MSE

MSE | MASTER OF SCIENCE IN ENGINEERING

Advanced Natural Language Processing: AdvNLP

Lesson 4: Non-contextual word vectors

Table of Contents

1. The vector space model (VSM) and information retrieval (IR)
2. Text similarity metrics
3. Word embeddings with matrix factorization (LSA & GloVe)
4. Word embeddings with neural networks (word2vec & fastText)

1. THE VECTOR SPACE MODEL (VSM) AND INFORMATION RETRIEVAL (IR)

The vector space model (VSM)

- The concept of representing words and documents as vectors in a multi-dimensional space was first introduced for developing information retrieval (IR) systems.
- Given a vocabulary of size T, i.e., a list of unique words appearing in the document collection of size D, we represent
  - A document d is as a vector in $\mathbb{N}^{|T|}$ with components counting occurrences of words
  - A word w or term t as a vector in $\mathbb{N}^{|D|}$
- This leads to the term-document matrix of size $\mathbb{N}^{|T| \, x \, |D|}$
- Notes
  - **pre-processing** the document collection by with stopword removal, case-folding, stemming or lemmatizing can be applied to **improve the efficiency and relevance of text representation**
  - under the **Bag of Word** (BoW) approach **word order**, **word distance** or **semantic relations** and **similarities between words** are **not considered**

Source: Salton et al. (1975) - A vector space model for automatic indexing

The vector space model (VSM)

- Database with nine very short documents (i.e., titles)
  - d1 : Human machine interface for ABC computer applications
  - d2 : A survey of user opinion of computer system response time
  - d3 : The EPS user interface management system
  - d4 : System and human system engineering testing of EPS
  - d5 : Relation of user perceived response time to error measurement
  - d6 : The generation of random, binary, ordered trees
  - d7 : The intersection graph of paths in trees
  - d8 : Graph minors IV: Widths of trees and well-quasi-ordering
  - d9 : Graph minors: A survey

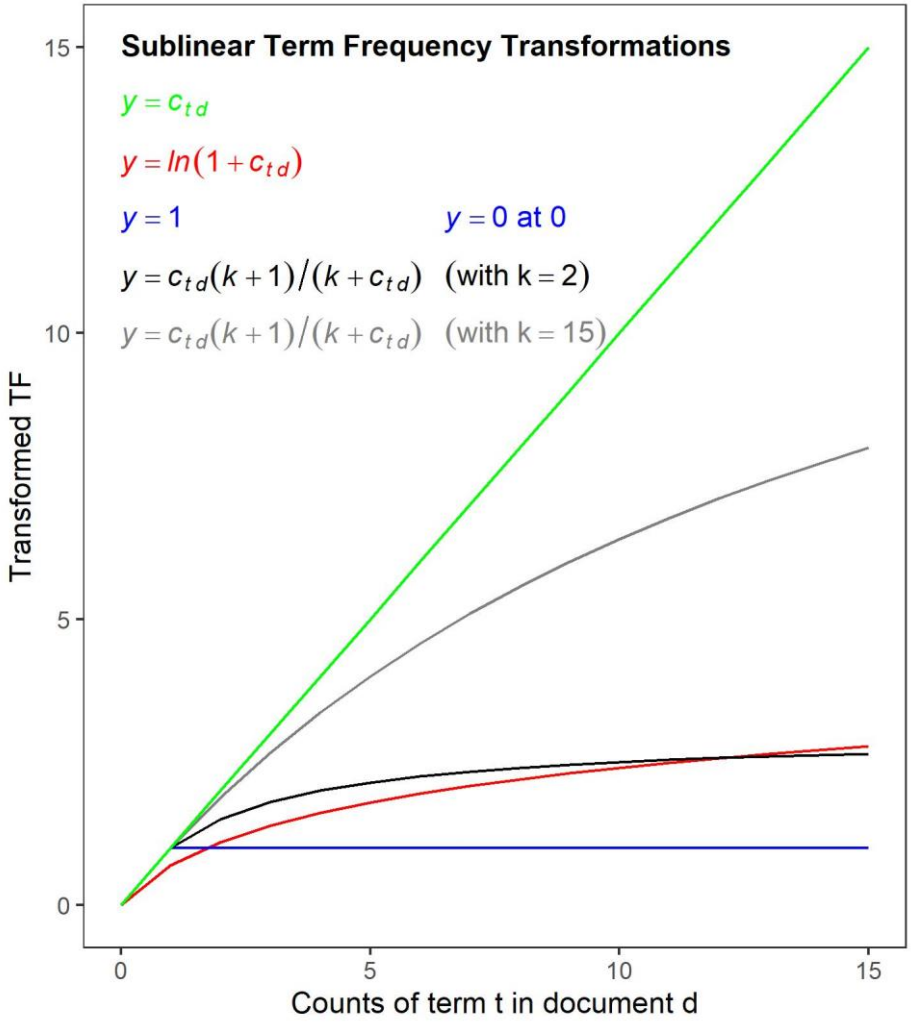
	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

- Vocabulary T pre-processed by removing articles, prepos. & words < 2 occurrences
  - {‘human’, ‘interface’, ‘computer’, ‘user’, ‘system’, ‘response’, ‘time’, ‘EPS’, ‘survey’, ‘trees’, ‘graph’, ‘minors’}

Source: Landauer, T. K., Foltz, P. W., & Laham, D. (1998). "Introduction to Latent Semantic Analysis", *Discourse Processes*, 25, p. 259-284.

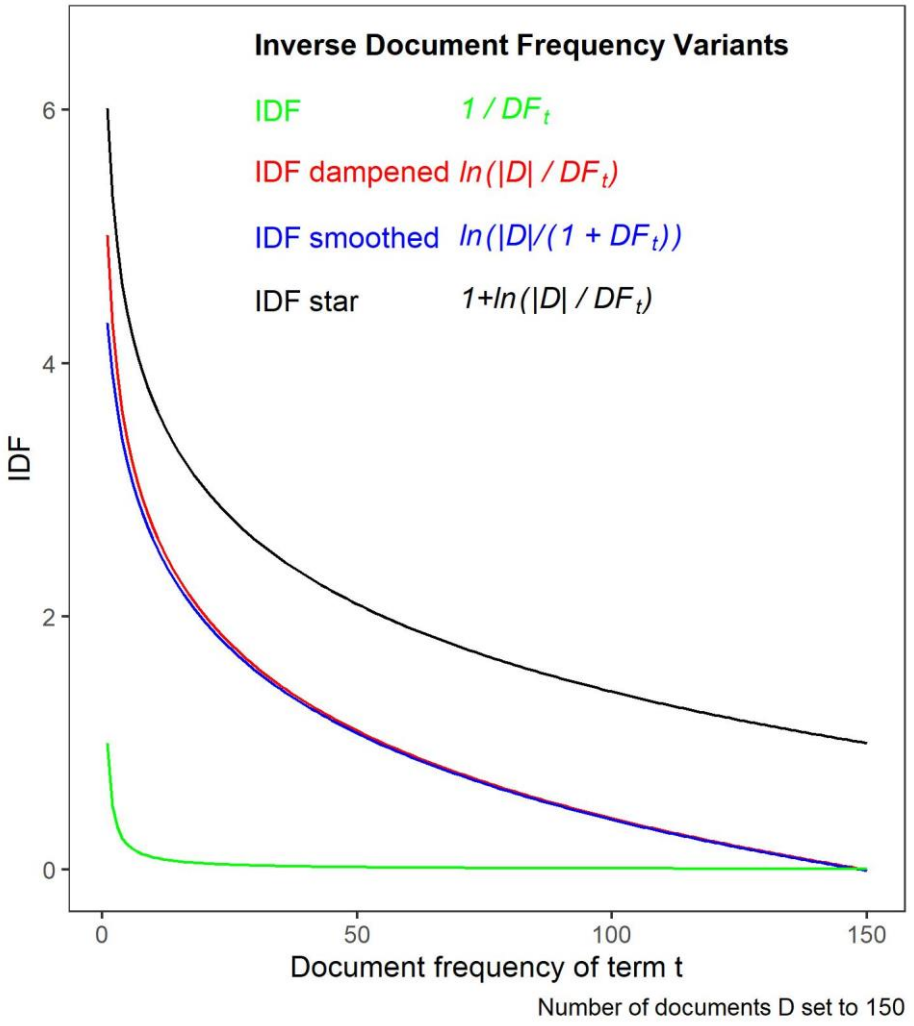
Term-Frequency (TF) Adjustments

- Relevance in fact increases sub-linearly with number of occurrences
  - If a **query word** occurs **5 times** in a document, then this document should be preferred over a document where it occurs only once … but if a **query word** occurs **50 times** in a document it should not be preferred 50 times more over a document where it occurs only once
- Term-frequency $TF_{term,docu}$ requires adjustment to avoid dominance of single words and comes in different flavors
- Note: $c_{t,d}$ is the number of occurrences of a term t document d



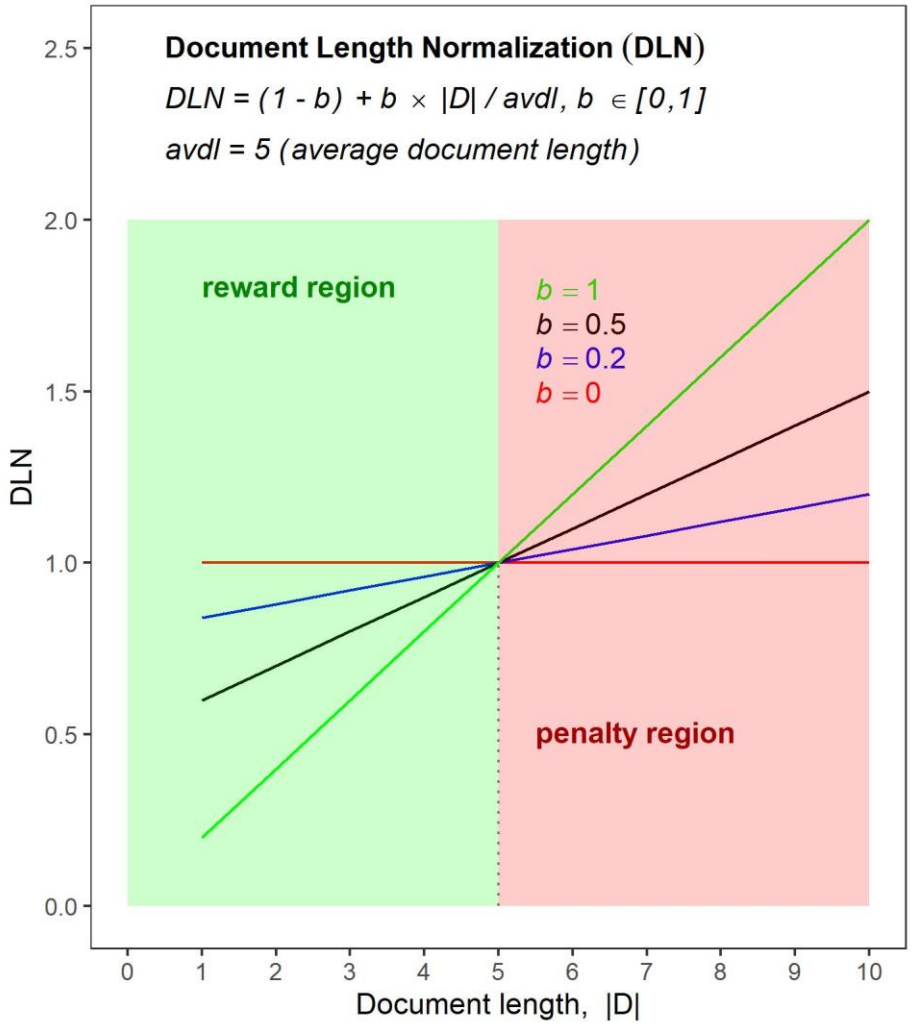
Inverse document frequency (IDF)

- Relevance in fact depends also on informative, e.g., domain-specific words even if they are rare
  - If a word occurs frequently across many/all documents (e.g., 'the', 'is', 'for', etc.) TF needs to be adjusted to suppress spurious relevance
- Document frequency related adjustment of term frequency $TF_{term,docu}$ comes in different flavors
- Approaches:
  - Naïve: IDF_{term} equals $1/DF_{term}$, i.e., the inverse of the number of documents in which the term occurs
  - Sophisticated: dampening with logs, $|D|$ number of documents in the collection
- Note: IDF_{term} can be pre-computed independent of queries



Document length normalization (DLN)

- Relevance requires careful consideration of document length
  - If a documents is long the chance to match any query increases and requires penalization… but a long document can objectively have more relevant content
- Document frequency related adjustment of term frequency $TF_{term,docu}$ uses document length normalizer applying the average document length as pivot
- Note: DLN plays a role in today's IR scoring systems (see Okapi BM-25 below)



The TF-IDF model

- Each cell of the term-document is transformed via TF-IDF with a multiplicator

$$TF - IDF_{term,docu} = TF_{term,docu}\ x\ IDF_{term,collection}$$

- TF-IDF increases each cell's value with the number of occurrences of given term within a document, and the rarity of term in the collection
- The TF-IDF transforms the term-document matrix from $\mathbb{N}^{|T|\ x\ |D|}$ to $\mathrm{R}^{|T|\ x\ |D|}$

	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

TF-IDF

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Term-document matrices for four words in four Shakespeare plays: counts (left) and TF-IDF weights (right). TF here uses a ´$\log_{10}(.) + 1$´ transform, IDF a ´$\log_{10}(.) + 1$´ transform based on 37 Shakespeare plays. The weights of 'good' ('fool') have been reduced as they appear 37 (36) times in all 37 plays. 'wit' occurring in 34 out of 37 plays yields IDF = $\log_{10}(37/34) = 0.037$, TF = $\log_{10}(20) + 1 = 2.301$

Source: Jurafsky, D. & Martin, J.H. (2008), Speech and Language Processing, 3rd edition, draft, https://web.stanford.edu/~jurafsky/slp3/, Chapter 6.3

Conclusion

- With document preprocessing and term-document matrix transformation, text can be analyzed algorithmically
- The coefficients of the term-document matrix can be fine-tuned in various ways, i.e., with different flavors of TF, IDF and DLN

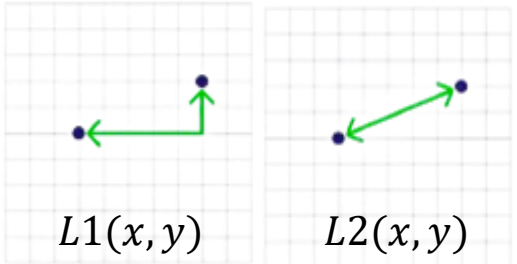
- However, different approaches are required to properly represent
  - Word order
  - Word distance
  - Semantic relations and similarities between words

2. TEXT SIMILARITY METRICS

Measuring similarity between two vectors

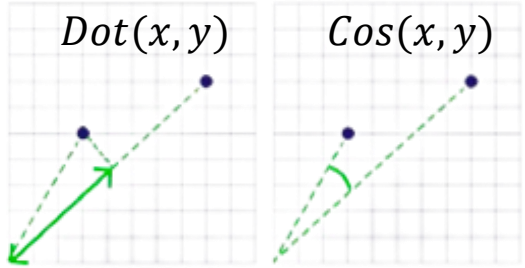
- Similarity can be measured by various metrics, the choice depending on your application, data and model. Similarity can be based on distance…

$$L1(x,y) = \sum_{i=1}^{n} |x_i - y_i|, \qquad Lp(x,y) = \sqrt[p]{\sum |x_i - y_i|^p}$$



- … or on orientation…

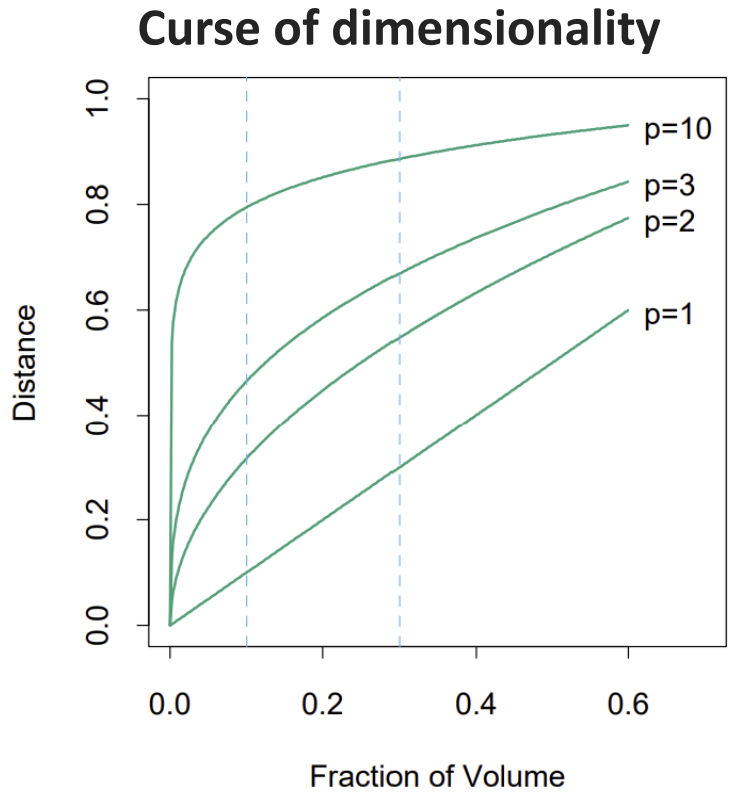
$$Dot - Product(x,y) = \sum_{i=1}^{n} x_i \cdot y_i \,, Cos(x,y) = \frac{\sum x_i\, y_i}{\sqrt[2]{\sum |x_i|^2}\,\sqrt[2]{\sum |y_i|^2}}$$



- … or on intersection/overlap (e.g., Jaccard), or on distributions (e.g., KL-divergence)

Distance-based metrics

- In high dimensions, distance is tricky, i.e., data tends to be concentrated around the surface, not the center, data is spread out and sparse
  - Reason: the ratio of the volume near the center to the rest of the volume becomes very small
- The ratio of the distances of nearest and farthest neighbors to a given point is ≈ 1 for a wide variety of data distributions and distance functions
- Implication: relative difference between distances becomes less significant in Euclidean terms, i.o.t. data points are effectively equidistant from one another



Side-length of a hypercube (y-axis) needed to capture a fraction of the volume (x-axis) of uniformly distributed data

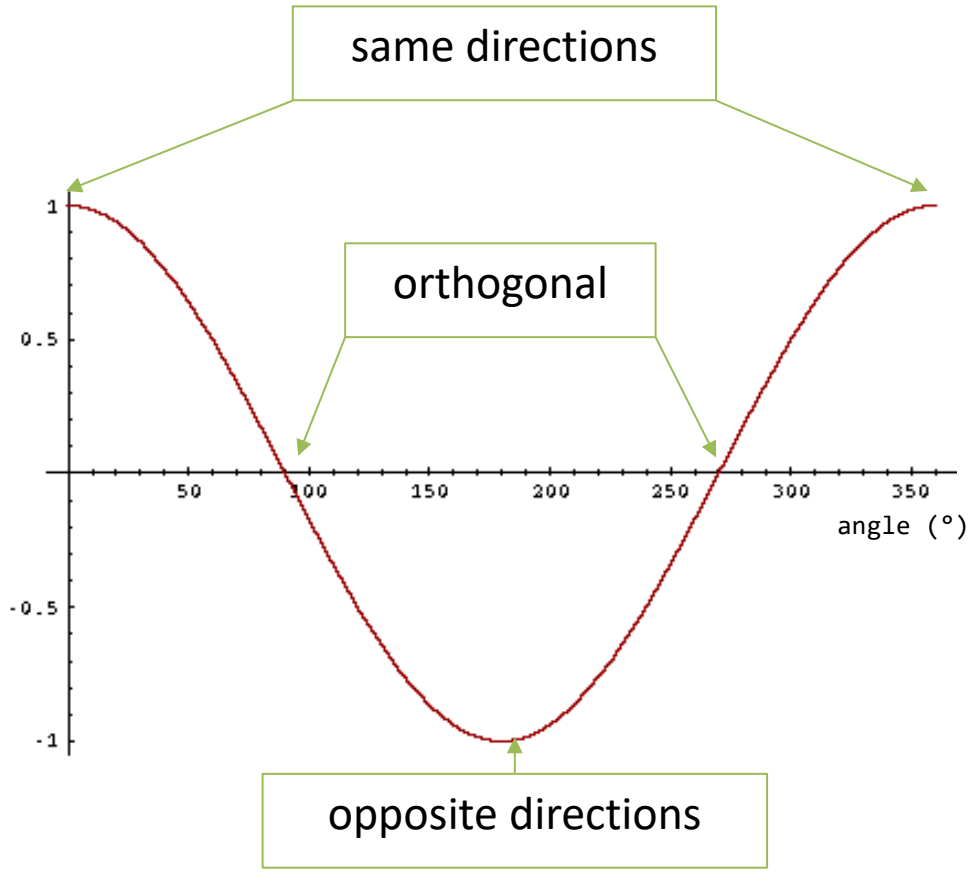
Source: Hastie, Trevor, et al. The elements of statistical learning: data mining, inference, and prediction. Vol. 2. New York: springer, 2009, p.23.

Orientation-based metrics

- Cosine similarity is determined by the orientation of vectors, not the Euclidean distance of data points
- Values of cosine similarity lie between -1 and 1
- Cosine similarity increases if the two vectors point in the same direction
- Interpretability of cosine similarity is clearer than for unbound $L1$ and $L2$ metrics

- Note: occasionally cosine distance is implemented (e.g., ChromaDB default)

$$Cos\ distance(x, y) = 1 - Cos(x, y)$$



Distance vs Orientation-based metrics

- Euclidean norm ($L2$) and Cosine similarity (Cos) time complexity is both $O(n)$, where n is the dimensionality of the vector space
- Dot product uses simpler operations (one multiplication and one addition per iteration) than the Euclidean norm (subtraction, squaring, and addition for each iteration and square root)

- Note: If vectors are normalized to length 1,
  - Cosine similarity can be computed faster via dot product
  - Cosine similarity and Euclidean distance are closely related via

$$[L2(x,y)]^2 = (x-y)^T(x-y) = xT\,x + yT\,y\ - 2\,x^T y = 1 + 1\ - 2\,xT\,y = 2\ - 2\cos(x,y)$$

  - Similarity-based ranking of vectors under both $L2$ and Cos metrics are equivalent!

Sources: Tom Hazledine, "Alternatives to cosine similarity", StackExchange: Is cosine similarity identical to l2-normalized euclidean distance?

Application of VSM and similarity metrics for IR

- Compute vector representation $v(D_i)$ for each document D_i. This can be applied off-line, i.e., $v(D_i)$ can be pre-computed, since it does not depend on the query
- Similarly, compute a vector representation $v(Q)$ for a given query Q
- Identify most relevant documents, e.g., by highest cosine similarity for a given query
- Practical implementation: from TF-IDF to Okapi BM25

$$score(Q, D_i) = \sum_{t\ in\ Q} \frac{TF - IDF(t, Q)}{\sqrt{\sum_{q_j}(TF - IDF\ (q_j, Q))^2}} \cdot \frac{TF - IDF(t, D_i)}{\sqrt{\sum_{d_j}(TF - IDF(d_j, D_i))^2}}$$

dot product

vector length

TF-IDF

$$score(Q, D_i) = \sum_{j=1}^{n} \log \frac{N - n(q_j) + 0.5}{n(q_j) + 0.5} \cdot \left[\frac{TF(q_j, D_i) \cdot (k_1 + 1)}{TF(q_j, D_i) + k_1 \cdot (1 - b + b \cdot \frac{|D_i|}{avg\ doc.\ len.})} + \delta \right]$$

Okapi BM25

IDF

DLN

Conclusion

- So far, we have been representing text as Bag of Words with the sparse Vector Space Model and applying similarity metrics
- Advantages of this approach are the following
  - Concept is simple and well-founded
  - Documents can be ranked according to relevance using similarity with the query (continuous scale)
  - Matching of terms between query and document can be partial (subset)
- Limitations of this approach include
  - Documents/queries with similar content but different term vocabularies (e.g., synonyms or plurals) will not be associated
  - Word order is ignored (“parking fine” vs. “fine parking”, “joint model” vs. “model joint”)

3. WORD EMBEDDINGS WITH MATRIX FACTORIZATION (LSA & GLOVE)

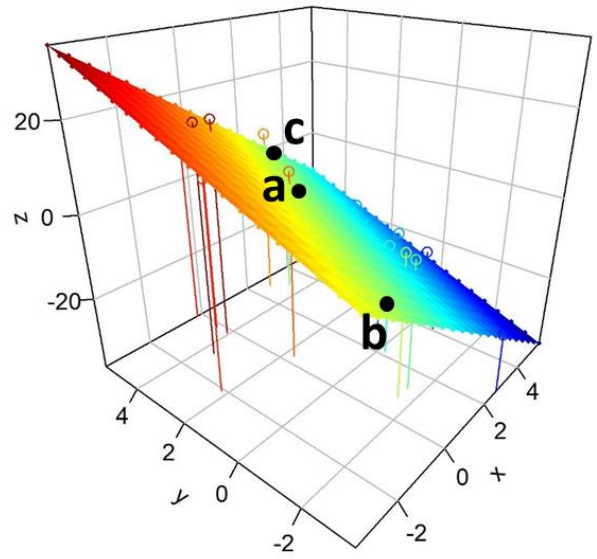
Modeling relatedness of word meanings

- Distributional semantics teaches us that words occurring often in similar contexts are likely to be similar ("*you shall know a word by the company it keeps*", J.R. Firth 1957)
- Relatedness of word meanings may be partially obtained via dictionaries, as far as finding synonyms is concerned…, but dictionaries are only available for a few languages and may be incomplete
- The more general topic of word similarity is preferably learnt from data covering similarity of types (car/bicycle, Paris/London) or topics (car/vehicle/engine/gas)

- Idea: transform high-dimensional space of words, where all words are perpendicular see TF-IDF VSM, into a low-dimensional, dense embedding space, where topically-similar words get closer

Intuition behind dimensionality reduction

- Instead of many unrelated vectors (e.g., word vectors), find a cluster of vectors that point in the same direction
  - Change the axis of the space to this direction
  - Represent each vector through its coordinate on this direction
  - Proceed orthogonally to the next direction, etc.
  - Stop at a much smaller number of new vectors than the original space → implies an approximation

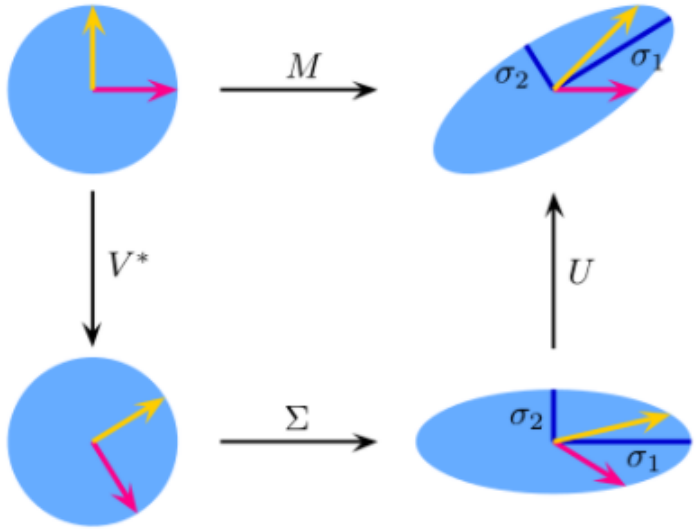


- Results: in the reduced space, similar original vectors are all reduced to the same principal component (plus some small "noise" on other dimensions)

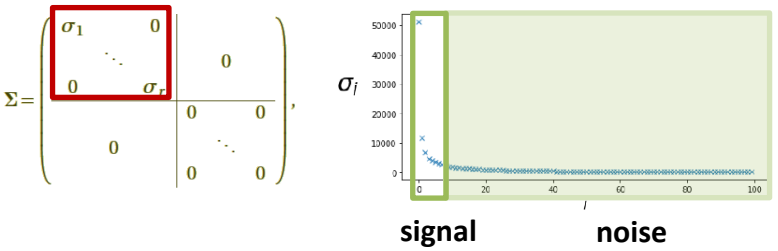
Base vectors	a	b	c	Dimensions
(x, y, z)	(1,2,1)	(-2,-3,1)	(3,5,0)	Full
(p, q)	(1,0)	(0,1)	(1,-1)	Reduced
2-D coordinates **p** = (1, 2, 1) ; **q** = (-2, -3, 1)				

Singular Value Decomposition (SVD)

- A matrix M of size m x n can be decomposed by 3 geometrical transformations $U \cdot \Sigma \cdot V^T$
  - Rotation U is an orthogonal matrix of size m x m with $U^T \cdot U = \mathbb{I}_m$
  - Coordinate-by-coordinate scaling Σ of size m x n is a matrix with null coefficients, except for non-negative diagonal elements σ_i sorted in decreasing order
    (diagonal elements σ_i are the singular values of M, i.e., the square roots of the eigenvalues of $M^T \cdot M$)
  - Rotation V^T of size n x n is the transpose of an orthogonal matrix (i.e., $V^T \cdot V = \mathbb{I}_n$) c



Coordinate-by-coordinate scaling & Singular values



[Relation between PCA and SVD]

SVD toy example

Original		Rotation		Scaling		Rotation
M (size 7x5)	=	U (size 7x7)	×	Σ (size 7x5)	×	V^T (size 5x5)

	Star Wars	Matrix	Star Trek	Australia	Out of Africa
Druid	1	1	1	0	0
Starship	3	3	3	0	0
Starfight	4	4	4	0	0
Power	5	5	5	0	0
Loneliness	0	2	0	4	4
Love	0	0	0	5	5
Desire	0	1	0	2	2

=

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7
Druid	-0.14	-0.02	-0.01	0.56	-0.38	-0.7	-0.19
Starship	-0.41	-0.07	-0.03	0.21	0.76	-0.26	0.38
Starfight	-0.55	-0.09	-0.04	-0.72	-0.18	-0.34	-0.09
Power	-0.69	-0.12	-0.05	0.34	-0.23	0.57	-0.12
Loneliness	-0.15	0.59	0.65	0	0.2	0	-0.4
Love	-0.07	0.73	-0.68	0	0	0	0
Desire	-0.08	0.3	0.33	0	-0.4	0	0.8

×

	Topic A	Topic B	Topic C	Topic D	Topic E
Topic 1	12.48	.	.	.	.
Topic 2	.	9.51	.	.	.
Topic 3	.	.	1.35	.	.
Topic 4	.	.	.	0	.
Topic 5	.	.	.	.	0
Topic 6	.	.	.	.	.
Topic 7	.	.	.	.	.

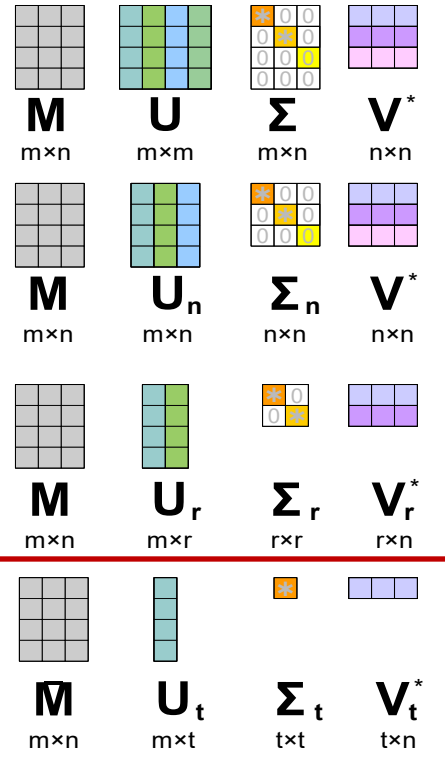
×

	Star Wars	Matrix	Star Trek	Australia	Out of Africa
Topic A	-0.56	-0.59	-0.56	-0.09	-0.09
Topic B	-0.13	0.03	-0.13	0.7	0.7
Topic C	-0.41	0.8	-0.41	-0.09	-0.09
Topic D	-0.71	0	0.71	0	0
Topic E	0	0	0	-0.71	0.71

Word-Document Matrix	Word-Topic Matrix	Topic Importance Matrix	Topic-Document Matrix

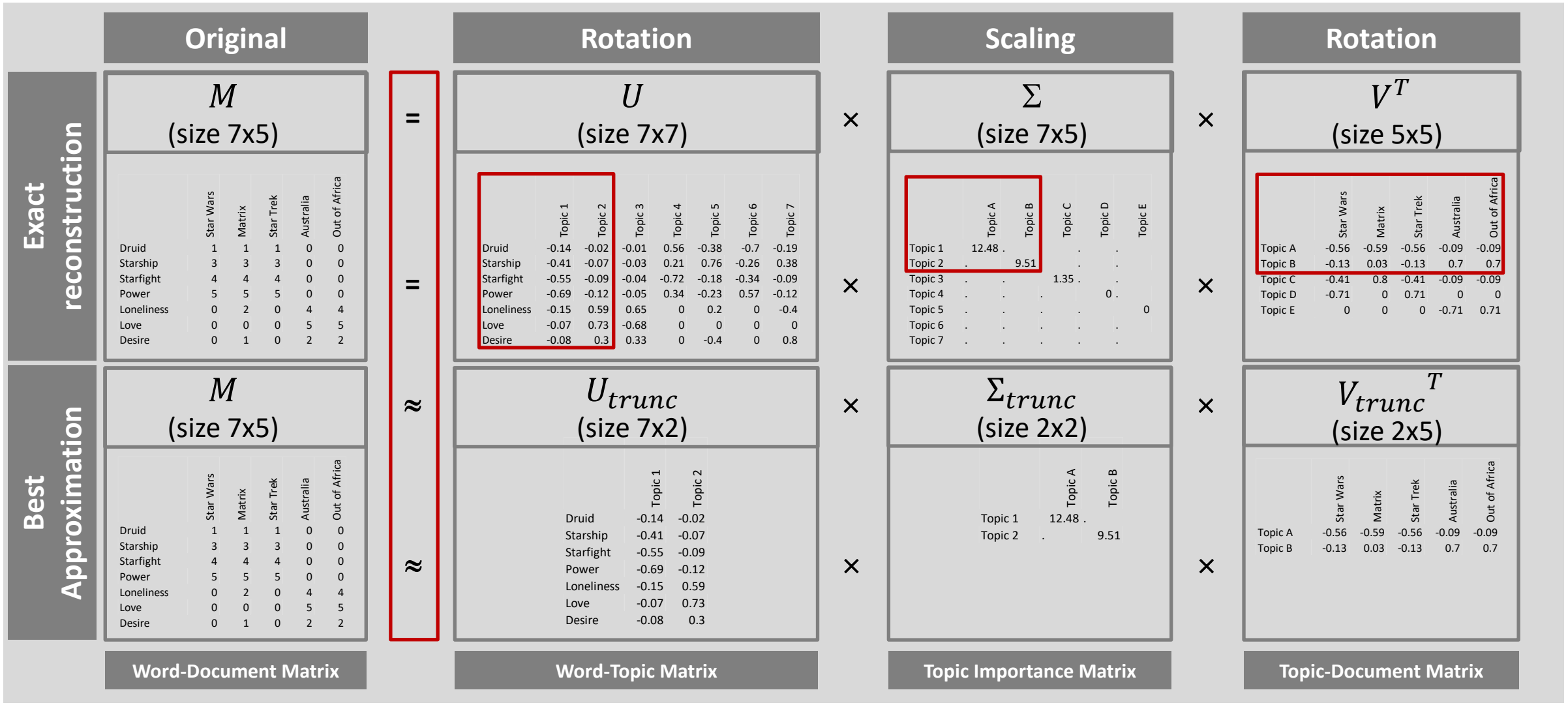
SVD theory & approximation by Truncated SVD

- Analytical solutions for the computation of SVD only exist in 2D, but algorithms allow to compute singular values and vectors with arbitrary precision
  - Full SVD described above
  - Thin SVD: remove columns of U not corresponding to rows of V^T
  - Compact SVD: remove zero singular values and the corresponding columns/rows in U and V^T
  - Truncated SVD: approximate M by keeping only the largest t singular values σ_1, …, σ_t and corresponding columns/rows in U & V^T
    w.r.t. the Frobenius norm, truncated SVD yields the closest approximation of M with rank r by another matrix M' of lower rank t $(t < r)$



Mercurysheet, CC BY-SA 4.0, Wikimedia Commons

SVD toy example continued



	Star Wars	Matrix	Star Trek	Australia	Out of Africa
Druid	1	1	1	0	0
Starship	3	3	3	0	0
Starfight	4	4	4	0	0
Power	5	5	5	0	0
Loneliness	0	2	0	4	4
Love	0	0	0	5	5
Desire	0	1	0	2	2

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7
Druid	-0.14	-0.02	-0.01	0.56	-0.38	-0.7	-0.19
Starship	-0.41	-0.07	-0.03	0.21	0.76	-0.26	0.38
Starfight	-0.55	-0.09	-0.04	-0.72	-0.18	-0.34	-0.09
Power	-0.69	-0.12	-0.05	0.34	-0.23	0.57	-0.12
Loneliness	-0.15	0.59	0.65	0	0.2	0	-0.4
Love	-0.07	0.73	-0.68	0	0	0	0
Desire	-0.08	0.3	0.33	0	-0.4	0	0.8

	Topic A	Topic B	Topic C	Topic D	Topic E
Topic 1	12.48	.	.	.	
Topic 2	.	9.51	.	.	
Topic 3	.	.	1.35	.	
Topic 4	.	.	.	0	.
Topic 5	.	.	.	.	0
Topic 6	.	.	.	.	
Topic 7	.	.	.	.	

	Star Wars	Matrix	Star Trek	Australia	Out of Africa
Topic A	-0.56	-0.59	-0.56	-0.09	-0.09
Topic B	-0.13	0.03	-0.13	0.7	0.7
Topic C	-0.41	0.8	-0.41	-0.09	-0.09
Topic D	-0.71	0	0.71	0	0
Topic E	0	0	0	-0.71	0.71

SVD applied to the word-document matrix

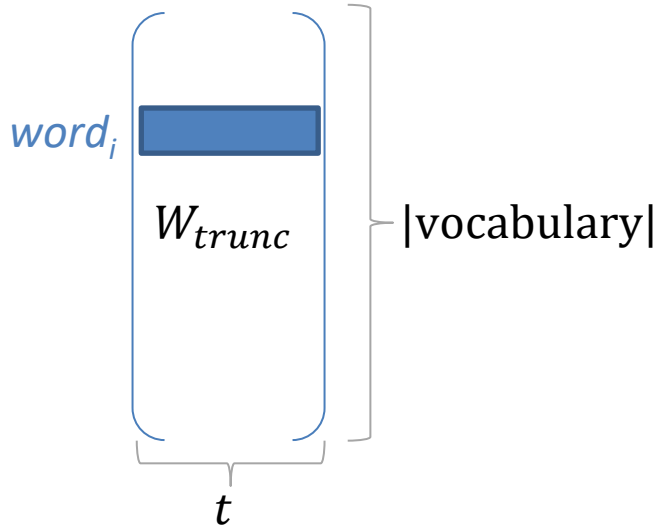
- Every matrix can be decomposed, i.e., $M = U \cdot \Sigma \cdot V^T$
- Orthogonal rotation matrix U with $U^T \cdot U = \mathbb{I}_m$
- Scaling matrix Σ of size m x n with null coefficients, except for non-negative diagonal elements σ_i, whose number corresponds to $rank(M)$
- Orthogonal rotation matrix V^T with $V^T \cdot V = \mathbb{I}_n$

- A word-document matrix can hence be re-written as $M = W \cdot S \cdot D$
- W : each row corresponds to a word in the vocabulary, and the m columns are dimensions of a "latent topic" space
- S : diagonal matrix with singular values, giving the importance of each dimension
- D : each column corresponds to a document in the database, the n rows are the dimensions of a "latent topic" space

Text embeddings from word-doc matrix with LSA

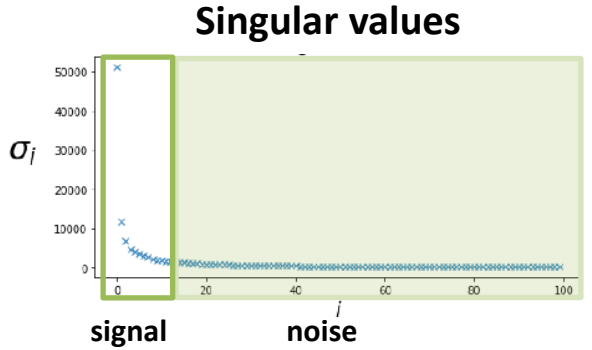
- Decompose the word-document matrix M with truncated SVD using dimensionality t
- Use W_{trunc} of size |vocabulary| x t , to identify low-dimensional embeddings, i.e., extract the i[th] row of W_{trunc} to represent a t–dimensional vector for the i[th] word of the word-document matrix M
  - alternatives used : $W_{trunc} \cdot \sqrt{S_{trunc}}$ or $W_{trunc} \cdot S_{trunc}$

LSA word embedding



- The dimensionality t is selected to keep 80%-90% of the total "energy" of the singular values, i.e.,
  - $\sum_{i=1}^{t} \sigma_i = \alpha \cdot \sum_{i=1}^{N} \sigma_i$ with $\alpha \in [0.8, 0.9]$
  - Heuristics from past: $50 < t < 1000$, frequently $t \approx 300$

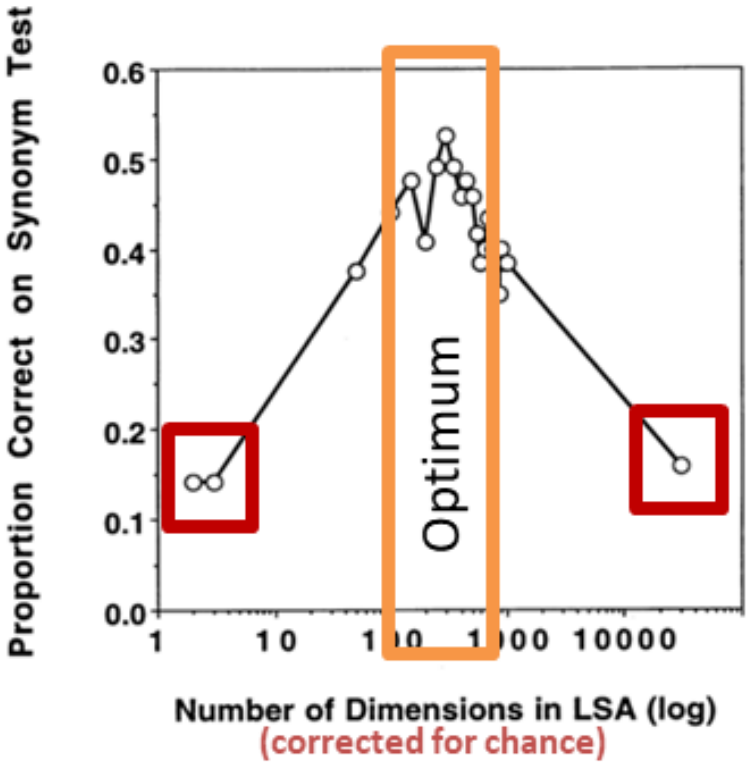
Singular values



Source: O. Levy et al. (2015), "Improving Distributional Similarity with Lessons Learned from Word Embeddings"

Applications of LSA-based text embeddings

- Identification of synonyms with LSA-based embeddings
- Dataset and task
  - TOEFL 80 x {target word, four candidate words}
  - select best synonym of the target word
- Method
  - LSA trained on newswire, encyclopedia, textbooks. i.e., 30k documents (first paragraphs with ca. 500 characters, total of 4.5 million words with vocabulary size 60k)
  - Synonyms identified by cosine similarity of LSA embeddings
- Results
  - 53% correct similar to students from non-English speaking countries applying to USA colleges, optimal dimensionality: ~300 (without SVD truncation: 16% correct!)



Source: Landauer, T. K., Foltz, P. W., & Laham, D. (1998). "Introduction to Latent Semantic Analysis", *Discourse Processes*, 25, p. 259-284.

Applications and limitations of LSA

- Additional applications include
  - Predict query-document topic similarity judgments
  - Simulate human choices on subject-matter multiple choice tests
  - Predict text coherence and resulting comprehension, subjective ratings of essays

- LSA enables the use of word co-occurrences to capture latent semantic associations of terms and thus addresses the first problem by detecting synonyms.
- However,
  - Dimensions of the LSA embedding space do not always have good interpretations
  - LSA has a significant computational cost for large corpora (due to SVD) which can be reduced by building the LSA embeddings from a document subset
  - LSA cannot express negations nor enforce Boolean conditions

Source: Manning, Raghavan, and Schütze (2008), *Introduction to Information Retrieval*, Cambridge University Press.

Extracting meaning from word co-occurrences

- Observations show that ratios of word-word co-occurrence probabilities have the potential to encode semantic meaning

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

solid related to *ice* , not *steam*

water related to *ice* & *steam*

fashion unrelated to *ice* & *steam*

- Using vectors for words (w_i, w_j) and context (w_k) the ratio can be expressed as

$$\frac{P_{ik}}{P_{jk}} = \frac{X_{ik}}{X_i} \cdot \frac{X_j}{X_{jk}} = F(w_i, w_j, w_k)$$

Source: J. Pennington, R. Socher, C. D. Manning, "GloVe: Global Vectors for Word Representation", EMNLP 2014

Global vectors (GloVe) for word representations

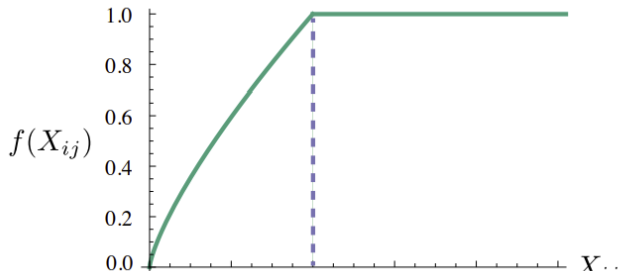
- GloVe is a matrix-based distributional semantic model trained in an unsupervised manner leveraging ratios of word co-occurrence probabilities
- The training uses
  - A log bi-linear regression objective function, i.e., word vectors align with the logarithm of the words co-occurrence via dot product
  - 5 large corpora extracting the 400k most frequent words; the co-occurrence matrix is constructed with a weighting factor $1/d$ reducing influence for word pairs d words apart in the total counts

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

X_{ij} : co-occurrence of word i and word j

w_i : embedding for word i

b_i, $\tilde{b}_j$: bias terms i



$f(X_{ij})$: weighting function that assigns lower weights to rare and frequent co-occurrences

Source: J. Pennington, R. Socher, C. D. Manning, "GloVe: Global Vectors for Word Representation", EMNLP 2014

GloVe model – Derivation of objective function

Learn word vector based on ratios of co-occurrence probabilities of words & context

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Use vector differences to express differences of co-occurrence probabilities

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Apply the dot-product to convert vectors (LHS) to scalars (RHS)

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \quad \textbf{(A)}$$

Determine a mapping of $\mathbf{R} \rightarrow \mathbf{R}_+$ which conserves structures (homomorphism)

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} \quad \textbf{(B)}$$

Combine the equation A and equation B

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}$$

Solve equation A with considering structural equation using $F = \exp$

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i)$$

Force symmetry between word and context

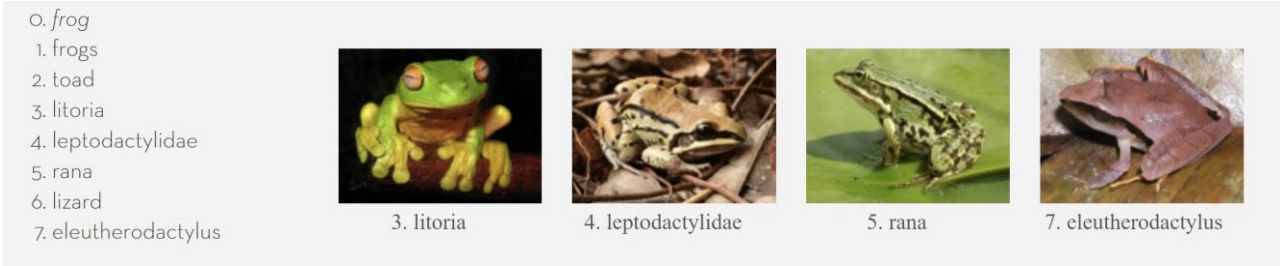
$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

Source: J. Pennington, R. Socher, C. D. Manning, "GloVe: Global Vectors for Word Representation", EMNLP 2014

GloVe - identifying word analogies & similarities

- Euclidean distance or cosine similarity between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words.

- Word analogies answer the question "a is to b as c is to ?" by finding the word d whose embedding w_d is closest to w_b - w_a + w_c according to the cosine similarity



Source: J. Pennington, R. Socher, C. D. Manning, "GloVe: Global Vectors for Word Representation", Stanford Homepage

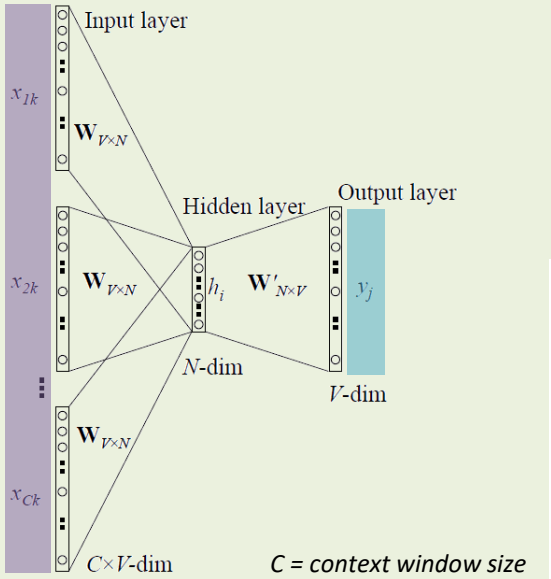
Conclusion

- The contextual understanding of LSA and GloVe builds upon of co-occurrence count statistics and leverages matrix factorization resulting in dense word embedding of dimensionality of a few 100's
- LSA efficiently leverages statistical information from the word-document matrix, but does relatively poorly on the word analogy task
- GloVe trained on global word-word co-occurrence counts makes efficient use of statistics and excels in word similarity and semantic and syntactic word analogy tasks
- While both LSA and GloVe use the Bag-of-Word approach, GloVe accounts for word pair distance when constructing the word co-occurrence matrix
- The unsupervised learning paradigm used word representations from large-scale corpora without requiring labeled data, making it scalable and applicable to a wide range of languages and domains

4. WORD EMBEDDINGS WITH NEURAL NETWORKS (WORD2VEC & FASTTEXT)

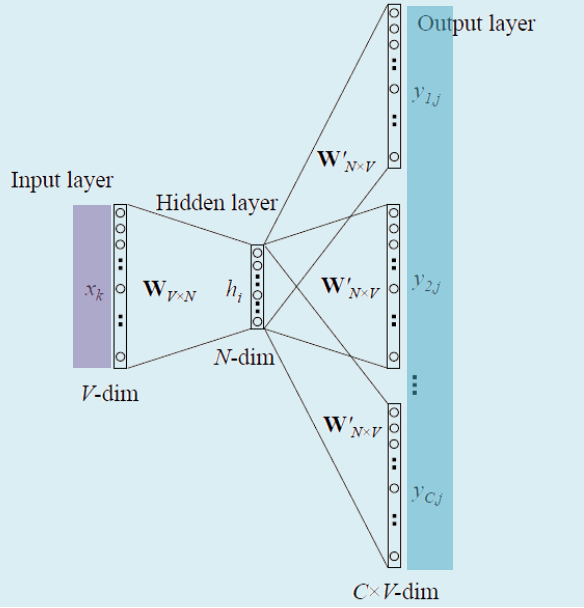
Computing word embeddings with neural nets

- Training a neural network language model, was observed to also yield learned word representations, which can be used for other, potentially unrelated, tasks
- Thus, novel model architectures for efficiently computing continuous vector representations of words from – at that time – very large data sets were developed



CBOW (context window size C = 2)
Focus: Predict one center word

Skip-gram (context window size C = 2)
Focus: Predict several context words



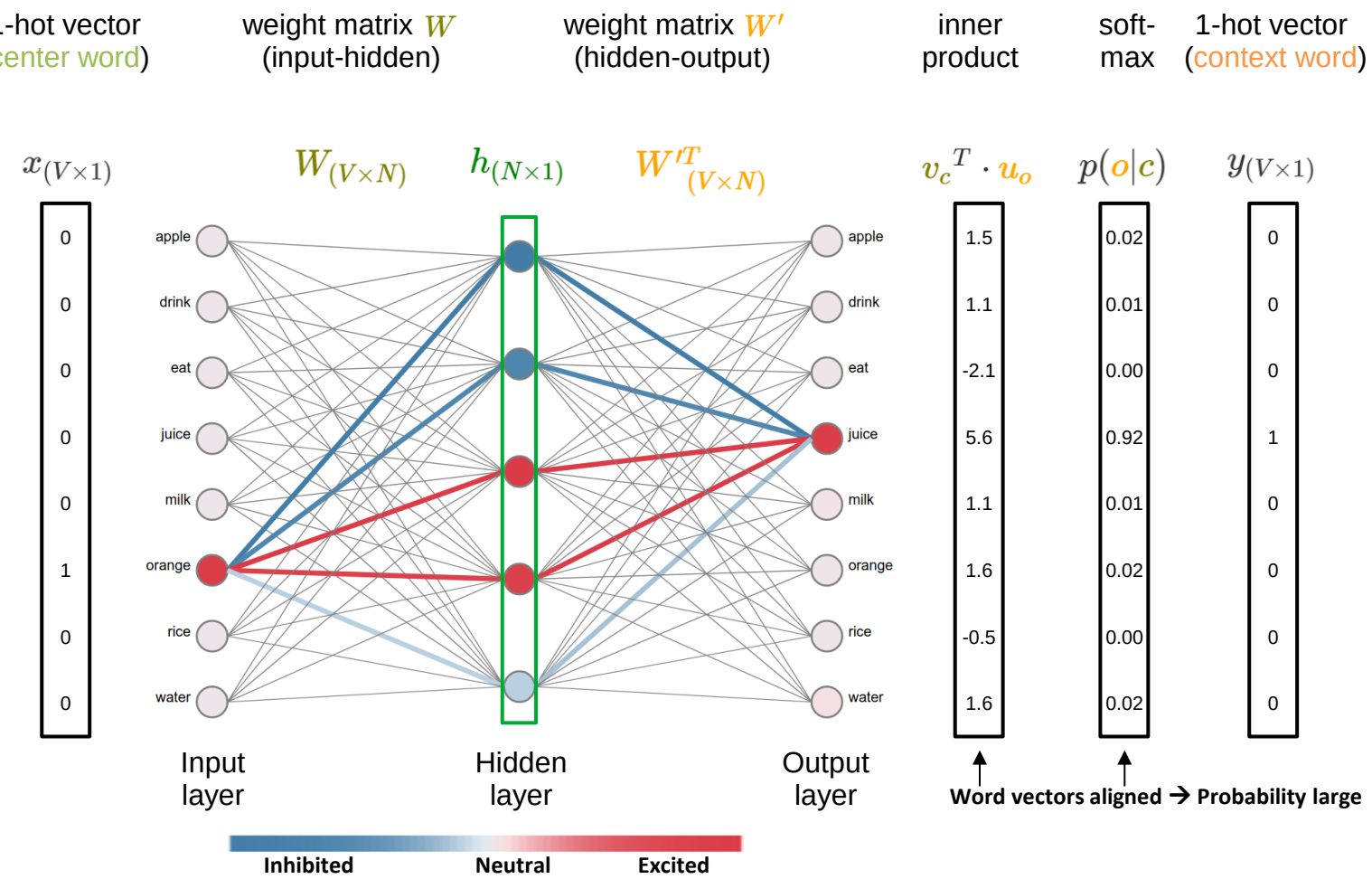
Sources: T. Mikolov et al. (2013) "Linguistic Regularities in Continuous Space Word Representations", "Efficient Estimation of Word Representations in Vector Space"

Computing word embeddings with word2vec

- The superficial classification task of predicting the most likely upcoming words is used to resolve the underlying objective, i.e., to learn dense word embeddings that reflect semantic relationships
- The architecture used is a multi-layer perceptron with three layers (input, hidden and output layer), the hidden layer uses a linear activation function!
- The dimensionality of word embeddings is defined by the size of the hidden layer and typically between 200 and 500
- Dense word embeddings are extracted from connections between input and hidden layer, connections between hidden and output layer represent context embeddings
- word2vec trains the neural network (much) more efficiently than Truncated SVD

Sources: T. Mikolov et al. (2013) "Linguistic Regularities in Continuous Space Word Representations", "Efficient Estimation of Word Representations in Vector Space"

Detailed view on the word2vec core network



- **Notes**

- the **probability calculation** is **based on embedding similarity** of center v_c and context word u_o expressed as **inner product** $v_c{}^T \cdot u_o$

Note $v_{w_i} = W^T \cdot x_{w_i} = h_{w_i}$

$W'^T \cdot h_{w_i} = u_{w_i}$

- The dot product "similarity" is **translated into probabilities** by applying the **softmax function** over all output words in the output layer

$$p(o|c) = \frac{e^{v_c{}^T \cdot u_o}}{\sum_{w \in V} e^{v_c{}^T \cdot u_w}}$$

Source: Rong, X. (2014) "word2vec Parameter Learning Explained."

Note: Mikolov et al (2013) initially used softmax function, whose denominator becomes computationally very expensive for large vocabularies!

Computing word embeddings with word2vec

- Both word2vec models, i.e., CBOW and Skip-Gram learn two embeddings for each word via W and W'. Word "i" is represented by the ith row of (a) W or (b) $W + W'$
- CBOW (Continuous Bag-of-words) predicts one center word
  - "Continuous" refers to the non-discrete representation used by the model
  - Context words on the input side are embedded into the hidden layer
  - Embedded context words are averaged before projecting them to the output layer
- SG (Skip-gram) predicts several context words
  - Word pairs in training are not all proper bi-grams, but often involve "skipping over" some words
  - Each context word on the output is treated equally in probability computations, the model only has one output layer (Note: on slide 35 , right panel, there is only one W')
  - Skip-gram takes much longer to train! It produces many more training samples than CBOW

Source: https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes01-wordvecs1.pdf

Training the Skip-gram word2vec model

- The Skip-gram model maximizes the probability with the context set of w, $C(w)$

$$argmax_{\Theta} \prod_{w \,\epsilon\, Text} \left[\prod_{c \,\epsilon\, C(w)} p(c|w,\theta) \right]$$

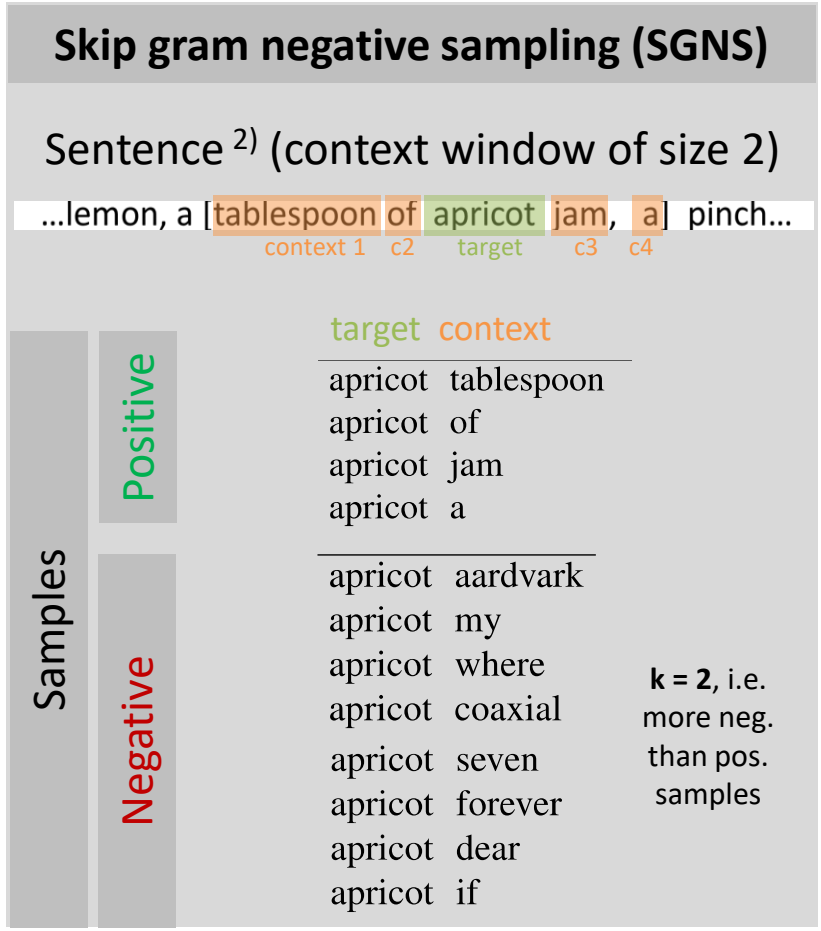
- Theoretical solution is found via gradient descent, i.e.,
  - Start from random weight matrices and consider the difference between desired and actual outputs over all training pairs
  - Compute the gradient for each parameter (weight) and back-propagate it to lower layers
  - Adjust coefficients of W', then W, s.t. difference desired vs. actual predictions is reduced

- The basic Skip-gram formulation defines $p(c|w,\theta)$ using the softmax function, which is computationally expensive, due to summation in the denominator

Sources: T. Mikolov et al. (2013) "Distributed Representations of Words and Phrases", Y. Goldberg & O. Levy (2014) "word2vec Explained"

Training the Skip-gram word2vec model

- The theoretical solution described above has further limitations
  - It is impossible to perform gradient descent for all data at once (approximation: do it for each data point individually)
  - Training converges towards a degenerate solution, i.e., $w = c$ and $w^T c = K$, where $K \approx 40$ or larger
- We resolve this issue by adding for each positive sample, negative samples, s.t. small fraction of model weights only are updated
  - Draw J random words w from the data, suppress influence of frequent words via $freq(w)^{3/4}$
  - Perform gradient descent for the data with positive and negative samples



Source: T. Mikolov et al. (2013) "Distributed Representations of Words and Phrases", Y. Goldberg & O. Levy (2014) "word2vec Explained"

Skip-gram model with negative sampling (SGNS)

- Given positive and negative instances and initial embedding vectors, the training goal is to adjust word vectors with Maximum Likelihood Estimation
  - Maximize similarity drawn from the positive data for the target word, context word pairs w, c_{pos}
  - Minimize similarity drawn from the negative data for the target word, context word pairs w, c_{neg}
- In other words, maximize dot product of the word with the actual context word and minimize dot products of the word with the k negative sampled non-neighbor words

Cross-Entropy loss for one word/context pair and k noise words

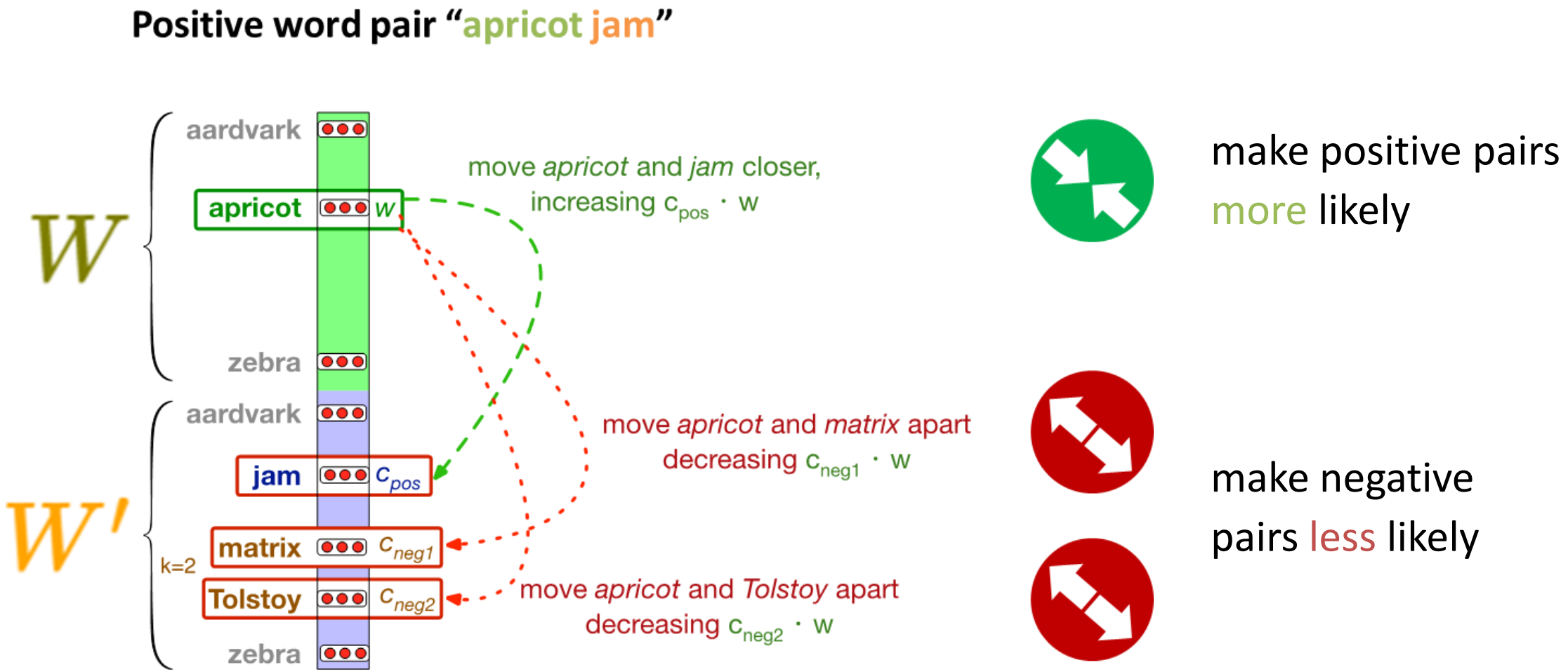
$$
\begin{aligned}
L_{\text{CE}} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log \sigma(c_{pos}\cdot w) + \sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

$P(+|w,c)$ **... probability that c is a context word**

$\sigma(x)=\frac{1}{1+\exp(-x)}$ **... transforms similarity to probability**

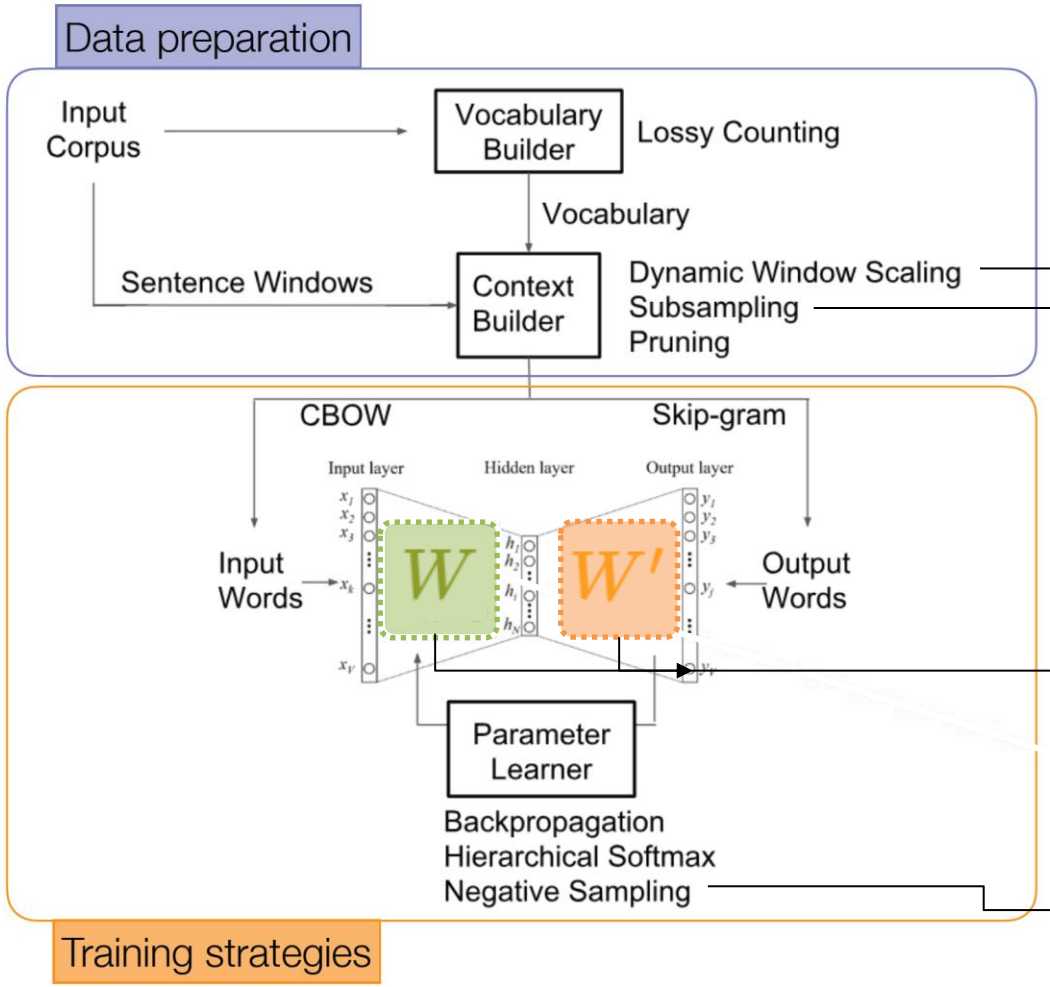
Source: Jurafsky, D. & Martin, J.H. (2008), "Speech and Language Processing", Chapter 6

SGNS - One step of stochastic gradient descent



Source: Jurafsky, D. & Martin, J.H. (2008), "Speech and Language Processing", Chapter 6

Word2vec – Design & tuning considerations



Dynamic Window Scaling
When the context window is moved, a random window size in the range [1, *C*] is selected to give more weight to context closer to target word (i.e., each context position has a proportional probability of being included)

Subsampling
Words with low information content and high frequency are subsampled (e.g., “the”)

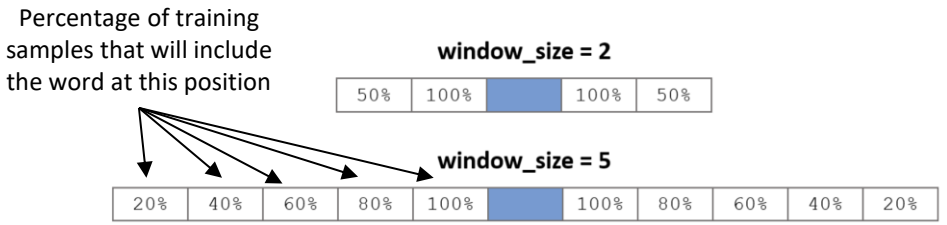
Final Product
We are no longer interested in the predictive aspects of the network, but only in the W and W' embeddings !

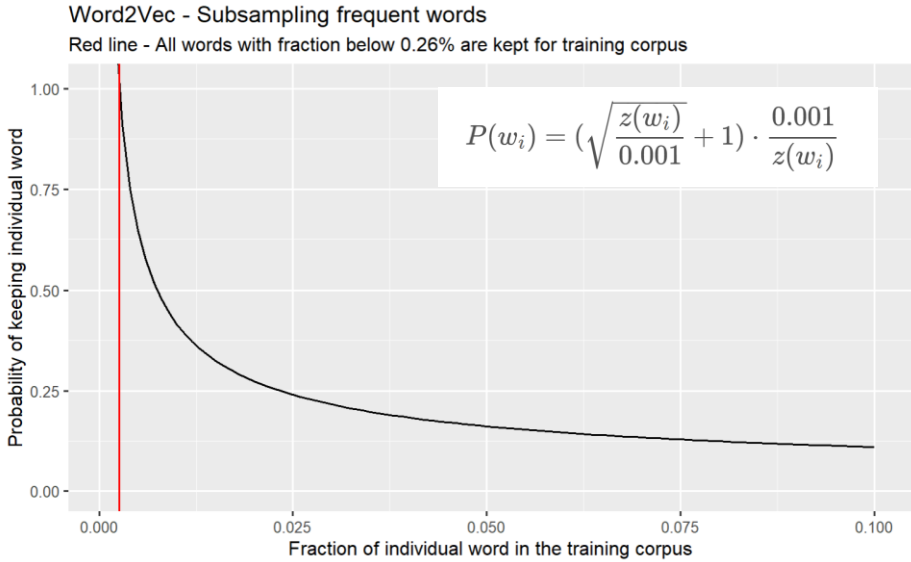
Tuning 3
Select 5-20 words for smaller data sets and 2-5 words for large data sets (parameter k)

Sources: Rong, X. (2014) "word2vec Parameter Learning Explained“ , McCormick, Ch. “The inner workings of word2vec”,

Word2vec – Design & tuning considerations

- Dynamic window scaling
  - Given a maximum context window of size C, a random window size in the range [1, C] is selected, each time the context window is moved to the next word in a sentence → more weight given to context closer to target word
- Subsampling frequent and irrelevant words
  - Frequent words are only kept in the training vocabulary with a certain probability; words which represent less than 0.26% of the total words will never be subsampled
  - Note: Do not mix up subsampling of the vocabulary with negative sampling adjustment based on $freq(w)^{3/4}$

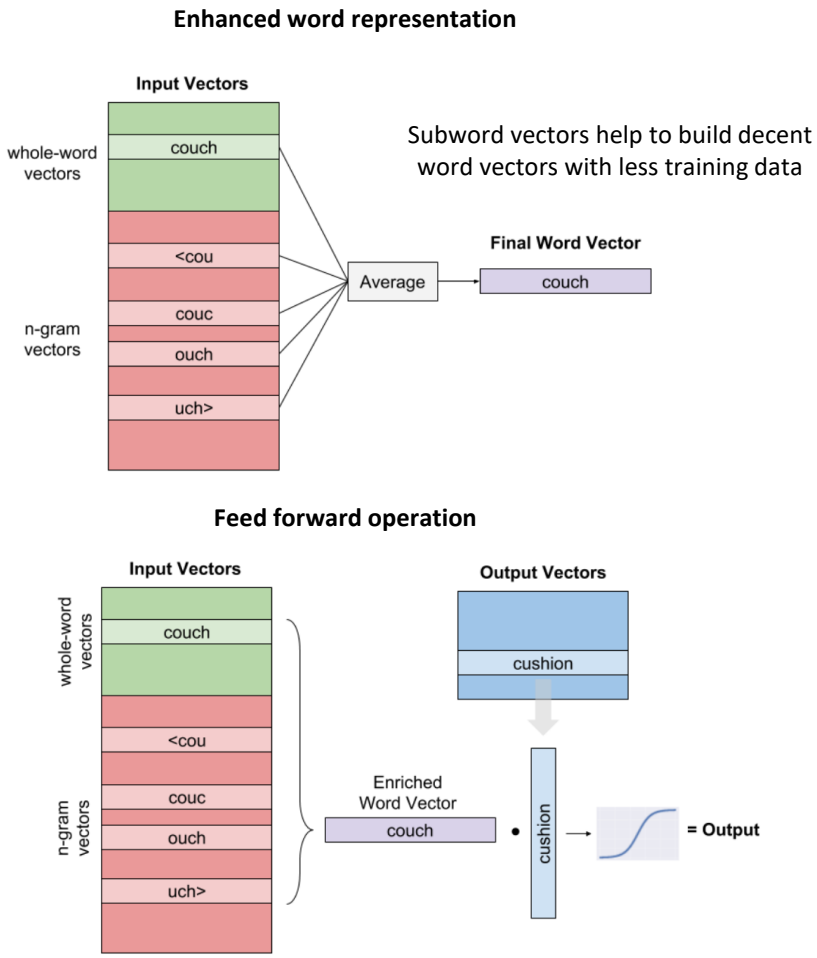




Source: McCormick, Ch. Word2Vec Tutorial Part 2 - Negative Sampling · Chris McCormick (mccormickml.com)

fastText – Solution for handling rare words

- fastText extends word2vec to learn from less data and represent rare words and typos, without major architectural changes
- fastText enhances vocabulary by adding a collection of character n-grams of which a word is composed; by default, n takes on values 3, …, 6
- At inference, rare words not present in the training data are represented by the average of underlying n-grams
- fastText needs to learn substantially more token vectors, hence more time and memory… and vectors for common words are not better necessarily



Source: fastText: Bojanowski, P. et al. “Enriching Word Vectors with Subword Information”, Transactions of the ACL, 2017, C. McCormick “The Inner Workings of word2vec”

Conclusion

- The presented methods using co-occurrence statistics (e.g., LSA & GloVe) or neural prediction models (e.g., word2vec & fastText) all use large corpora's word distributions, taking into account local (e.g., LSA) or global context windows and considering distance-dependence (e.g., GloVe)
- All of these methods yield static or non-contextual word embeddings, i.e., a single vector for ambiguous words like "bank", "bat" or "apple"
- Word embeddings revolutionized NLP applications, like information retrieval, similarity analysis, word clustering, sentiment analysis, topic classification
- Word embeddings can be used to construct sentence & document embeddings