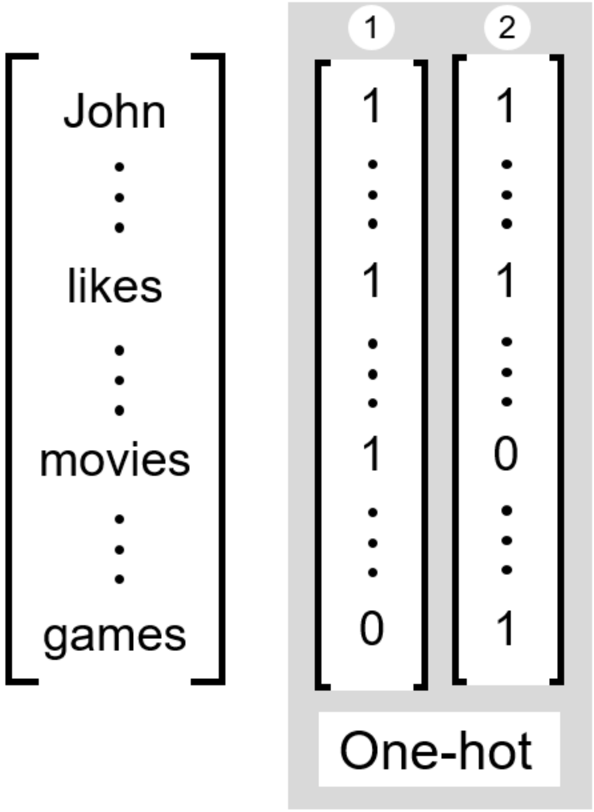
University of Ljubljana, Faculty of Computer and Information Science

Sparse text representations



Prof Dr Marko Robnik-Šikonja

Natural language processing, Edition 2023

Contents

- distributional semantics
- bag-of-words
- document similarity
- tf-idf weighting
- PPMI weighting
- use in information retrieval
- information retrieval evaluation

Distributional semantics

“You shall know a word by the company it keeps”

Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In Studies in Linguistic Analysis, p. 11. Blackwell, Oxford.

"The meaning of a word is its use in the language“

Ludwig Wittgenstein, PI #43

Distributional semantics

- a baseline for a distributional word similarity
- first-order co-occurrence of words (syntagmatic association), words that are typically nearby each other: wrote, book, or poem
- second-order co-occurrence (paradigmatic association), words with similar neighbors: wrote, said, or remarked

We define a word as a vector

- Called an "embedding" because it's embedded into a space
- The standard way to represent meaning in NLP
- Fine-grained model of meaning for similarity
  - NLP tasks like sentiment analysis
    - With words, requires **same** words to be in training and test
    - With embeddings: ok if **similar** words occurred!!!
  - Question answering, conversational agents, etc

Two kinds of embeddings

- sparse, e.g., tf-idf
  - A common baseline model
  - Sparse vectors
  - Words are represented by a simple function of the counts of nearby words
- dense, e.g., word2vec
  - Dense vectors
  - Representation is created by training a classifier to distinguish nearby and far-away words

Word embeddings

- embeddings shall transform syntactic and semantic similarity of words into vector space (as distances and directions)

Sparse vector representation

- *An elephant is a mammal. Mammals are animals. Humans are mammals, too. Elephants and humans live in Africa.*

Africa	animal	be	elephant	human	in	live	mammal	too
1	1	3	2	2	1	1	3	1

9 dimensional vector (1,1,3,2,2,1,1,3,1)

In reality this is sparse vector of dimension |V| (vocabulary size in order of 10,000 dimensions)

Similarity between documents and queries in vector space.

Vectors and documents

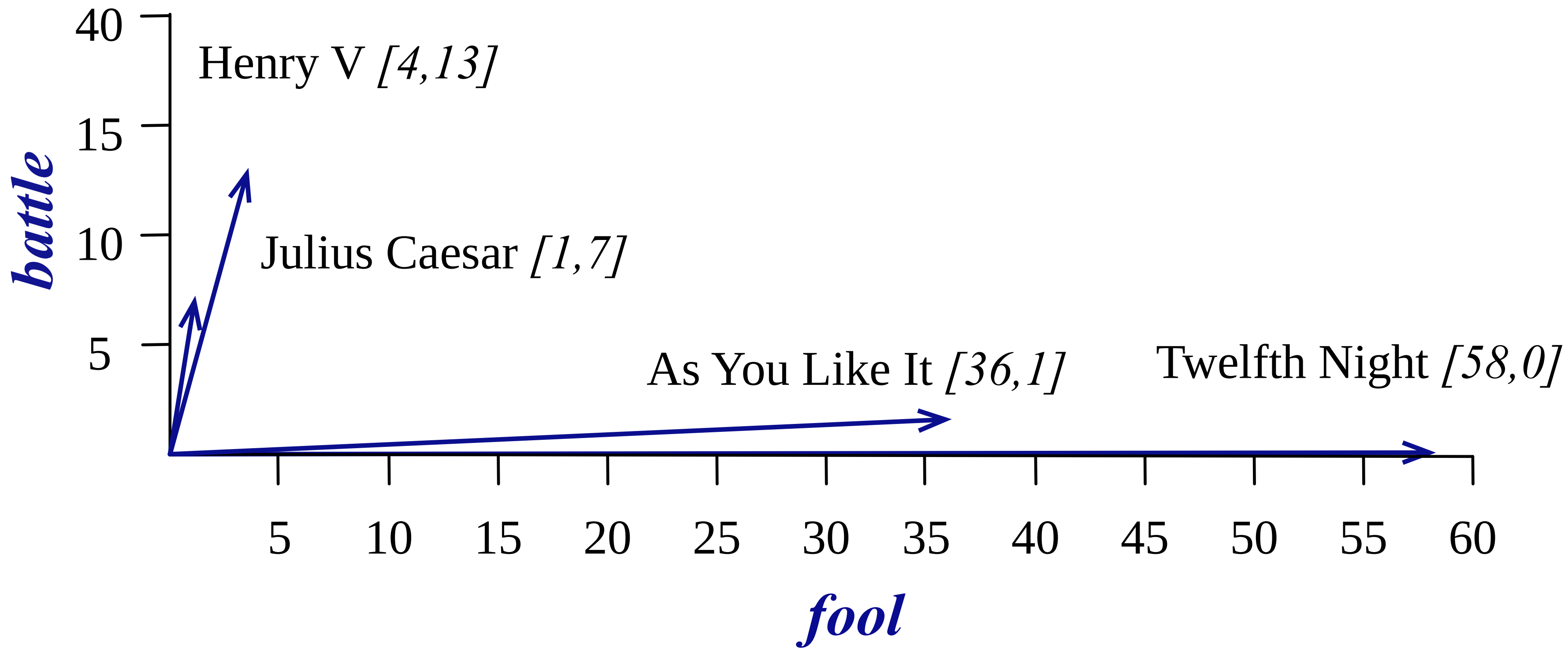
- a word occurs in several documents
- a document contains several words
- both words and documents are vectors
- an example: Shakespeare

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- term-document matrix, dimension |V| x |D|
- a sparse matrix

Visualizing document vectors

- e.g., in two dimensional space



- the difference between dramas and comedies

Vectors are the basis of information retrieval

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Vectors are similar for the two comedies
- Different than the history

- Comedies have more fools and wit and fewer battles.

Words can be vectors too

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle is "the kind of word that occurs in Julius Caesar and Henry V"

fool is "the kind of word that occurs in comedies, especially Twelfth Night"

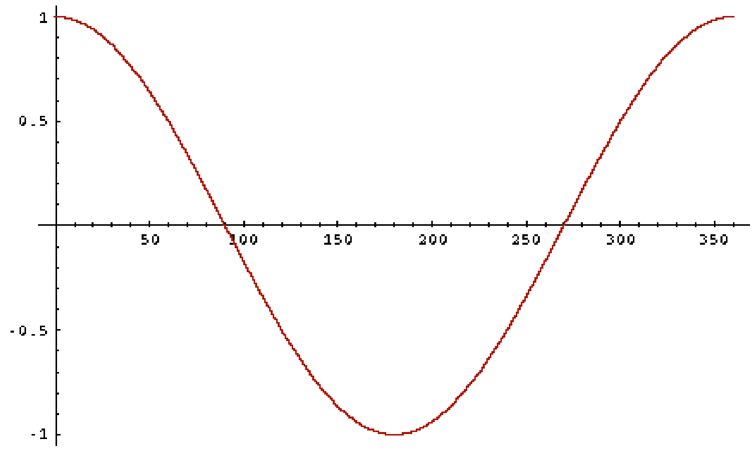
Reminders from linear algebra

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

vector length $\quad |\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$

Cosine as a similarity metric

- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal



- Frequency is non-negative, so cosine range 0-1

Document similarity

- Assume orthogonal dimensions
- Cosine similarity
- Dot (scalar) product of vectors

$$\cos(\Theta) = \frac{A \cdot B}{|A||B|}$$

Weighted similarity

- Between query and document

$$sim(q,d) = \frac{\sum_b w_{b,d} \cdot w_{b,q}}{\sqrt{\sum_b w_{b,d}^2} \cdot \sqrt{\sum_b w_{b,q}^2}}$$

- Ranking by the decreasing similarity

But raw frequency is a bad representation

- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.
- But overly frequent words like *the*, *it,* or *they* are not very informative about the context
- Need a function that resolves this frequency paradox!

tf-idf: combine two factors

- **tf: term frequency**. frequency count (usually log-transformed):

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Idf: inverse document frequency: tf-**

Total # of docs in collection

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Words like "the" or "good" have very low idf

# of docs that have word i

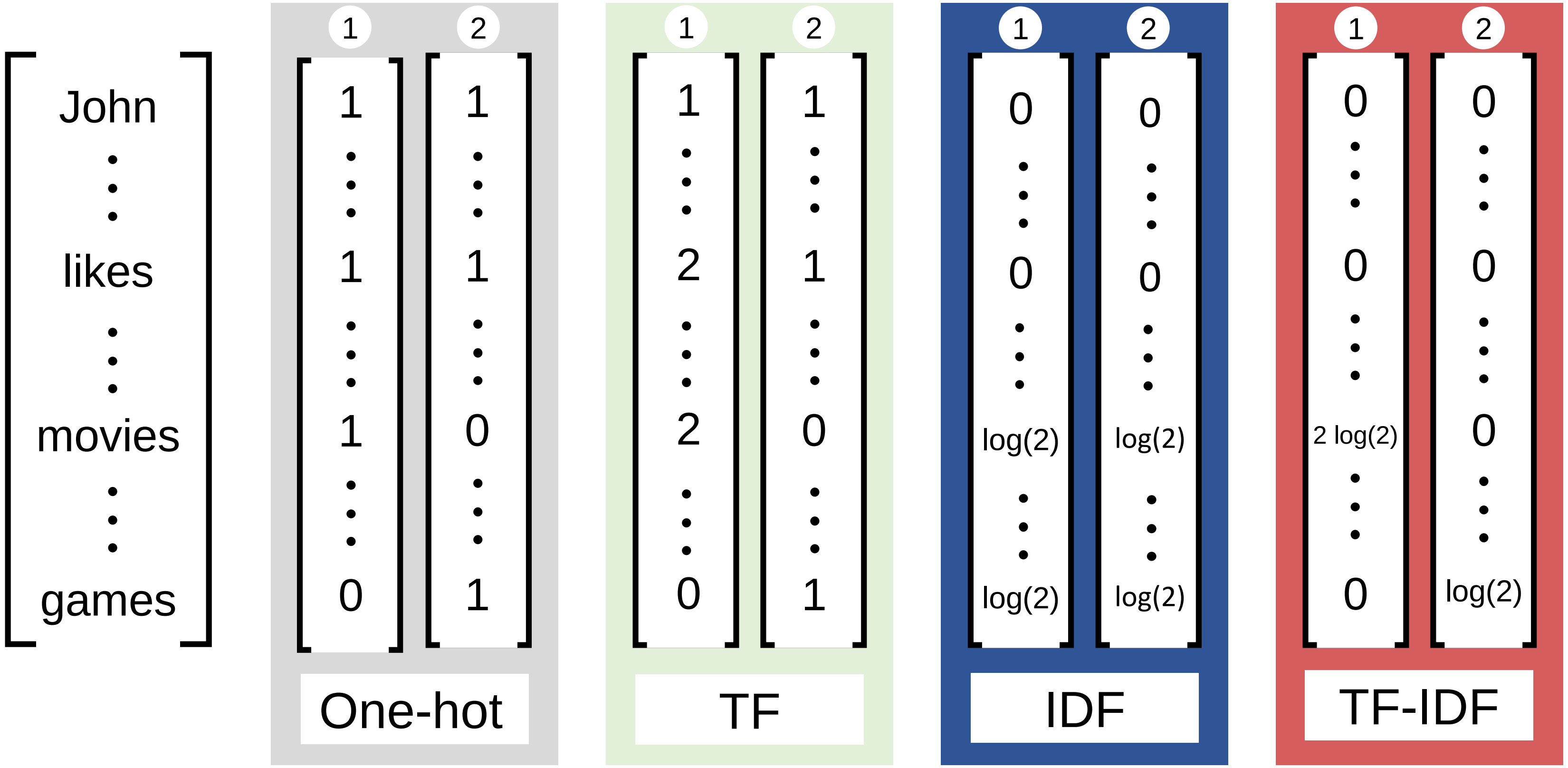
tf-idf value for word t in document d:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

Weights in bag-of-words text representations

Sentences

1. "John likes to watch movies. Mary likes movies too."
2. "John also likes to watch football games."



Evaluation of information retrieval

- How to compare different text representations, weightings, algorithms, etc?

Performance measures for information retrieval

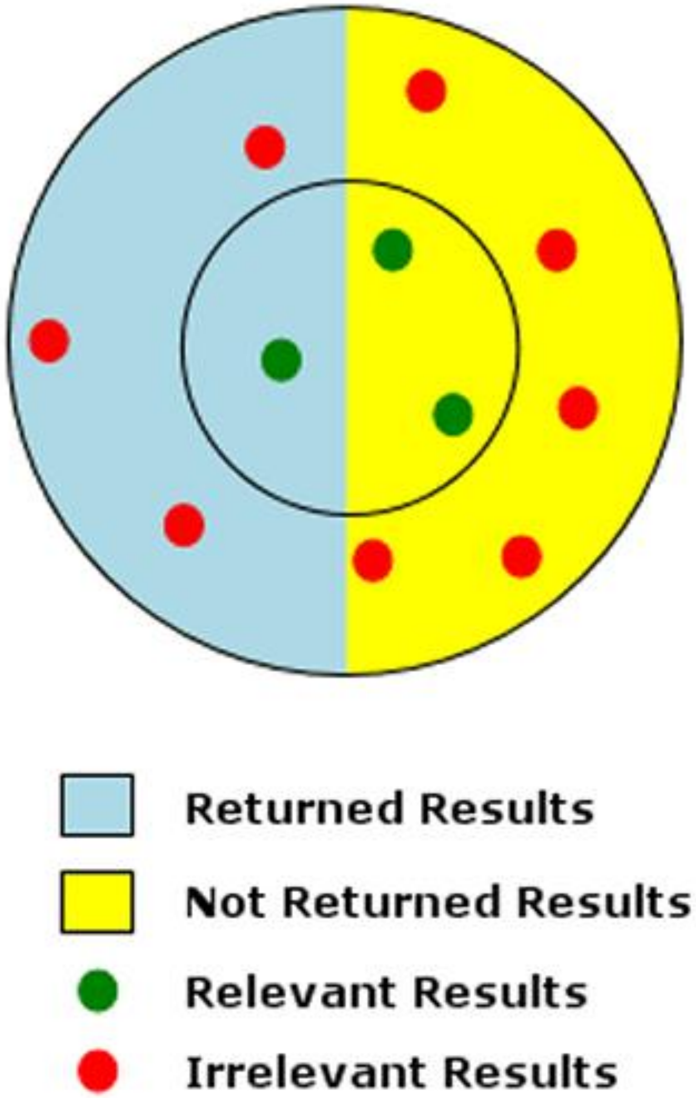
- Subjective measures
- Statistical measures
- Precision, recall
- A contingency table analysis of precision and recall

	Relevant	Non-relevant	
Retrieved	a	b	$a + b = m$
Not retrieved	c	d	$c + d = N - m$
	$a + c = n$	$b + d = N - n$	$a + b + c + d = N$

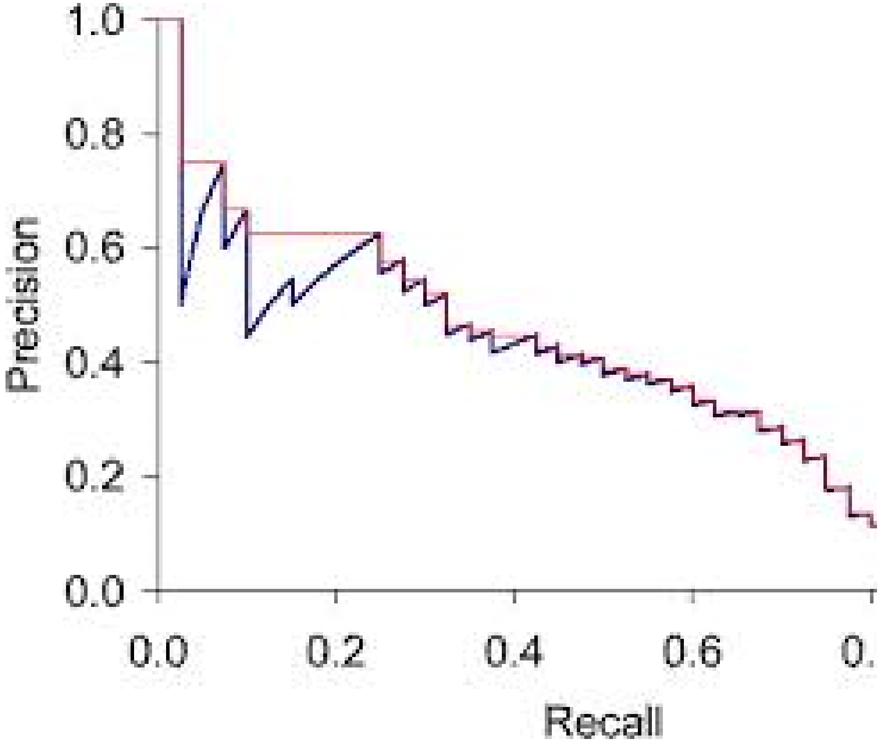
Precision and recall

- N = number of documents in collection
- n = number of important documents for given query q
- m = number of retreived documents
- Search returns m documents including a relevant ones
- Precision $P = a/m$
  proportion of relevant document in the obtained ones
- recall $R = a/n$
  proportion of obtained relevant documents in all relevant documents

An example: low precision, low recall



Precision-recall graphs



F-measure

- combine both P and R
- $$F_\beta = \frac{(1+\beta^2)\cdot P \cdot R}{\beta^2 P + R} \text{ for } \beta > 0$$

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

- Weighted precision and recall
- β=1 weighted harmonic mean
- Also used β=2 or β= 0.5

measure@k

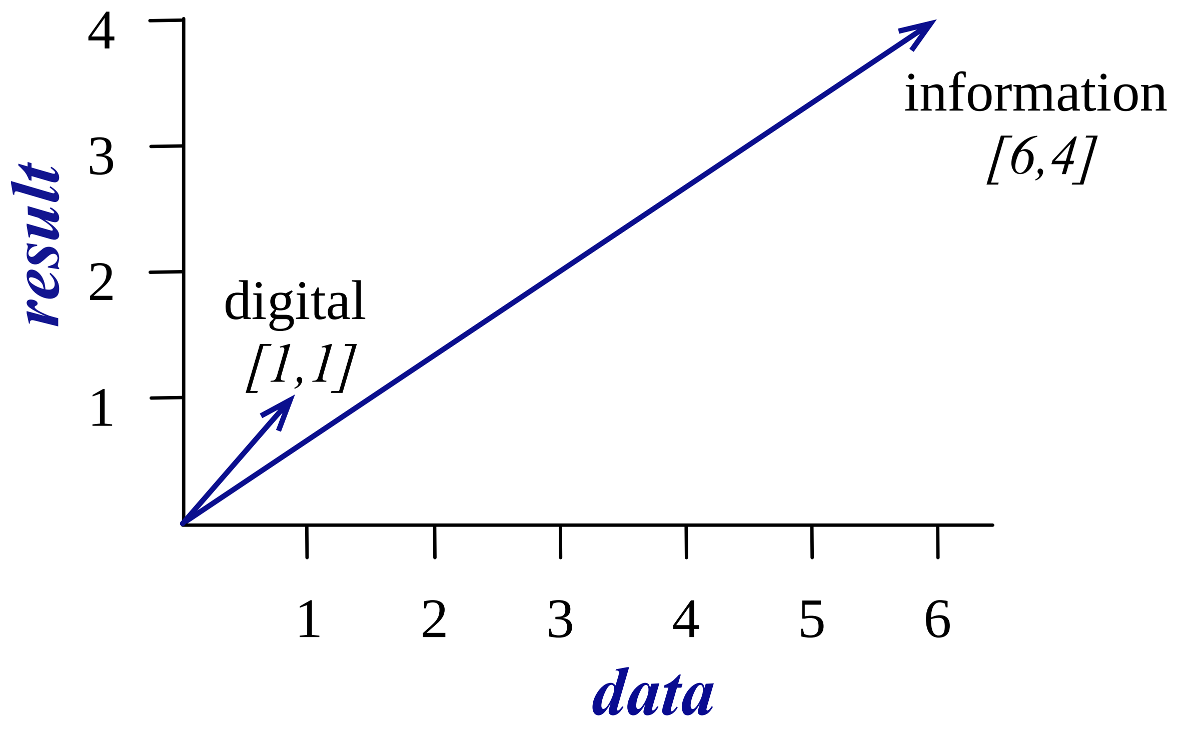
- for large number of returned items, the precision may no longer be a relevant measure (why)
- Precision@k is a precision achieved for the first k returned items
- Recall@k is a recall achieved for the first k returned documents
- Analogously, F1@k
- Weakness: Recall@k increases with larger k

Word similarity with word-word matrix (or "term-context matrix")

- Two **words** are similar in meaning if their context vectors are similar

sugar, a sliced lemon, a tablespoonful of	**apricot**	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	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	


4
3
2
1
result
digital
[1,1]
information
[6,4]
1
2
3
4
5
6
data

Word weighting

- Ask 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?

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

PMI between two words: (Church & Hanks 1989)

Do words x and y co-occur more than if they were independent?

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

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 w_1 and w_2 whose probability is each 10^{-6}
    - Hard to be sure $p(w_1,w_2)$ 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:

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

Computing PPMI on a term-context matrix

- Matrix F with W rows (words) and C columns (contexts)
- f_{ij} is # of times w_i occurs in context c_j

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

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

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

Weighting PMI

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values
- Two solutions:
  - Give rare words slightly higher probabilities
  - Use add-one smoothing (which has a similar effect)

Weighting PMI: Giving rare context words slightly higher probability

- Raise the context probabilities to $\alpha = 0.75$:

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

- This helps because $P_\alpha(c) > P(c)$ for rare *c*
- Consider two events, P(a) = .99 and P(b)=.01
- $P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97 \;\; P_\alpha(b) = \frac{.01^{.75}}{.01^{.75} + .01^{.75}} = .03$

Use Laplace (add-1) smoothing

- relative frequency based probability

$$P(w_i) = \frac{c_i}{N}$$

- Laplace smoothing (add-1)

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

- Add-k smoothing
  k=0.5, 0.1, 0.05, 0.01 ?

Add-2 Smoothed Count(w,context

	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

p(w,context) [add-2] **p(w)**

	computer	data	pinch	result	sugar	p(w)
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
p(context)	0.19	0.25	0.17	0.22	0.17	

Document similarity based on words

- Compare two words using cosine similarity to see if they are similar
- Compare two documents
  - Take the centroid of vectors of all the words in the document
  - Centroid document vector is:

$$d = \frac{w_1 + w_2 + ... + w_k}{k}$$