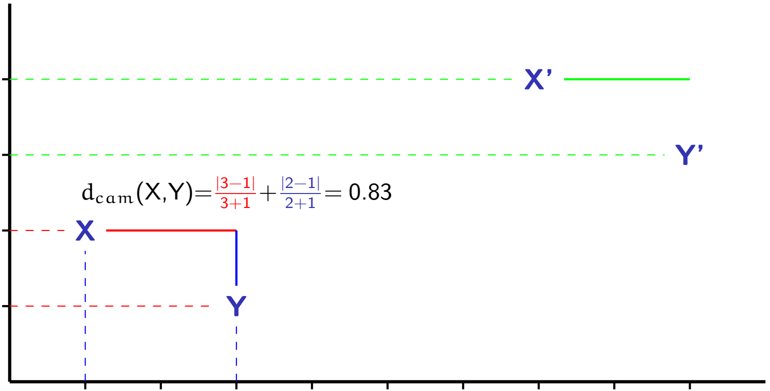
$$d_{cam}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$



Vector similarities/distances

- Camberra distance is similar to L_1 but relative to the distance to origin:

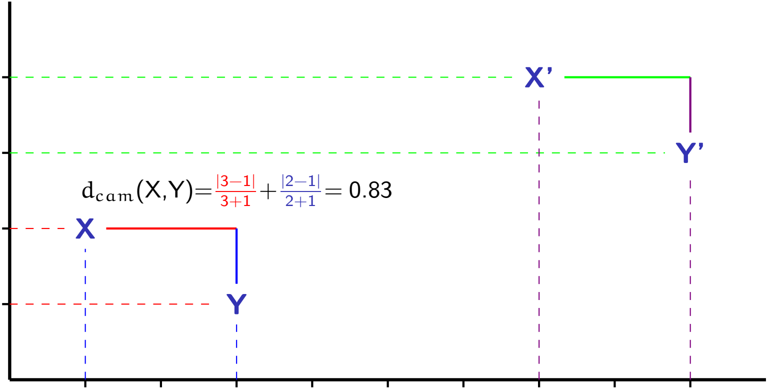
$$d_{cam}(\vec{x},\vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$



Vector similarities/distances

- Camberra distance is similar to L_1 but relative to the distance to origin:

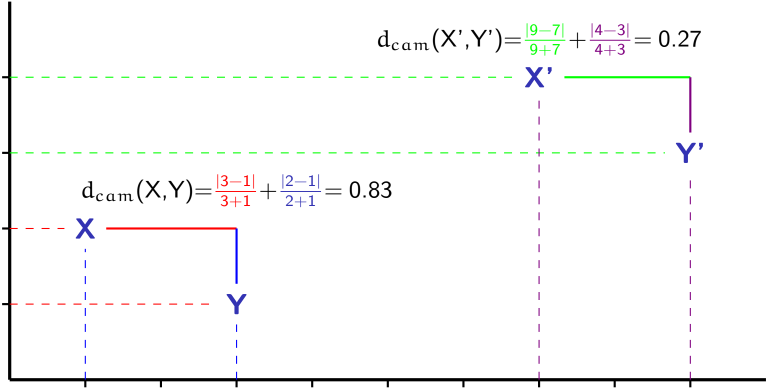
$$d_{cam}(\vec{x},\vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$



Vector similarities/distances

- Camberra distance is similar to L_1 but relative to the distance to origin:

$$d_{cam}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{|x_i + y_i|}$$



Example

s_1 = *Spokesman confirms senior government advisor was shot*
s_2 = *The spokesman said the senior advisor was shot dead*
s_3 = *Spokesman said the shot government advisor was dead*

	Spokesman	confirms	said	the	senior	government	advisor	was	shot	dead
s_1	1	1	0	0	1	1	1	1	1	0
s_2	1	0	1	2	1	0	1	1	1	1
s_3	1	0	1	1	0	1	1	1	1	1

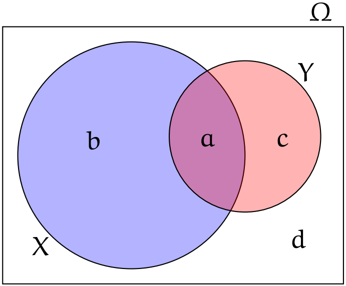
	d_{man}	d_{euc}	d_{che}	d_{cam}	sim_{dot}	sim_{cos}
$s_1 \leftrightarrow s_2$	6	$\sqrt{8} = 2.83$	2	5	5	$\frac{5}{\sqrt{7}\sqrt{11}} = 0.57$
$s_1 \leftrightarrow s_3$	5	$\sqrt{5} = 2.24$	1	5	5	$\frac{5}{\sqrt{7}\sqrt{8}} = 0.67$
$s_2 \leftrightarrow s_3$	3	$\sqrt{3} = 1.73$	1	2.33	8	$\frac{8}{\sqrt{8}\sqrt{11}} = 0.85$

Outline

Set similarities/distances

- When objects are represented as [feature] sets (or binary-valued vectors) we can use set similarity measures
- These similarities are in $[0, 1]$ and can be converted to distances simply substracting: $d(X, Y) = 1 - sim(X, Y)$
- Easily computable using a contingency table:

		Y		
		1	0	
X	1	a	b	$a + b$
	0	c	d	$c + d$
		$a + c$	$b + d$	



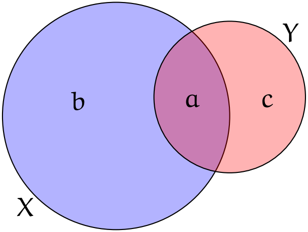
Set similarities/distances

- Dice.

$$\mathrm{sim}_{dic}(X, Y) = \frac{2 \cdot |X \cap Y|}{|X| + |Y|} = \frac{2a}{2a + b + c}$$

- Jaccard.

$$\mathrm{sim}_{jac}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{a}{a + b + c}$$



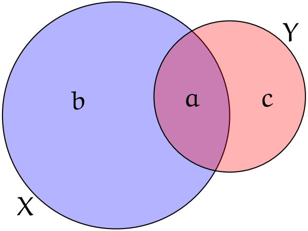
Set similarities/distances

- Overlap.

$$sim_{ovl}(X,Y) = \frac{|X \cap Y|}{\min(|X|,|Y|)} = \frac{a}{\min(a+b, a+c)}$$

- Cosine.

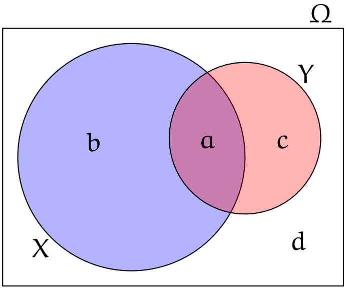
$$sim_{cos}(X,Y) = \frac{|X \cap Y|}{\sqrt{|X|} \cdot \sqrt{|Y|}} = \frac{a}{\sqrt{(a+b)}\sqrt{(a+c)}}$$



Set similarities/distances

- Matching Coefficient

$$\mathrm{sim}_{mc}(X, Y) = \frac{|X \cap Y| + |(\Omega - X) \cap (\Omega - Y)|}{|\Omega|} = \frac{a + d}{a + b + c + d}$$



Example

$s_1 =$ *Spokesman confirms senior government advisor was shot*
$s_2 =$ *The spokesman said the senior advisor was shot dead*
$s_3 =$ *Spokesman said the shot government advisor was dead*

	Spokesman	confirms	said	the	senior	government	advisor	was	shot	dead
s_1	1	1	0	0	1	1	1	1	1	0
s_2	1	0	1	**1**	1	0	1	1	1	1
s_3	1	0	1	1	0	1	1	1	1	1

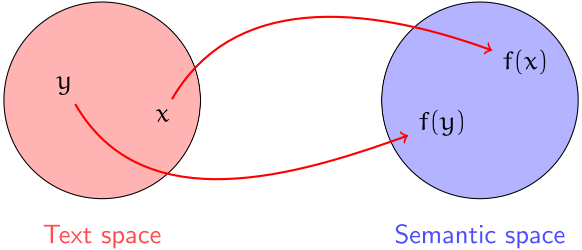
	$\mathrm{sim}_{\mathrm{dic}}$	$\mathrm{sim}_{\mathrm{jac}}$	$\mathrm{sim}_{\mathrm{ovl}}$	$\mathrm{sim}_{\mathrm{cos}}$	$\mathrm{sim}_{\mathrm{mc}}$
$s_1 \leftrightarrow s_2$	0.33	0.50	0.71	0.67	0.50
$s_1 \leftrightarrow s_3$	0.33	0.50	0.71	0.67	0.50
$s_2 \leftrightarrow s_3$	0.87	0.78	0.87	0.87	0.80

Outline

Knowledge-based Approaches

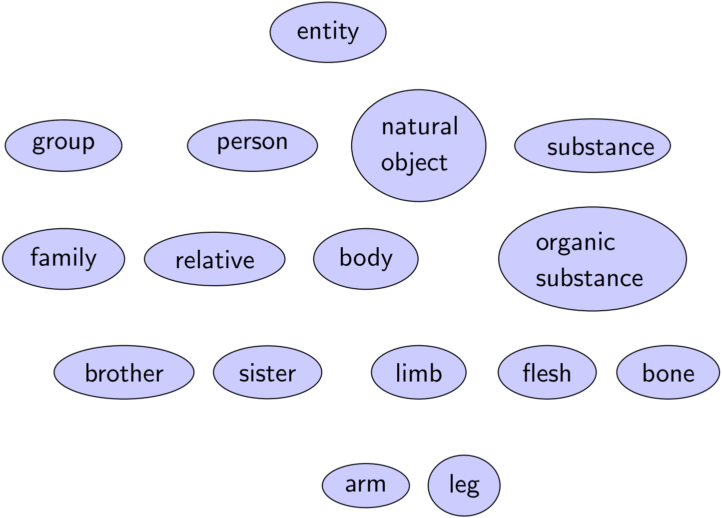
Project objects onto a knowledge-based semantic space:

$$d(x, y) = d_{sem}(f(x), f(y))$$

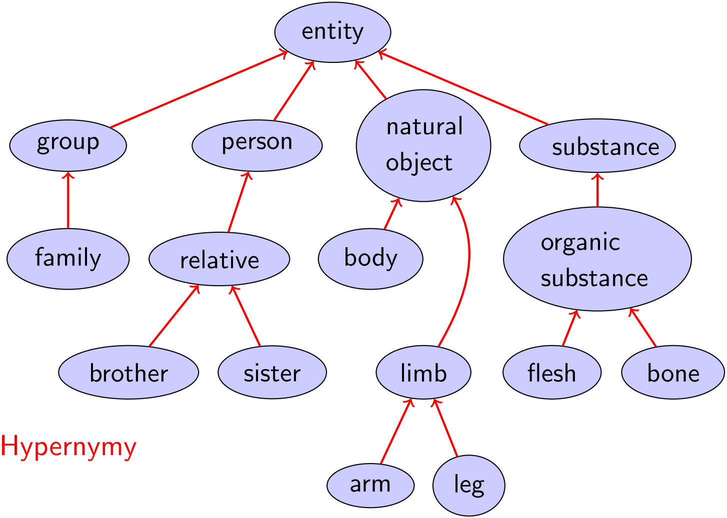


- Semantic spaces may be ontologies (e.g. WordNet, CYC, SUMO, ...) or graph-shaped knowledge bases (e.g. Wikipedia, DBPedia, ...).
- Projection function $f(x)$ is not trivial, since each word may map to more than one concept in semantic space.

WordNet

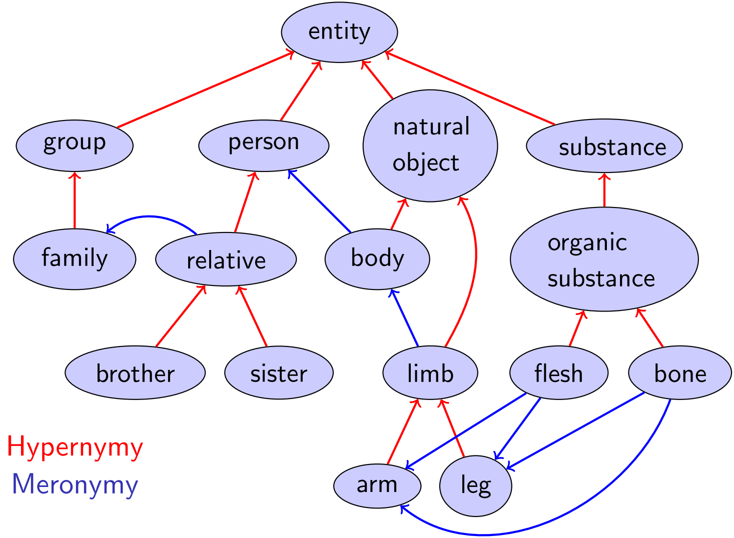

entity
group
person
natural object
substance
family
relative
body
organic substance
brother
sister
limb
flesh
bone
arm
leg

WordNet

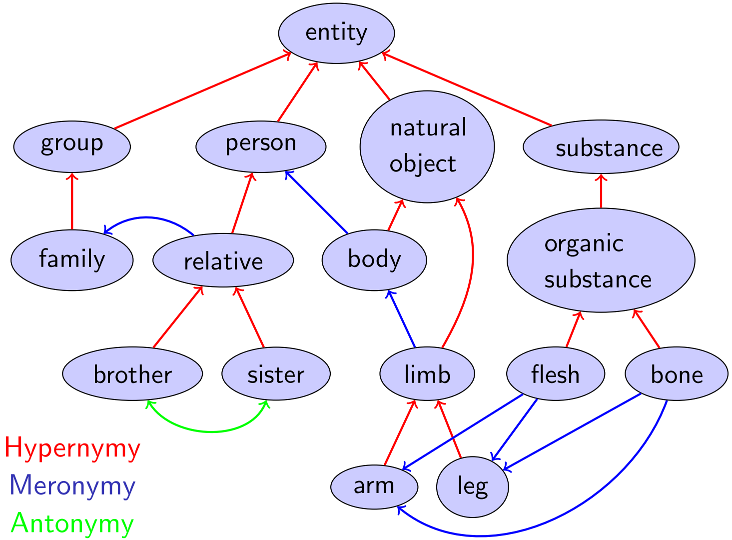


Hypernymy

WordNet


entity
group
person
natural object
substance
family
relative
body
organic substance
brother
sister
limb
flesh
bone
arm
leg
Hypernymy
Meronymy

WordNet



entity
group
person
natural object
substance
family
relative
body
organic substance
brother
sister
limb
flesh
bone
arm
leg
Hypernymy
Meronymy
Antonymy

WordNet distances

Based on graph structure:

- Shortest Path Length:

$$d(s_1, s_2) = \mathrm{SLP}(s_1, s_2)$$

- Leacock & Chodorow (similarity, $[0, \infty)$):

$$s(s_1, s_2) = -\log \frac{\mathrm{SLP}(s_1, s_2)}{2 \cdot \mathrm{MaxDepth}}$$

- Wu & Palmer (similarity, $(0, 1]$):

$$d(s_1, s_2) = \frac{2 \cdot \mathrm{depth}(\mathrm{LCS}(s_1, s_2))}{\mathrm{depth}(s_1) + \mathrm{depth}(s_2)}$$

WordNet distances

Based on Information Content

$$IC(c) = -\log P(c) = -\log \frac{freq(c)}{N}$$

$freq(c)$: number of occurrences of *any instance* of concept c.
N: total number of observed instances.

- Resnik (similarity, $[0, \infty)$)

$$s(s_1, s_2) = IC(LCS(s_1, s_2))$$

- Jiang & Conrath (distance, $[0, \infty)$)

$$d(s_1, s_2) = IC(s_1) + IC(s_2) - 2 \cdot IC(LCS(s_1, s_2))$$

- Lin (similarity, $[0, 1]$):

$$s(s_1, s_2) = \frac{2 \cdot IC(LCS(s_1, s_2))}{IC(s_1) + IC(s_2)}$$

WordNet distances

Based on sense information (not relations/structure)

- Gloss overlap: Any vector/set similary measure applied to words in sense glosses.

Distances in Wikipedia

- Graph-based distances (e.g Shortest Path Length, Page Rank, ...)
- Link-based similarities (some set similarity measure applied to the set of links of each page)
- Category-based similarities (some set similarity measure applied to the set of categories of each page)
- Text-based similarities (some text similarity measure applied to the texts of the pages)
- Heterogenous measures (combining several of the above in a weighted average)

Outline

Corpus based representations

Vectors to represent linguistic objects may be build using the distributional behaviour of the contexts they appear in.

E.g.:

- Represent words depending on the distribution of words frequently appearing nearby.
- Represent documents depending on the [general] distribution of words they contain.

Large corpus are required to pre-compute this distributions.

Corpus based representations

Vectors representing words or document contexts can be obtained in a variety of ways.

- Sparse vector representations
  - PMI
  - TF-IDF
- Dense vector representations
  - LSI
  - LDA
  - Word Embeddings

Outline

PMI - Pointwise Mutual Information

- Mutual Information of two random variables X, Y measures the amount of information about one random variable obtained observing the other.

$$MI(X, Y) = \sum_{x \in X} \sum_{y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$

- Pointwise MI measures the ratio between the expected co-occurrence of events x and y, and their actual co-ocurrence.

$$PMI(x, y) = \log \frac{P(x, y)}{P(x)P(y)}$$

Term-Term Matrix

PMI (or any other term-term relatedness feature, e.g. co-occurrence frequency) may be used to build a Term-Term Matrix.

Co-ocurrence is typically defined as *co-occurrence in a window of size* n. In this example $n = 2$ (i.e. we count only consecutive words co-occurrences).

d_1: "Two for tea and tea for two"
d_2: "Tea for me and for you"
d_3: "You and me for tea"

	two	for	tea	and	me	you	#occ.
two	0	2	0	0	0	0	2
for	-	0	4	1	2	1	5
tea	-	-	0	2	0	0	4
and	-	-	-	0	2	1	3
me	-	-	-	-	0	0	2
you	-	-	-	-	-	0	2

size-2 window co-occurrence absolute frequency term-term matrix

Term-Term Matrix

We need to compute the occurrence probability of a single word $P(x)$, and the co-occurrence probability of two words $P(x, y)$.

d_1: "Two for tea and tea for two"
d_2: "Tea for me and for you"
d_3: "You and me for tea"

Total words: 18
Total size-2 windows: 15

$P(x, y)$	two	for	tea	and	me	you	$P(x)$
two	0	2/15	0	0	0	0	2/18
for	-	0	4/15	1/15	2/15	1/15	5/18
tea	-	-	0	2/15	0	0	4/18
and	-	-	-	0	2/15	1/15	3/18
me	-	-	-	-	0	0	2/18
you	-	-	-	-	-	0	2/18

size-2 window co-occurrence probability term-term matrix

Term-Term Matrix

We can compute PMI for each pair, obtaining a PMI Term-Term Matrix

d_1: "Two for tea and tea for two"
d_2: "Tea for me and for you"
d_3: "You and me for tea"

Total words: 18
Total bigrams: 15

$PMI(x, y)$	two	for	tea	and	me	you	$P(x)$
two	$-\infty$	2.11	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0.11
for	-	$-\infty$	2.11	0.53	2.11	1.11	0.28
tea	-	-	$-\infty$	1.85	$-\infty$	$-\infty$	0.22
and	-	-	-	$-\infty$	2.85	0.56	0.17
me	-	-	-	-	$-\infty$	$-\infty$	0.11
you	-	-	-	-	-	$-\infty$	0.11

PMI term-term matrix

Term-Term Matrix

- Entries in the Term-Term Matrix can directly be used to compare two terms (higher PMI - higher relatedness)
- Rows (or columns) in the Matrix can be used as term representations, and compared with vector similarity measures (to find terms with similar co-occurence patterns).

- Negative PMI represent terms that *repel* each other (co-occur less than expected).
- Very low frequency terms may have negative PMI just because they have less chances to co-occur.
- Negative PMI values are often replaced by zero (PPMI - Positive PMI)

TF-IDF

TF-IDF (*Term Frequency* $\times$ *Inverse Document Frequency*) is a measure of relevance (or relatedness) between a term and a document, very commonly used in Information Retrieval.

$$\mathrm{TF\text{-}IDF}(t, d, \mathcal{D}) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t, \mathcal{D})$$

where:

- $\mathcal{D}$ is a collection (set) of documents, $\mathcal{D} = \{d_1, d_2, \dots, d_n\}$
- $d_i \in \mathcal{D}$ is a document, represented as a multiset (i.e. set with repetitions) of terms, $d_i = \{t_1, t_2, \dots, t_{m_i}\}$
- t is a term that may appear (or not) in documents in $\mathcal{D}$.

TF-IDF

- $\mathrm{TF}(t, d)$: Frequency of a term t in a document d, relative to the lenght of the document

$$\mathrm{TF}(t, d) = \frac{|\{x \in d : x = t\}|}{|d|}$$

- $\mathrm{IDF}(t, \mathcal{D})$: Inverse of the proportion of documents containing term t in a document collection $\mathcal{D}$.

$$\mathrm{IDF}(t, \mathcal{D}) = \log\left(\frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}|}\right)$$

TF-IDF score for a term t and a document d is rewarded when the term is frequent in the document (high TF), and is penalized when the term appears in many documents (low IDF).

Term-Document Matrix

TF-IDF (or any other term-document relatedness measure, e.g. plain frequency) can be used to build a Term-Document Matrix:

d_1: "Two for tea and tea for two"
d_2: "Tea for me and for you"
d_3: "You and me for tea"

	two	for	tea	and	me	you	$\lvert d_i \rvert$
d_1	2	2	2	1	0	0	7
d_2	0	2	1	1	1	1	6
d_3	0	1	1	1	1	1	5

Absolute frequency term-document matrix

Term-Document Matrix

TF-IDF (or any other term-document relatedness measure, e.g. plain frequency) can be used to build a Term-Document Matrix:

d_1: "Two for tea and tea for two"
d_2: "Tea for me and for you"
d_3: "You and me for tea"

	two	for	tea	and	me	you	$\lvert d_i \rvert$
d_1	2/7	2/7	2/7	1/7	0	0	7
d_2	0	2/6	1/6	1/6	1/6	1/6	6
d_3	0	1/5	1/5	1/5	1/5	1/5	5

TF term-document matrix

Term-Document Matrix

TF-IDF (or any other term-document relatedness measure, e.g. plain frequency) can be used to build a Term-Document Matrix:

d_1: "Two for tea and tea for two"
d_2: "Tea for me and for you"
d_3: "You and me for tea"

two	for	tea	and	me	you
$\log(3/1) = 1.58$	$\log(3/3) = 0$	$\log(3/3) = 0$	$\log(3/3) = 0$	$\log(3/2) = 0.58$	$\log(3/2) = 0.58$

IDF for each term in the collection

Term-Document Matrix

TF-IDF (or any other term-document relatedness measure, e.g. plain frequency) can be used to build a Term-Document Matrix:

d_1: "Two for tea and tea for two"
d_2: "Tea for me and for you"
d_3: "You and me for tea"

	two	for	tea	and	me	you
d_1	$2/7 \cdot 1.58$	$2.7 \cdot 0$	$2/7 \cdot 0$	$1/7 \cdot 0$	$0 \cdot 0.58$	$0 \cdot 0.58$
d_2	$0 \cdot 1.58$	$2/6 \cdot 0$	$1/6 \cdot 0$	$1/6 \cdot 0$	$1/6 \cdot 0.58$	$1/6 \cdot 0.58$
d_3	$0 \cdot 1.58$	$1/5 \cdot 0$	$1/5 \cdot 0$	$1/5 \cdot 0$	$1/5 \cdot 0.58$	$1/5 \cdot 0.58$

TF-IDF term-document matrix

Term-Document Matrix

TF-IDF (or any other term-document relatedness measure, e.g. plain frequency) can be used to build a Term-Document Matrix:

d_1: "Two for tea and tea for two"
d_2: "Tea for me and for you"
d_3: "You and me for tea"

	two	for	tea	and	me	you	$\|d_i\|$
d_1	0.45	0	0	0	0	0	7
d_2	0	0	0	0	0.097	0.097	6
d_3	0	0	0	0	0.117	0.117	5

TF-IDF term-document matrix

Term-Document Matrix

TF-IDF table entries contain the *relevance* (or *relatedness*, or *similarity*, ...) between terms and documents, and can be used for IR.

- A query with the term *two* will retrieve document d_1 with high relevace.
- A query with the term *me* or *you* will retrieve documents d_2 and d_3 with moderate relevance.
- Terms *for*, *and*, or *tea* would be filtered out from the index.

	two	for	tea	and	me	you	$\|d_i\|$
d_1	0.45	0	0	0	0	0	7
d_2	0	0	0	0	0.097	0.097	6
d_3	0	0	0	0	0.117	0.117	5

TF-IDF term-document matrix

Term-Document Matrix

The Term-Document matrix may also be used as a representation of terms/documents:

- Row vectors in the matrix represent documents.

  We can use vector distances/similarities to compare row vectors and find similar documents.

- Column vectors in the matrix represent terms.

  We can use vector distances/similarities to compare column vectors and find similar terms.

Term-Document Matrix

In the running example:

- Documents d_2 and d_3 are similar documents, quite different from d_1.
- Terms *me* and *you* behave similarly (wrt the documents where they appear).
- Terms *and*, *for*, and *tea* behave similarly (wrt the documents where they appear).

	two	for	tea	and	me	you	$\lvert d_i \rvert$
d_1	0.45	0	0	0	0	0	7
d_2	0	0	0	0	0.097	0.097	6
d_3	0	0	0	0	0.117	0.117	5

TF-IDF term-document matrix

Outline

Sparse *vs.* Dense Representations

Term-Term and Term-Document Matrices are typically sparse:

- A term co-occurs with only a small subset of all possible terms.
- A document contains only a small subset of all possible terms.

Dense representations are preferred:

- Lower dimensionality spaces, less features to deal with.
- Better generalization:
  E.g., better handling of synonyms (*car* and *automobile* are different dimensions in a sparse representation, but may be combined into one dimension in a dense representation.)

Dimensionality Reduction

To obtain dense representations, a dimensionality reduction must be performed.

Distributional semantics methods are appropriate:

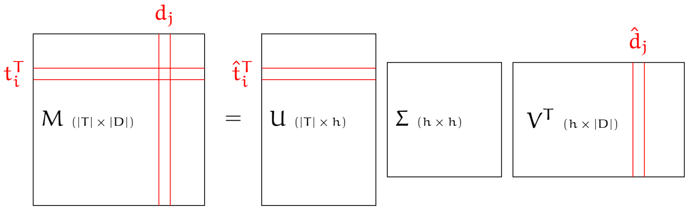
- Latent Semantic Analysis (LSA, a.k.a. Latent Semantic Indexing, LSI)
- Word Embeddings

Latent Semantic Analysis

Goal: Reduce dimensionality of Term-Document matrix M.
Method: Apply SVD (Singular Value Decomposition):

$$M = U\Sigma V^T$$

basically, apply PCA (Principal Component Analysis) to Term-Document co-ocurrence matrices.

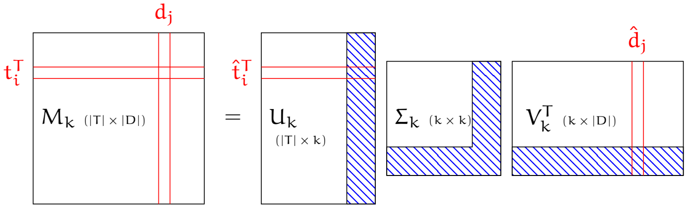


Σ is a diagonal matrix containing the singular values, and U,V are orthonormal matrices ($UU^T = U^TU = I; VV^T = V^TV = I$)

Latent Semantic Analysis (2)

Reduce M rank selecting the k largest singular values, obtaining M_k, a low-rank approximation of M:

$$M \approx M_k = U_k \Sigma_k V_k^T$$



Latent Semantic Analysis (3)

We can then compute low rank representations for document and term vectors:

- low-rank term vector: $\hat{t}_i = \Sigma_k^{-1} V_k^T t_i$ (see proof 1)
- low-rank document vector: $\hat{d}_j = \Sigma_k^{-1} U_k^T d_j$ (see proof 2)

And use them to compute similarities:

- Term-term similarity: Entry ij in $M_k M_k^T$, i.e. dot product of $\Sigma_k \hat{t}_i$ and $\Sigma_k \hat{t}_j$ (see proof 3)
- Doc-doc similarity: Entry ij in $M_k^T M_k$), i.e. dot product of $\Sigma_k \hat{d}_i$ and $\Sigma_k \hat{d}_j$ (see proof 4)
- Query-doc similarity: Convert query (seen as a mini-document vector) to low-rank space $\hat{q} = \Sigma_k^{-1} U_k^T q$ and compare with known documents.

Latent Semantic Analysis (proofs)

- Proof 1:

$$t_i^T = \hat{t}_i^T \Sigma_k V_k^T \;\rightarrow\; t_i^T V_k = \hat{t}_i^T \Sigma_k \;\rightarrow\; t_i^T V_k \Sigma_k^{-1} = \hat{t}_i^T \;\rightarrow\; \hat{t}_i = \Sigma_k^{-1} V_k^T t_i$$

- Proof 2:

$$d_j = U_k \Sigma_k \hat{d}_j \;\rightarrow\; U_k^T d_j = \Sigma_k \hat{d}_j \;\rightarrow\; \Sigma_k^{-1} U_k^T d_j = \hat{d}_j$$

- Proof 3:

$$M_k M_k^T = U_k \Sigma_k V_k^T (U_k \Sigma_k V_k^T)^T = U_k \Sigma_K V_k^T (V_k \Sigma_k^T U_k^T) =$$
$$= U_k \Sigma_K I \Sigma_k^T U_k^T = U_k \Sigma_K (U_k \Sigma_K)^T$$

Thus, element ij in the matrix is:

$$t_i^T \Sigma_k (t_j^T \Sigma_k)^T = t_i^T \Sigma_k \Sigma_k^T t_j = \Sigma_k^T t_i \Sigma_k^T t_j = \Sigma_k t_i \Sigma_k t_j$$

- Proof 4:

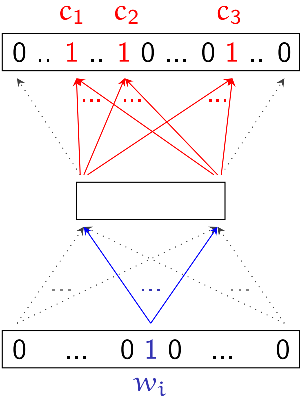
$$M_k^T M_k = (U_k \Sigma_k V_k^T)^T U_k \Sigma_k V_k^T = (V_k \Sigma_k^T U_k^T) U_k \Sigma_K V_k^T =$$
$$= V_k \Sigma_k^T I \Sigma_K V_k^T = V_k \Sigma_k \Sigma_K^T V_k^T = V_k \Sigma_K (V_k \Sigma_K)^T$$

Thus, element ij in the matrix is:

$$d_i^T \Sigma_k (d_j^T \Sigma_k)^T = d_i^T \Sigma_k \Sigma_k^T d_j = \Sigma_k^T d_i \Sigma_k^T d_j = \Sigma_k d_i \Sigma_k d_j$$

Word Embeddings

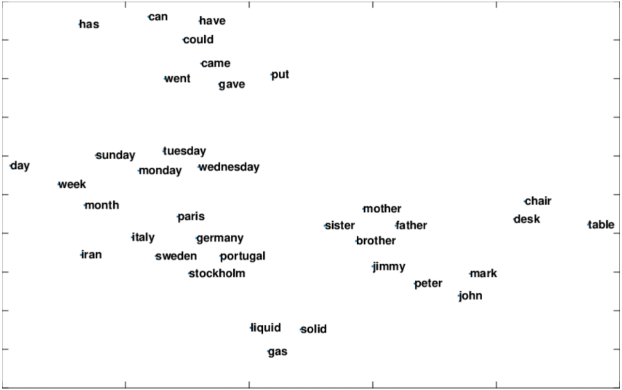
Goal: Find a low-rank representation for terms.
Method: Train a neural network to learn appropriate low-rank vectors for each term.



- Word w_i appearing near context words c_1, c_2, c_3 is used as a training example.
- The NN learns to relate words to their usual context words.
- The hidden layer input weights encode the usual contexts of each input word.
- Words usually appearing in similar context will have similar hidden layer weights.

LSA *vs* Word Embeddings

Distributional semantics methods produce close vectors for words in similar contexts.



Source: Ali Basirat 2018, *Principal Word Vectors*, PhD Thesis, Uppsala Univ.

LSA *vs* Word Embeddings

- LSA
  - Allows comparing not only words, but also documents
  - Requires managing documents
  - Traditionally used in IR
- WE
  - Allows comparing only words, but not documents (may be tricked to, though)
  - No need to manage/represent documents
  - Learned vectors show analogy properties (man $\rightarrow$ king, woman $\rightarrow$ X?)
  - Natural approach when using NN