Distances and distributional models

Marta R. Costa-jussà
Universitat Politècnica de Catalunya

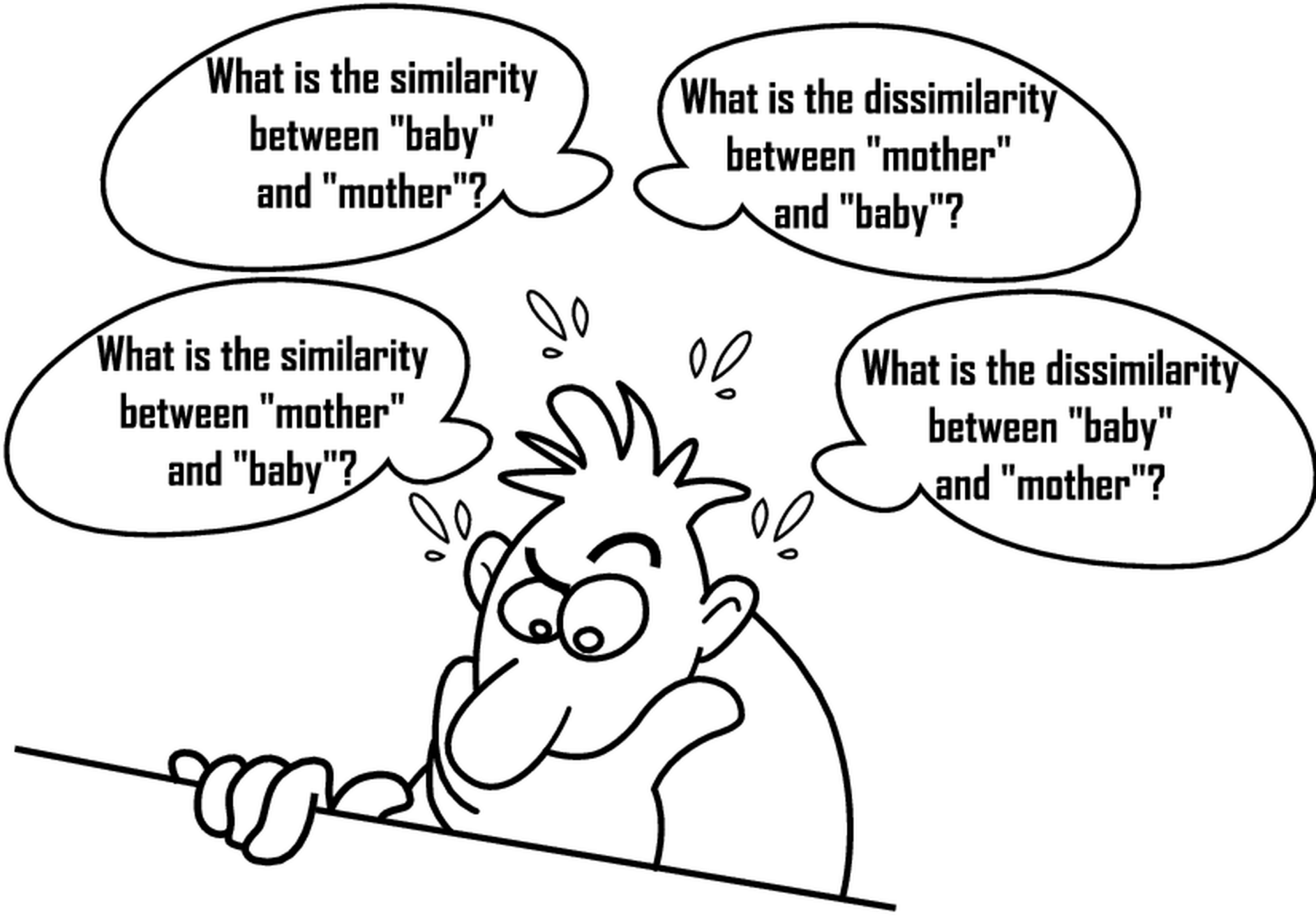
with slides from Christopher Manning, Stanford University, adapted from CS224n lecture 10th Jan 2019
and some slides from Dan Jurafsky, Stanford University, adapted from Speech and Language Processing course

Outline

- What to read
- String-based distances: character vs term-based
- Corpus-based/distributional models: vector-space model, ppmi, lsa

What to read

- Gomaa, Fahmy, 2013 (general distances)
- Manning,Schütze, 1999 (general NLP)
- Budanitsky, 1999 (semantic relatedness)
- NLTK WordNet
- Nice introduction in Turney & Pantel, 2010. From frequency to meaning: Vector Space Models of Semantics
  - Topic Modeling
  - Probabilistic Topic Models, Blei, 2012
  - Camacho-Collados, 2018. From Word to Sense Embeddings: A Survey on Vector Representations of Meaning


What is the similarity between "baby" and "mother"?
What is the dissimilarity between "mother" and "baby"?
What is the similarity between "mother" and "baby"?
What is the dissimilarity between "baby" and "mother"?

Why word similarity : NLP tasks and applications

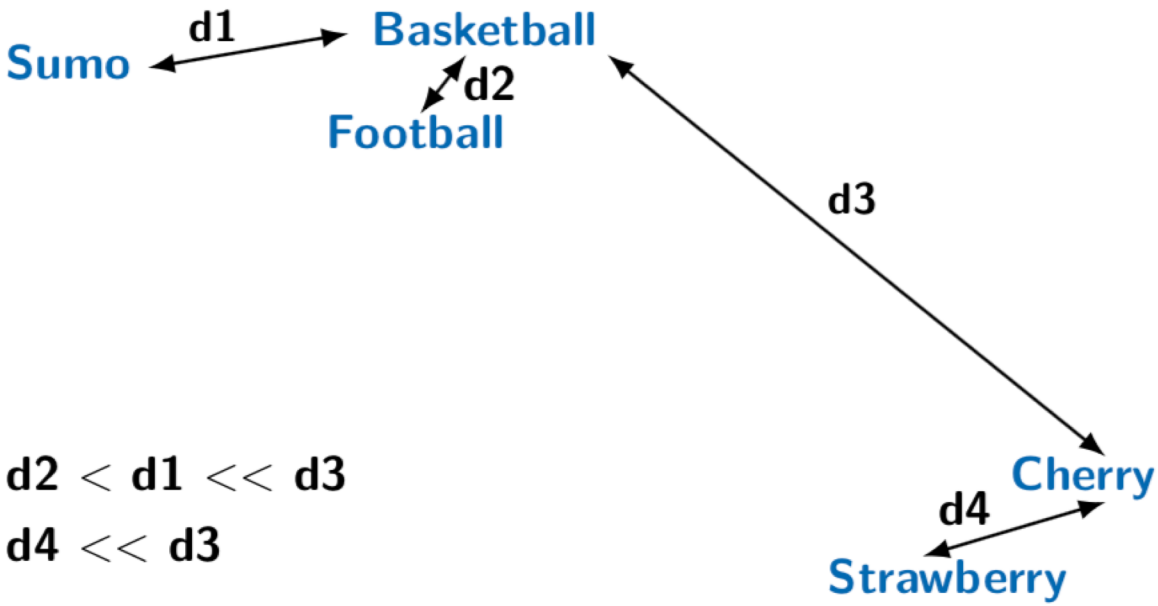
- *Information retrieval*
- *Question answering*
- *Machine translation*
- *Natural language generation*
- *Language modeling*
- *Automatic essay grading*
- *Plagiarism detection*
- *Document clustering*

Similarity vs Distance

- From distances to similarities

$$sim_{dist}(A,B) = \frac{1}{1 + dist(A,B)}$$

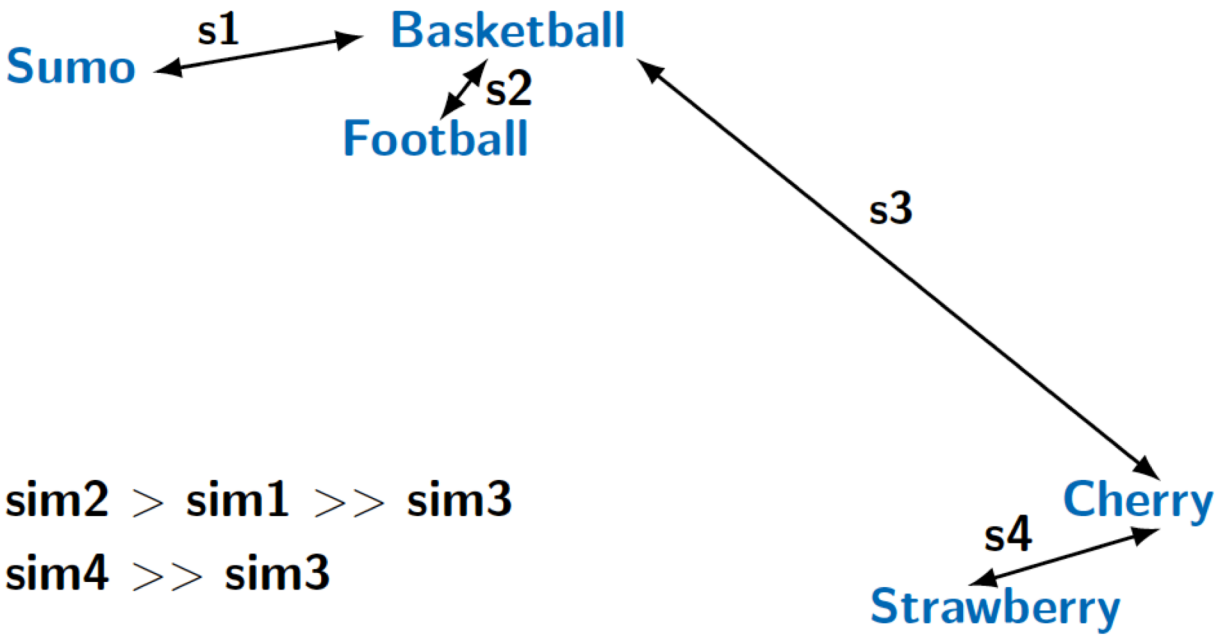
Similarity vs Distance



$d2 < d1 << d3$

$d4 << d3$

Similarity vs Distance



sim2 > sim1 >> sim3

sim4 >> sim3

Distances in metric spaces

- $d(x,y) \geq 0$
- $d(x,x) = 0$
- $d(x,y) > 0$ when $x \neq y$
- $d(x,y) = d(y,x)$ (symmetry)
- $d(x,z) \leq d(x,y) + d(y,z)$ (triangular inequality)

Information for computing similarities

- Content: string/text
- External resources:
  - Dictionaries
  - Knowledge bases:
    - WordNet
    - Wikipedia
- Context

Outline

- What to read
- String-based distances: character vs term-based
- Corpus-based/distributional models: vector-space model, pmi, lsa, psli, lda

String-based approaches

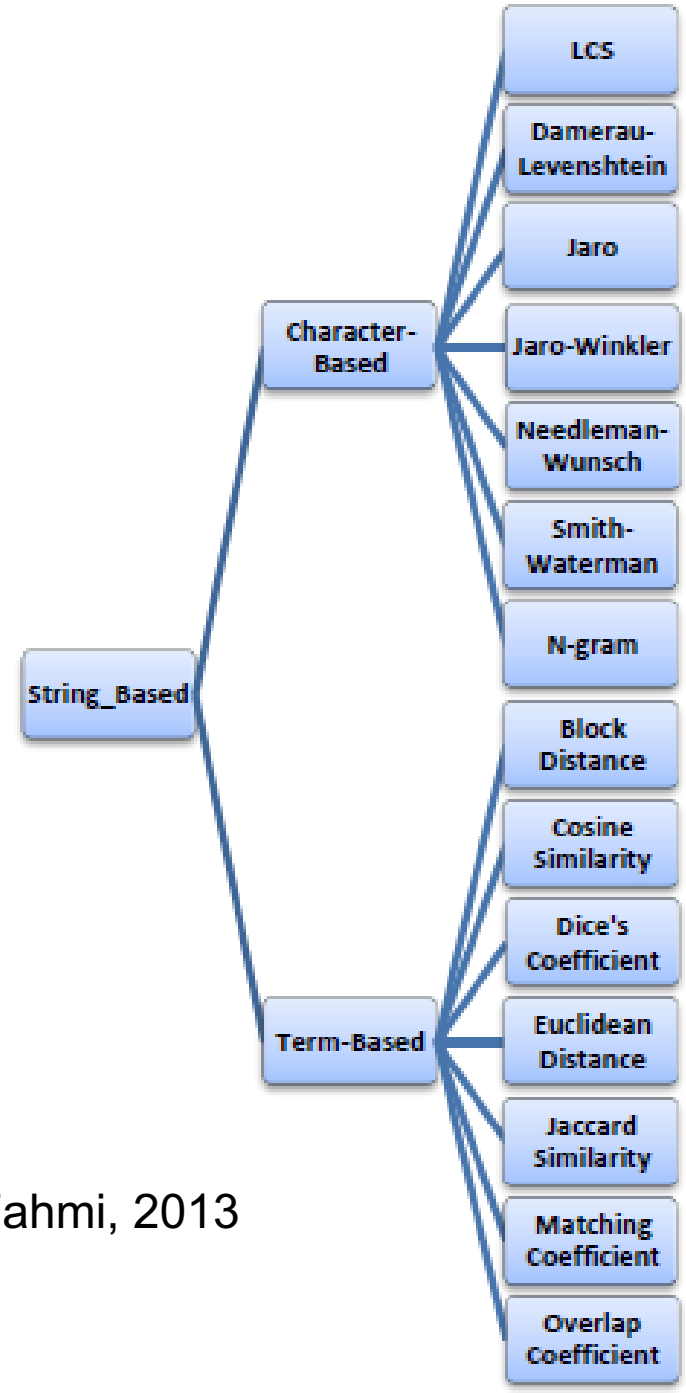


Figure and text from Gomaa, Fahmi, 2013

Character-based approaches

- LCS :
  - length of contiguous chain of characters that exist in both strings.
  - the longest common substring of the strings "ABABC", "BABCA" and "ABCBA" is string "ABC" of length 3. Other common substrings are "A", "AB", "B", "BA", "BC" and "C".
- Damerau-Levenshtein (edit distance):
  - counting the minimum number of operations needed to transform one string into the other, where an operation is defined as an insertion, deletion, or substitution of a single character, or a transposition of two adjacent characters.
  - the edit distance between “Hello” and “Jello” is 1. The edit distance between “good” and “goodbye” is 3. The edit distance between any string and itself is 0.

Details on Edit Distance

- *The minimum edit distance between two strings*
- *Is the minimum number of editing operations*
  - Insertion
  - Deletion
  - Substitution
- *Needed to transform one into the other*

Minimum Edit Distance

- *Two strings and their **alignment**:*

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

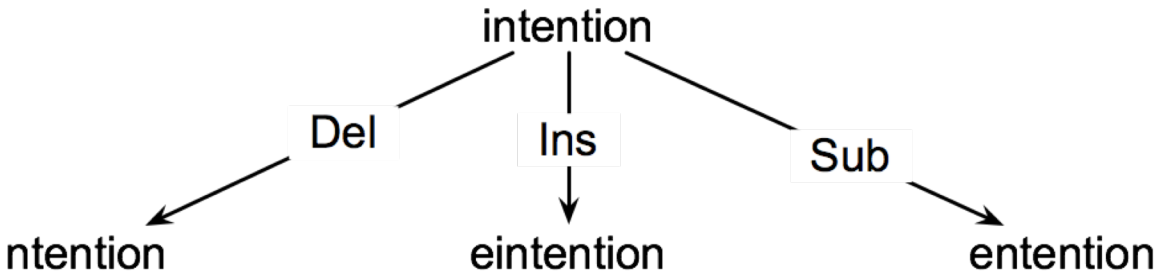
Minimum Edit Distance

- *If each operation has cost of 1*
  - Distance between these is 5
- *If substitutions cost 2 (Levenshtein)*
  - Distance between them is 8

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

How to find the Min Edit Distance?

- *Searching for a path (sequence of edits) from the start string to the final string:*
  - **Initial state**: the word we're transforming
  - **Operators**: insert, delete, substitute
  - **Goal state**: the word we're trying to get to
  - **Path cost**: what we want to minimize: the number of edits



Exercise: compute the edit distance

- *Compute the edit distance on the following examples:*
  - Intention / intentions
  - intention / execution
  - inclusive / plausive

Character-based approaches: Jaro

Transpositions= The number of matching (but different sequence order) characters divided by 2

The Jaro distance is a measure of similarity between two strings. The higher the Jaro distance for two strings is, the more similar the strings are. The score is normalized such that ***0*** *equates to no similarity and* ***1*** *is an exact match.*

Definition

The Jaro distance d_j of two given strings s_1 and s_2 is

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

Where:

- m is the number of *matching characters*;
- t is half the number of *transpositions*.

Two characters from s_1 and s_2 respectively, are considered *matching* only if they are the same and not farther than $\left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1$.

Each character of s_1 is compared with all its matching characters in s_2.

The number of matching (but different sequence order) characters divided by 2 defines the number of *transpositions*.

Example: DWAYNE vs DUANE

Character-based approaches: N-gram

– sub-sequence of n items from a given sequence of text. N-gram similarity algorithms compare the n-grams from each character or word in two strings. Distance is computed by dividing the number of similar n-grams by maximal number of n-grams.

$$similarity(s1, s2) = \frac{2 \times |pairs(s1) \cap pairs(s2)|}{|pairs(s1)| + |pairs(s2)|}$$

$$similarity(FRANCE, FRENCH)$$

Let’s play

- *Compute edit distance*
- *Compute other distances*

Term-based: Assuming…

- *Strings have the information of terms in a vector*

- *S1= The house is big*
- *S2= The cat is brown*

Term	S1	S2
The	1	1
house	1	0
is	1	1
big	1	0
cat	0	1
brown	0	1

Term-based approaches: vectors

- Cosine:

$$cos(\vec{x}, \vec{y}) = \frac{\sum_i x_i . y_i}{\sqrt{\sum_i x_i{}^2} . \sqrt{\sum_i y_i{}^2}}$$

- Scalar product :

$$\vec{x} . \vec{y} = \sum_i x_i . y_i$$

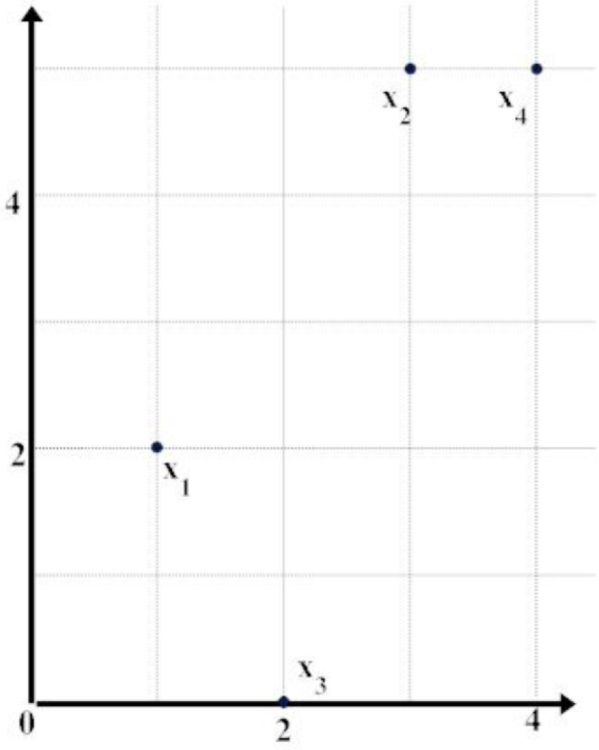
- L_1 norm (also named Manhattan, taxi-cab o city-block):

$$L_1(\vec{x}, \vec{y}) = \sum_i |x_i - y_i|$$

- L_2 norm (Euclidean distance):

$$L_2(\vec{x}, \vec{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

Examples



Data Matrix

point	attribute1	attribute2
x1	1	2
x2	3	5
x3	2	0
x4	4	5

Distance Matrix (Manhattan)

	x1	*x2*	*x3*	*x4*
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Distance Matrix (Euclidean)

	x1	*x2*	*x3*	*x4*
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Term-based approaches: Minkowski

Minkowski distances

Minkowski distances are a family of distances induced by ℓ_p norms:

$$d_p(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_p = \left(\sum_{i=1}^{d} |x_i - x_i'|^p \right)^{1/p},$$

for $p \geq 1$. We can recover three widely used distances:

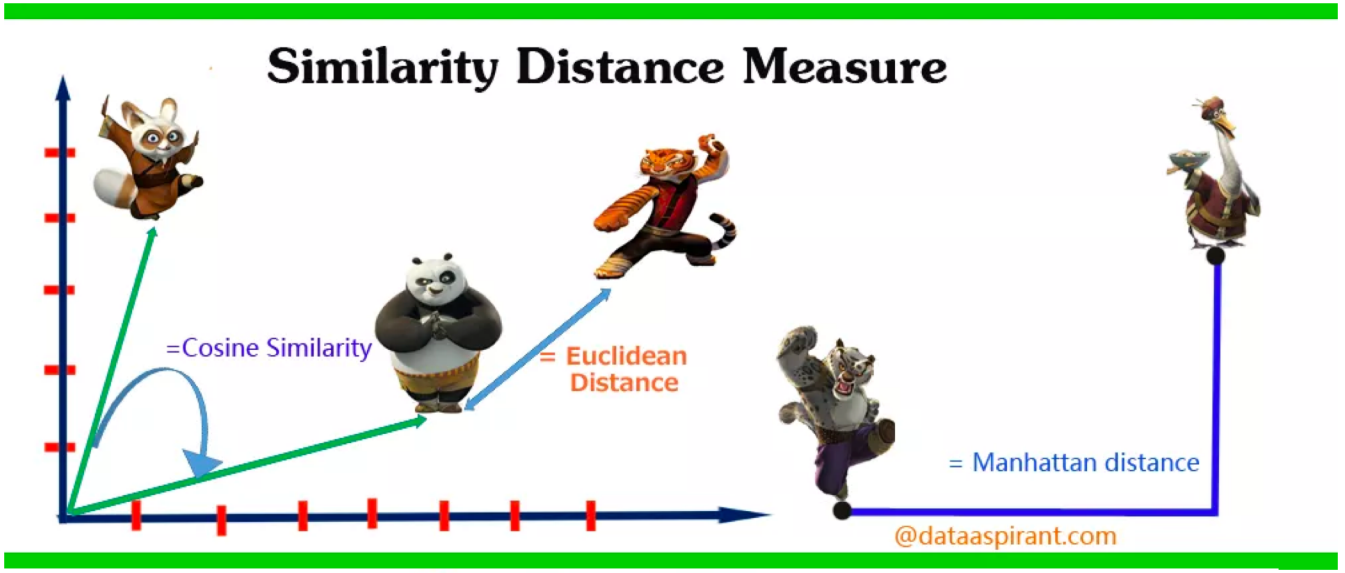
- For $p = 1$, the Manhattan distance $d_{man}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} |x_i - x_i'|$.
- For $p = 2$, the "ordinary" Euclidean distance:

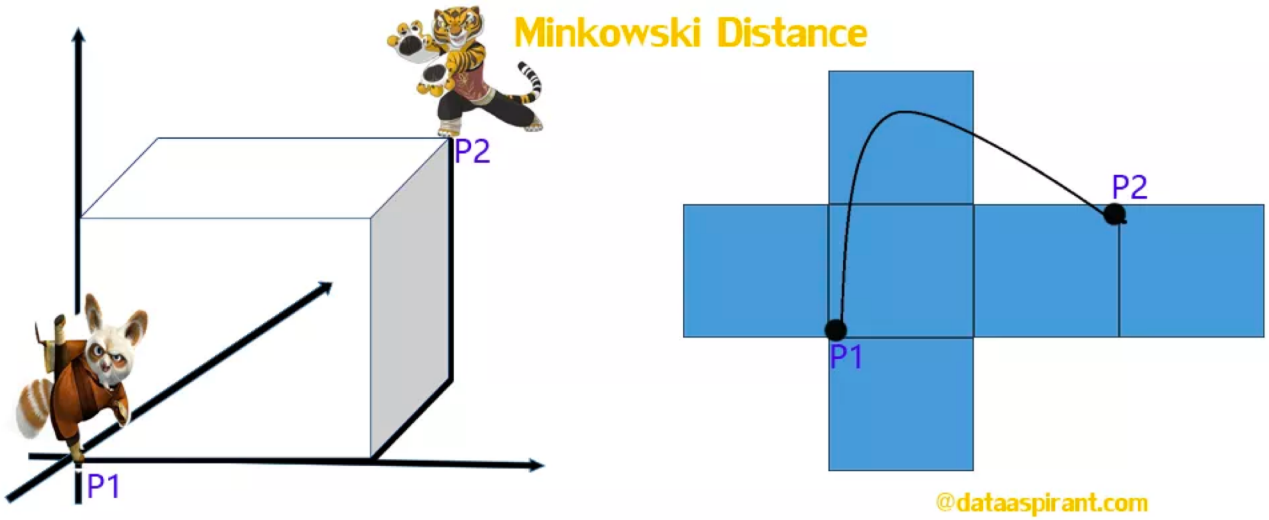
$$d_{euc}(\mathbf{x}, \mathbf{x}') = \left(\sum_{i=1}^{d} |x_i - x_i'|^2 \right)^{1/2} = \sqrt{(\mathbf{x} - \mathbf{x}')^T (\mathbf{x} - \mathbf{x}')}$$

.

- For $p \to \infty$, the Chebyshev distance $d_{che}(\mathbf{x}, \mathbf{x}') = \max_i |x_i - x_i'|$.

Visual comparison





Euclidean vs Cosine

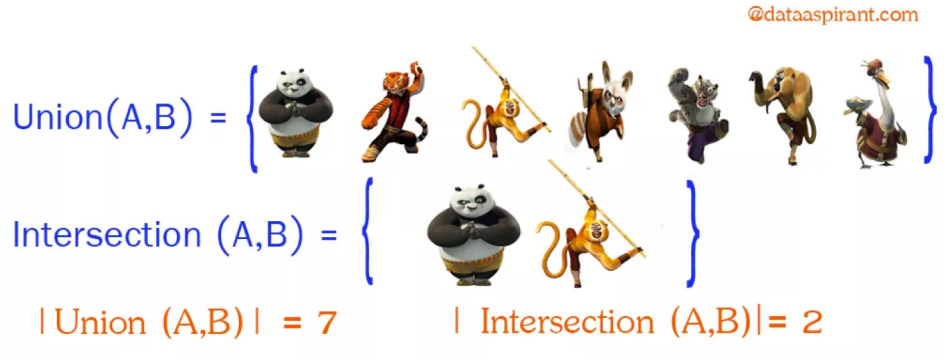
- *While cosine looks at the **angle** between vectors (thus not taking into regard their weight or magnitude), euclidean distance is similar to using a ruler to actually measure the distance.*

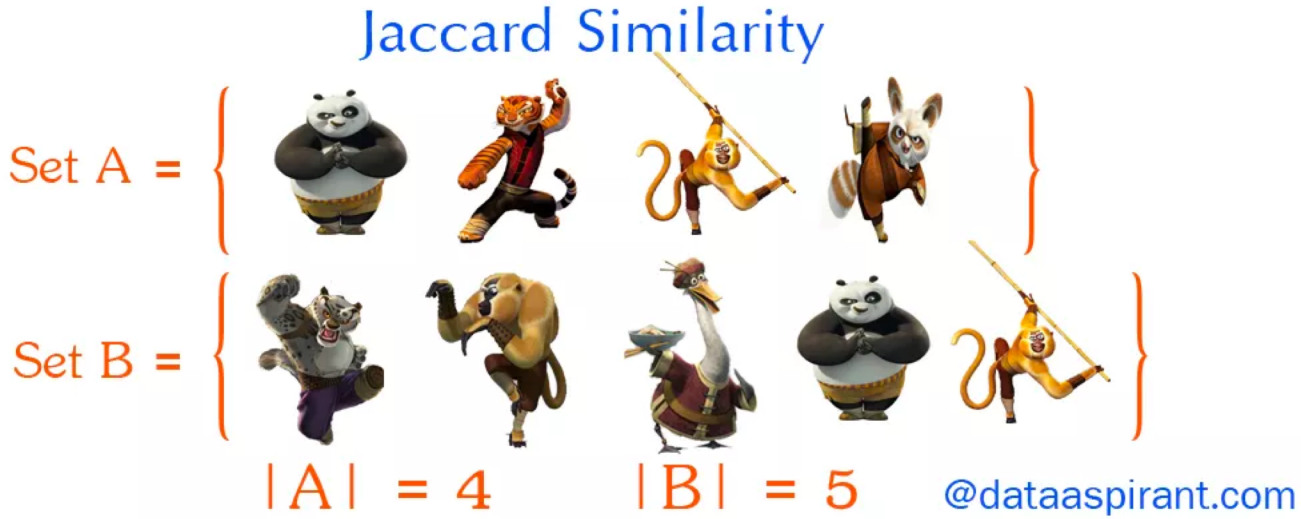
- ***Cosine** similarity is generally used as **a metric for measuring distance when the magnitude of the vectors does not matter**. This happens for example when working with text data represented by **word counts**. We could assume that when a word (e.g. science) occurs more frequent in document 1 than it does in document 2, that document 1 is more related to the topic of science.*

Term-based approaches: sets

$$matching\_coefficient = |X \cap Y|$$

$$Jaccard = \frac{|X \cap Y|}{|X \cup Y|}$$





= | Intersection (A,B) | / | Union (A,B) |

= 2 / 7

= 0.286

@dataaspirant.com

Example Jaccard vs Cosine

- *text1 = 'How can I be a geologist?'*
- *text2 = 'What should I do to be a geologist?'*

Therefore, for $\mathbf{S_1}$

HOW	1
CAN	1
I	1
BE	1
A	1
GEOLOGIST	1
WHAT	0
SHOULD	0
DO	0
TO	0

$\mathbf{S_1}$: [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

For, $\mathbf{S_2}$

HOW	0
CAN	0
I	1
BE	1
A	1
GEOLOGIST	1
WHAT	1
SHOULD	1
DO	1
TO	1

$\mathbf{S_2}$: [0, 0, 1, 1, 1, 1, 1, 1, 1, 1]

Cosine

$\mathbf{S_1} \cdot \mathbf{S_2} = |S_1||S_2|\cos\theta$

So, $\cos\theta$ (which is our cosine similarity) = $(\mathbf{S_1} \cdot \mathbf{S_2}) / (|S_1||S_2|)$

$\mathbf{S_1} \cdot \mathbf{S_2} = 4$

$|S_1| = \sqrt{(1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2 + 0^2 + 0^2)} \cong 2.44949$

$|S_2| = \sqrt{(0^2 + 0^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2)} \cong 2.82843$

$\cos\theta = 4/(2.44949 * 2.82843) \cong 0.5773496$

So, *our cosine similarity between these two sentence vectors is* ***0.5773496.***

Jaccard

- *A = "How can I be a geologist?"*
- *B = "What should I do to be a geologist?"*
- *A ∩ B = ['a', 'be', 'geologist?', 'i']*
- *A U B = ['a', 'be', 'geologist?', 'to', 'do', 'i', 'what', 'should', 'how', 'can']*
- *|A ∩ B| = 4 (Cardinality of the elements in the intersection of A and B)*
- *|A ∪ B| = 10 (Cardinality of the elements in the union of A and B)*
- *Jaccard Similarity = |A ∩ B| : |A U B|*
- *= 4/10*
- *= 0.40*

Outline

- What to read
- String-based distances: character vs term-based
- Corpus-based/distributional models: vector-space model, pmi, lsa, psli, lda

Corpus-based methods

Corpus-based semantic similarity measure that determines the similarity between words/documents according to information gained from large corpora

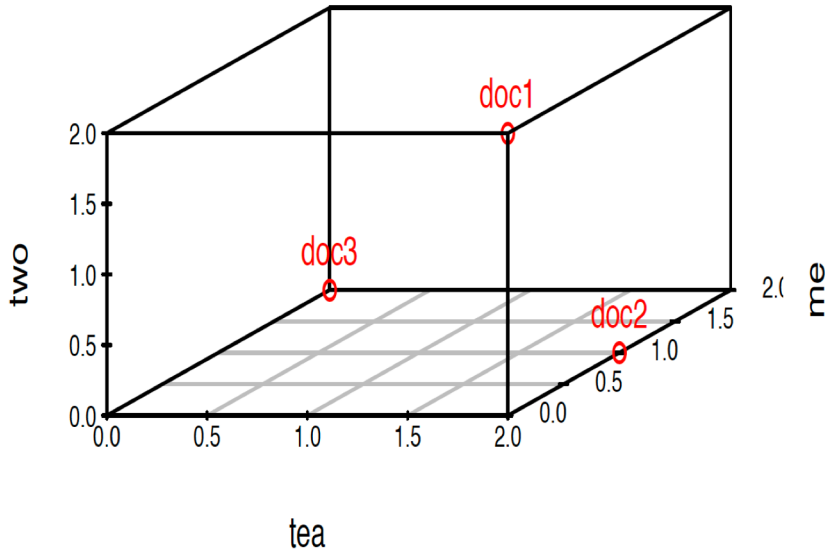
Corpus-based methods: similarity of documents

- Vector-space model
  - TF-IDF : similarity of terms or documents

Vector Space Model: Term-document matrix

doc1	Two for tea and tea for two
doc2	Tea for me and tea for you
doc3	You for me and me for you

	two	tea	me	you
doc1	2	2	0	0
doc2	0	2	1	1
doc3	0	0	2	2



Term Frequency

Term Frequency

How frequently a term occurs in a document d normalised to account for d length

$$\mathrm{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in a document } d}{\text{Total number of terms in } d}$$

Inverse Document Frequency

Measures how important a term is (low weight for stop words)

$$\text{IDF}(t, D) = \log_e \left(\frac{\text{Total number of documents } D}{\text{Number of documents with term } t \text{ in it}} \right)$$

TF-IDF

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

Question

- *Is TF-IDF domain independent?*
- *Suggestions to work with different domains? Advantages and disadvantages*

Exercise/Example: Toy Corpus

d1: We like to play some sport in the afternoon, I like basketball but John likes sumo more.
d2: Sumo is a sport where a rikishi attempts to force another wrestler out of a circular ring.
d3: Messi scored 4 goals yesterday and kept the ball as a memory of this fantastic sports afternoon!
d4: I ate too many cherries yesterday.

$$TF(ball) = ?\,; IDF(ball) = ?$$
$$TF - IDF(ball)_3 = ?$$

Example: Toy Corpus

d1: We like to play some sport in the afternoon, I like basketball but John likes sumo more.
d2: Sumo is a sport where a rikishi attempts to force another wrestler out of a circular ring.
d3: Messi scored 4 goals yesterday and kept the ball as a memory of this fantastic sports afternoon!
d4: I ate too many cherries yesterday.

$$TF(ball) = \left(0,0,\frac{1}{17},0\right); IDF(ball) = log_e\left(\frac{4}{1}\right)$$

$$TF - IDF(ball)_3 = \frac{1}{17} x log_e(4) = 0.08$$

Example: Toy Corpus

d1: We like to play some sport in the afternoon, I like basketball but John likes sumo more.
d2: Sumo is a sport where a rikishi attempts to force another wrestler out of a circular ring.
d3: Messi scored 4 goals yesterday and kept the ball as a memory of this fantastic sports afternoon!
d4: I ate too many cherries yesterday.

$$TF(a) = \left(0, \frac{3}{17}, \frac{1}{17}, 0\right); IDF(a) = log_e\left(\frac{4}{2}\right)$$

$$TF - IDF(a)_2 = \frac{3}{17} x log_e(2) = 0.12; TF - IDF(a)_3 = 0.04$$

TF-IDF Exercise

D1 = *"If it walks like a duck and quacks like a duck, it must be a duck."*
D2 = *"Beijing Duck is mostly prized for the thin, crispy duck skin with authentic versions of the dish serving mostly the skin."*
D3 = *"Bugs' ascension to stardom also prompted the Warner animators to recast Daffy Duck as the rabbit's rival, intensely jealous and determined to steal back the spotlight while Bugs remained indifferent to the duck's jealousy, or used it to his advantage. This turned out to be the recipe for the success of the duo."*
D4 = *"6:25 PM 1/7/2007 blog entry: I found this great recipe for Rabbit Braised in Wine on cookingforengineers.com."*
D5 = *"Last week Li has shown you how to make the Sechuan duck. Today we'll be making Chinese dumplings (Jiaozi), a popular dish that I had a chance to try last summer in Beijing. There are many recipies for Jiaozi."*

Task 1. *For the query Q = "Beijing duck recipe", find the two top ranked documents according to the TF/IDF rank. Assume the cosine similarity measure and the culinary term set T = {beijing, dish,duck, rabbit, recipe, roast}. Are the top ranked documents relevant to the query?*

Task 2. *Assume that the author of the document D5 goes on to tell more about her summer trip to China before doing the cooking and uses the word Beijing 3 times, instead of just once. What happens to the rank of D5? How can this be interpreted in the vector retrieval model (vectors and angles between them)? Is this change in the ranking of D5 a desirable property of TF/IDF? Why?*

Corpus-based methods: similarity of words

- PMI
  - Pointwise Mutual Information is a method for computing the similarity between pairs of words. The more often two words co-occur near each other on a document, the higher is their PMI similarity score.

PMI using the word-context matrix

- *Instead of entire documents, use smaller contexts*
  - Paragraph
  - Window of ± 4 words
- *A word is now defined by a vector over counts of context words*
- *Instead of each vector being of length D*
- *Each vector is now of length |V|*
- *The word--word matrix is |V|x|V|*

Example

- *Sample context 7 words*

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** necessary for the study authorized in the

	aardvark	computer	pinch	result	sugar	…
apricot	0.	0	1	0	1	
pineapple …	0	0	0	0	0	

Word-Word matrix

- *We showed only small example, but the real matrix is 50,000 x 50,000*

  So it's very sparse. Most values are 0.

  That's OK, since there are lots of efficient algorithms for sparse matrices.

- *The size of windows depends on your goals*

- *The shorter the windows , the more syntactic the representation:* ± *1--3 very syntacticy*

- *The longer the windows, the more semantic the representation:* ± *4--10 more semanticy*

2 kinds of co-ocurrence between 2 words

- *First--order co--occurrence (syntagmatic association):*

They are typically nearby each other.

wrote is a first--order associate of book or poem.

- *Second--order co--occurrence (paradigmatic association):*

They have similar neighbors.

wrote is a second-- order associate of words like said or remarked.

Positive Pointwise Mutual Information (PPMI)

- *Raw word frequency is not a great measure of association between words*
  - *It's very skewed*
  - *"the" and "of" are very frequent, but maybe not the most discriminative*

- *We'd rather have a measure that asks whether a context word is particularly informative about the target word.*

Positive Pointwise Mutual Information (PPMI)

Pointwise Mutual Information

- *Pointwise mutual information:*
  - Do **events** x and y **co-occur more than if they were independent**?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- *PMI between two words: (Church & Hanks 1989)*
  - Do **words** 1 and 2 co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora
    - Imagine w1 and w2 whose probability is each 10^{-6}
    - Hard to be sure p(w1,w2) is significantly different than 10^{-12}
  - Plus it's not clear people are good at "unrelatedness"
- So we just replace negative PMI values by 0
- Positive PMI (PPMI) between word1 and word2:

$$\text{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

example

Computing PPMI on a term-context matrix

Matrix F with W rows (words) and C columns (contexts) ij is # of times wi occurs in context cj

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

...

	Count(w,context)				
	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

...

Count(w,context)

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{N}$$

p(w)

$$p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{N}$$

p(w,context)

p(w=information) = 11/19 = .58

p(c=data) = 7/19 = .37

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

Count(w,context)

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

p(w=information,c=data) = 6/19 = .32

p(w=information) = 11/19 = .58

p(c=data) = 7/19 = .37

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{N} \qquad p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{N}$$

p(w,context)

	computer	data	pinch	result	sugar	**p(w)**
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}}$$

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

pmi(information,data) = $\log_2$ (.32 / (.37*.58)) = .58

PPMI Exercise

Let's consider two examples:

- *x is "eat is the first word in a bigram" and y is "pizza is the second word in a bigram".*
- *X is "happy occurs in a Tweet" and y is "pizza occurs in a Tweet".*

(A) For each example, what does P(x,y) represent?

(B) What do negative, zero, and positive PMI values represent in terms of the statistical independence of x and y? (Hint:consider what must be true of the relationship betweenP(x,y)andP(x)P(y)for the PMI to be negative, zero, or positive.) Give some example pairs of words that you would expect to have negative or positive PMI (in either the bigram or Tweets cenario).

(C) Normally in NLP we do not know the true probabilities of x and y so we must estimatethem from data. Assume we use MLE to estimate probabilities. Write down an equation to compute PMI in terms of counts rather tan probabilities. Use N to represent the total number of observations (e.g.,the total number of bigrams in the first example, or the total number of Tweets in the second example).

(D) Now, using your MLE version of PMI and the following toy dataset, compute PMI(x,y), PMI(y,z), and PMI(x,z).

N=13

C(x)=6	*C(x,y)=2*
C(y)=4	*C(x,z)=1*
C(z)=3	*C(y,z)=2*

Corpus-based methods: similarity of words/documents with dense representations

- LSA
  - Latent Semantic Analysis (dimensionality reduction through SVD)
- pLSA
  - Probabilistic Latent Semantic Analysis (topic-based dimensionality reduction)
- LDA
  - Latent Dirichlet Annotation (topic-based dimensionality reduction)

Sparse versus dense vectors

PPMI vectors are

- *long (length $|V| = 20{,}000$ to $50{,}000$)*
- *sparse (most elements are zero)*

Alternative: learn vectors which are

- *short (length 200-1000)*
- *dense (most elements are non-zero)*

Sparse versus dense vectors

Why dense vectors?

Short vectors may be easier to use as features in machine learning (less weights to tune)

Dense vectors may generalize better than storing explicit counts

They may do better at capturing synonymy:

car and automobile are synonyms; but are represented as distinct dimensions; this fails to capture similarity between a word with car as a neighbor and a word with automobile as a neighbor

Dimensionality reduction

- *Reducing the size of semantic spaces*
  - Dimensionality reduction
    - Latent Semantic Indexing, LSA (LSA, PCA, …) Deerwester et al, 1988
    - pLSA Hoffman, 1999
  - Latent Dirichlet Allocation, LDA Blei et al, 2003

Intuition

Approximate an N-dimensional dataset using fewer dimensions

By first rotating the axes into a new space

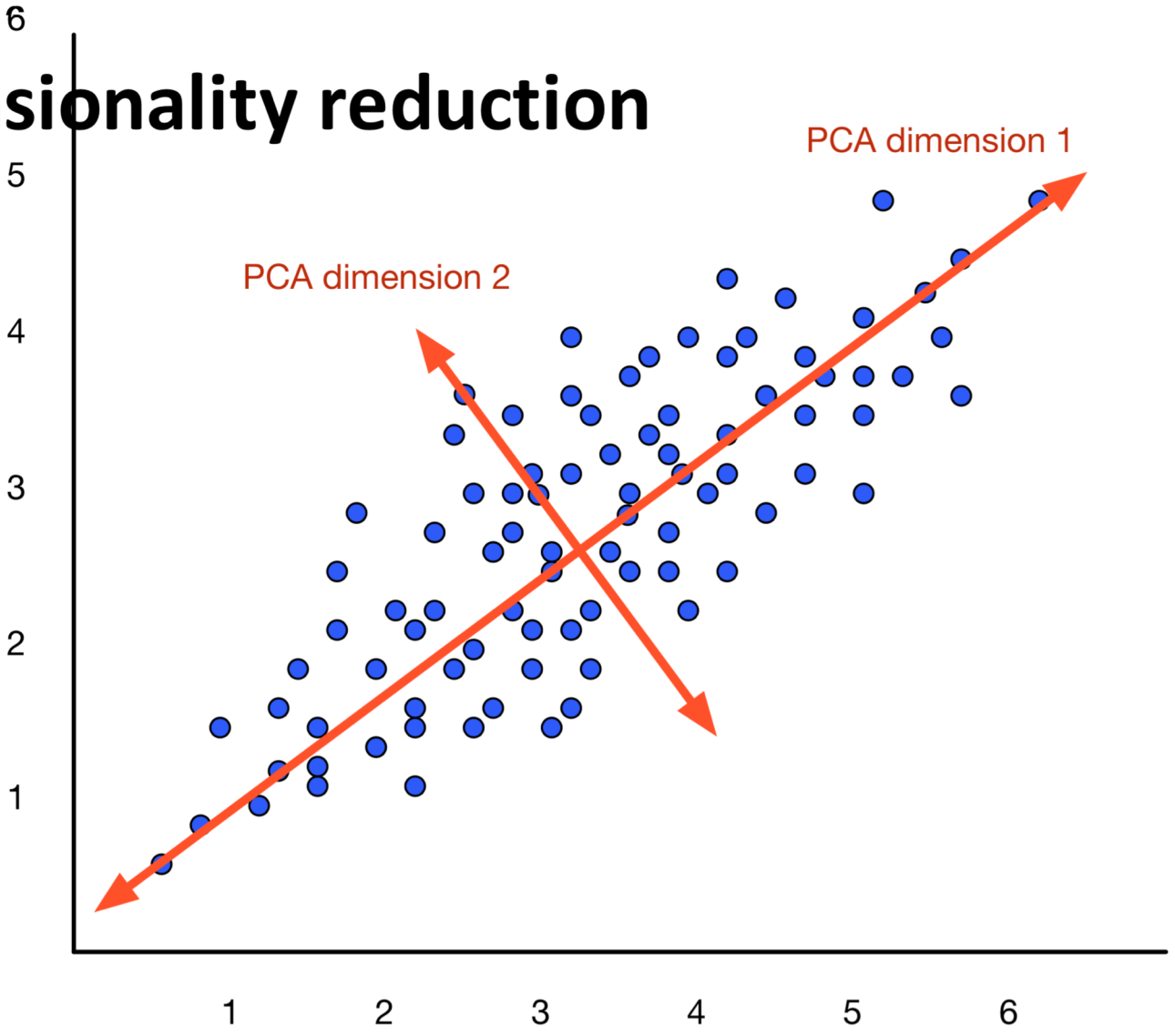
In which the highest order dimension captures the most variance in the original dataset

And the next dimension captures the next most variance, etc.

Many such (related) methods:

- *PCA–principle components analysis*
- *Factor Analysis*
- *SVD*

Dimensionality reduction

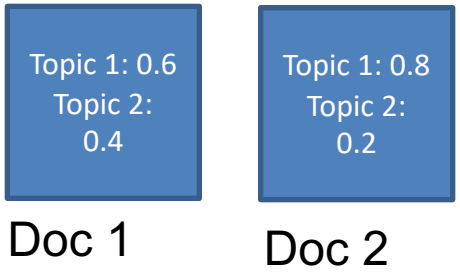


Reasons for performing dimensionality reduction:

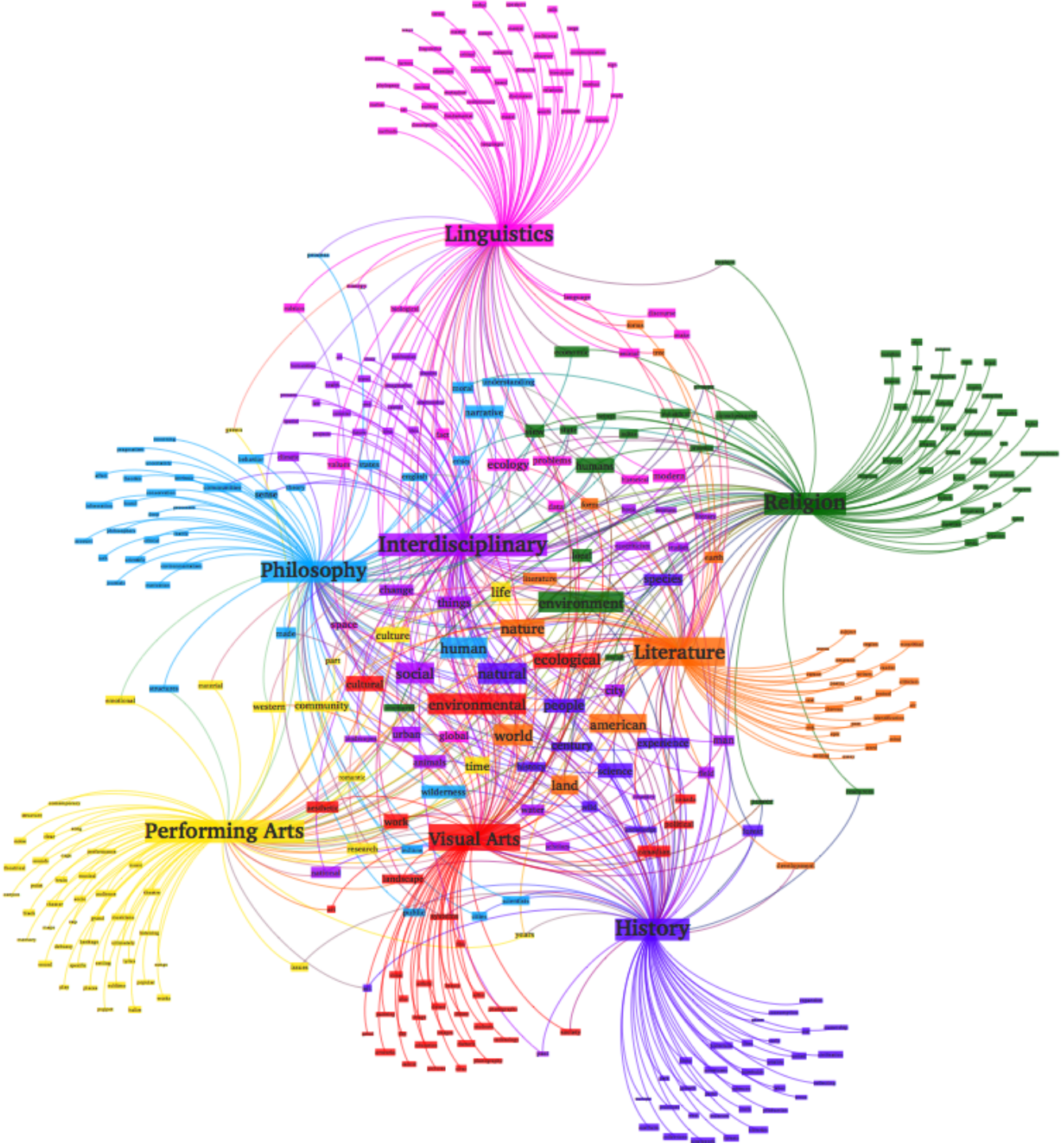
- *Intractable computations*
  - When number of elements and number of features is too large,similarity computations may become intractable
  - reduction of the number of features makes computationtractable again
- *Generalization capacity*
  - the dimensionality reduction is able to describe the databetter, or is able to capture intrinsic semantic features
  - dimensionality reduction is able to improve the results (counterdata sparseness and noise

Priors to LSA, pLSA, LDA

- *Documents with similar topics will use simlar groups of words*
- *Documents are probability distribution over latent topics*



- *Topics are probability distribution over words*


Linguistics
Religion
Interdisciplinary
Philosophy
Literature
Performing Arts
Visual Arts
History
ecology
problems
human
modern
narrative
understanding
life
environment
nature
culture
space
things
change
species
ecological
social
natural
cultural
environmental
people
city
american
world
global
urban
century
expedition
time
animals
science
land
wilderness
work
research
landscape
western
community
earth
literature

Latent Semantic Analysis

- *Application of a mathematical/statistical technique tos imulate how humans learn the semantics of words*
- *LSA finds 'latent semantic dimensions' according to which words and documents can be identified*
- *Goal: counter data sparseness and get rid of noise*
- *What is latent semantic analysis technically speaking?*
  - *The application of singular value decomposition to a term-document matrix o improve similarity calculations*

Singular Value Decomposition

- *Find the dimensions that explain most variance by solving number of eigenvector problems*
- *Only keep the n most important dimensions (n= 50−300)*

Aside note: eigenvector and eigenvalue

- ***Eigenvector:*** *Every vector (list of numbers) has a direction when it is plotted on a XY chart. Eigenvectors are those vectors when a linear transformation (such as multiplying it to a scalar) is performed on them then their direction does not change.* ***The direction of an eigenvector does not change when a linear transformation is applied to it.***

- ***Eigenvalue****—The scalar that is used to transform (stretch) an Eigenvector.*

Aside note: computing eigenvectors and eigenvalues

- Equation to solve:

$$A * x - lambda * x = 0$$

A matrix
x eigenvectors
Lambda eigenvalues

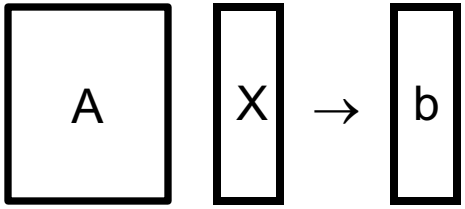
The above equation states that we need to multiply a scalar lambda (eigenvalue) to the vector **x** such that it is equal to the linear transformation of matrix **A** once it is scaled by vector **x** (eigenvector).

As the above equation should not be invertible, we need to ensure that the determinant of the matrix is 0.

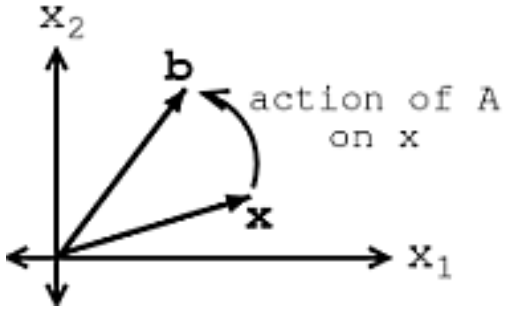
Once we have the Eigenvalues, we can find Eigenvector for each of the Eigenvalues. We can substitute the eigenvalue in the lambda and we will achieve an eigen vector.

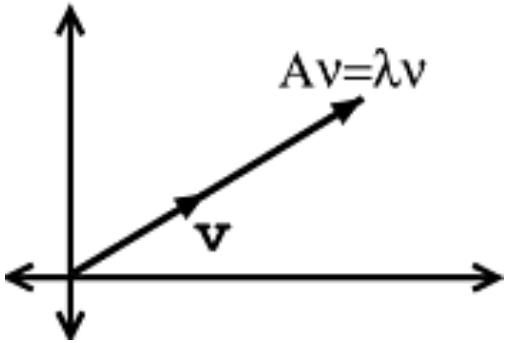
Singular Value Decomposition

- For a square matrix *A*:
- $A\mathbf{x} = \mathbf{b}$



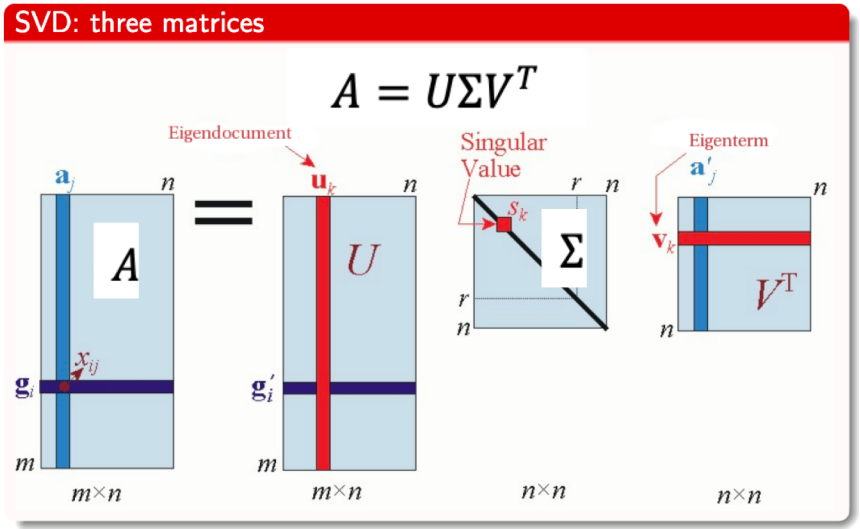
$A\mathbf{x} = \lambda\mathbf{x}$
where **x** is a vector (eigenvector), and λ a scalar (eigenvalue)





SVD: A least-squares method for dimension reduction

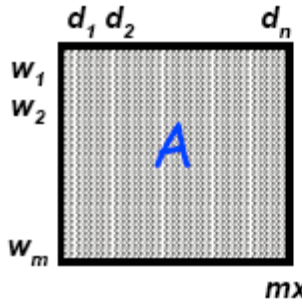
In both *U* and *V*, the columns correspond to one of our *t* topics. In *U,* rows represent document vectors expressed in terms of topics; in *V*, rows represent term vectors expressed in terms of topics.



Computing similarity in LSA

CLASSIC EXAMPLE

- Fundamental comparisons based on SVD
  - The original word-document matrix (A)



- compare two terms → dot product of two rows of A
  - or an entry in AA^T
- compare two docs → dot product of two columns of A
  - or an entry in A^TA
- compare a term and a doc → each individual entry of A

  - The new word-document matrix (A')

$U'=U_{mxk}$
$\Sigma'=\Sigma_k$
$V'=V_{nxk}$

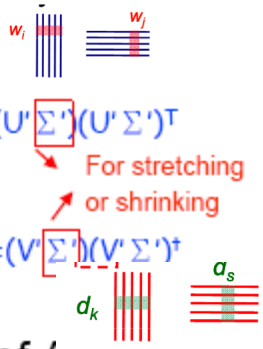
- compare two terms → dot product of two rows of $U'\Sigma'$
- compare two docs → dot product of two rows of $V'\Sigma'$
- compare a query and a doc → each individual entry of A

$A'A'^T=(U'\Sigma'V'^T)(U'\Sigma'V'^T)^T=U'\Sigma'V'^TV'\Sigma'^TU'^T=(U'\Sigma')(U'\Sigma')^T$

$A'^TA'=(U'\Sigma'V'^T)^T(U'\Sigma'V'^T)=V'\Sigma'^TU^TU'\Sigma'V'^T=(V'\Sigma')(V'\Sigma')^T$

I

For stretching or shrinking



31

scaled by the square root of singular values

$$\hat{q}_{1\times k} = \left(q^T\right)_{1\times m} U_{m\times k} \Sigma^{-1}_{k\times k}$$

LSA Exercise

- *A "collection" consists of the following "documents":*
  - d1: Shipment of gold damaged in a fire.
  - d2: Delivery of silver arrived in a silver truck.
  - d3: Shipment of gold arrived in a truck

- *Suppose that we use the term frequency as term weights and query weights. The following document indexing rules are also used:*
  - stop words were not ignored
  - text was tokenized and lowercased
  - no stemming was used
  - terms were sorted alphabetically
- *Problem: Use Latent Semantic Indexing (LSI) to rank these documents for the query "gold silver truck"*

Simple SVD word vectors in Python

Corpus:
I like deep learning. I like NLP.I enjoy flying.

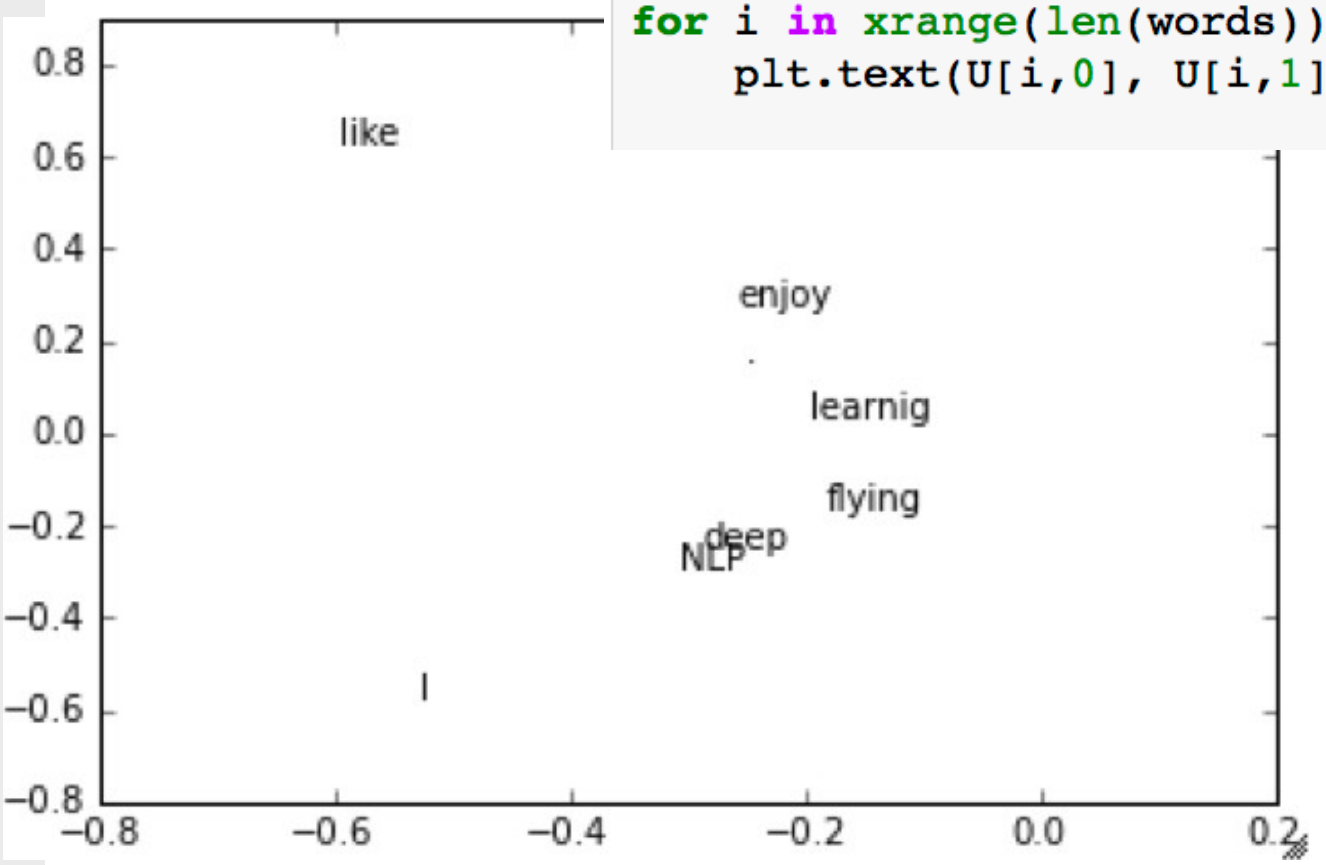
```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep","learnig","NLP","flying","."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

U, s, Vh = la.svd(X, full_matrices=False)
```

Simple SVD word vectors in Python

Corpus: I like deep learning. I like NLP. I enjoy flying.
Printing first two columns of U corresponding to the 2 biggest singular values

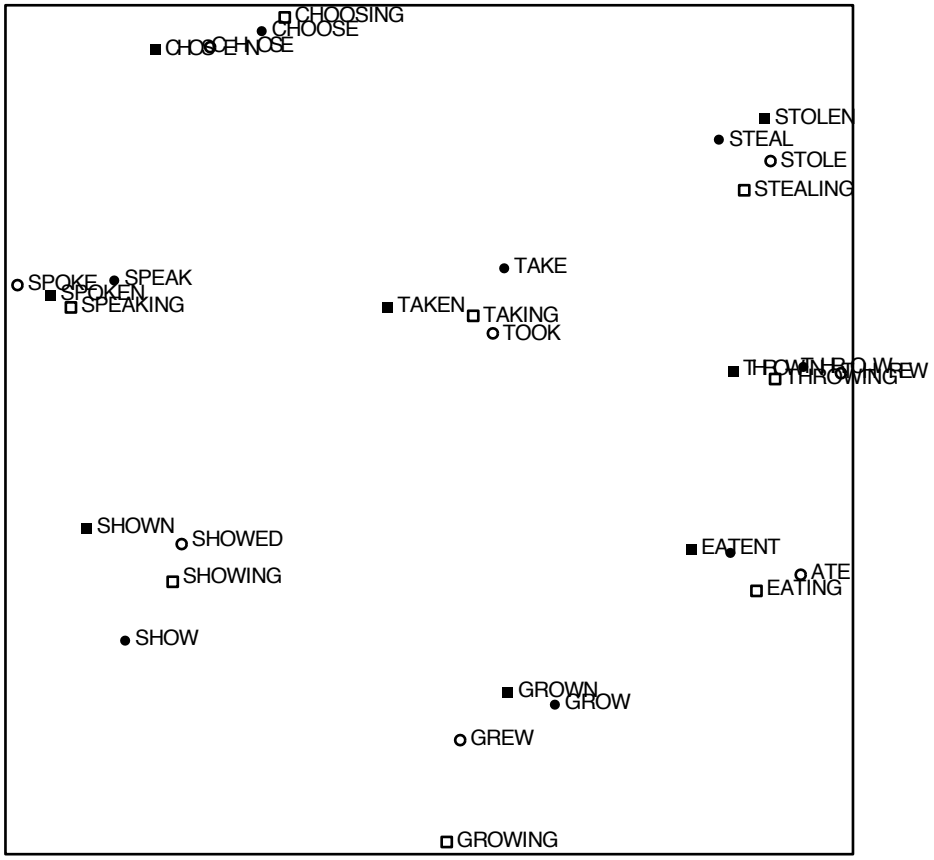
```
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```



Hacks to X (several used in Rohde et al. 2005)

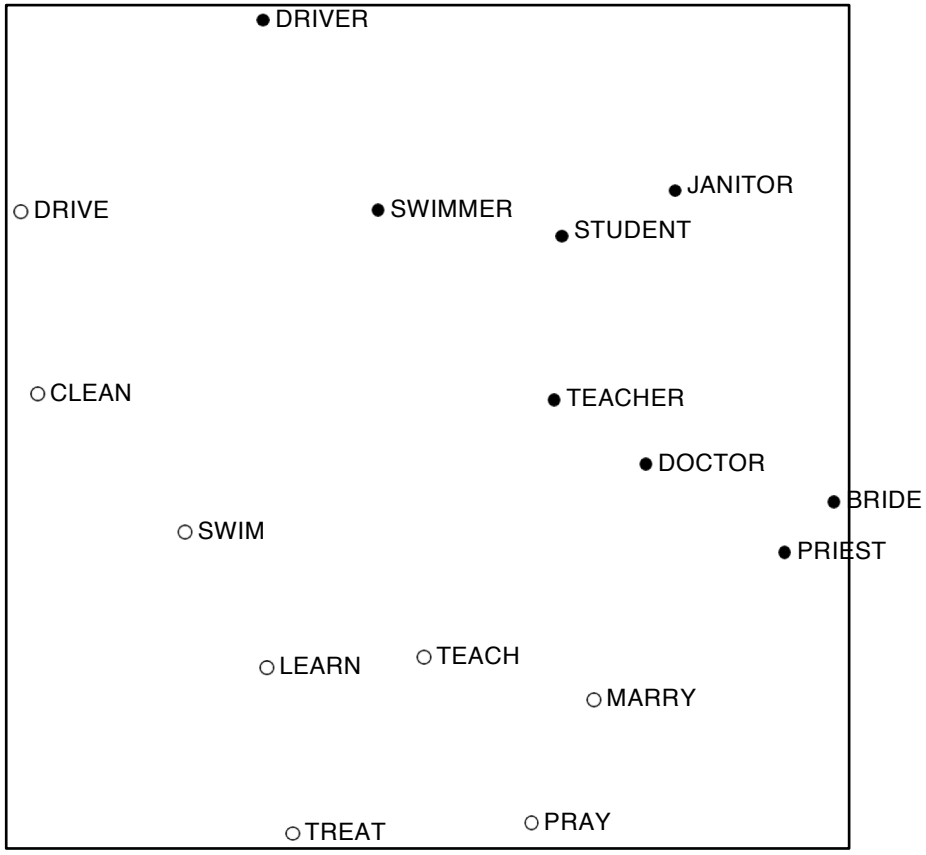
- Scaling the counts in the cells can help a lot
- Problem: function words (the, he, has) are too frequent, syntax has too much impact. Some fixes:
  - min(X,t), with t ≈ 100
  - Ignore them all
  - Ramped windows that count closer words more
  - Use Pearson correlations instead of counts, then set negative values to 0
  - Etc.

Interesting syntactic patterns emerge in the vectors



COALS model from
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence Rohde et al. ms., 2005

Interesting semantic patterns emerge in the vectors



COALS model from
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. ms., 2005

LSA: shortcommings

- *LSA criticized for a number of reasons:*
  - dimensionality reduction is best fit in least-square sense, but this is only valid for normally distributed data; language data is not normally distributed
  - Dimensions may contain negative values; it is not clear what negativity on a semantic scale should designate; lack of interpretable embeddings (we don't know what the topics are, and the components may be arbitrarily positive/negative)
  - need for *really* large set of documents and vocabulary to get accurate results
  - less efficient representation

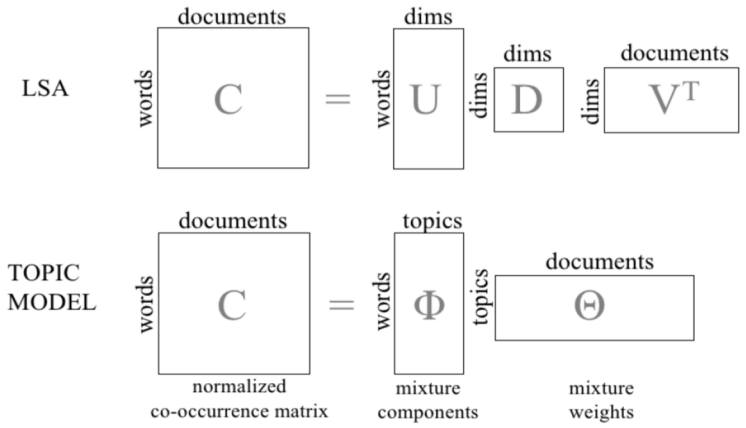
- *Shortcomings are remedied by subsequent techniques (PLSA,LDA, NMF, . . .)*

LSA and pLSA (topic model)

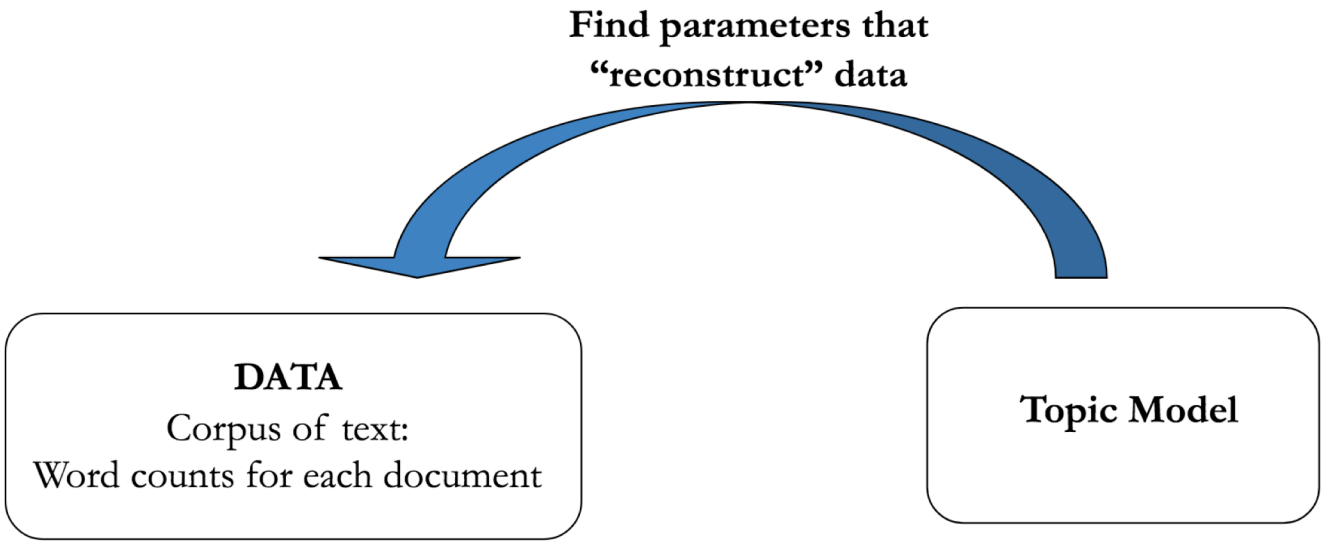
- *LSA: find the k-dimensions that Minimizes the Frobenius norm of A-A'.*

$$\|A\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2 = \operatorname{trace}(A^*A) = \sum_{i=1}^{\min\{m,n\}} \sigma_i^2$$

- *pLSA: defines one's own objective function to minimize (maximize)*



Generative model



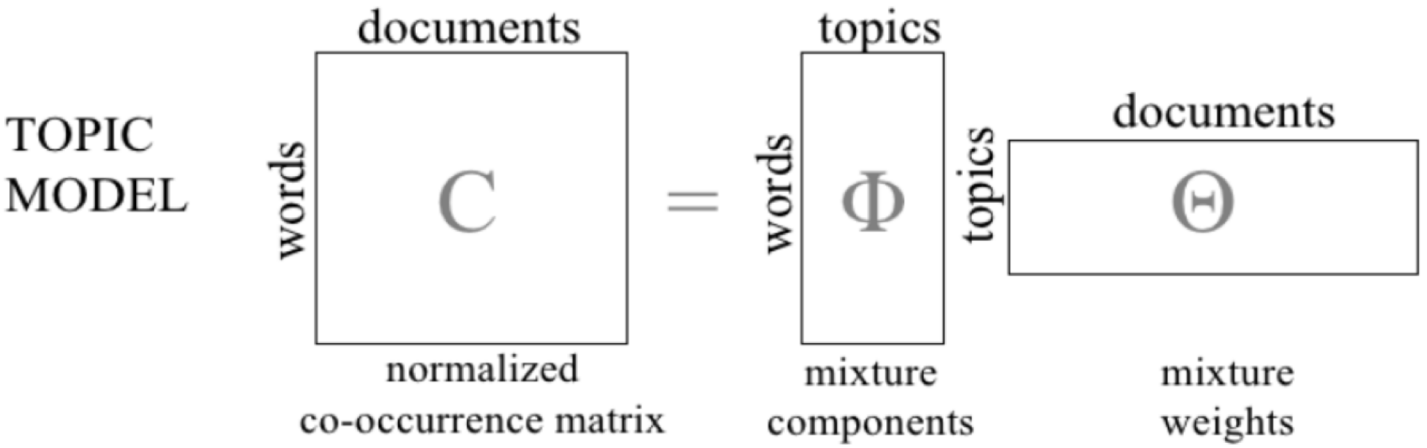
Examples

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

pLSA for K topics and M documents

- *given a document d, topic z is present in that document with probability P(z|d)*
- *given a topic z, word w is drawn from z with probability P(w|z)*



$$P(d_i, w_j) = P(d_i)P(w_j \mid d_i), \quad P(w_j \mid d_i) = \sum_{k=1}^{K} P(w_j \mid z_k)P(z_k \mid d_i).$$

$$P(d_i, w_j) = \sum_{k=1}^{K} P(z_k)P(d_i \mid z_k)P(w_j \mid z_k),$$

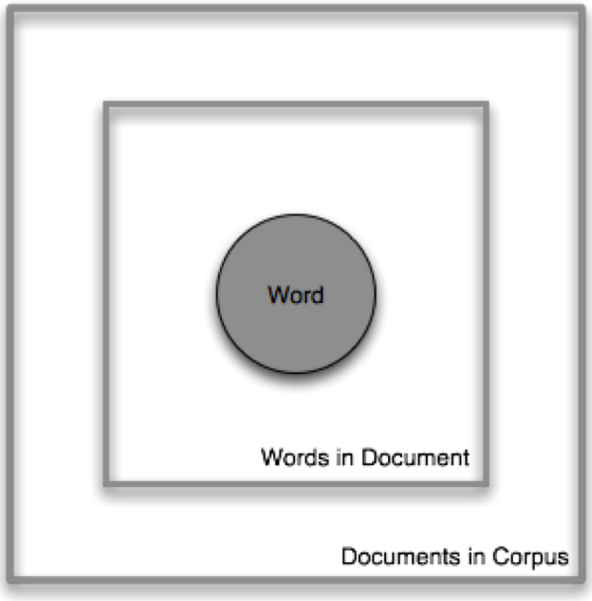
- *Distribution of topics (z)*

Intuitively, the right-hand side of this equation is telling us *how likely it is see some document*, and then based upon the distribution of topics of that document, *how likely it is to find a certain word within that document*.

$$P(d_i, w_j) = \sum_{k=1}^{K} P(z_k) P(d_i \mid z_k) P(w_j \mid z_k),$$

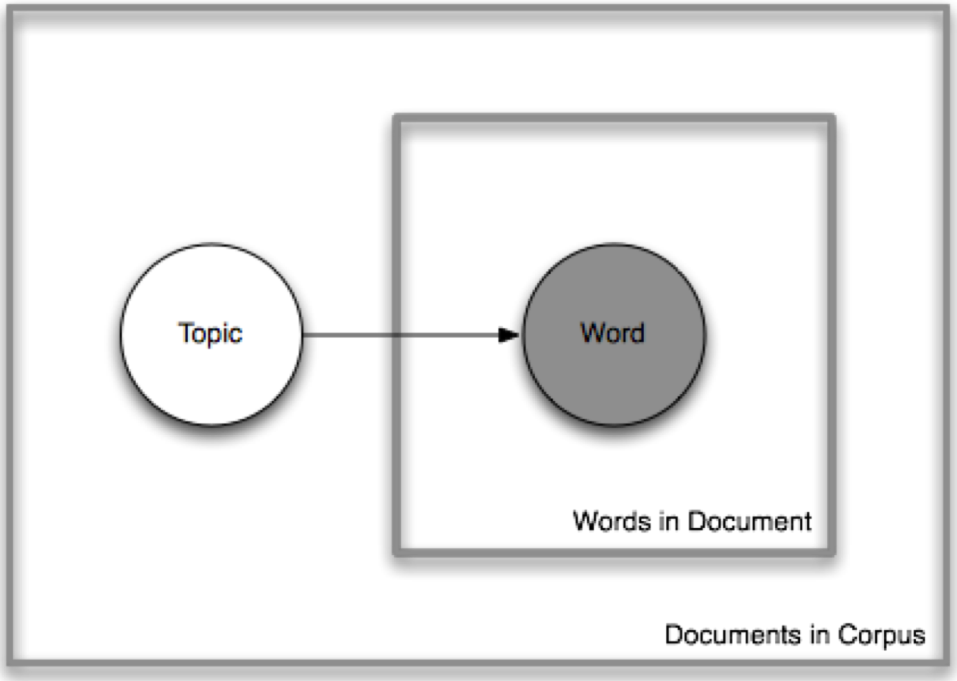
- *Distribution of topics (z)*

pLSA: generative process of the unigram model



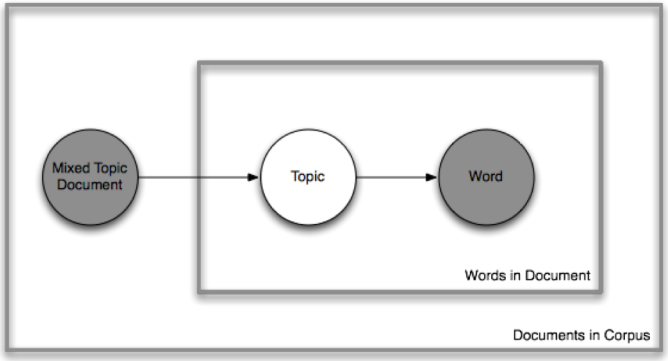
The image above describes the model using plate notation. The outer square represents an iteration over every single document. The inner square represents an iteration over every word for each document. The grey circle in the middle represents the observed variable, in this case the words in each document. The shaded circle is the observed word token we select from the word distribution, a.ka. topic (below), and it is encapsulated by two squares meaning it is nested within two loops. The outer square loops over every document in the corpus and the inner square loops over every word in a document.

pLSA: generative process of the mixture of unigrams model

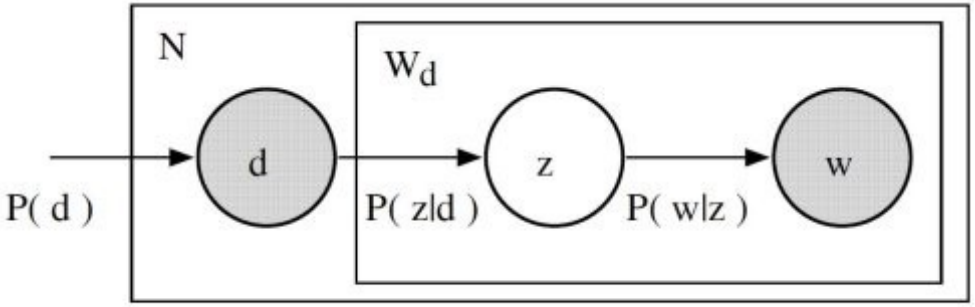


The mixture of unigrams model is represented in plate notation above. The mixture of unigrams adds a new latent, or unobserved, variable to the model that represents the topic, the word distribution, from which each document will be drawing words. Because the latent variable is outside the inner square, there is only one topic per document. This is better than the unigram model which only allows one topic per corpus. This adds a bit more diversity to the model of the corpus, but not necessarily much diversity to any individual document. This means, while the corpus might be about food and books, a single document is about either food or books, but not both.

pLSA

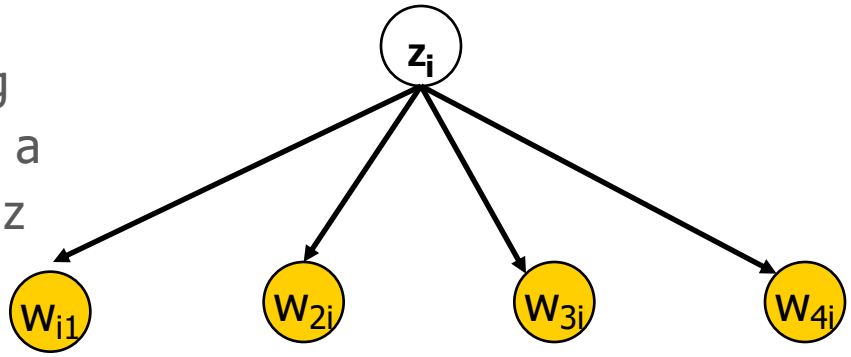


In this case, P(D), P(Z|D), and P(W|Z) are the parameters of our model. P(D) can be determined directly from our corpus. P(Z|D) and P(W|Z) are modeled as multinomial distributions, and can be trained using the expectation-maximization algorithm (EM).



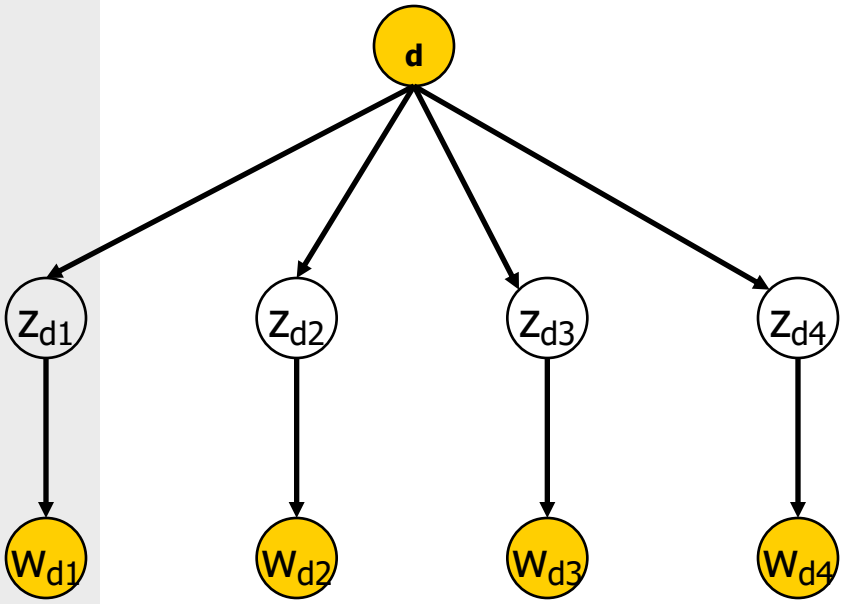
pLSA: step by step

- *P(w|z) ; for each M document:*
  - Choose a topic x
  - Choose N words by drawing each one independent from a multinomial conditioned on z



In the Mixture of Unigrams model, we can only have one topic per document!

pLSA: step by step



For each word of document d in the training set,

- *Choose a topic z according to a multinomial conditioned on the index d.*
- *Generate the word by drawing from a multinomial conditioned on z.*

In pLSA, documents can have multiple topics.

LSA vs pLSA

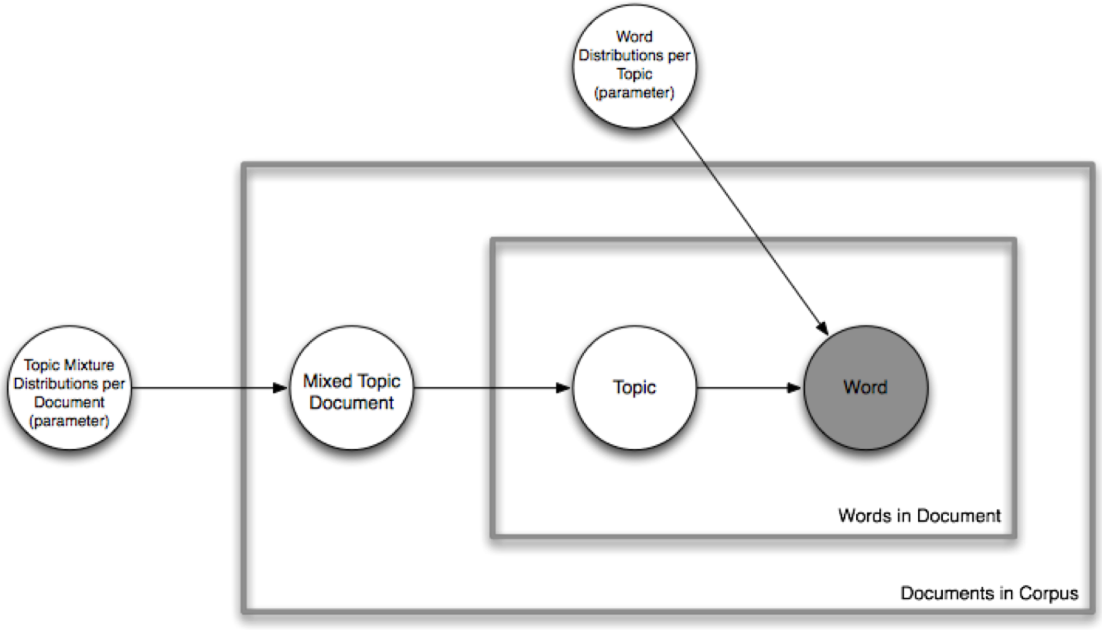
- *Each of them is used to describe incoming set of bag-of-word distribution data into a lower dimensional bag-of-topic data.*

    - LSA: uses SVD, keeps k singular values, and as a result the topics are assumed to be orthogonal.

    - pLSA: has k aspects, treats topics as word distributions, uses probabilistic methods, and topics are allowed to be non-orthogonal.

LDA

- In the LDA model, the topic mixture proportions for each document are drawn from some distribution, i.e. a Dirichlet prior.
  - Dirichlet is a multivariate generalization of Beta distribution.
- pLSA can be considered a LDA under an uniform Dirichlet prior distribution
- The distribution is based on multinomials. That is, k-tuples of non-negative numbers that sum to one.



Summary: LSA vs pLSA vs LDA

- *Each of them is used to describe incoming set of bag-of-word distribution data into a lower dimensional bag-of-topic data.*

  - LSA: uses SVD, and as a result the topics are assumed to be orthogonal.
  - pLSA: Treats topics as word distributions, uses probabilistic methods, and topics are allowed to be non-orthogonal.
  - LDA: similar to pLSA, but with dirichlet priors for the document-topic and topic-word distributions. This prevents over-fitting, and gives better results.