Non-Standard Datenbanken und Data Mining

From Clustering to Embedding

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Übersicht

- Semistrukturierte Datenbanken (JSON, XML) und Volltextsuche
- Information Retrieval
- Mehrdimensionale Indexstrukturen
- Cluster-Bildung
- Einbettungstechniken
- First-n-, Top-k-, und Skyline-Anfragen
- Probabilistische Datenbanken, Anfragebeantwortung, Top-k-Anfragen und Open-World-Annahme
- Probabilistische Modellierung, Bayes-Netze, Anfragebeantwortungsalgorithmen, Lernverfahren,
- Temporale Datenbanken und das relationale Modell,
- Probabilistische Temporale Datenbanken
- SQL: neue Entwicklungen (z.B. JSON-Strukturen und Arrays), Zeitreihen (z.B. TimeScaleDB)
- Stromdatenbanken, Prinzipien der Fenster-orientierten inkrementellen Verarbeitung
- Approximationstechniken für Stromdatenverarbeitung, Stream-Mining
- Probabilistische raum-zeitliche Datenbanken und Stromdatenverarbeitungsssysteme: Anfragen und Indexstrukturen, Raum-zeitliches Data Mining, Probabilistische Skylines
- Von NoSQL- zu NewSQL-Datenbanken, CAP-Theorem, Blockchain-Datenbanken

Word-Word Associations in Document Retrieval

Recap bag-of-words approaches
- LSI: Documents as vectors, dimension reduction

Words are not independent of each other
- Word similarity measures
- Extend query with similar words automatically
- Extend query with most frequent followers/predecessors
- Insert words in anticipated gaps in a string query

Need to represent some aspects of word semantics

Approaches for Representing Word Semantics

Beyond bags of words

Distributional Semantics (*Count*)

- Used since the 90's
- Sparse word-context PMI/PPMI matrix
- Decomposed with SVD

Word Embeddings (*Predict*)

- Inspired by deep learning
- `word2vec` *(Mikolov et al., 2013)*
- GloVe *(Pennington et al., 2014)*

Underlying Theory: **The Distributional Hypothesis** *(Harris, '54; Firth, '57)*

"Similar words occur in similar contexts"

https://www.tensorflow.org/tutorials/word2vec

https://nlp.stanford.edu/projects/glove
/

References

- Harris 54

  Harris, Zellig. Distributional structure. *Word* 10(23). 146–162. **1954**.

- Micholov et al. 13

  Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13). **2013**.

- Firth 57

  Firth, John R. A synopsis of linguistic theory 1930–1955. In *Studies in linguistic analysis*, 1–32. Oxford: Blackwell. **1957**.

- Pennington et al. 14

  Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. **2014**.

Point(wise) Mutual Information: PMI

- Measure of association used in information theory and statistics

$$\operatorname{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}$$

- Positive PMI: PPMI(x, y) = max(pmi(x, y), 0)
- Quantifies the discrepancy between the probability of their coincidence given their joint distribution and their individual distributions, assuming independence
- Finding collocations and associations between words
- Countings of occurrences and co-occurrences of words in a text corpus can be used to approximate the probabilities p(x) or p(y) and p(x,y) respectively

[Wikipedia]

PMI – Example

word 1	word 2	count word 1	count word 2	count of co-occurrences	PMI
puerto	rico	1938	1311	1159	10.0349081703
hong	kong	2438	2694	2205	9.72831972408
los	angeles	3501	2808	2791	9.56067615065
carbon	dioxide	4265	1353	1032	9.09852946116
prize	laureate	5131	1676	1210	8.85870710982
san	francisco	5237	2477	1779	8.83305176711
nobel	prize	4098	5131	2498	8.68948811416
ice	hockey	5607	3002	1933	8.6555759741
star	trek	8264	1594	1489	8.63974676575
car	driver	5578	2749	1384	8.41470768304
it	the	283891	3293296	3347	-1.72037278119
are	of	234458	1761436	1019	-2.09254205335
this	the	199882	3293296	1211	-2.38612756961
is	of	565679	1761436	1562	-2.54614706831
and	of	1375396	1761436	2949	-2.79911817902
a	and	984442	1375396	1457	-2.92239510038
in	and	1187652	1375396	1537	-3.05660070757
to	and	1025659	1375396	1286	-3.08825363041
to	in	1025659	1187652	1066	-3.12911348956
of	and	1761436	1375396	1190	-3.70663100173

- Counts of pairs of words getting the most and the least PMI scores in the first 50 millions of words in Wikipedia (dump of October 2015)
- Filtering by 1,000 or more co-occurrences.
- The frequency of each count can be obtained by dividing its value by 50,000,952. (Note: natural log is used to calculate the PMI values in this example, instead of log base 2)

[Wikipedia]

Applications of PMI Data

- Extend query with most frequent followers/predecessors
- Insert words in anticipated gaps in a string query

Church, K.W., Hanks, P.: Word association norms mutual information, and lexicography. Comput. Linguist. 1(1), 22–29, **1990**

PMI – Co-occurrence Matrix

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

PPMI(w,context)

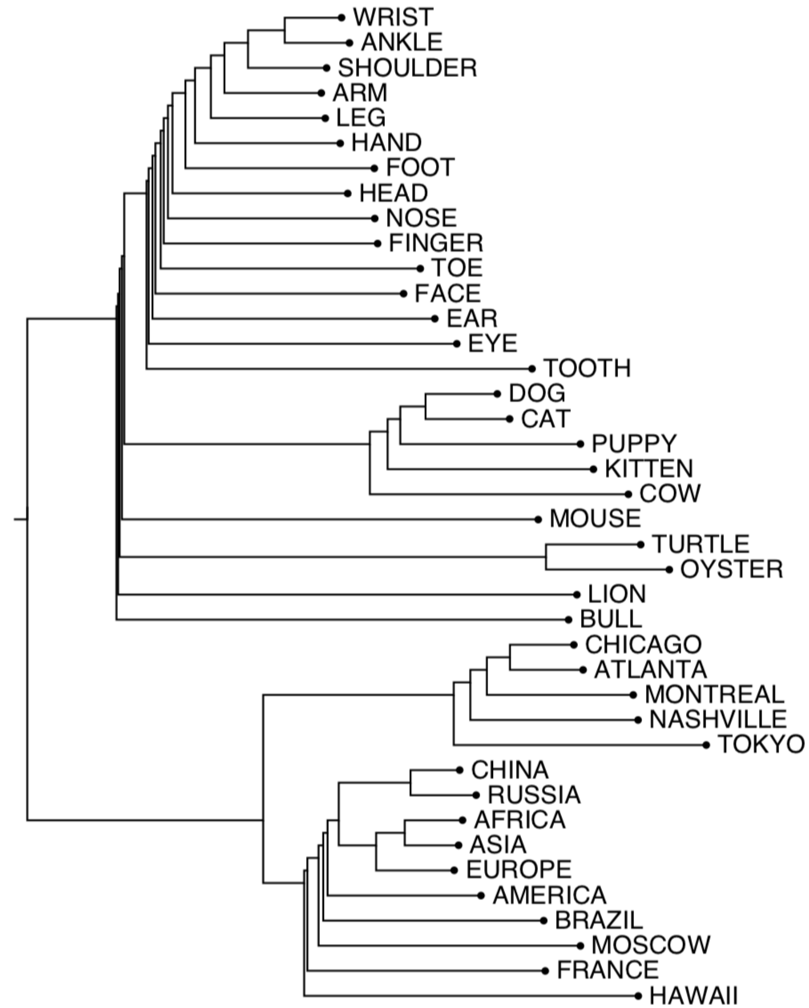
	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

1. Clustering Approach to Word Semantics

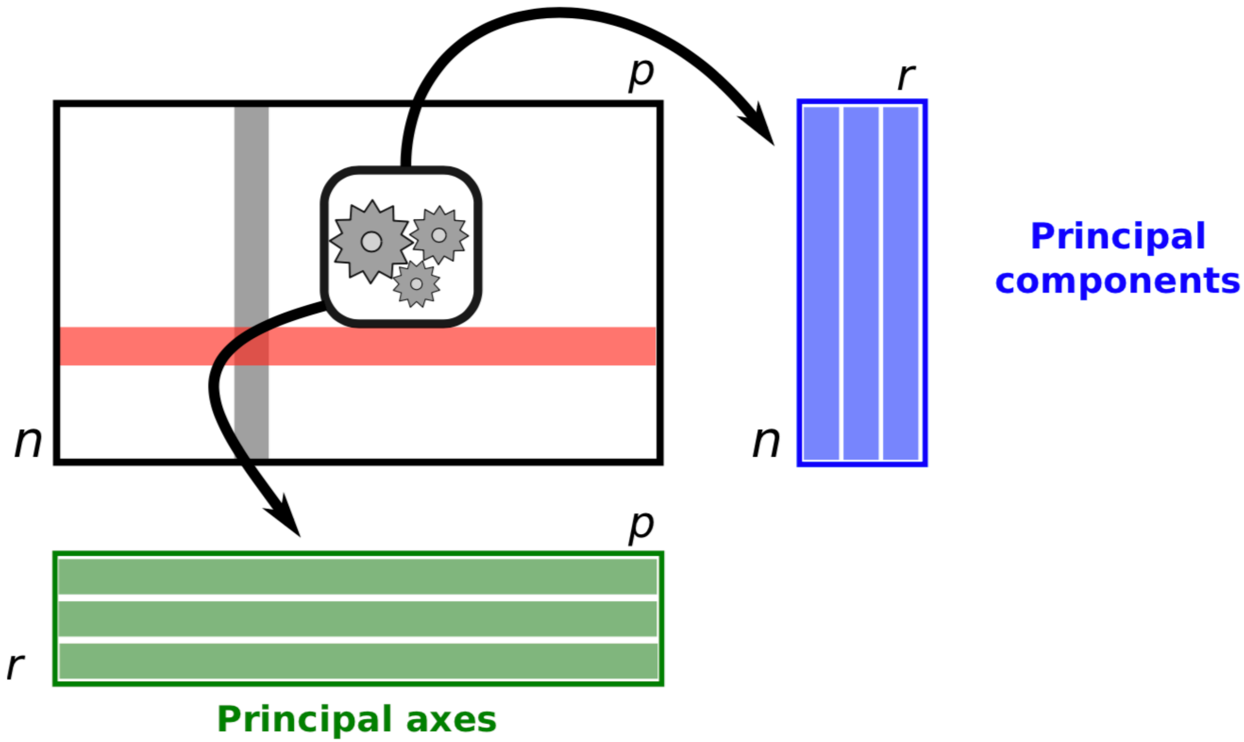
Clustering vectors to visualize similarity in co-occurrence matrices (Rohde et al. 2005)

Use whatever clustering algorithm you prefer to determine "related" words

Application:
Extend query with related words automatically



Apply SVD-based Dimension Reduction



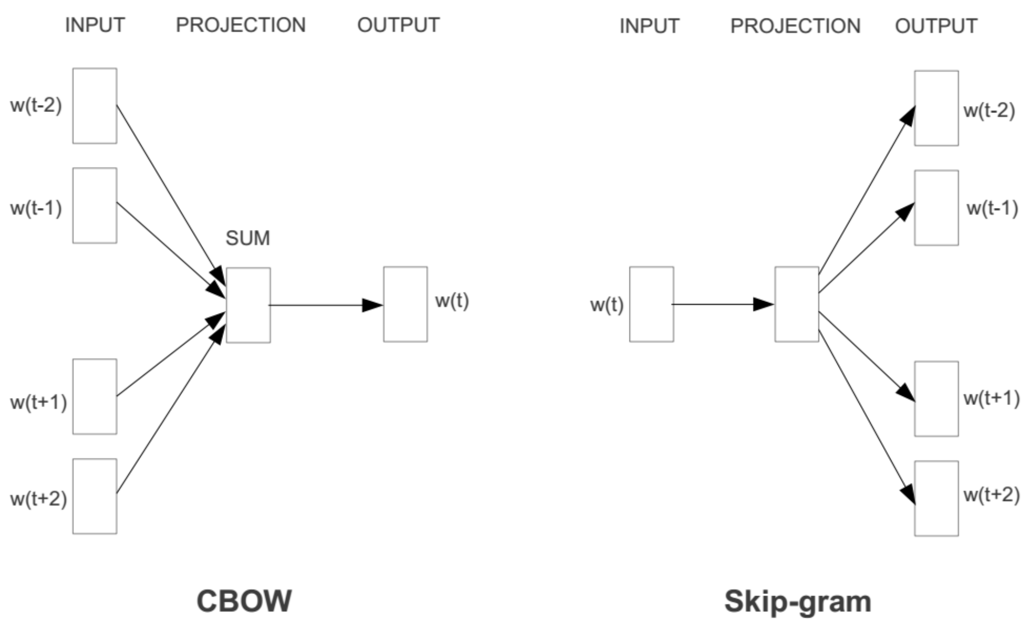
- principal components: Word context vectors
- principal axes: Number of clusters

2. Embedding Approaches to Word Semantics

- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key idea: Predict surrounding words of every word

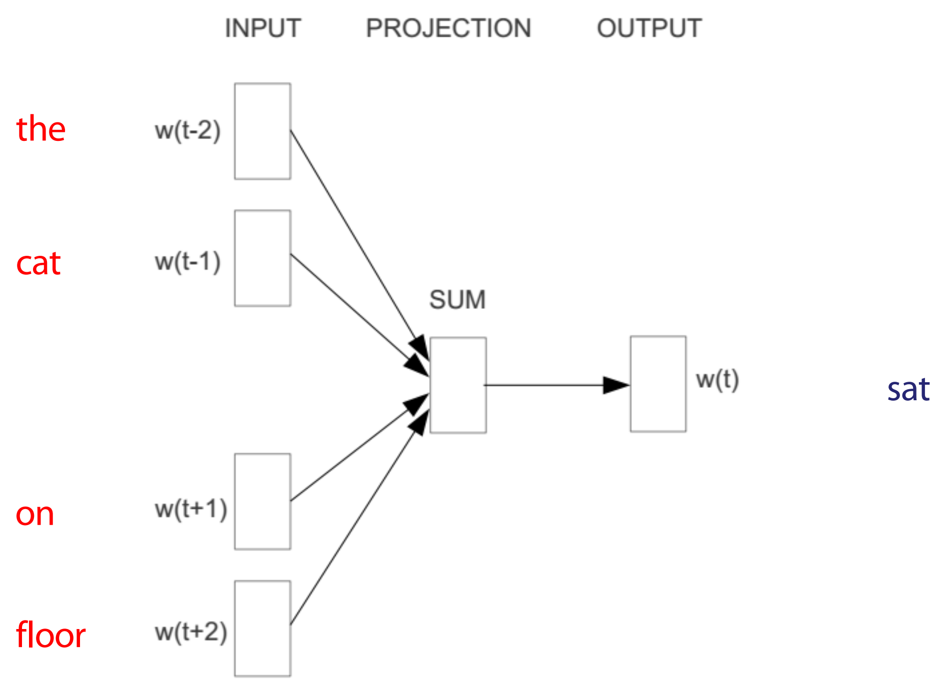
Represent the meaning of **words** – word2vec

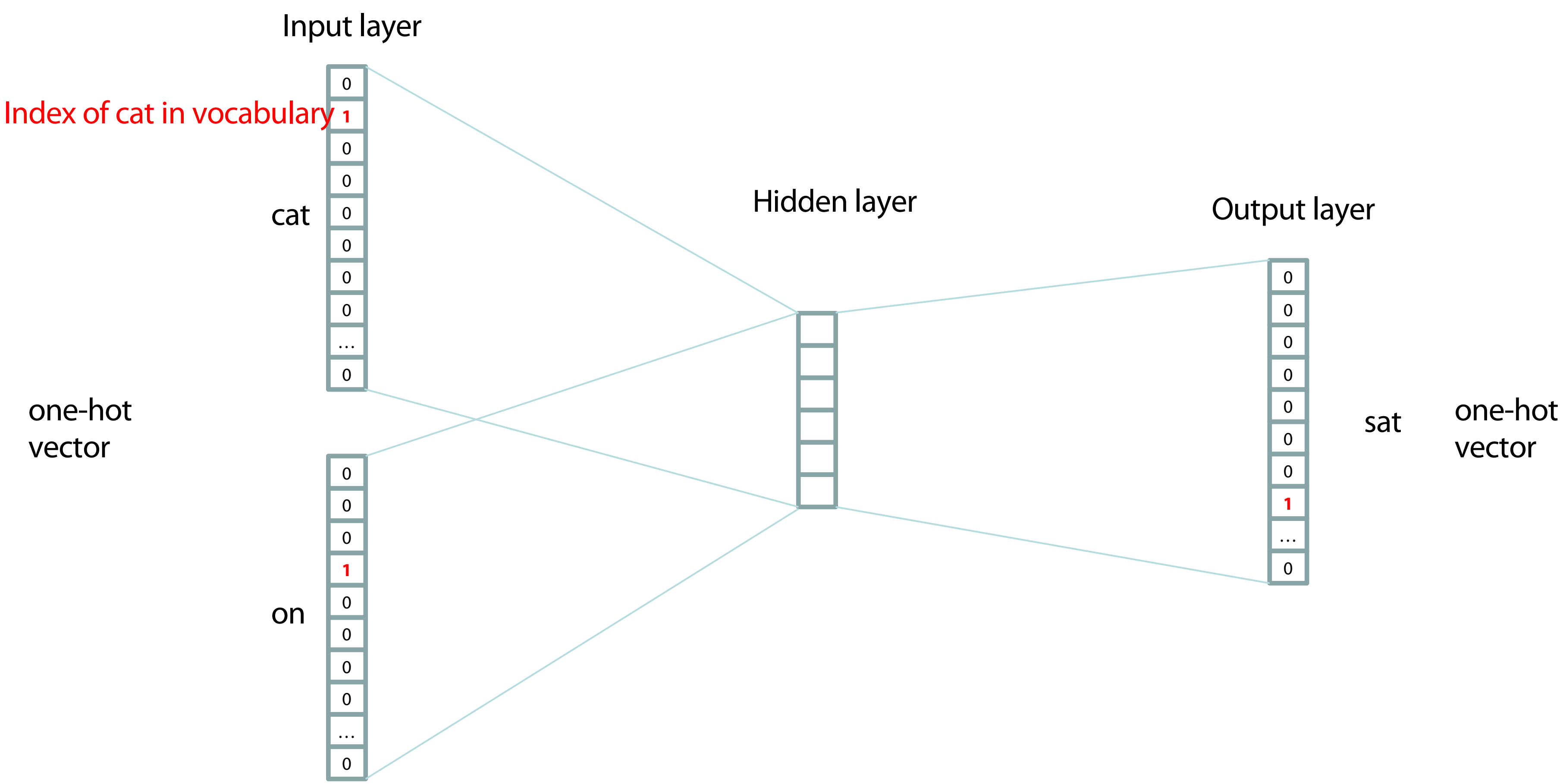
- 2 basic structural models:
  - Continuous Bag of Words (CBOW): use a window of words to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.

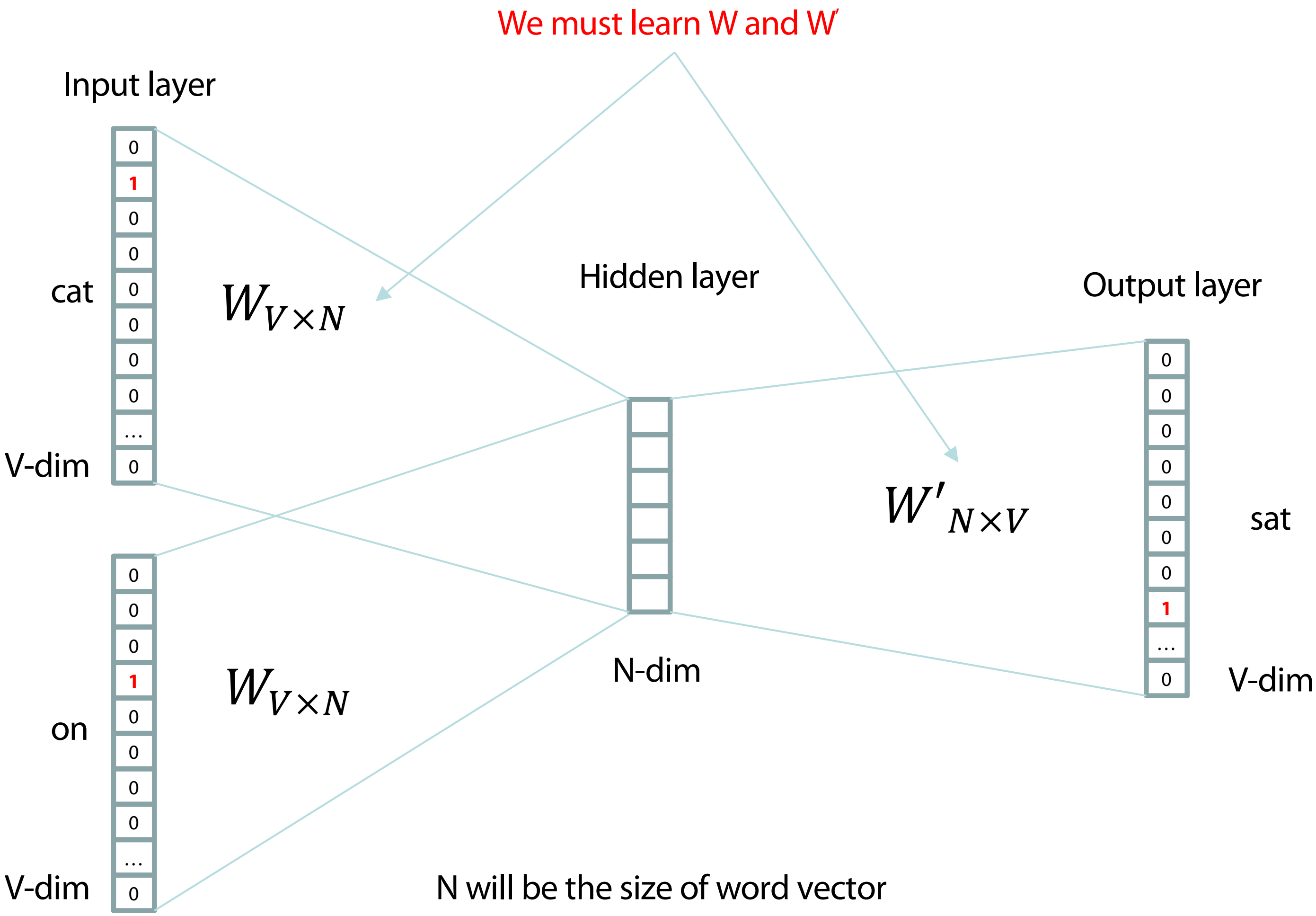


Word2vec – Continuous Bag of Word

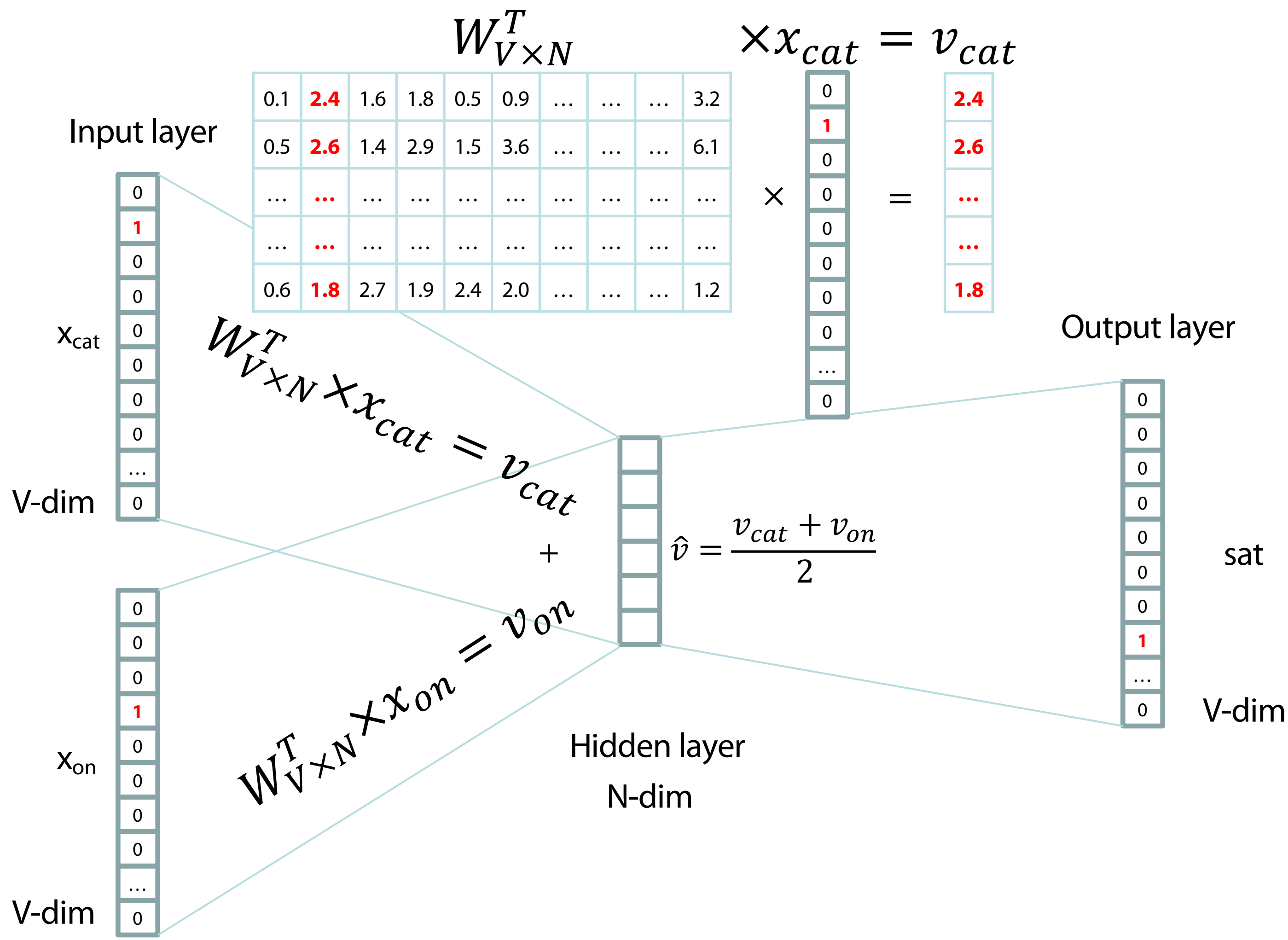
- E.g. “The cat <sat> on floor”
  - Window size = 2

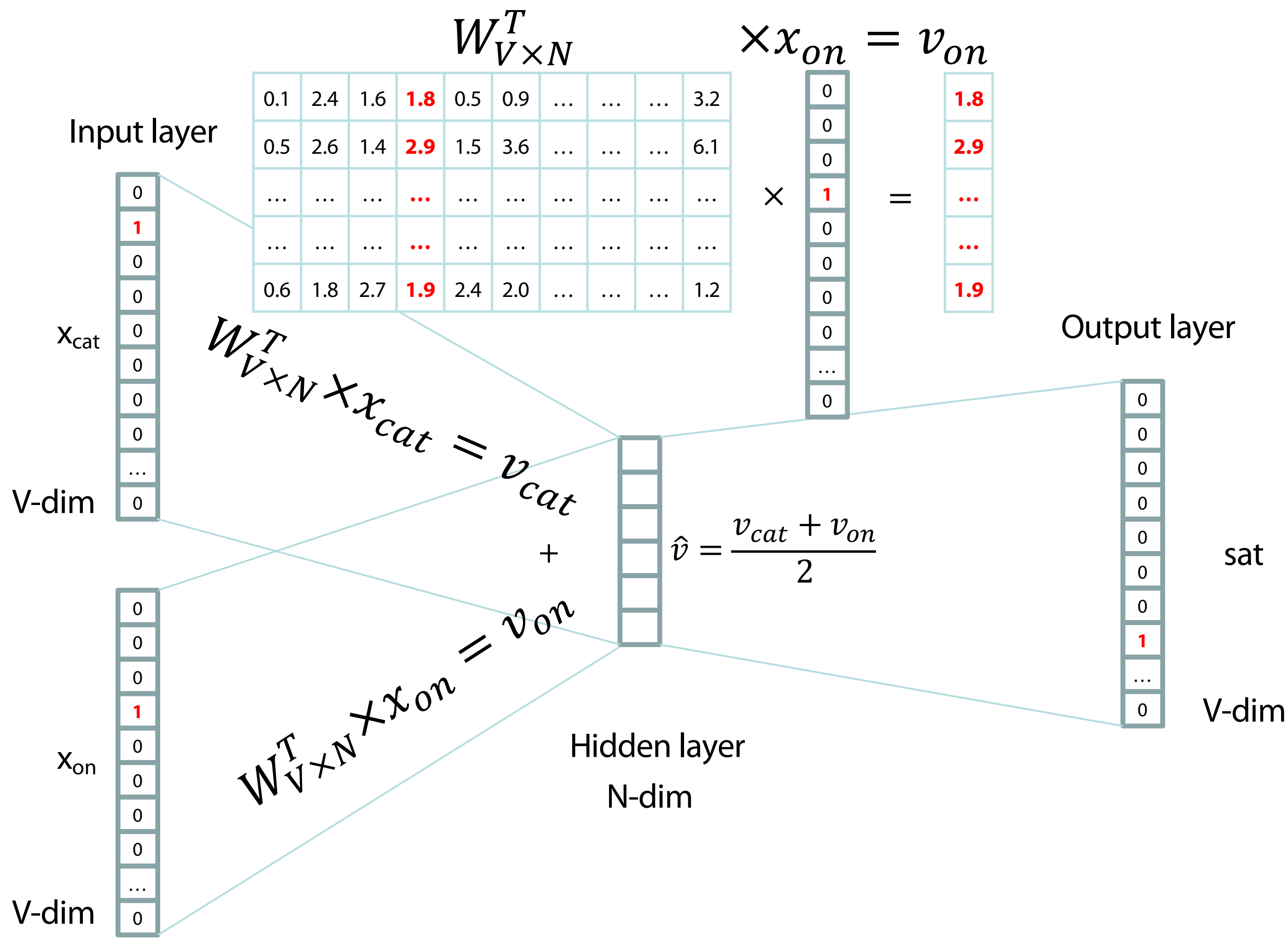



Input layer
Index of cat in vocabulary
cat
one-hot
vector
on
Hidden layer
Output layer
sat
one-hot
vector

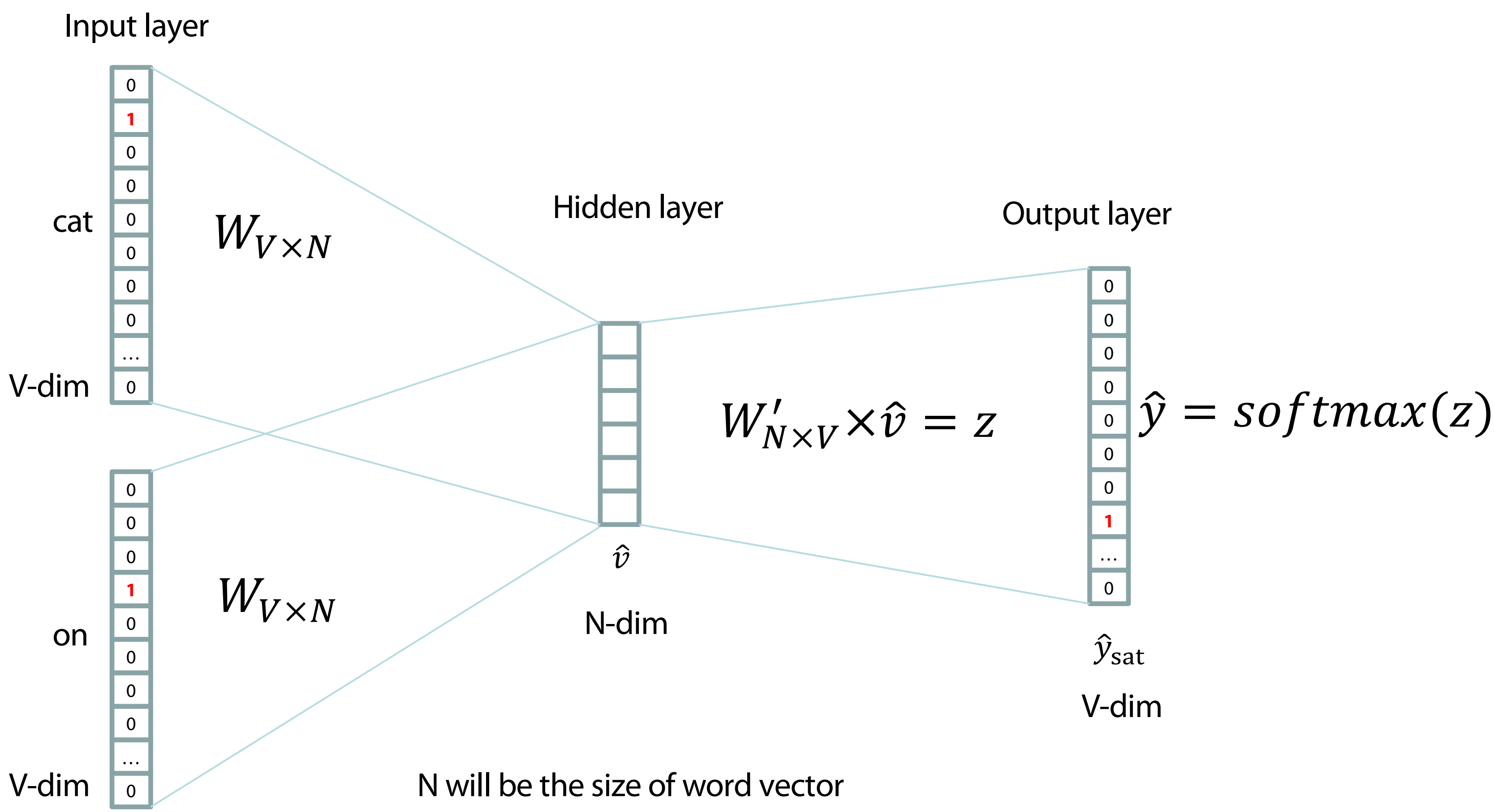


N will be the size of word vector


$W^T_{V\times N}$
$\times x_{cat} = v_{cat}$
Input layer
0.1 2.4 1.6 1.8 0.5 0.9 … … … 3.2
0.5 2.6 1.4 2.9 1.5 3.6 … … … 6.1
0.6 1.8 2.7 1.9 2.4 2.0 … … … 1.2
$\times$
=
2.4
2.6
1.8
x_{cat}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$W^T_{V\times N} \times x_{on} = v_{on}$
x_{on}
V-dim
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
Hidden layer
N-dim
Output layer
sat
V-dim



$W^T_{V\times N}$
$\times x_{on} = v_{on}$
Input layer
0.1 2.4 1.6 1.8 0.5 0.9 … … … 3.2
0.5 2.6 1.4 2.9 1.5 3.6 … … … 6.1
0.6 1.8 2.7 1.9 2.4 2.0 … … … 1.2
1.8 2.9 … … 1.9
x_{cat}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$W^T_{V\times N} \times x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
x_{on}
V-dim
Output layer
sat
V-dim
Hidden layer
N-dim


Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
Output layer
$W'_{N\times V} \times \hat{v} = z$
$\hat{y} = softmax(z)$
$\hat{y}_{\text{sat}}$
V-dim
N will be the size of word vector

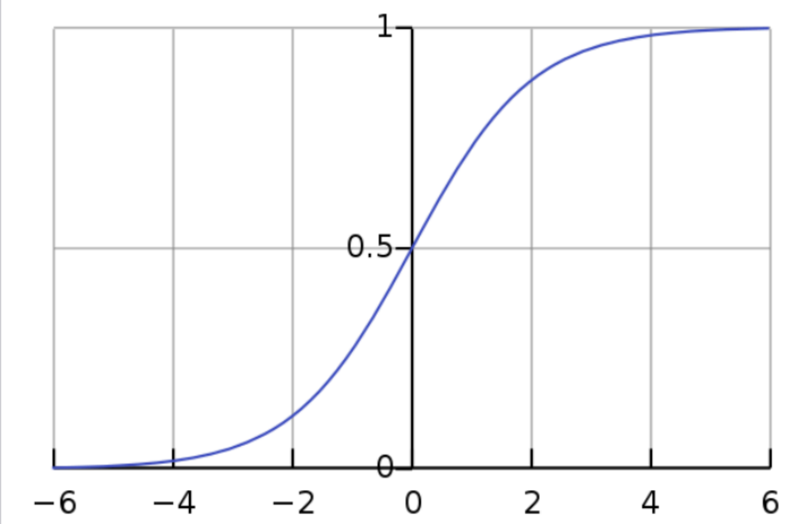
Logistic function

A **logistic function** or **logistic curve** is a common "S" shape (sigmoid curve), with equation:

$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}}$$

where

- e = the natural logarithm base (also known as Euler's number),
- x_0 = the x-value of the sigmoid's midpoint,
- L = the curve's maximum value, and
- k = the steepness of the curve.[1]



Standard logistic sigmoid function i.e. $L = 1, k = 1, x_0 = 0$

[Wikipedia]

softmax($\mathbf{z}$)

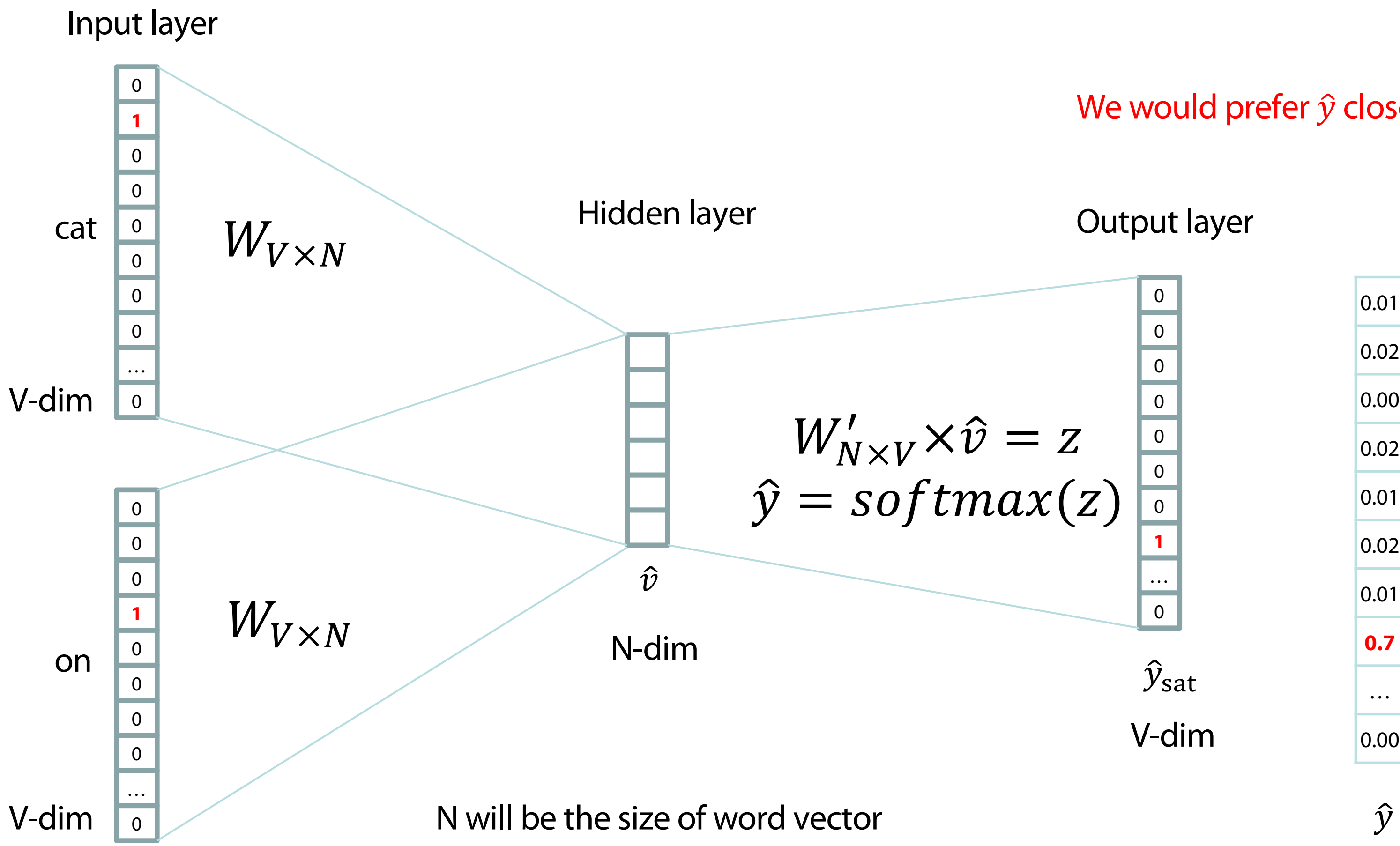
The **softmax function**, or **normalized exponential function**, is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range [0, 1] that add up to 1. The function is given by

$$\sigma : \mathbb{R}^K \to [0,1]^K$$

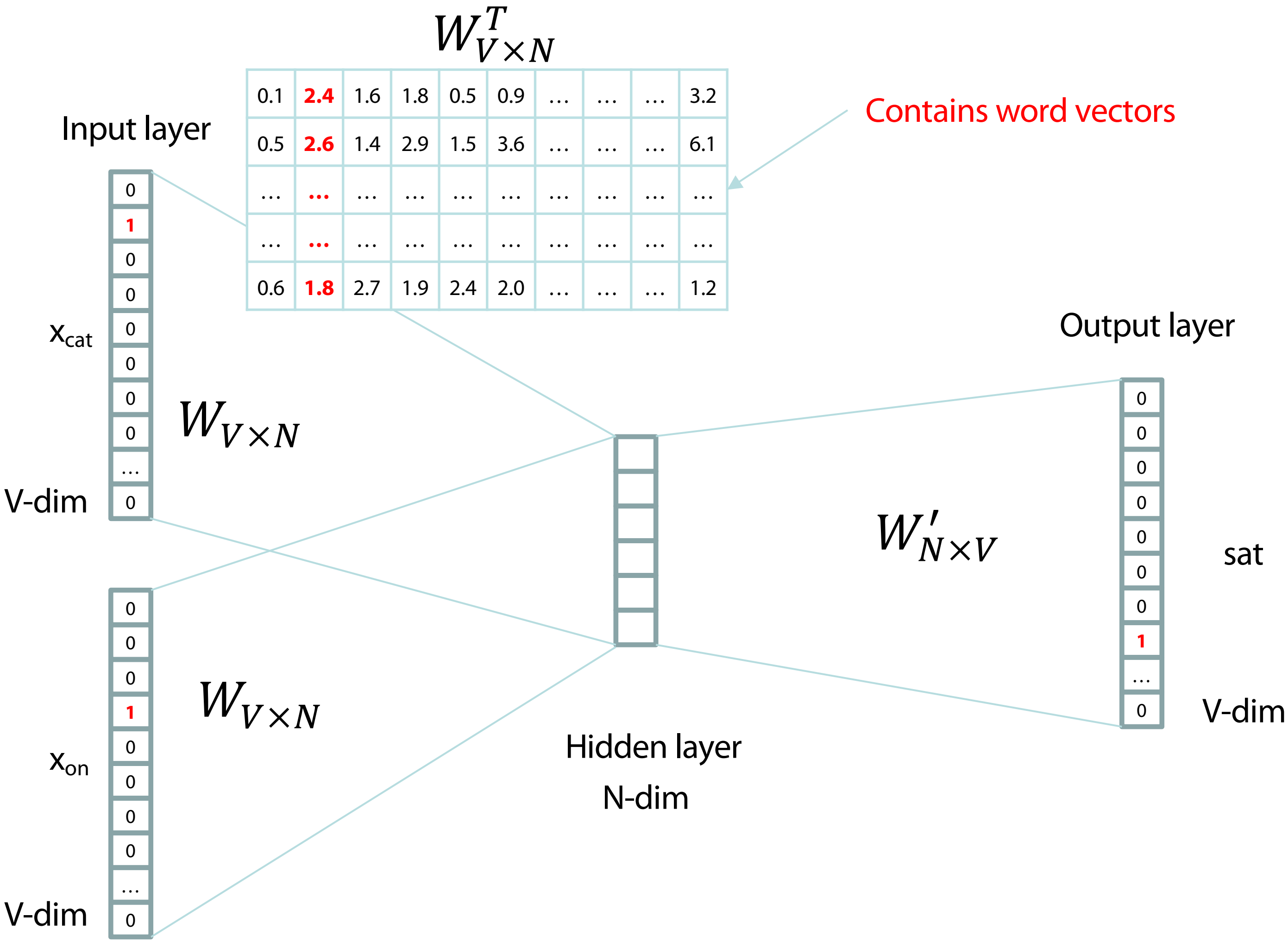
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

In probability theory, the output of the softmax function can be used to represent a categorical distribution – that is, a probability distribution over K different possible outcomes.

We would prefer $\hat{y}$ close to $\hat{y}_{sat}$



N will be the size of word vector



We can consider either W or W' as the word's representation.

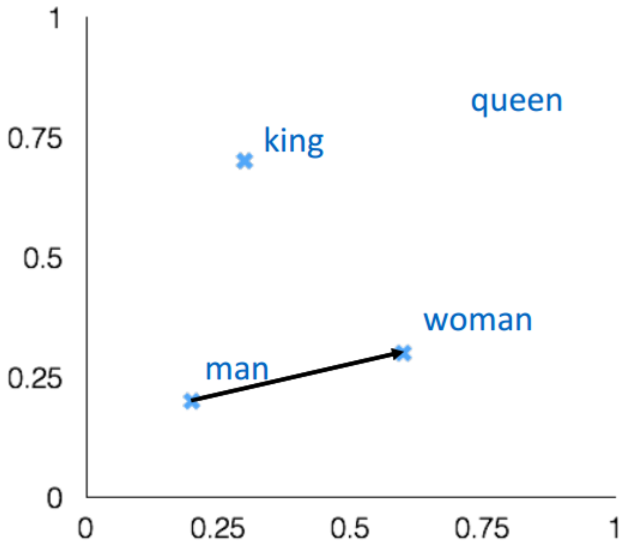
Word Analogies

Test for linear relationships, examined by Mikolov et al. (2014)

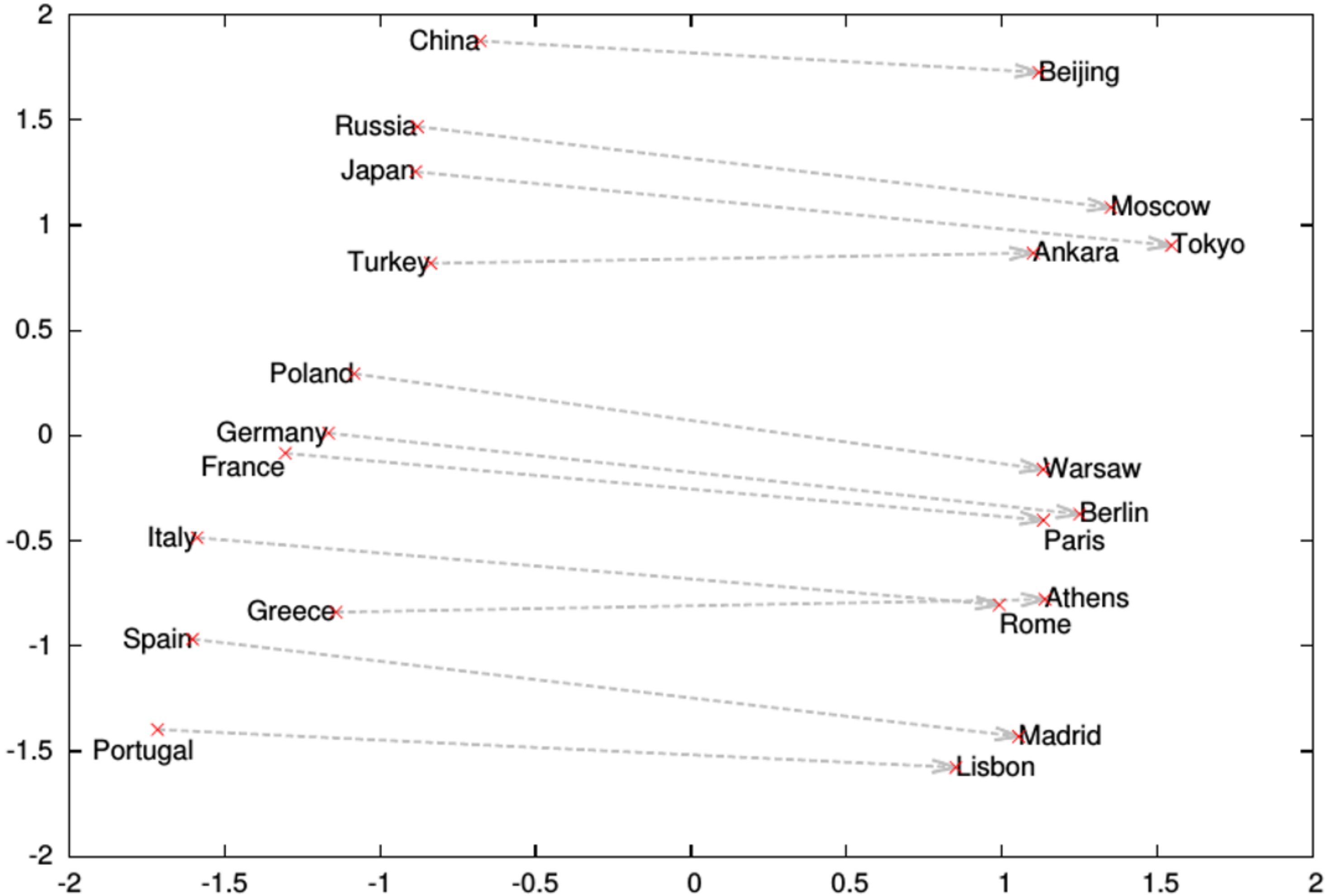
a:b :: c:? → $$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c|| ||w_x||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



Word Analogies



What is `word2vec`?

- `word2vec` is **not** a single algorithm
- It is a **software package** for representing words as vectors, containing:
  - Two distinct models
    - CBoW
    - **Skip-Gram** **(SG)**
  - Various training methods
    - **Negative Sampling** **(NS)**
    - Hierarchical Softmax
  - A rich preprocessing pipeline
    - Dynamic Context Windows
    - Subsampling
    - Deleting Rare Words

Skip-Grams with Negative Sampling (SGNS)

Marco saw a furry little wampimuk hiding in the tree.

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

Skip-Grams with Negative Sampling (SGNS)

Marco saw a furry little wampimuk hiding in the tree.

“word2vec Explained…”
Goldberg & Levy, arXiv 2014

Skip-Grams with Negative Sampling (SGNS)

Marco saw a furry little wampimuk hiding in the tree.

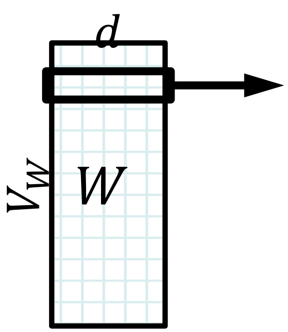
words	**contexts**
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in
…	…

D (data)

“word2vec Explained…”
Goldberg & Levy, arXiv 2014

Skip-Grams with Negative Sampling (SGNS)

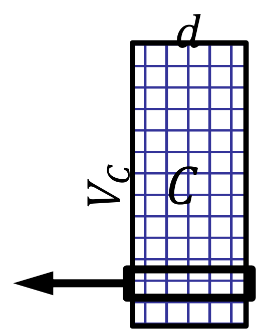
- SGNS finds a vector $\vec{w}$ for each word w in our vocabulary V_W
- Each such vector has d latent dimensions (e.g. $d = 100$)
- Effectively, it learns a matrix W whose rows represent V_W
- **Key point:** it also derives a similar auxiliary matrix C of context vectors
- In fact, each word has two embeddings



w:wampimuk $=$
$(-3.1, 4.15, 9.2, -6.5, \dots)$

$\neq$

c:wampimuk $=$
$(-5.6, 2.95, 1.4, -1.3, \dots)$



"word2vec Explained…"
Goldberg & Levy, arXiv 2014

Skip-Grams with Negative Sampling (SGNS)

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

Skip-Grams with Negative Sampling (SGNS)

- **Maximize:** $\sigma(\vec{w} \cdot \vec{c})$
  - c was **observed** with w

words	**contexts**
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

Skip-Grams with Negative Sampling (SGNS)

- **Maximize:** $\sigma(\vec{w} \cdot \vec{c})$
  - c was **observed** with w

words	**contexts**
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in

- **Minimize:** $\sigma(\vec{w} \cdot \vec{c}')$
  - c' was **hallucinated** with w

words	**contexts**
wampimuk	Australia
wampimuk	cyber
wampimuk	the
wampimuk	1985

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

Skip-Grams with Negative Sampling (SGNS)

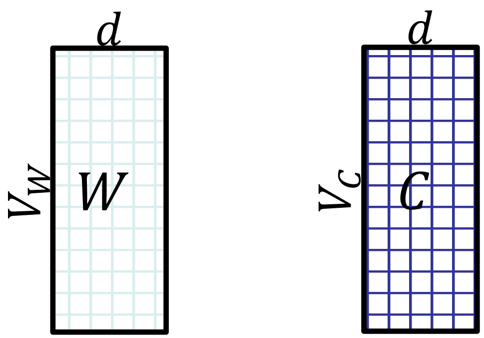
- "Negative Sampling"
- SGNS samples k contexts c' **at random** as **negative** **examples**
- "Random" = unigram distribution

$$P\,(c) = \frac{\#c}{|D|}$$

- **Spoiler:** Changing this distribution has a significant effect

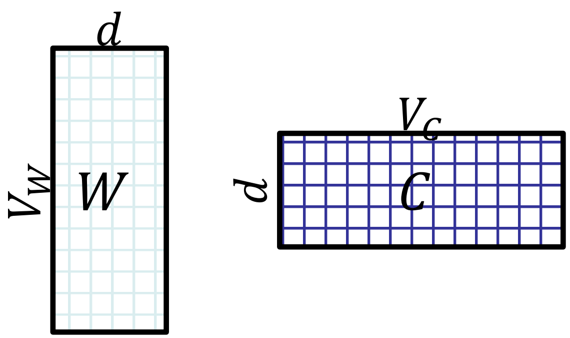
What is SGNS learning?

- Take SGNS's embedding matrices (W and C)



What is SGNS learning?

- Take SGNS's embedding matrices (W and C)
- Multiply them
- What do you get?

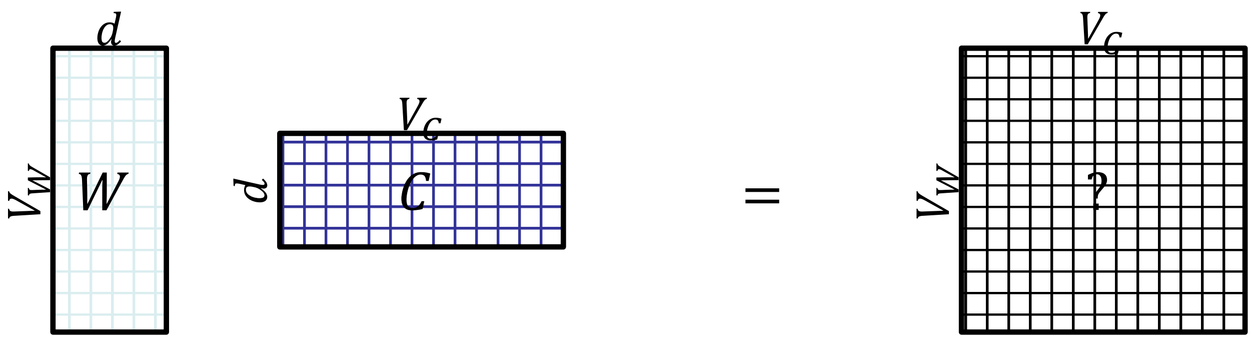


"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- A $V_W \times V_C$ matrix
- Each cell describes the relation between a specific word-context pair

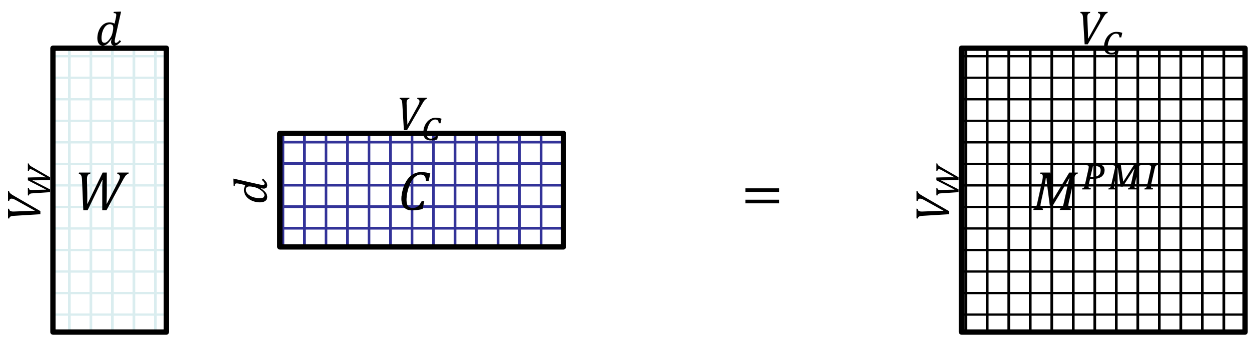
$$\vec{w} \cdot \vec{c} = ?$$



“Neural Word Embeddings as Implicit Matrix Factorization”
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- Levy&Goldberg [2014] **proved** that for large enough d and enough iterations …
- … one obtains the word-context PMI matrix

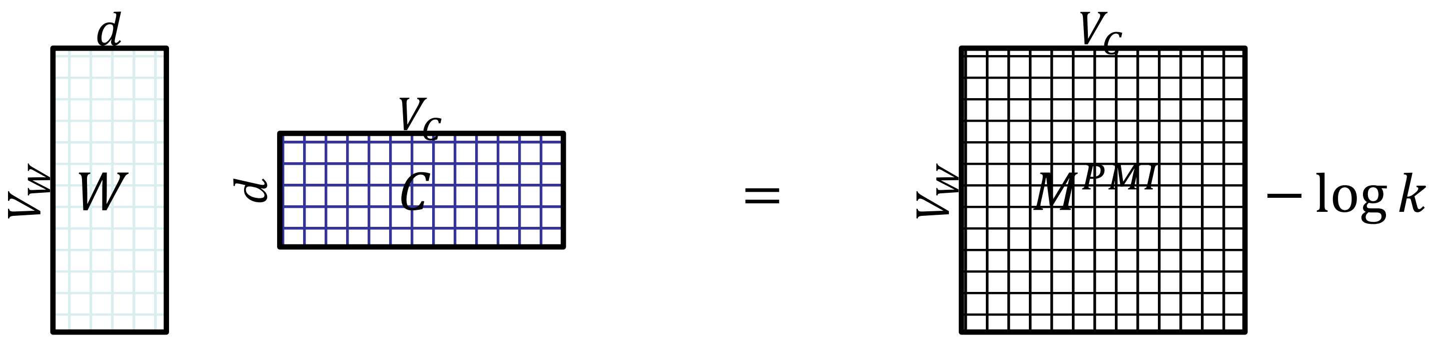


“Neural Word Embeddings as Implicit Matrix Factorization”
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- Levy&Goldberg [2014] **proved** that for large enough d and enough iterations …
- … one obtains the word-context PMI matrix …
- shifted by a global constant

$$Opt(\vec{w} \cdot \vec{c}) = PMI(w, c) - \log k$$



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- SGNS is doing something very similar to the older approaches

- SGNS factorizes the traditional word-context PMI matrix

- So does SVD!

- GloVe factorizes a similar word-context matrix

But embeddings are still better, right?

- Plenty of evidence that embeddings outperform traditional methods
  - "Don't Count, Predict!" (Baroni et al., ACL 2014)
  - GloVe (Pennington et al., EMNLP 2014)

- How does this fit with our story?

The Big Impact of "Small" Hyperparameters

- `word2vec` & GloVe are more than just algorithms…
- Introduce **new hyperparameters**
- May seem minor, but **make a big difference** in practice

New Hyperparameters

- **Preprocessing** **(word2vec)**
  - Dynamic Context Windows
  - Subsampling
  - Deleting Rare Words

- **Postprocessing** **(GloVe)**
  - Adding Context Vectors

- **Association Metric** **(SGNS)**
  - Shifted PMI
  - Context Distribution Smoothing

Dynamic Context Windows

Marco saw a furry little wampimuk hiding in the tree.

Dynamic Context Windows

saw a furry little wampimuk hiding in the tree

Dynamic Context Windows

saw a furry little wampimuk hiding in the tree

Word2vec:	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{3}{4}$	$\frac{4}{4}$		$\frac{4}{4}$	$\frac{3}{4}$	$\frac{2}{4}$	$\frac{1}{4}$
GloVe:	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{1}$		$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$
Aggressive:	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{1}$		$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

The Word-Space Model *(Sahlgren, 2006)*

Adding Context Vectors

- SGNS creates word vectors $\vec{w}$
- SGNS creates auxiliary context vectors $\vec{c}$
  - So do GloVe and SVD

Adding Context Vectors

- SGNS creates word vectors $\vec{w}$
- SGNS creates auxiliary context vectors $\vec{c}$
  - So do GloVe and SVD

- Instead of just $\vec{w}$
- Represent a word as: $\vec{w} + \vec{c}$

- Introduced by Pennington et al. (2014)
- Only applied to GloVe

Context Distribution Smoothing

- SGNS samples $c' \sim P$ to form **negative** (w, c') examples

- Our analysis assumes P is the unigram distribution

$$P\left(c\right) = \frac{\#c}{\sum_{c' \in V_C} \#c'}$$

Context Distribution Smoothing

- SGNS samples $c' \sim P$ to form **negative** (w, c') examples

- Our analysis assumes P is the unigram distribution

- In practice, it's a **smoothed** unigram distribution

$$P^{0.75}(c) = \frac{(\#c)^{0.75}}{\sum_{c' \in V_C} (\#c')^{0.75}}$$

- This little change makes a big difference

Context Distribution Smoothing

- We can **adapt** context distribution smoothing to PMI!

- Replace $P(c)$ with $P^{0.75}(c)$:

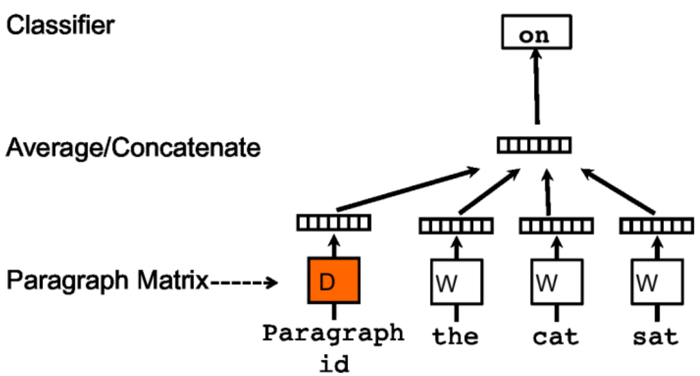
$$PMI^{0.75}(w,c) = \log \frac{P(w,c)}{P(w) \cdot \boldsymbol{P^{0.75}(c)}}$$

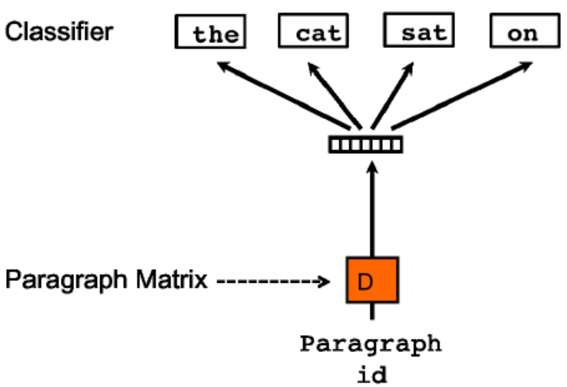
- Consistently improves **PMI** on **every task**

- **Always use Context Distribution Smoothing!**

Represent the meaning of **sentence/text**

- Paragraph vector (2014, Quoc Le, Mikolov)
  - Extend word2vec to text level
  - Also two models: add paragraph vector as the input





Don’t Count, Predict! [Baroni et al., 2014]

- “word2vec is better than count-based methods”
- **Hyperparameter settings** account for most of the reported gaps
- Embeddings do **not** really outperform count-based methods
- No unique conclusion available

The Contributions of Word Embeddings

Novel Algorithms

(objective + training method)

- Skip Grams + Negative Sampling
- CBOW + Hierarchical Softmax
- Noise Contrastive Estimation
- GloVe
- …

New Hyperparameters

(preprocessing, smoothing, etc.)

- Subsampling
- Dynamic Context Windows
- Context Distribution Smoothing
- Adding Context Vectors
- …

What's really improving performance?

Improving Distributional Similarity with Lessons Learned from Word Embeddings, Omer Levy, Yoav Goldberg, Ido Dagan