Intelligent Agents

Word Semantics, Embeddings and Latent Sequential Structures

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

Word-Word Associations in Document Retrieval

Recap

- LSI: Documents as vectors, dimension reduction
- Topic Modeling
  - Topic = Word distribution
  - From LDA-Model: P(Z | **w**)
  - Assumption: Bag of words model (independence, naïve Bayes, unigram distribution)

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 more 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/

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

Kenneth Ward Church and Patrick Hanks. "Word association norms, mutual information, and lexicography". Comput. Linguist. 16 (1): 22–29. **1990**.

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]

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 of Frequent Words
- Dynamic Context Windows
- Context Distribution Smoothing
- Adding Context Vectors
- …

What's really improving performance?

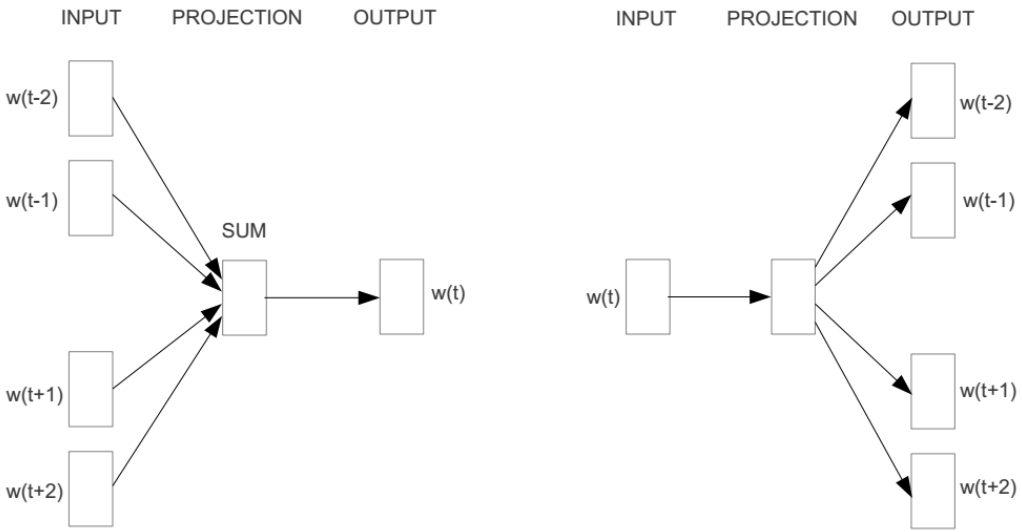
Improving Distributional Similarity with Lessons Learned from Word Embeddings, Omer Levy, Yoav Goldberg, Ido Dagan, Transactions of the Association for Computational Linguistics, Volume 3, **2015**.

Embedding Approaches

- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key idea: Predict surrounding words of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

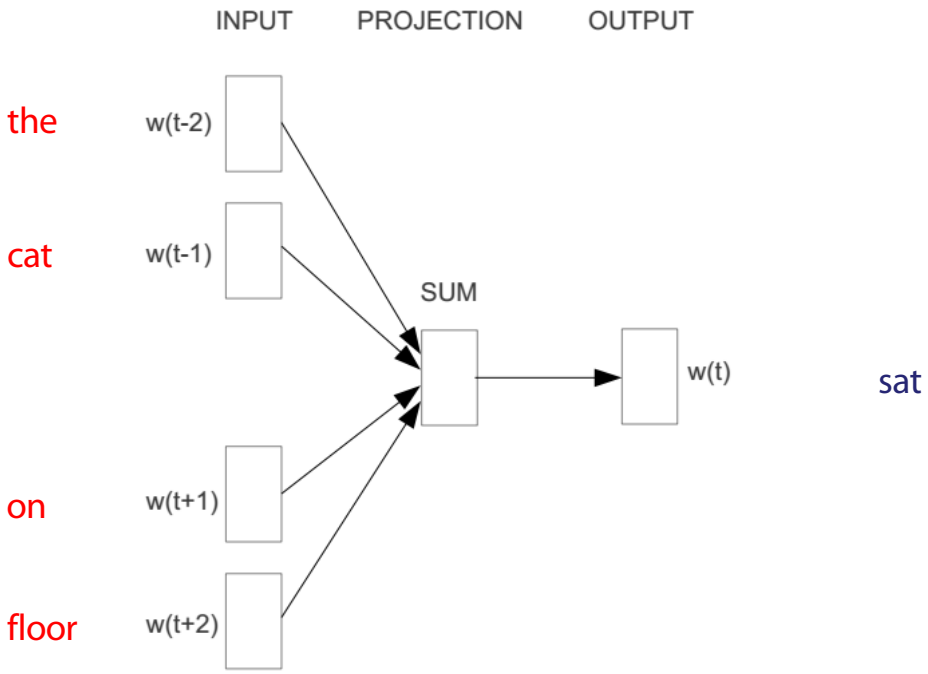
Represent the meaning of **word** – word2vec

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

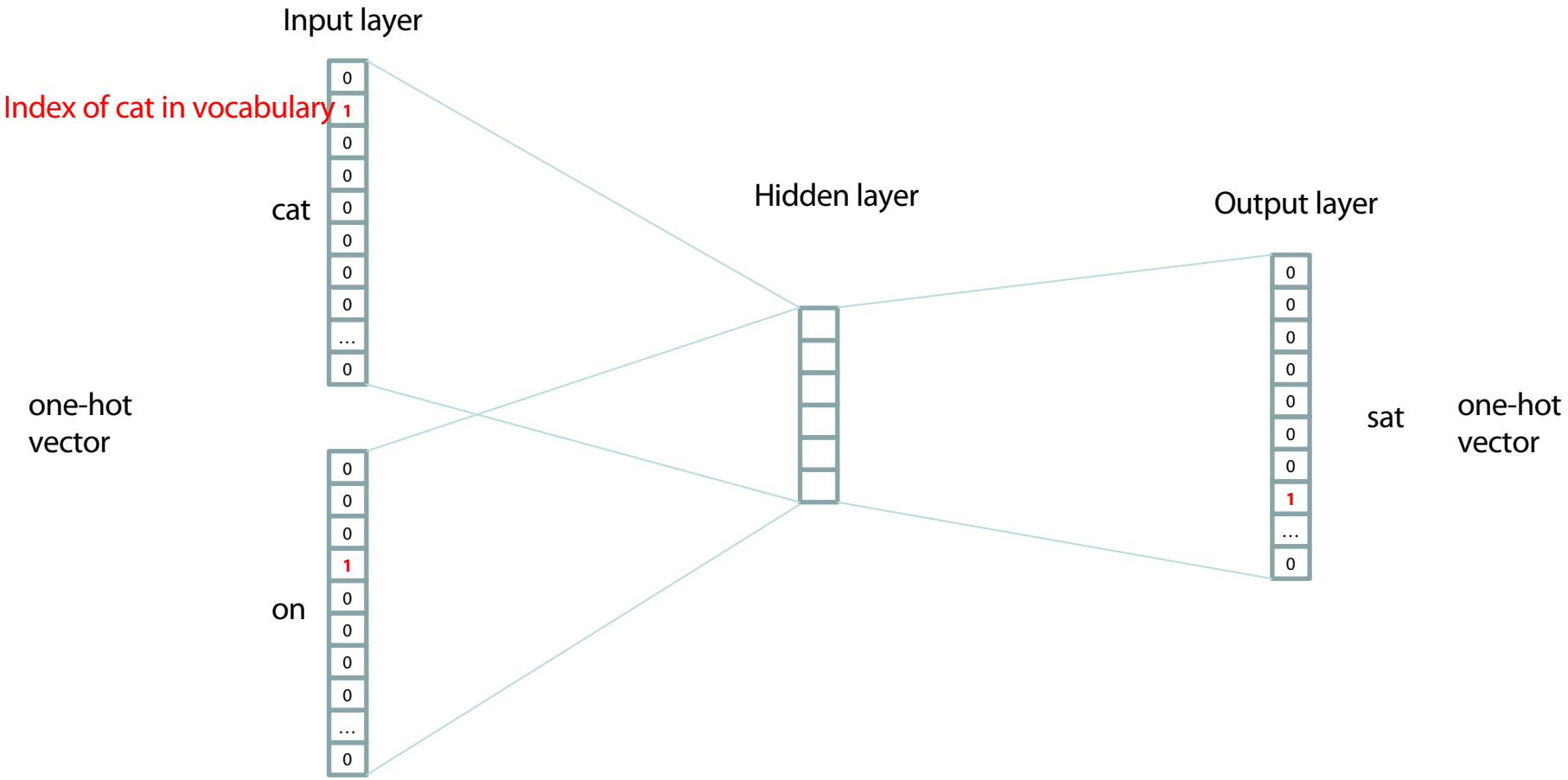


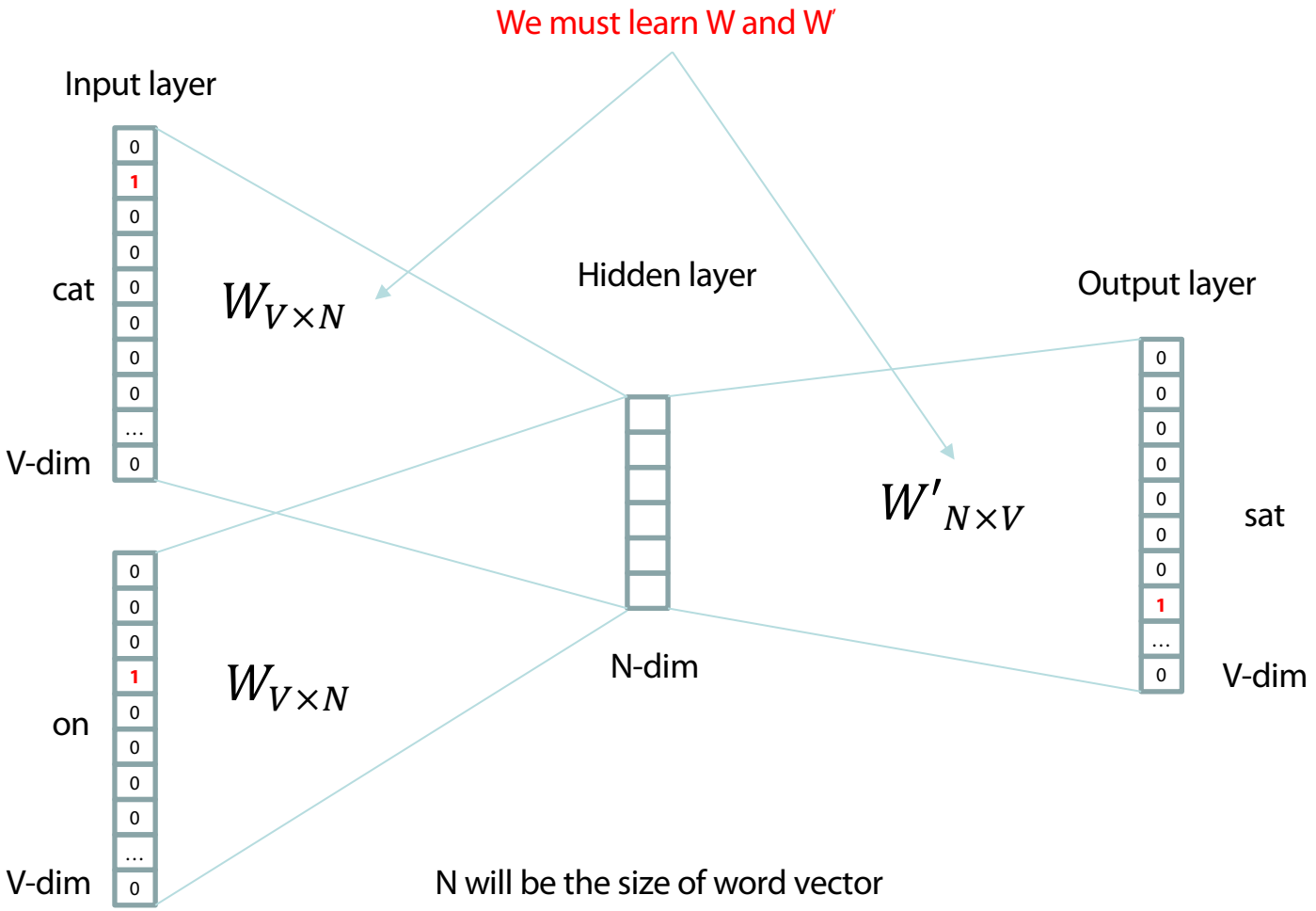
Word2vec – Continuous Bag of Words

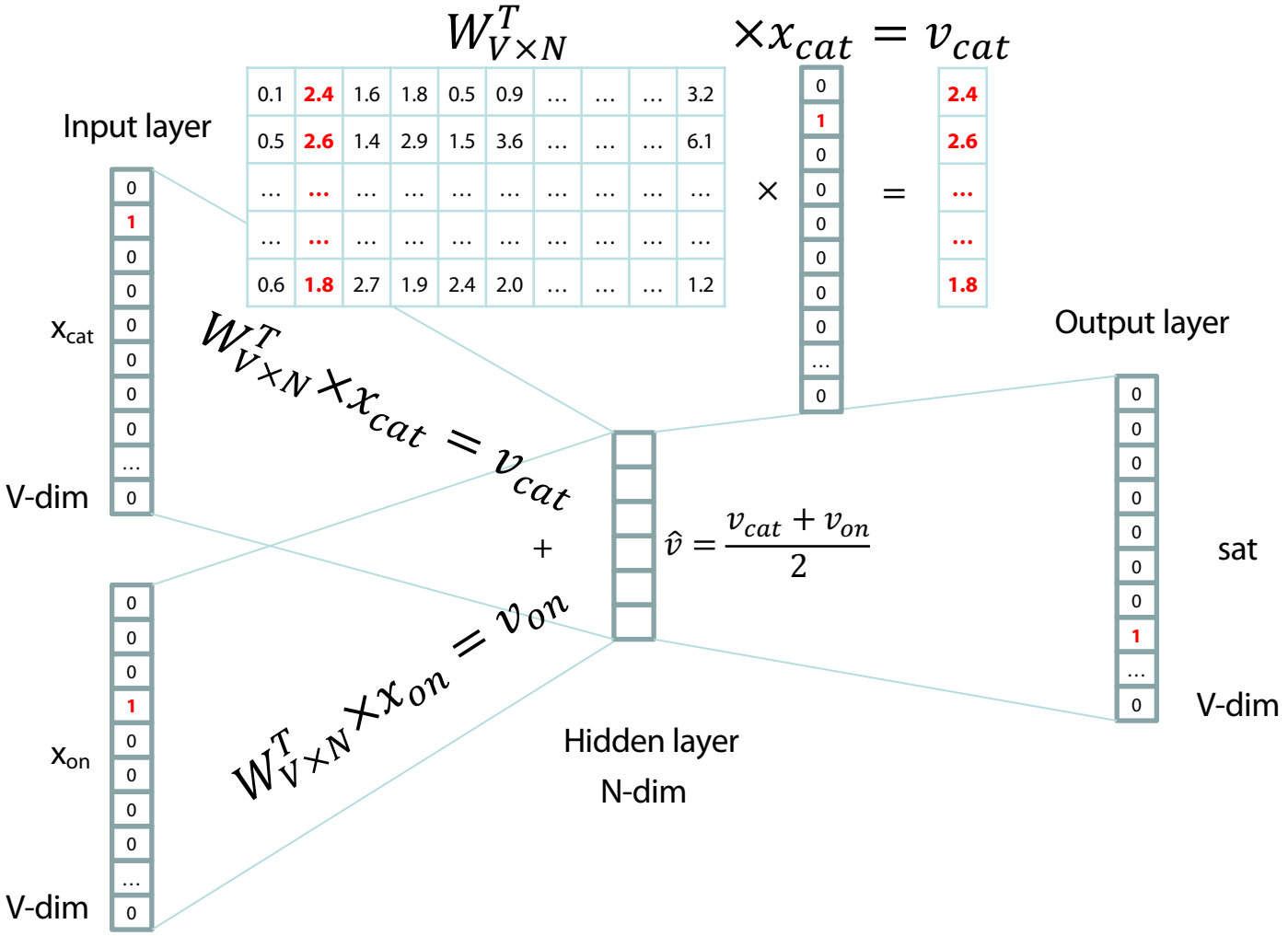
- E.g. “The cat sat on floor”
  - Window size = 2

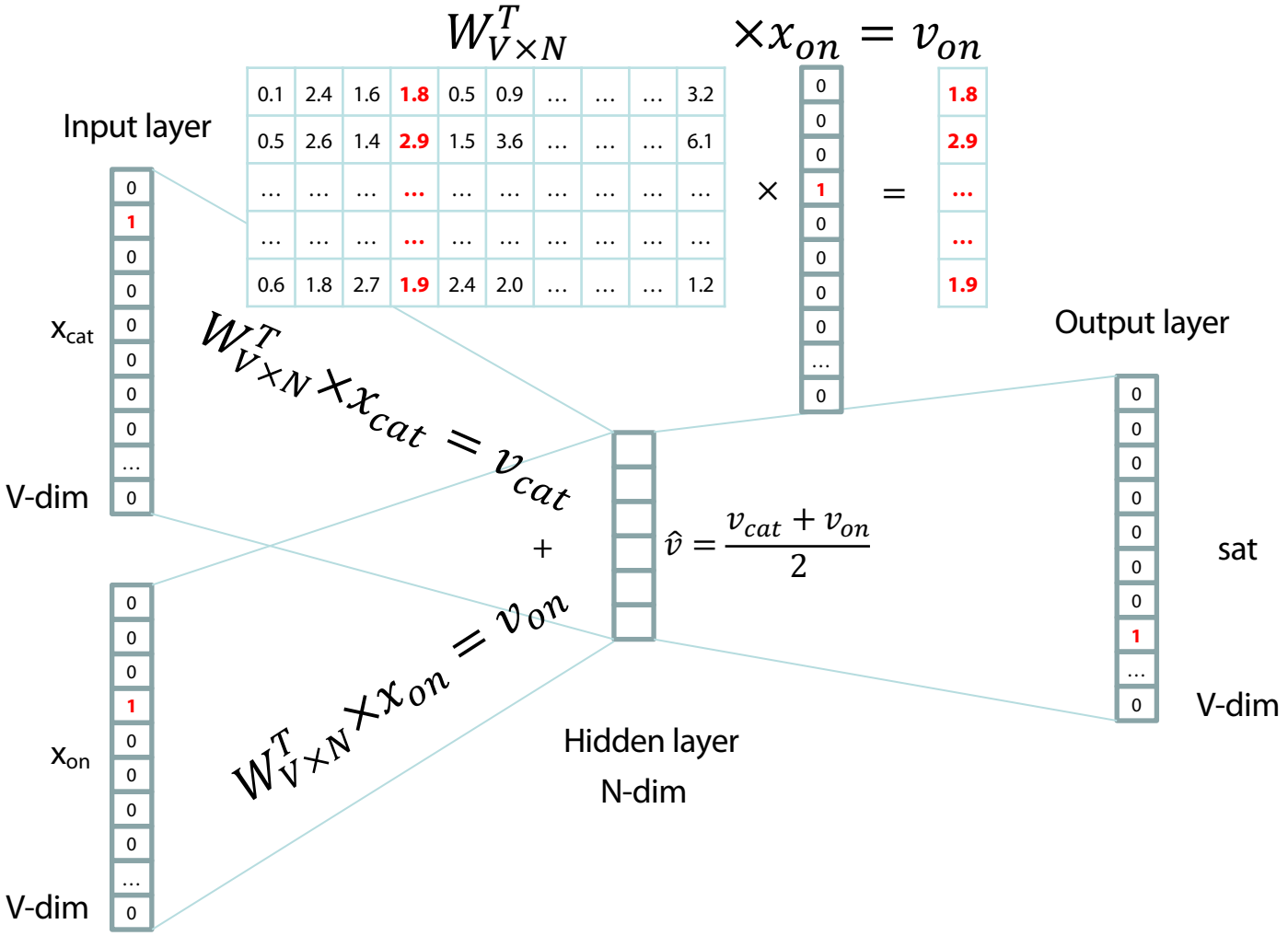


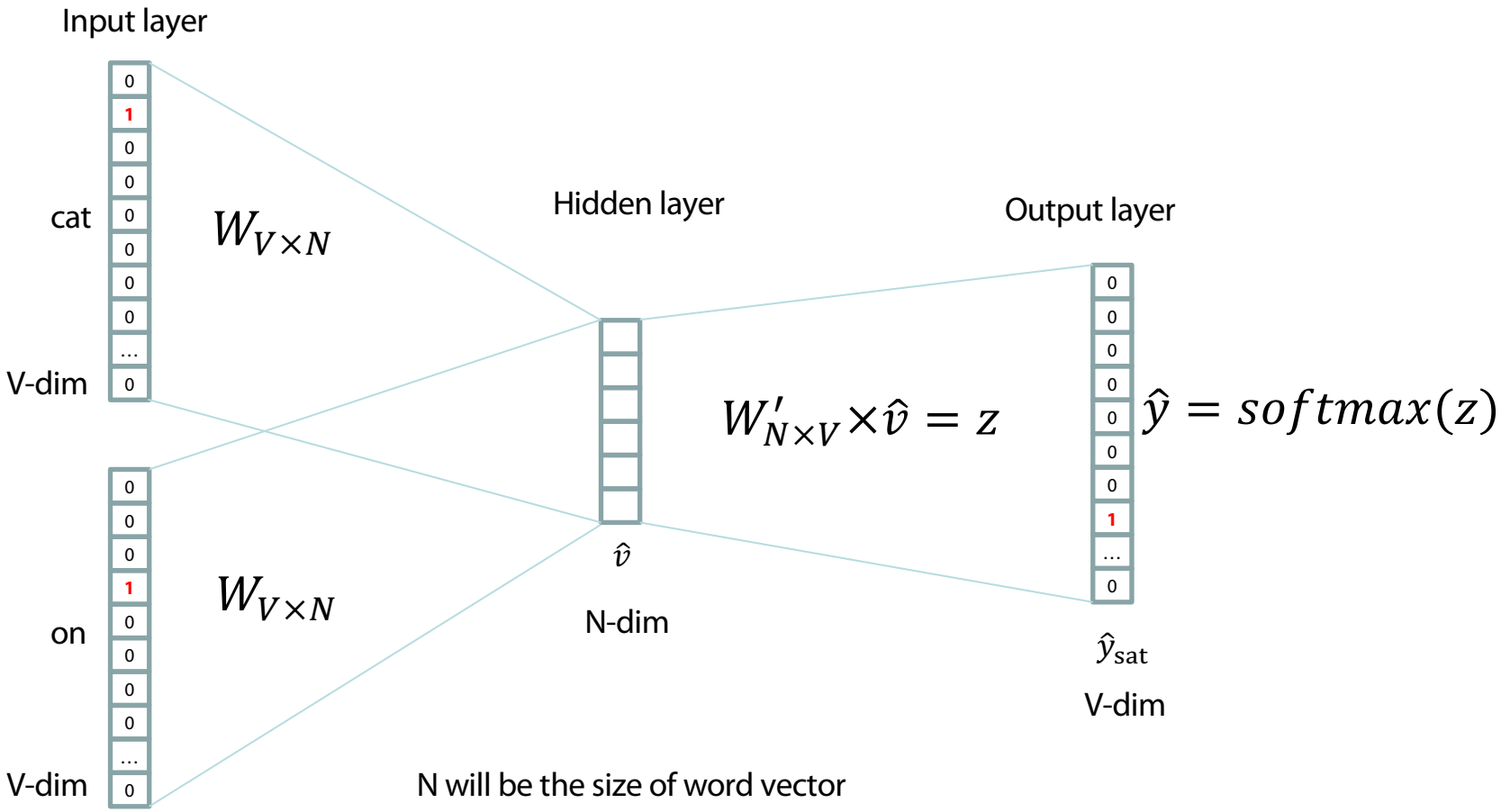
Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS **2013**


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


We must learn W and W'
Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
N-dim
$W'_{N\times V}$
Output layer
sat
V-dim
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
×
=
2.4
2.6
…
…
1.8
Output layer
x_{cat}
V-dim
$W^T_{V\times N} x_{cat} = v_{cat}$
+
$W^T_{V\times N} x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
sat
x_{on}
V-dim
Hidden layer
N-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}
$W^T_{V\times N} x_{cat} = v_{cat}$
Output layer
V-dim
+
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
sat
x_{on}
$W^T_{V\times N} x_{on} = v_{on}$
Hidden layer
N-dim
V-dim


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

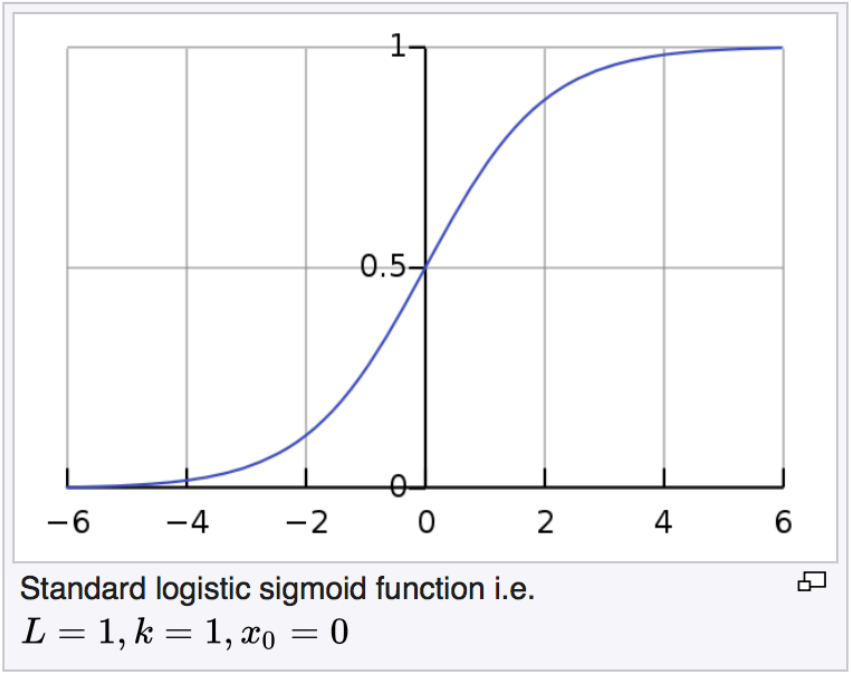
Logistic function

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(**z**)

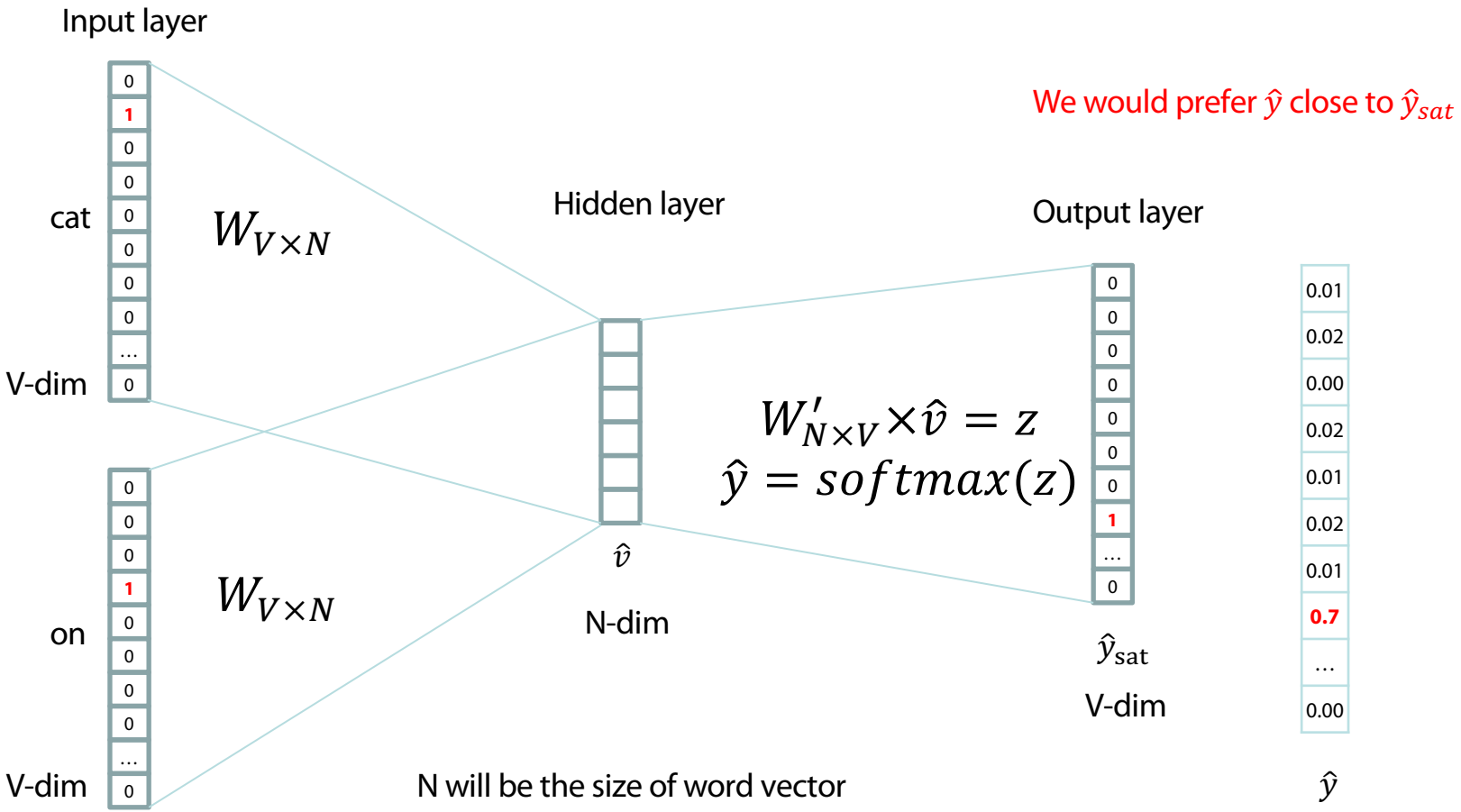
The **softmax function**, or **normalized exponential function**, is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range [0, 1] that add up to 1. The function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

[Wikipedia]


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

CBOW Model

Objective: Given $w_{c-k} \ \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+k}$, predict w_c

Training data: Given sequence of words $< w_1, w_2, \ldots wn >$,
extract context and target: $(w_{c-k} \ \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+k}; w_c)$

Knowns:

- Training data $\{(w_{c-k} \ \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+k}; w_c)\}$
- Vocabulary $\{w_1, w_2, \ldots wV\}$ of the training corpus

Unknowns:

- Word embedding matrices $W_{V \text{ x } N}$ and $W'_{N \text{ x } V}$ with N being a hyperparameter

Loss Function for Learning

- How to determine word embedding matrices?
- Cross entropy for comparing probability distributions
  - $H(\hat{y}, y) = -\sum_{j=1}^{V} y_j \, log(\hat{y}_j)$
- y is a one-hot vector with a "one" at position i
  - $H(\hat{y}, y) = -yi \, log(\hat{y}_i) = -log(\hat{y}_i)$

In this formulation, c is the index where the correct word's one hot vector is 1. We can now consider the case where our prediction was perfect and thus $\hat{y}_c = 1$. We can then calculate $H(\hat{y}, y) = -1 \log(1) = 0$. Thus, for a perfect prediction, we face no penalty or loss. Now let us consider the opposite case where our prediction was very bad and thus $\hat{y}_c = 0.01$. As before, we can calculate our loss to be $H(\hat{y}, y) = -1 \log(0.01) \approx 4.605$. We can thus see that for probability distributions, cross entropy provides us with a good measure of distance.

CBOW: Derivation of Learning Procedure

$$\textit{Minimize} \quad -\log P(w_c \mid w_{c-k}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+k})$$

$$= -\log P(W'[c] \mid \hat{v}) \quad \text{(and due to the softmax)}$$

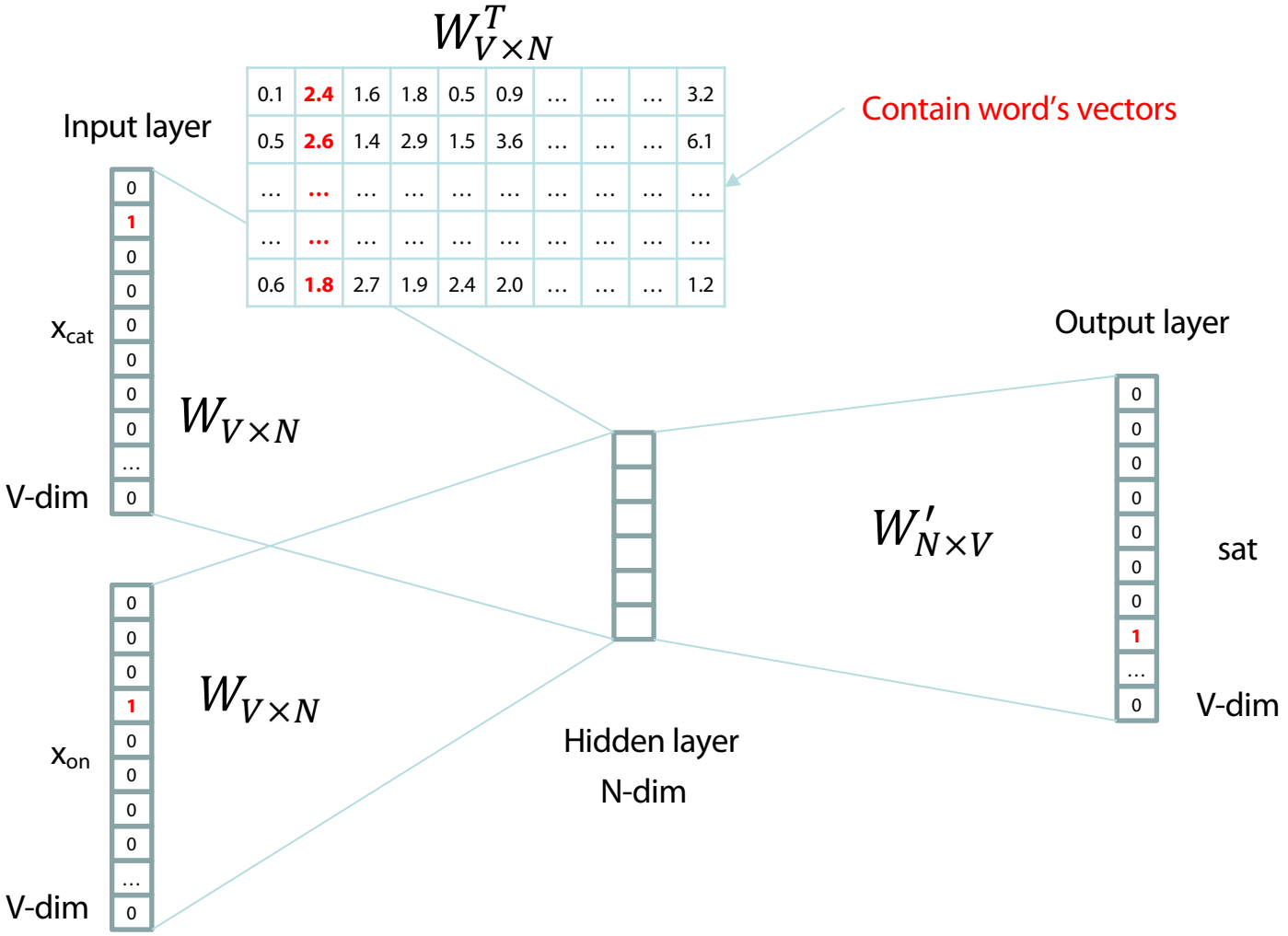
$$= -\log \frac{e^{W\prime[c]^T \hat{v}}}{\sum_{j=1}^{V} e^{W\prime[c]^T{}_j \hat{v}}}$$

$$= -W'[c]^T \hat{v} + \log \sum_{j=1}^{V} e^{W\prime[j]^T \hat{v}}$$

where

$$\hat{v} = (2k)^{-1} \sum_{i=-k}^{k} W^T w_{c+i}$$

Use gradient decent to update word vectors



We can consider either W or W' as the word's representation.
Or even take the average.

Intrinsic Evaluation

Word Analogies

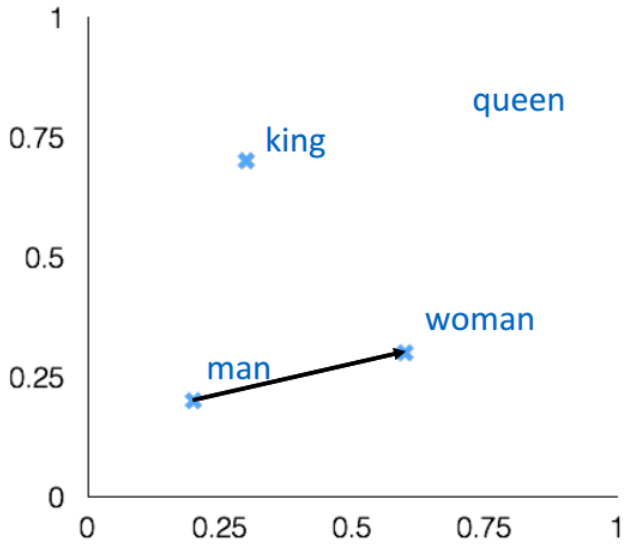
Test for linear relationships, examined by Mikolov et al. (2014)

a:b :: c:?

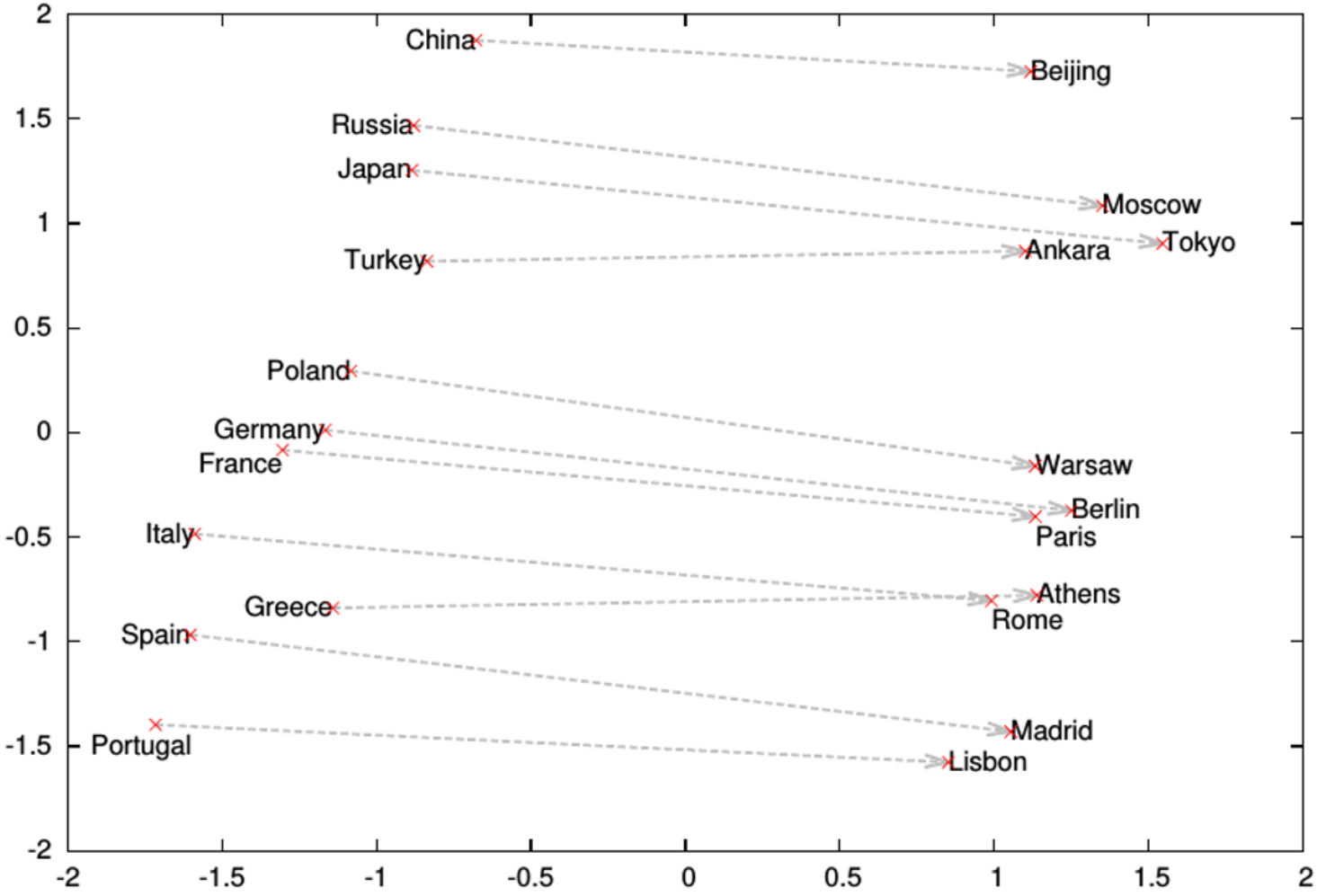
$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

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



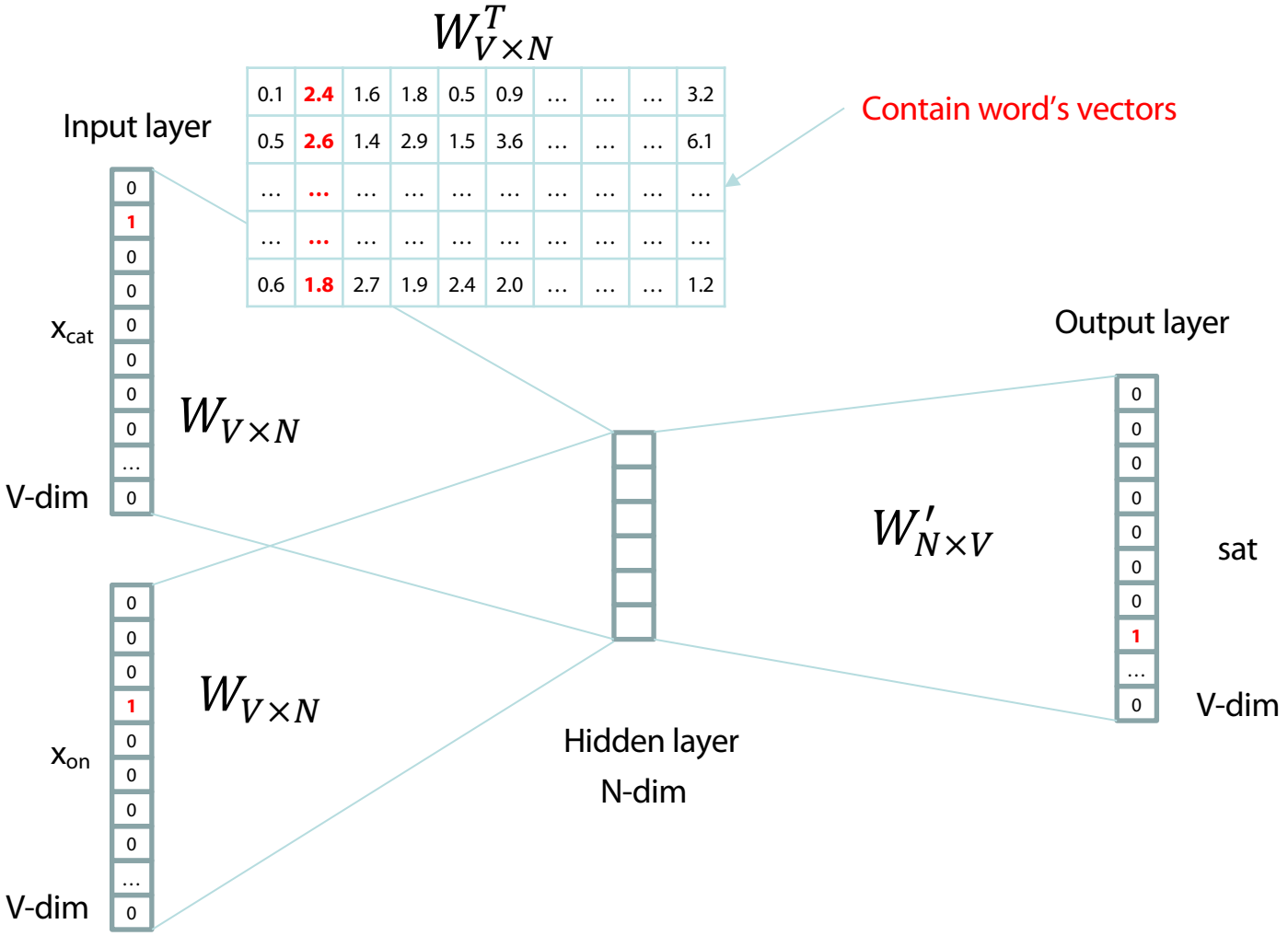
Extrinsic Evaluation

- Evaluate in applications
  - Sentiment analysis
  - …

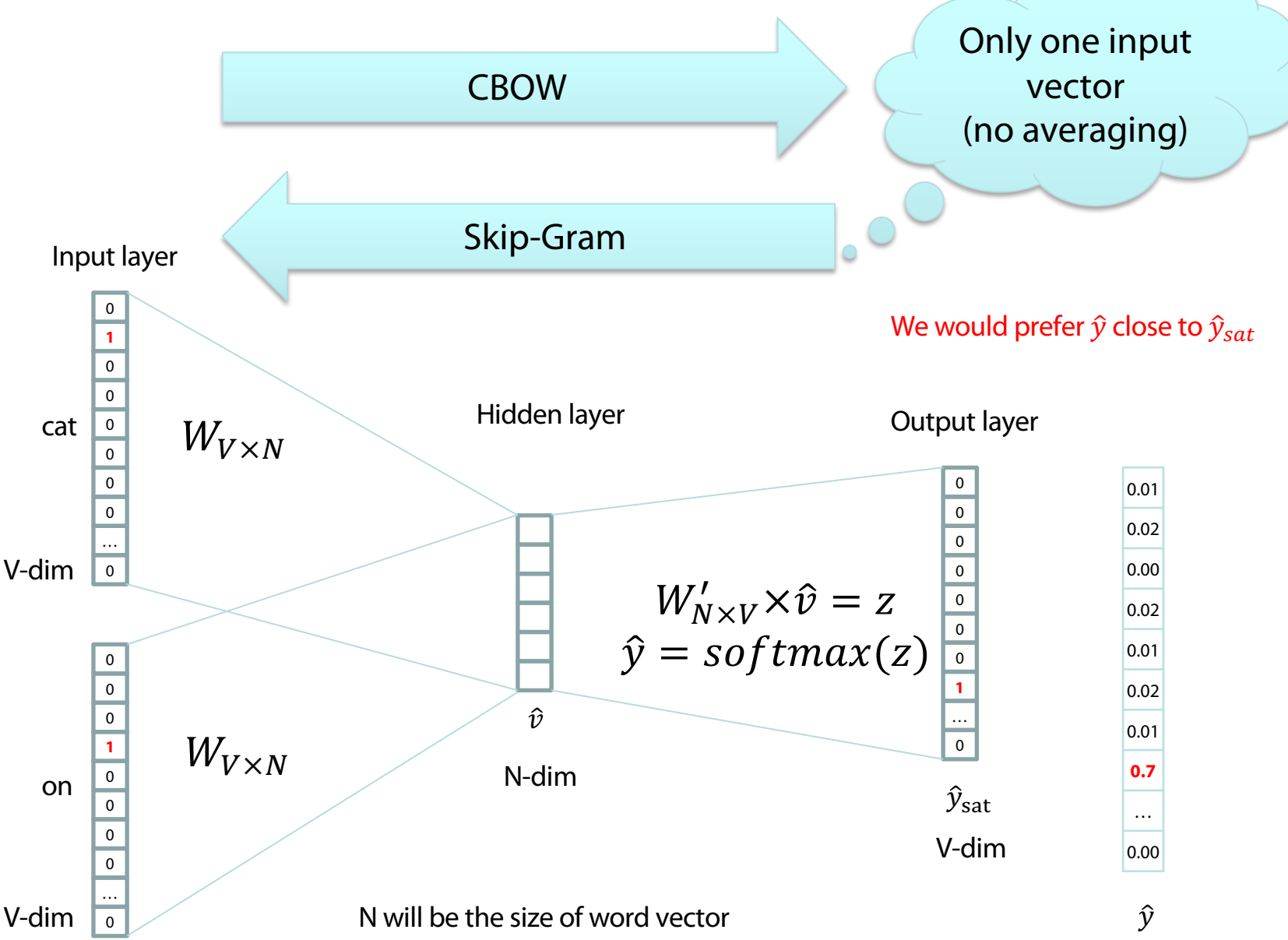
Intelligent Agents

Word Semantics, Embeddings and Latent Sequential Structures

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



We can consider either W or W' as the word's representation.
Or even take the average.


CBOW
Only one input vector (no averaging)
Skip-Gram
Input layer
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
$W'_{N\times V}\times\hat{v}=z$
$\hat{y}=softmax(z)$
Output layer
$\hat{y}_{\text{sat}}$
V-dim
0.01
0.02
0.00
0.02
0.01
0.02
0.01
0.7
0.00
$\hat{y}$
N will be the size of word vector

Skip-Gram Model

Objective: Given w_c, predict $w_{c-k} \ \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+k}$

Training data: Given sequence of words $< w_1, w_2, \ldots w_n >$,
extract input and output: $(w_c \ ; \ w_{c-k} \ \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+k})$

Knowns:

- Training data $\{(w_c \ ; \ w_{c-k} \ \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+k})\}$
- Vocabulary $\{w_1, w_2, \ldots w_V\}$ of the training corpus

Unknowns:

- Word embedding matrices $W_{V \times N}$ and $W'_{N \times V}$ with N being a hyperparameter

Skip-Gram: Derivation of Learning Procedure

$$\textit{Minimize} \quad -\log P(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m} | \, w_c)$$

$$= -\log \prod_{j=0, j\neq m}^{2m} P\left(w_{c-m+j} \,|v_c\right) \quad \text{(and due to softmax)}$$

$$= -\log \prod_{j=0, j\neq m}^{2m} \frac{e^{W_{c-m+j} v_c}}{\sum_{k=1}^{V} e^{W_k v_c}}$$

$$= -\sum_{j=0, j\neq m}^{2m} W_{c-m+j} \, v_c \quad + 2m \log \sum_{k=1}^{V} e^{W_k v_c}$$

where $v_c = W' w_c$
(no averaging for skip-gram)

Use stochastic gradient decent to minimize and then to update word vectors

word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method, Yoav Goldberg and Omer Levy, arxiv, **2014**.

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
  - Various training methods
    - Softmax is a bottleneck (discussed next)
  - A rich preprocessing pipeline
    - Dynamic Context Windows
    - Subsampling of Frequent Words
    - Deleting Rare Words (left out)

Softmax is a Bottleneck (CBOW and Skip-Gram)

- The denominator is a sum across entire vocabulary
- $-\sum_{j=0,j\neq m}^{2m} W_{c-m+j}\, v_c \quad + 2m \log \sum_{k=1}^{V} e^{W_k\, v_c}$
- To be computed for every window
- Too expensive
- Single update of parameters requires iteration of entire vocabulary (which usually is in millions)
- Various optimized training methods
  - Hierarchical Softmax
  - Noise Contrastive Estimation (left out)
  - Negative Sampling

Rong, X. word2vec Parameter Learning Explained (cite arxiv:1411.2738). **2014**.

Hierarchical softmax using Trees

target word $w(t)$

word history $\mathbf{w}_1^{t-1}$

path to target word at node j $n(w(t), j)$

predicted word vector $\hat{\mathbf{z}}(t)$

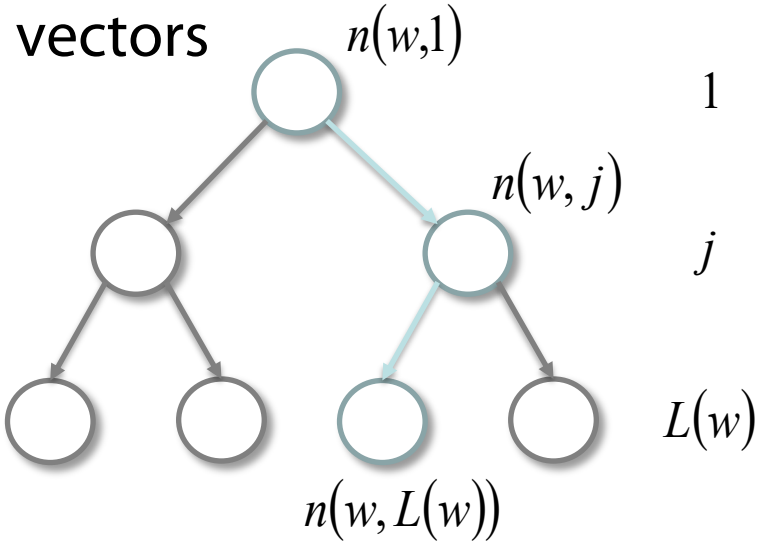
vector at node j of target word $\mathbf{z}_{n(w,j)}$

sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

$$P(w_t = v \mid \mathbf{w}_1^{t-1}) = \prod_{j=1}^{L(w)-1} \sigma\left(\pm 1_{n(w,j+1)=ch(n(w,j))} \mathbf{z}_{n(w,j)}^T \hat{\mathbf{z}}\right)$$

ch(n) = left(+) or right child(-)

- Replace computation with *V* vectors of target words by computation with log(*V*) vectors

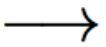


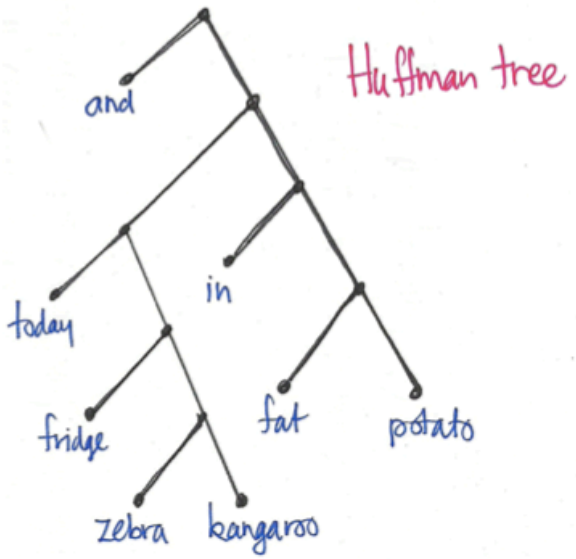
- Need to learn $\mathbf{z}_{n(w,j)}$

Mikolov, T., Chen, K., Corrado, G., Dean, J.:
Efficient Estimation of Word Representations in Vector Space.
In: ICLR Workshop. **2013**.

Huffman Trees instead of Balanced Trees

word	count
fat	3
fridge	2
zebra	1
potato	3
and	14
in	7
today	4
kangaroo	2





Skip-Grams with Negative Sampling (SGNS)

Marco saw a furry little wampimuk hiding in the tree.

Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS **2013**

Skip-Grams with Negative Sampling (SGNS)

Marco saw a furry little wampimuk hiding in the tree.

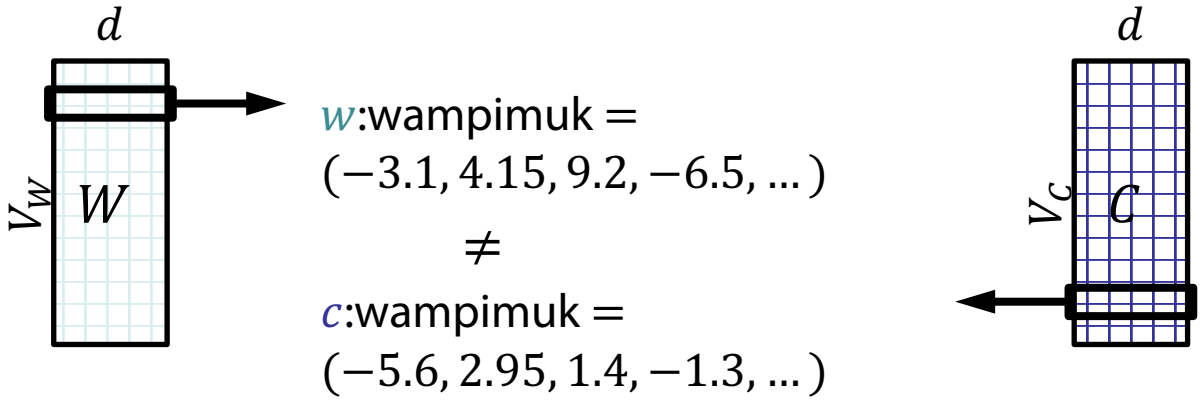
words	**contexts**
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in
…	…

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 learns a similar auxiliary matrix C of context vectors
- In fact, each word has two embeddings



d was called N before

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

Coming back to Negative Sampling

- Given (w, c): word and context
- Let $P(D = 1| w, c)$ be the probability that (w, c) came from the corpus data
- $P(D = 0| w, c)$ = probability that (w, c) are not from the corpus data
- Let us model $P(D = 1| w, c)$ with $sigmoid$
- $P(D = 1| w, c) = sigmoid(u_w^T v_c) = \frac{1}{1+e^{-u_w^T v_c}}$

  $u_v = Ww \quad v_c = Cc$
- Objective:
  - Maximize $P(D = 1| w, c)$ if (w, c) is in the corpus data
  - Minimize$P(D = 1| w, c)$ if (w, c) not in the corpus data

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

Math behind Negative Sampling

Maximum Likelihood approach for learning θ

$$\theta = \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta)$$

$$= \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} (1-P(D=1|w,c,\theta))$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log P(D=1|w,c,\theta) + \sum_{(w,c)\in \tilde{D}} \log(1-P(D=1|w,c,\theta))$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log(1-\frac{1}{1+\exp(-u_w^T v_c)})$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log(\frac{1}{1+\exp(u_w^T v_c)})$$

$\theta = (\mathrm{W}, \mathrm{C}), \qquad u_v = Ww \quad v_c = Cc$

Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS **2013**

Math behind Negative Sampling

- Maximize log likelihood = minimize –log likelihood

$$-\sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} - \sum_{(w,c)\in \tilde{D}} \log\left(\frac{1}{1+\exp(u_w^T v_c)}\right)$$

- $\widetilde{D}$ is the negative corpus with wrong contexts
- Generate $\widetilde{D}$ on the fly by randomly sampling from the vocabulary
- New objective function for observing context word w_{c-m+j} $(j = 0..2m)$ given the center word w_c would be

$$-\log\sigma(u_{c-m+j}^T \cdot v_c) - \sum_{k=1}^{K} \log\sigma(-\tilde{u}_k^T \cdot v_c)$$

$$-u_{c-m+j}^T v_c + \log\sum_{k=1}^{|V|} \exp(u_k^T v_c)$$

regular softmax loss for skip-gram

$$-\sum_{j=0,j\neq m}^{2m} W_{c-m+j}\, v_c \quad + 2m \log\sum_{k=1}^{V} e^{W_k v_c}$$

Skip-Grams with Negative Sampling (SGNS)

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

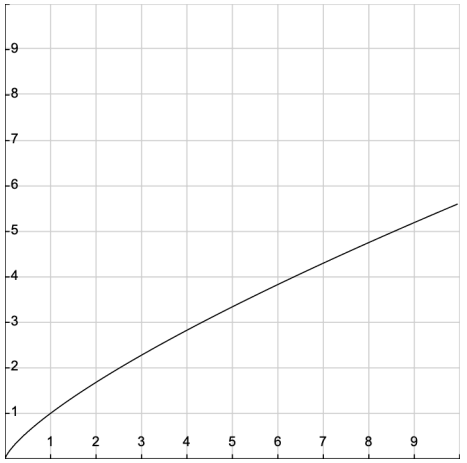
$$P(c) = \frac{\#c}{\sum_{c' \in V_C} (\#c')}$$

- Changing this distribution has a significant effect

Context Distribution Smoothing

- 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

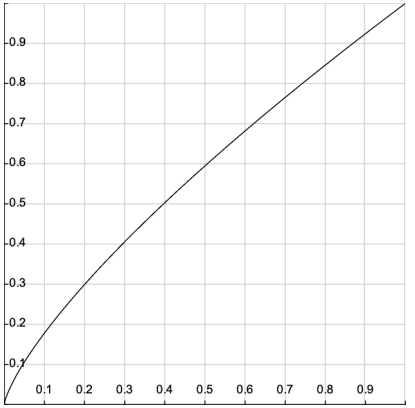
Context Distribution Smoothing

- 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!**



Math behind CBOW with Negative Sampling

- Likewise for CBOW $\hat{v} = \frac{v_{c-m} + v_{c-m+1} + \ldots + v_{c+m}}{2m}$

- Objective:

$$-\log \sigma(u_c^T \cdot \hat{v}) - \sum_{k=1}^{K} \log \sigma(-\tilde{u}_k^T \cdot \hat{v})$$

where $\{\tilde{u}_k \mid k = 1..\mathrm{K}\}$ is sampled from vocabulary (also use context distribution smoothing)

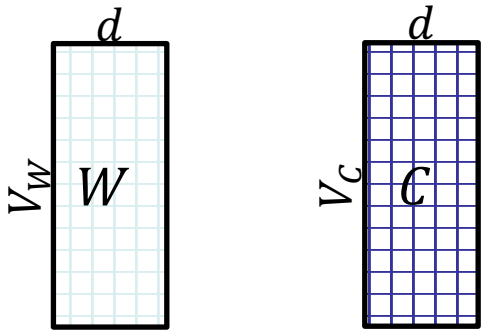
- Rather than:

$$-u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})$$

regular softmax loss for CBOW

What is SGNS learning?

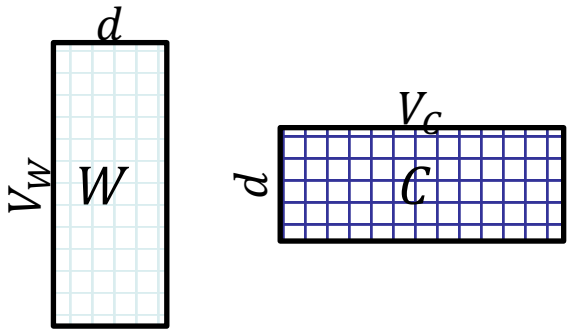
- Take SGNS's embedding matrices (W and C)



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

What is SGNS learning?

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

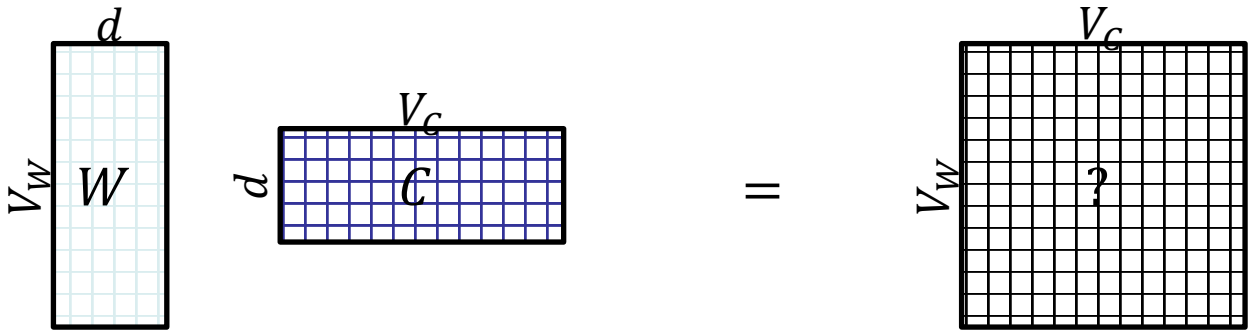


"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

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

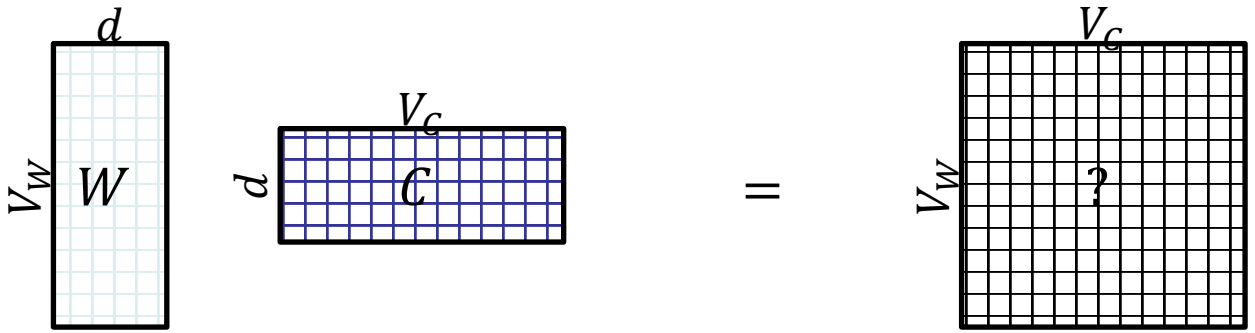
$$\vec{w} \cdot \vec{c} = ?$$



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

What is SGNS learning?

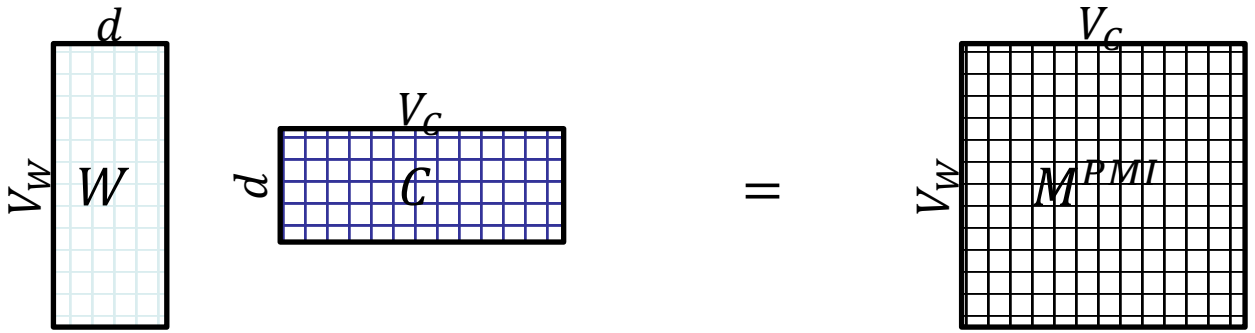
- Levy&Goldberg [2014] **proved** that for large enough d and enough iterations …



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

What is SGNS learning?

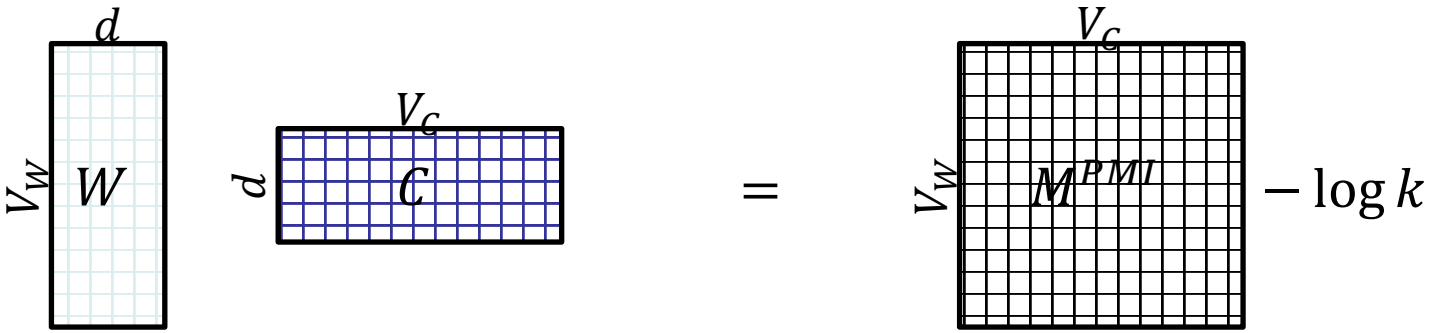
- Levy&Goldberg [2014] **proved** that for large enough d and enough iterations …
- … one obtains the word-context PMI matrix



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

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



where k is the number of negative examples

"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?

Marco Baroni, Georgiana Dinu, Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: Proc. ACL-14, 238–247, **2014**.

Jeffrey Pennington, Richard Socher, Christopher Manning. GloVe: Global Vectors for Word Representation. In: Proc. EMNLP-.14, 1532–1543, **2014**.

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 of Frequent Words
  - 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)*

Magnus Sahlgren, The Word-Space Model, Dissertation, Stockholm Univ., **2006**.

Subsampling of Frequent Words

- Counter imbalance of rare and frequent words
- Each word in the training set is discarded with a probability computed by

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

- where $f(w_i)$ is the frequency of word w_i and t is a chosen threshold

Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS **2013**

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

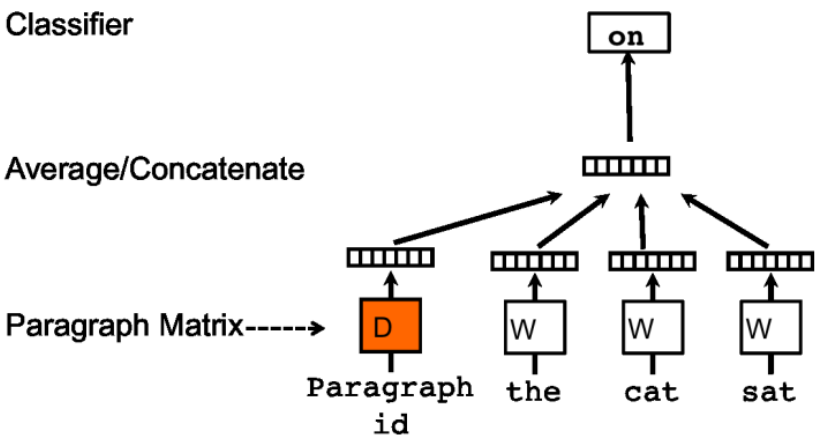
Don’t Count, Predict! ?

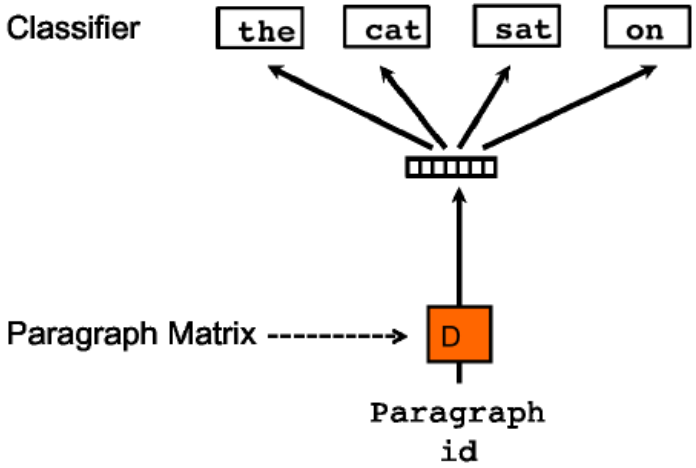
- “word2vec is better than count-based methods” [Baroni et al., 2014]
- **Hyperparameter settings** account for most of the reported gaps in count-based approaches
- Embeddings do **not** really outperform count-based methods
- No unique conclusion available

Marco Baroni, Georgiana Dinu, Germán Kruszewski. Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: Proc. ACL-14, 238–247, **2014**.

Represent the meaning of sentence/paragraph/doc

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





Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Proceedings ICML'14. **2014**.

Problem

- Learn low-dimensional, dense representations (or embeddings) for documents.
- Document embeddings can be used off-the-shelf to solve many IR applications such as,
  - Document Classification
  - Document Retrieval
  - Document Ranking

Power of 2Vec Representations

- Bag-of-words (BOW) or Bag-of-n-grams
  - Data sparsity
  - High dimensionality
  - Not/hardly capturing word order
- Latent Dirichlet Allocation (LDA)
  - Computationally inefficient for larger dataset.
- Paragraph Vector
  - Dense representation
  - Compact representation
  - Captures word order
  - Efficient to estimate

Paragraph Vector

- Learn document embedding by predicting the next word in the document using the context of the word and the ('unknown') document vector as features.
- Resulting vector captures the topic of the document.
- Update the document vectors, but not the word vectors [Le et al.]
- Update the document vectors, along with the word vectors [Dai et al.]
  - Improvement in the accuracy for document similarity tasks.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Proceedings ICML'14. **2014**.

Dai, A.M., Olah, C., Le, Q.V., Corrado, G.S.: Document embedding with paragraph vectors. In: NIPS Deep Learning Workshop. **2014**

Doc2Sent2Vec Idea - Being granular helps

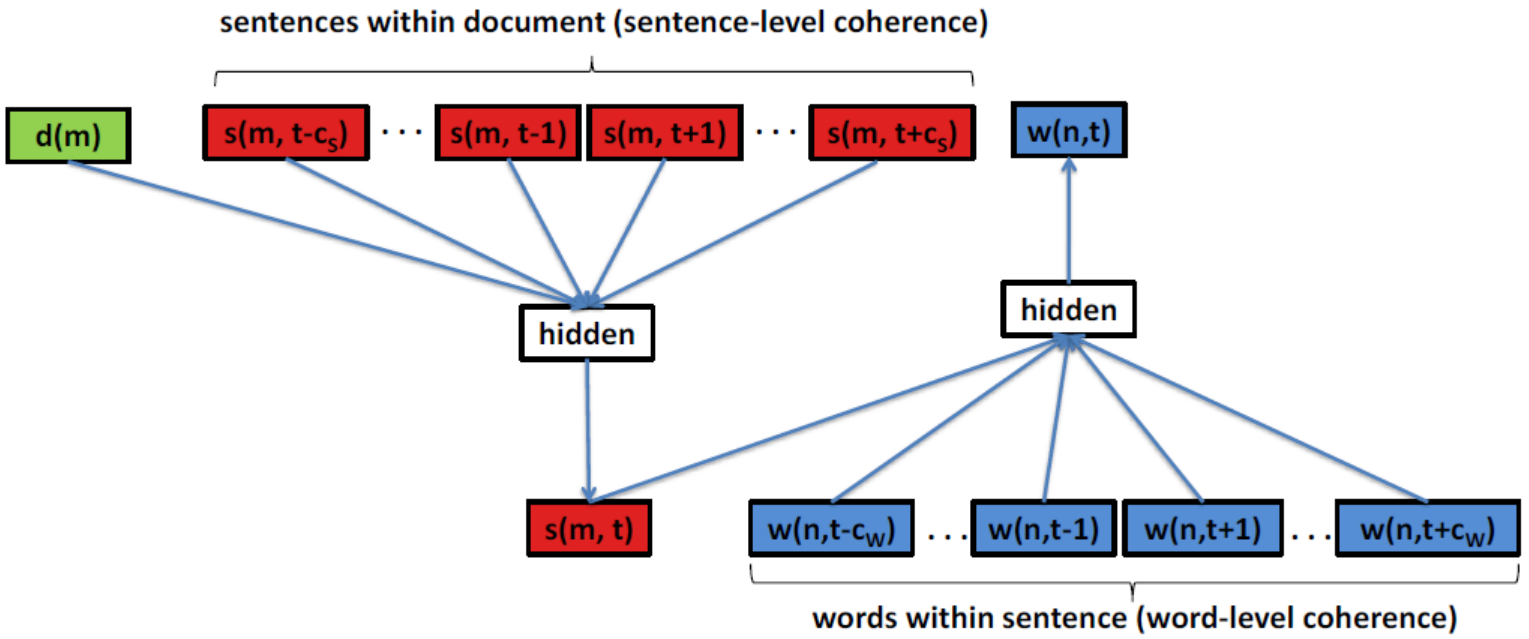
- Should we learn the document embedding from the word context directly?
- Can we learn the document embedding from the sentence context?
  - Explicitly exploit the sentence-level and word-level coherence to learn document and sentence embedding respectively.

Notation

- Document Set: $D = \{d_1, d_2, \ldots, d_M\}$; 'M' documents;
- Document: $d_m = \{s(m,1), s(m,2), \ldots, s(m,T_m)\}$; '$T_m$' sentences;
- Sentence: $s(m,n) = \{w(n,1), w(n,2), \ldots, w(n,T_n)\}$; '$T_n$' words;
- Word: w(n,t);

Doc2Sent2Vec's goal is to learn low-dimensional representations of words, sentences and documents as a continuous feature vector of dimensionality D_w, D_s and D_d respectively.

Architecture Diagram



Phase 1: Learn Sentence Embedding

Idea: Learn sentence representation from the word sequence within the sentence.

Input Features:

- Context words for target word w(n,t): $w(n,t-c_w)$, …, $w(n,t-1)$, $w(n,t+1)$, …, $w(n,t+c_w)$ (where 'c_w' is the word context size)
- Target Sentence: s(m,n) (where 'm' is the document id)

Output: w(n,t)

Task: Predict the target word using the concatenation of word vectors of context words along with the sentence vector as features.

- Maximize the word likelihood:

$$L_{word} = P(w(n,t) \mid w(n,t-c_w), \ldots, w(n,t-1), w(n,t+1), \ldots, w(n,t+c_w), s(m,n))$$

Phase 2: Learn Document Embedding

Idea: Learn document representation from the sentence sequence within the document.

Input Features:

- Context sentences for target sentence s(m,t): $s(m,t-c_s)$, …, $s(m,t-1)$, $s(m,t+1)$, …, $s(m,t+c_s)$ (where 'c_s' is the sentence context size)
- Target Document: d(m)

Output: s(m,t)

Novel Task: Predict the target sentence using the concatenation of sentence vectors of context sentences along with the document vector as features.

- Maximize the sentence likelihood:

$$L_{sent} = P(s(m,t) | s(m,t-c_s), \ldots, s(m,t-1), s(m,t+1), \ldots, s(m,t+c_s), d(m))$$

Training

- Overall objective function: $L = L_{word} + L_{sent}$
- Use Stochastic Gradient Descent (SGD) to learn parameters
- Use Hierarchical Softmax (Mikolov et al.) to facilitate faster training

Latent Relational Structures

Processing natural language data:

- ✓ Tokenization/Sentence Splitting
- ✓ Part-of-speech (POS) tagging
- Phrase chunking
- Named entity recognition
- Coreference resolution
- Semantic role labeling

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In Proceedings ICML '08. pp. 160–167. **2008**.

Phrase Chunking

- Identifies phrase-level constituents in sentences

  [NP Boris] [ADVP regretfully] [VP told] [NP his wife] [SBAR that] [NP their child] [VP could not attend] [NP night school] [PP without] [NP permission] .

- Useful for filtering: identify e.g. only noun phrases, or only verb phrases
- Used as source of features, e.g. distance, (abstracts away determiners, adjectives, for example), sequence,…
  - More efficient to compute than full syntactic parse
  - Applications in e.g. Information Extraction – getting (simple) information about concepts of interest from text documents
- Hand-crafted chunkers (regular expressions/finite automata)
- HMM/CRF-based chunk parsers derived from training data

Named Entity Recognition

- Identifies and classifies strings of characters representing proper nouns
- **[PER Neil A. Armstrong]** , the 38-year-old civilian commander, radioed to earth and the mission control room here: "**[LOC Houston]** , **[ORG Tranquility]** Base here; the Eagle has landed."
- Useful for filtering documents
  - "I need to find news articles about organizations in which Bill Gates might be involved…"
- Disambiguate tokens: "Chicago" (team) vs. "Chicago" (city)
- Source of abstract features
  - E.g. "Verbs that appear with entities that are Organizations"
  - E.g. "Documents that have a high proportion of Organizations"

Named Entity Recogniton: Definition

- NE involves identification of *proper names* in texts, and classification into a set of predefined categories of interest
  - Three universally accepted categories: person, location and organisation
  - Other common tasks: recognition of date/time expressions, measures (percent, money, weight etc), email addresses etc.
  - Other domain-specific entities: names of drugs, medical conditions, names of ships, bibliographic references etc

- NER ist not easy

Named Entity Classification

- Category definitions are intuitively quite clear, but there are many grey areas.
- Many of these grey area are caused by metonymy.
  - Person vs. Artefact: "The ham sandwich wants his bill." vs "Bring me a ham sandwich."
  - Organisation vs. Location : "England won the World Cup" vs. "The World Cup took place in England".
  - Company vs. Artefact: "shares in MTV" vs. "watching MTV"
  - Location vs. Organisation: "she met him at Heathrow" vs. "the Heathrow authorities"

Basic Problems in NE

- Variation of NEs – e.g. John Smith, Mr Smith, John.
- Ambiguity of NE types
  - John Smith (company vs. person)
  - May (person vs. month)
  - Washington (person vs. location)
  - 1945 (date vs. time)
- Ambiguity with common words, e.g. “may”

More complex problems in NER

- Issues of style, structure, domain, genre etc.
  - Punctuation, spelling, spacing, formatting, ….all have an impact

Dept. of Computing and Maths
Manchester Metropolitan University
Manchester
United Kingdom

> Tell me more about Leonardo
> Da Vinci

List Lookup Approach

- System that recognises only entities stored in its lists (gazetteers).
- Advantages - Simple, fast, language independent, easy to retarget
- Disadvantages – collection and maintenance of lists, cannot deal with name variants, cannot resolve ambiguity

Shallow Parsing Approach

- Internal evidence – names often have internal structure. These components can be either stored or guessed.

location:

CapWord + {City, Forest, Center}

e.g. Sherwood Forest

Cap Word + {Street, Boulevard, Avenue, Crescent, Road}

e.g. *Portobello Street*

Shallow Parsing Approach

- External evidence - names are often used in very predictive local contexts

Location:

"to the" COMPASS "of" CapWord

e.g. *to the south of* ***Loitokitok***

"based in" CapWord

e.g. *based in* ***Loitokitok***

CapWord "is a" (ADJ)? GeoWord

e.g. ***Loitokitok*** *is a friendly city*

Difficulties in Shallow Parsing Approach

- **Ambiguously capitalised words** (first word in sentence)

  [All American Bank] vs. All [State Police]

- **Semantic ambiguity**

  "John F. Kennedy" = airport (location)

  "Philip Morris" = organisation

- **Structural ambiguity**

  [Cable and Wireless] vs. [Microsoft] and [Dell]

  [Center for Computational Linguistics] vs. message from [City Hospital] for [John Smith].

Coreference

- Identify all phrases that refer to each entity of interest – i.e., group mentions of concepts
- **[Neil A. Armstrong]** , **[the 38-year-old civilian commander]**, radioed to **[earth]**. **[He]** said the famous words, "**[the Eagle]** has landed"."
- The Named Entity Recognizer only gets us part-way…
- …if we ask, "what actions did Neil Armstrong perform?", we will miss many instances (e.g., "He said…")
- Coreference resolver **abstracts over different ways of referring to the same person**
  - Useful in feature extraction, information extraction

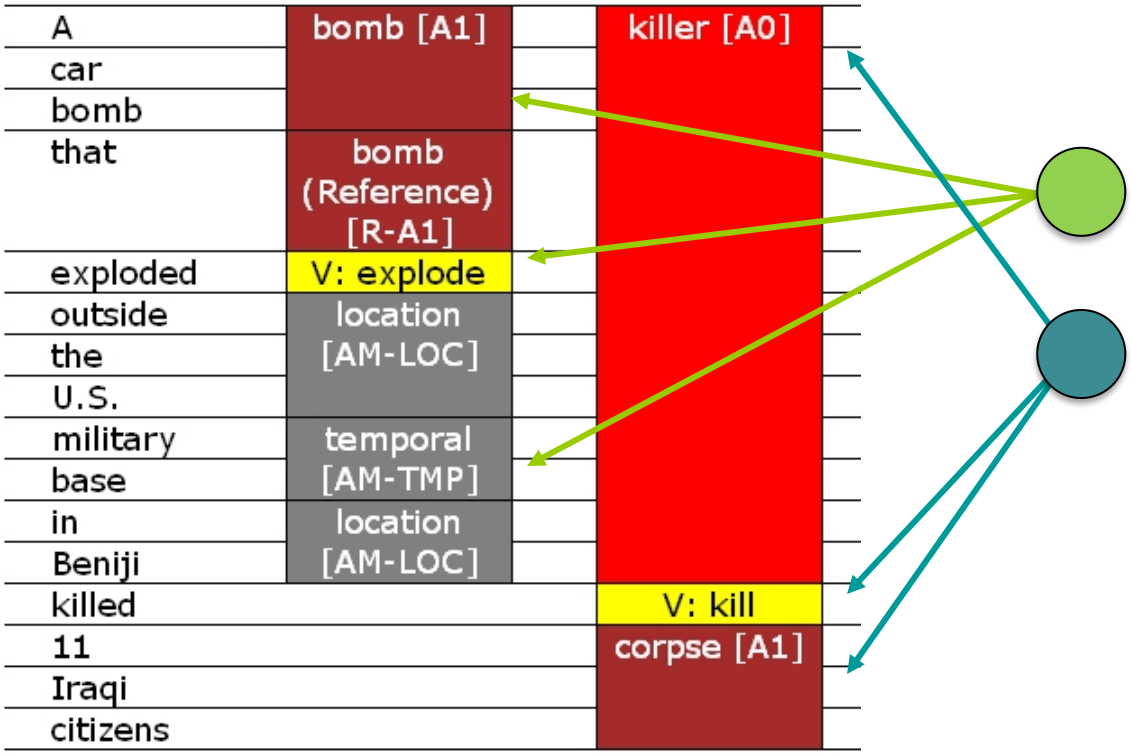
Semantic Role Labeling (SRL)

Input Text:

A car bomb that exploded outside the U.S. military base in Beniji killed 11 Iraqi citizens.

Result: Complete!

⊞ General Explanation of Argument Labels



- SRL reveals relations and arguments in the sentence (where relations are expressed as verbs)
- Cannot abstract over variability of expressing the relations – e.g. kill vs. murder vs. slay…

Why is SRL Important – *Applications*

- Question Answering
  - Q: When was Napoleon defeated?
  - Look for: [PATIENT Napoleon] [PRED **defeat-synset**] [ARGM-TMP *ANS*]

- Machine Translation

English (SVO)	Farsi (SOV)	
[AGENT The little boy]	[AGENT pesar koocholo]	boy-little
[PRED **kicked**]	[THEME toop germezi]	ball-red
[THEME the red ball]	[ARGM-MNR moqtam]	hard-adverb
[ARGM-MNR hard]	[PRED **zaad-e**]	hit-past

- Document Summarization
  - Predicates and Heads of Roles summarize content

- Information Extraction
  - SRL can be used to construct useful rules for IE

Some History

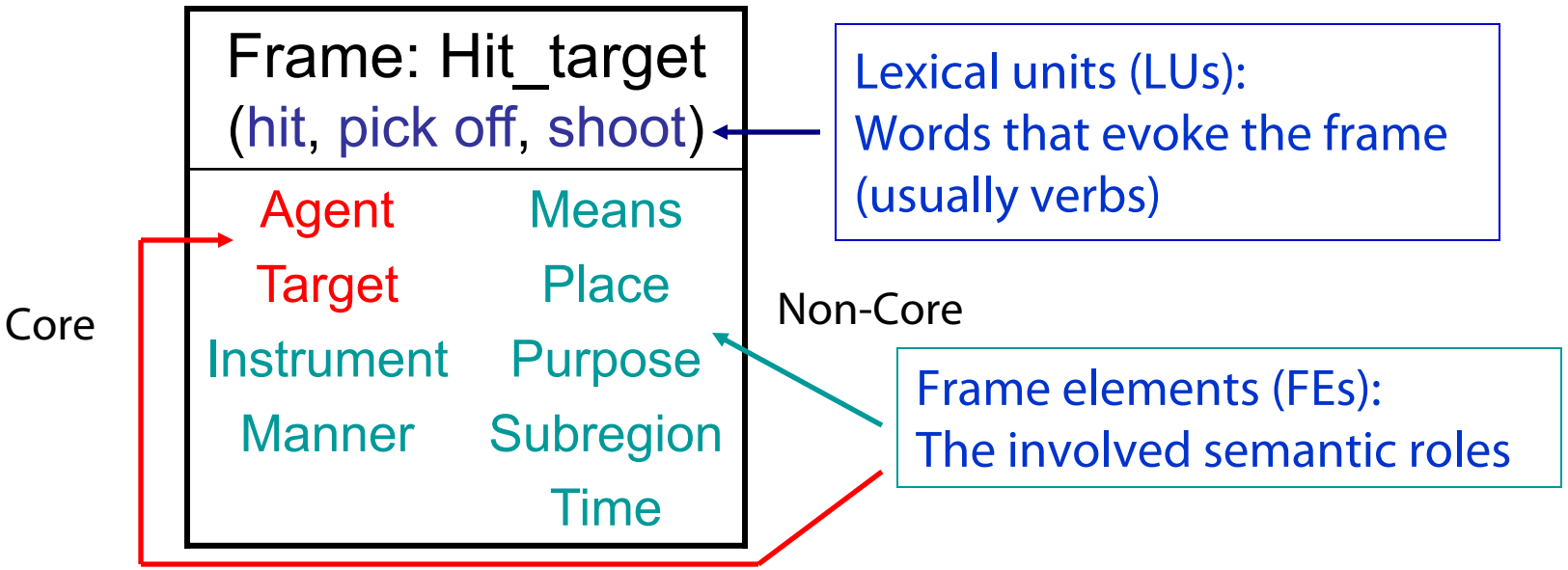
- Minsky 74, Fillmore 1976: *Frames* describe events or situations
  - Multiple participants, "props", and "conceptual roles"
  - E.g., agent, instrument, target, time, …
- Levin 1993: verb class defined by sets of frames (meaning-preserving alternations) a verb appears in
  - {*break,shatter,..*}: *Glass X's easily; John Xed the glass, …*
  - *Cut* is different: *The window broke; *The window cut.*
- FrameNet, late '90s: based on Levin's work: large corpus of sentences annotated with *frames*
- PropBank

Marvin Minky. A Framework for Representing Knowledge Marvin Minsky, MIT-AI Laboratory Memo 306, June, **1974**.

Charles J. Fillmore, Frame semantics and the nature of language Annals of the New York Academy of Sciences 280(1):20 – 32, **1976**.

Levin, B. English Verb Classes and Alternations: A Preliminary Investigation, University of Chicago Press, Chicago, IL. **1993**.

FrameNet



[**Agent** *Kristina*] ***hit*** [**Target** *Scott*] [**Instrument** *with a baseball*] [**Time** *yesterday*].

Gildea, Daniel; Jurafsky, Daniel. "Automatic Labeling of Semantic Roles". Computational Linguistics. 28 (3): 245–288. **2002**.

Proposition Bank (PropBank)

- Transfer sentences to propositions
  - Kristina hit Scott → hit(Kristina,Scott)
- Penn TreeBank → PropBank
  - Add a semantic layer on Penn TreeBank
  - Define a set of semantic roles for each verb
  - Each verb's roles are numbered

…[**A0** the company] to … *offer* [**A1** a 15% to 20% stake] [**A2** to the public]
…[**A0** Sotheby's] … *offered* [**A2** the Dorrance heirs] [**A1** a money-back guarantee]
…[**A1** an amendment] *offered* [**A0** by Rep. Peter DeFazio] …
…[**A2** Subcontractors] will be *offered* [**A1** a settlement] …

Palmer M, Kingsbury P, Gildea D. "The Proposition Bank: An Annotated Corpus of Semantic Roles". Computational Linguistics. 31 (1): 71–106. **2005**.

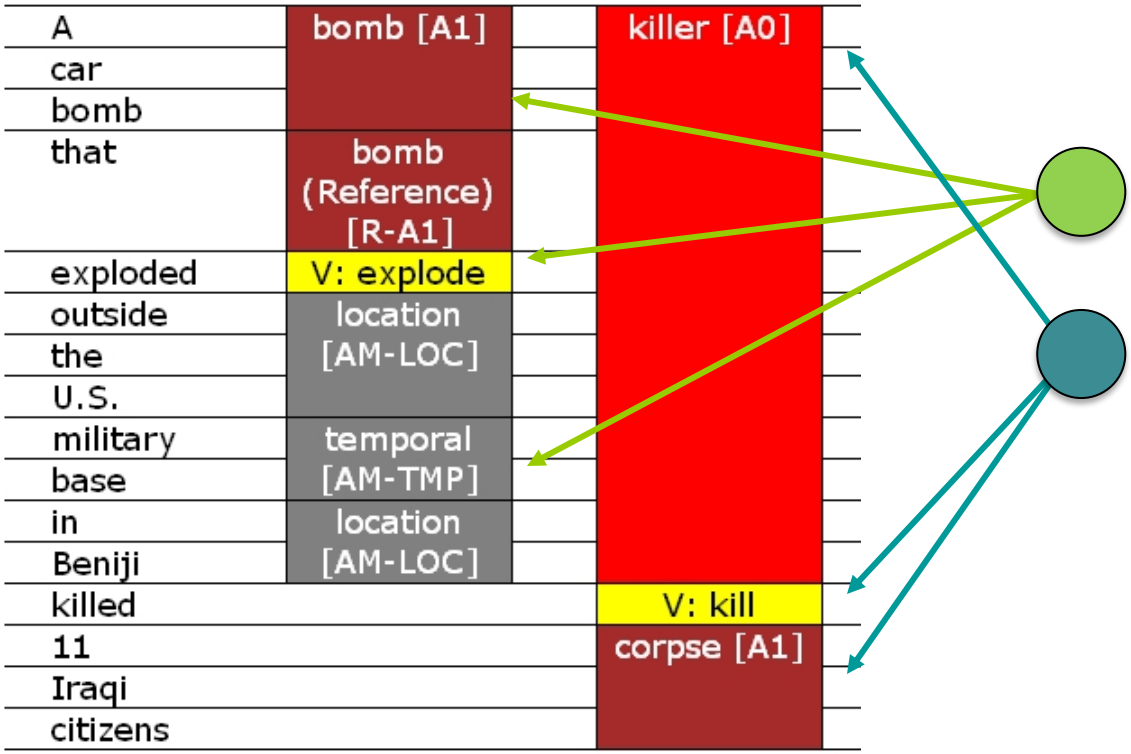
Semantic Role Labeling (SRL)

Input Text:

A car bomb that exploded outside the U.S. military base in Beniji killed 11 Iraqi citizens.

Result: Complete!

⊞ General Explanation of Argument Labels



- SRL reveals relations and arguments in the sentence (where relations are expressed as verbs)
- Cannot abstract over variability of expressing the relations – e.g. kill vs. murder vs. slay…