Web-Mining Agents

Word Semantics and Latent Relational Structures

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

Tanya Braun (Übungen)

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 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

[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]

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?

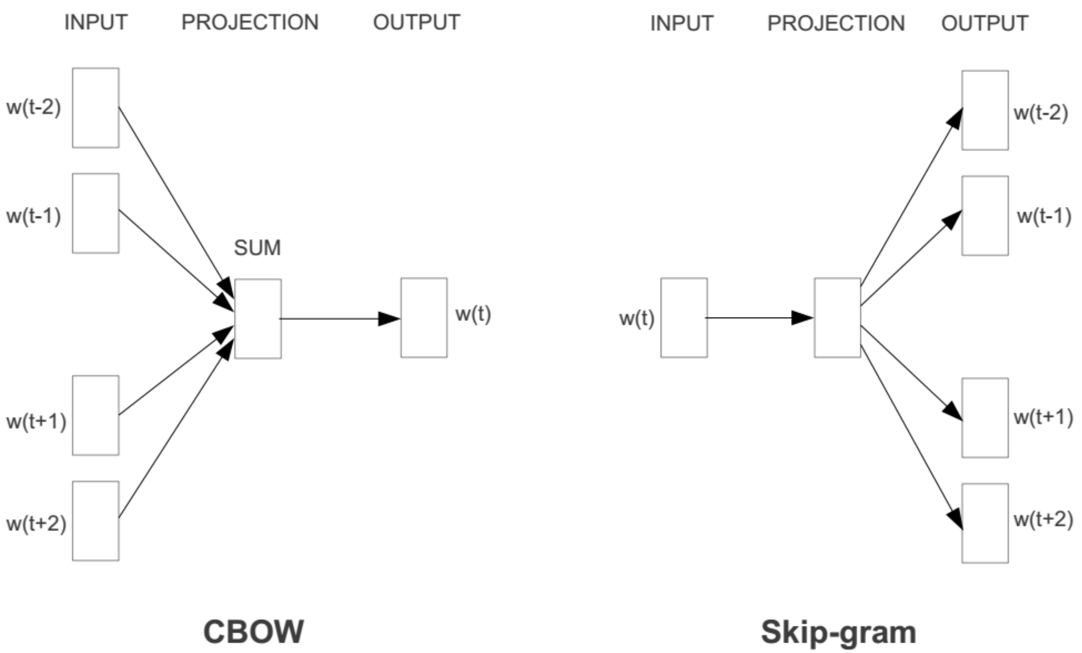
Improving Distributional Similarity with Lessons Learned from Word Embeddings, Omer Levy, Yoav Goldberg, Ido Dagan

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

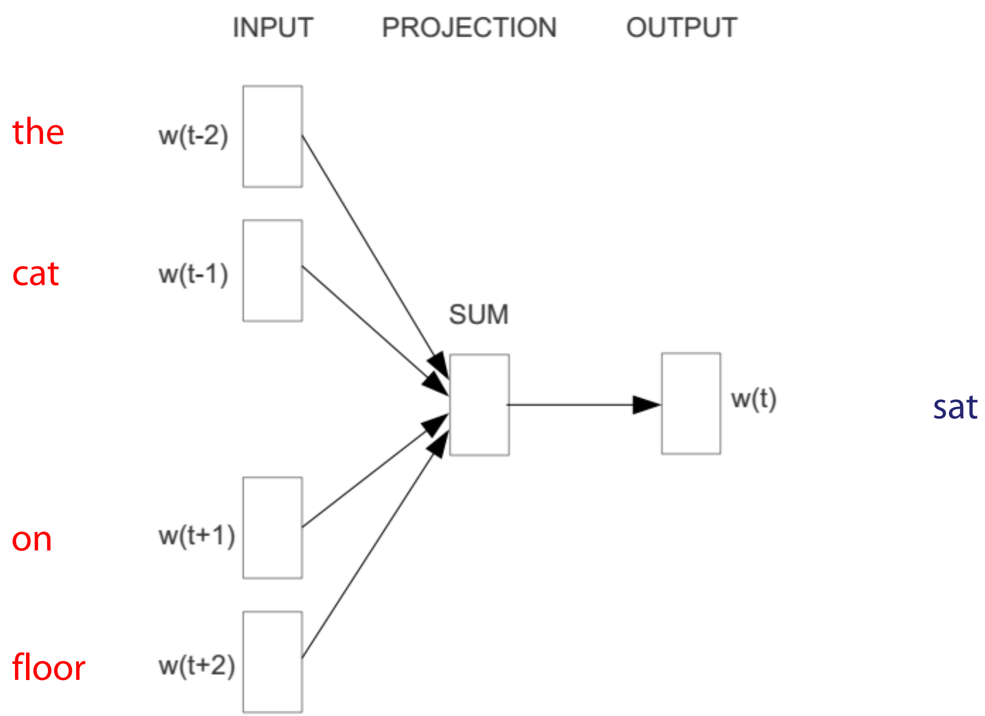
Represent the meaning of **word** – word2vec

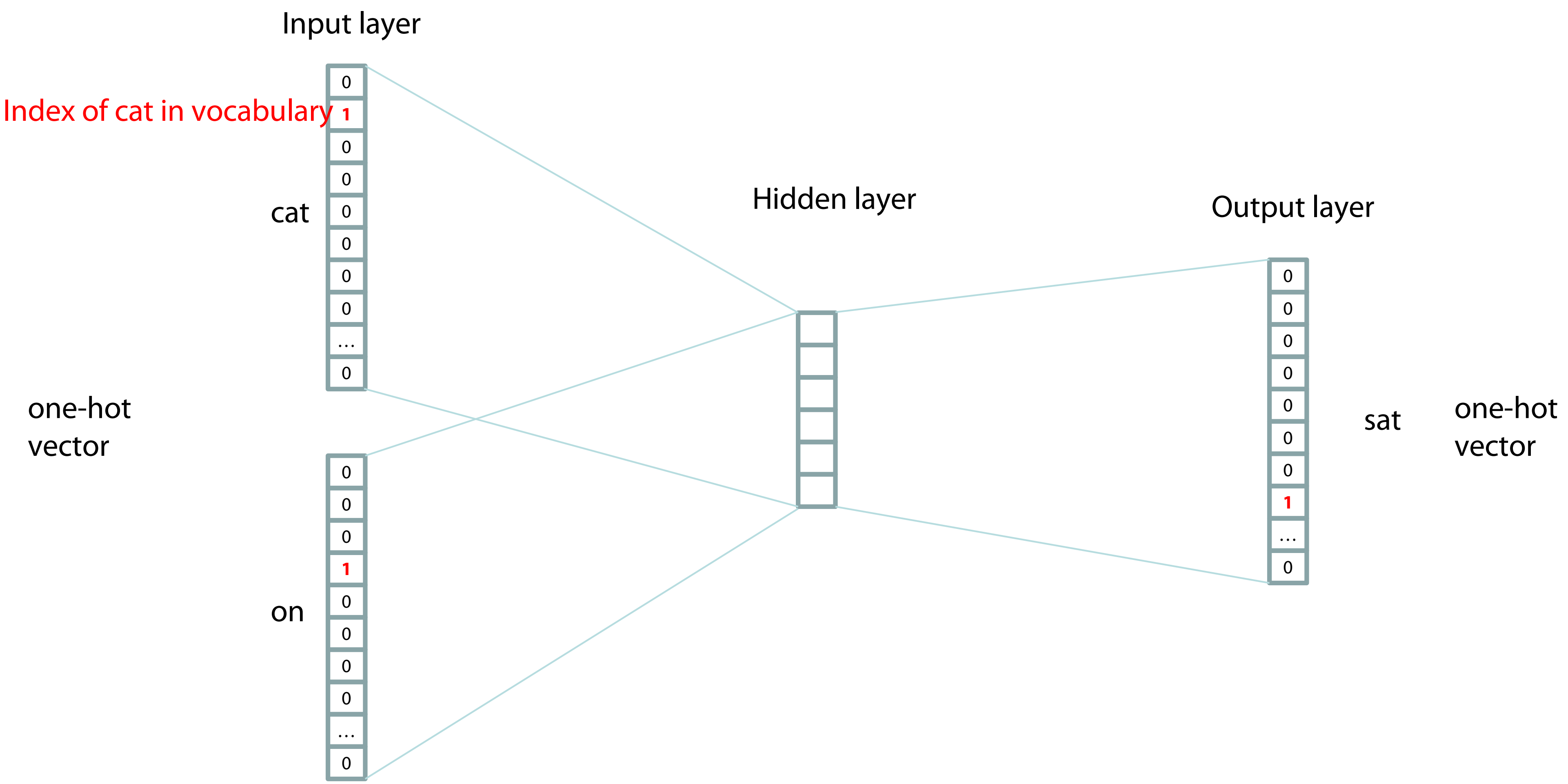
- 2 basic network models:
  - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.

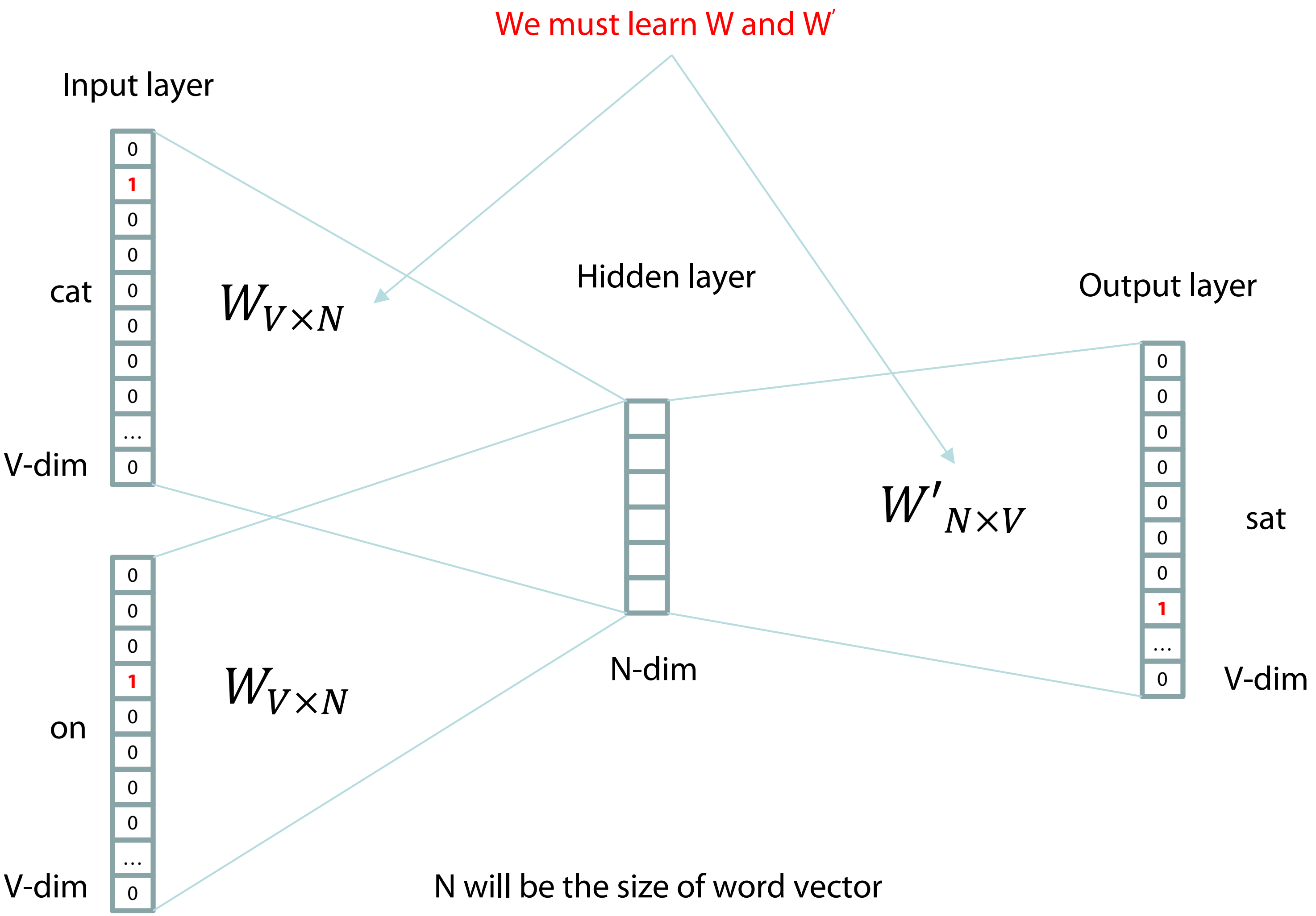


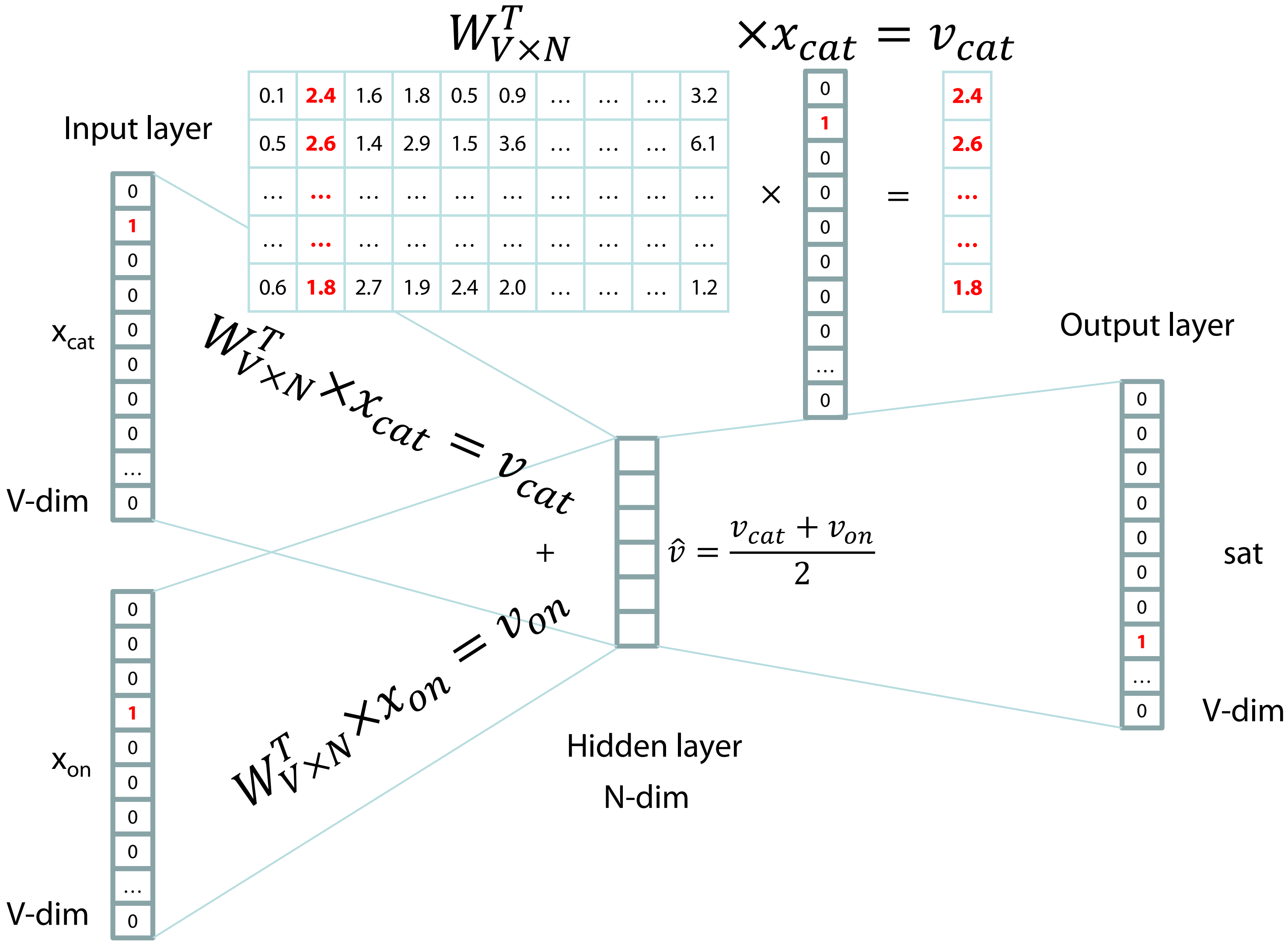
Word2vec – Continuous Bag of Word

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

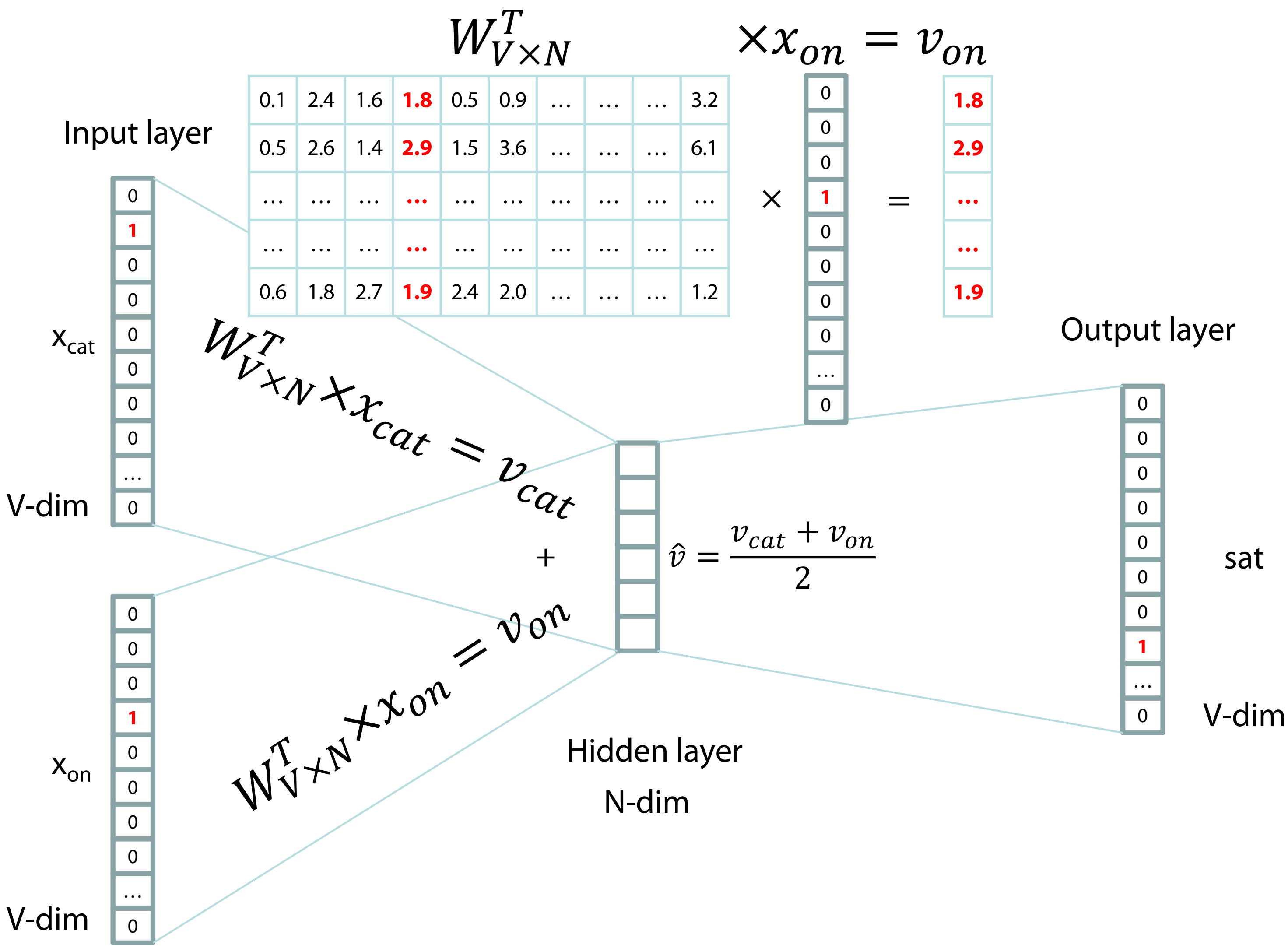



Input layer
Index of cat in vocabulary
0
1
0
0
0
0
0
0
…
0
cat
Hidden layer
Output layer
0
0
0
0
0
0
0
1
…
0
sat
one-hot
vector
0
0
0
1
0
0
0
0
…
0
on
one-hot
vector

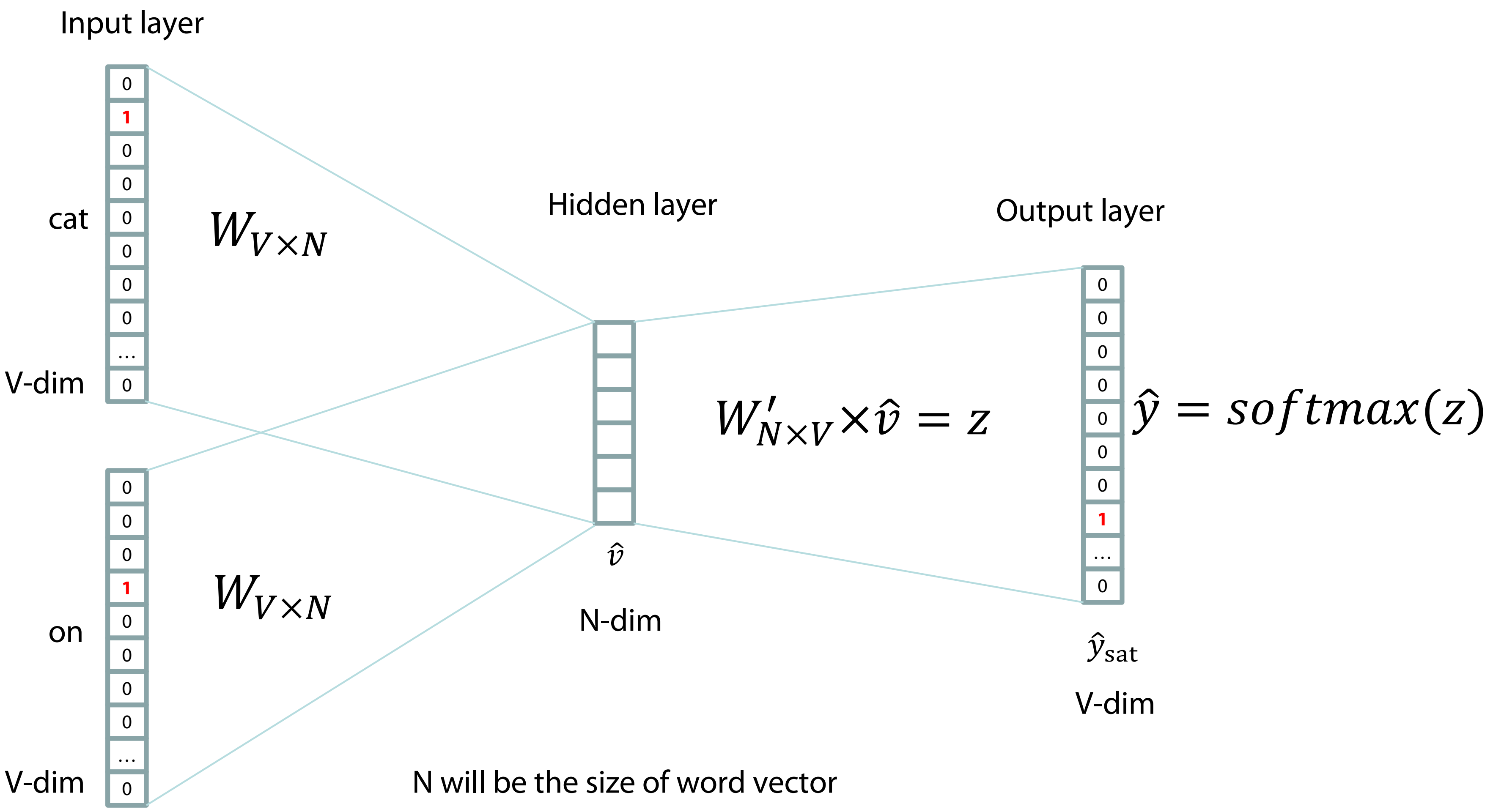

We must learn W and W′
Input layer
Hidden layer
Output layer
cat
on
sat
V-dim
V-dim
N-dim
V-dim
$W_{V\times N}$
$W_{V\times N}$
$W'_{N\times V}$
N will be the size of word vector



$W_{V\times N}^{T} \times x_{cat} = v_{cat}$
Input layer
x_{cat}
V-dim
x_{on}
V-dim
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
$W_{V\times N}^{T} \times x_{cat} = v_{cat}$
+
$W_{V\times N}^{T} \times x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
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
Output layer
x_{cat}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$W^T_{V\times N} \times x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
sat
V-dim
x_{on}
V-dim
Hidden layer
N-dim



N will be the size of word vector

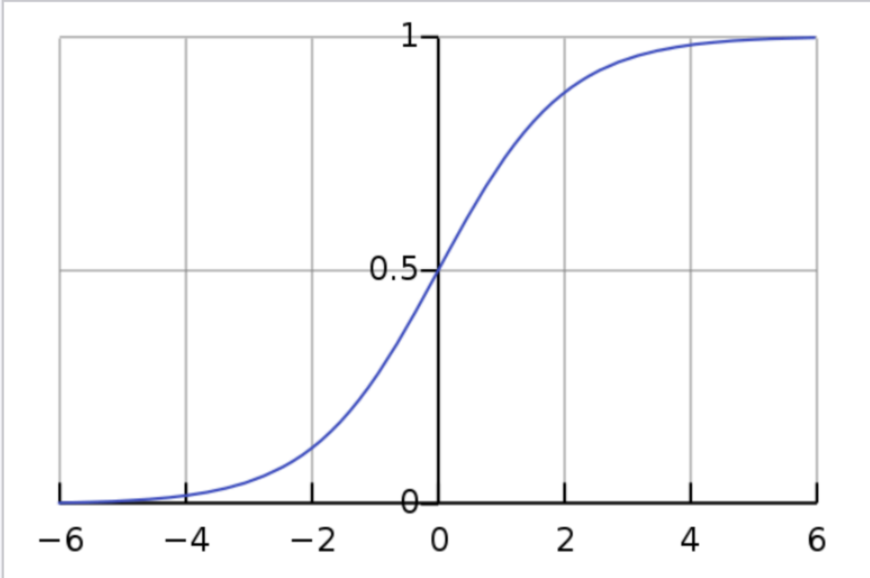
Logistic function

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. $L = 1, k = 1, x_0 = 0$

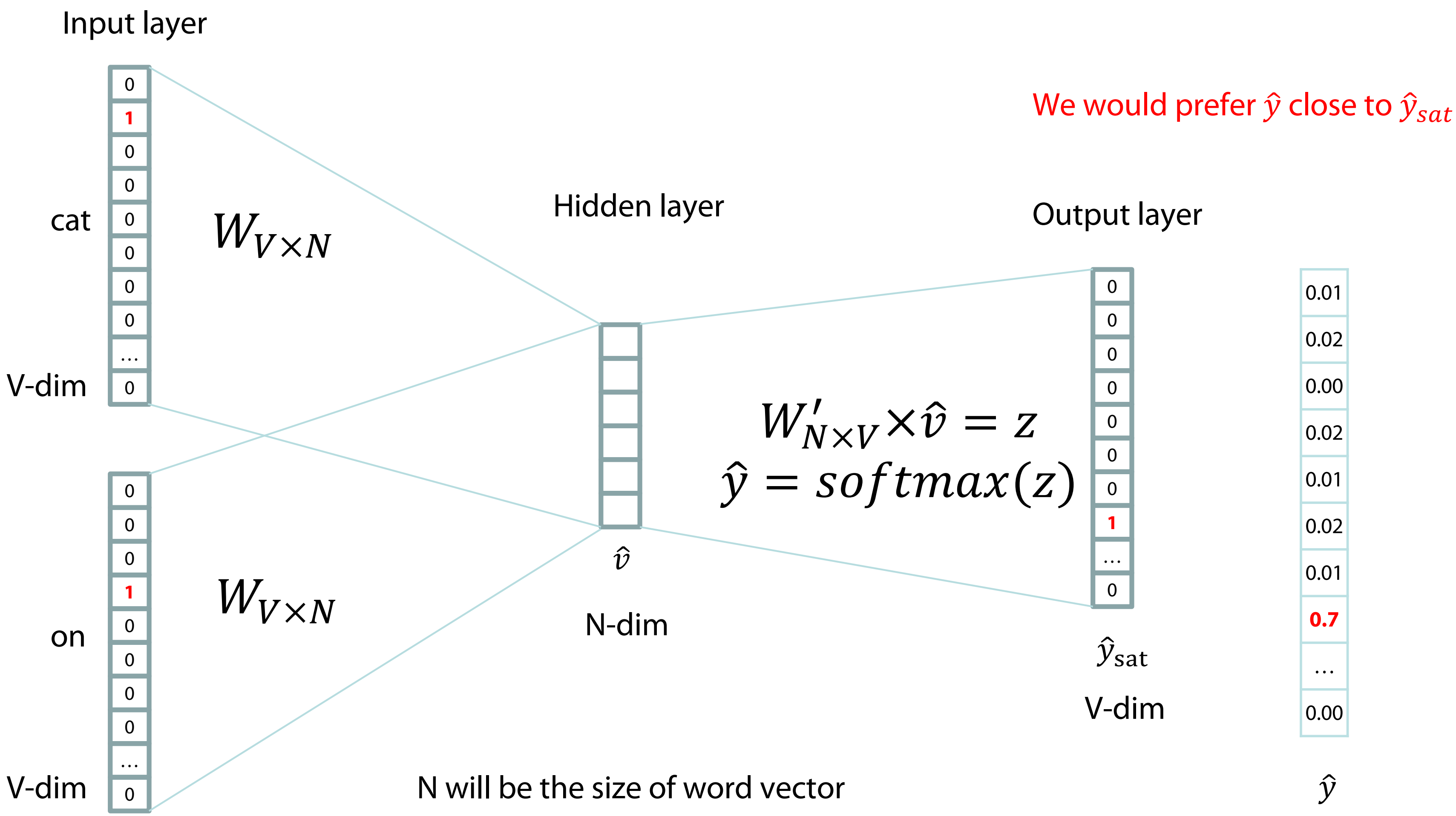
softmax($\mathbf{z}$)

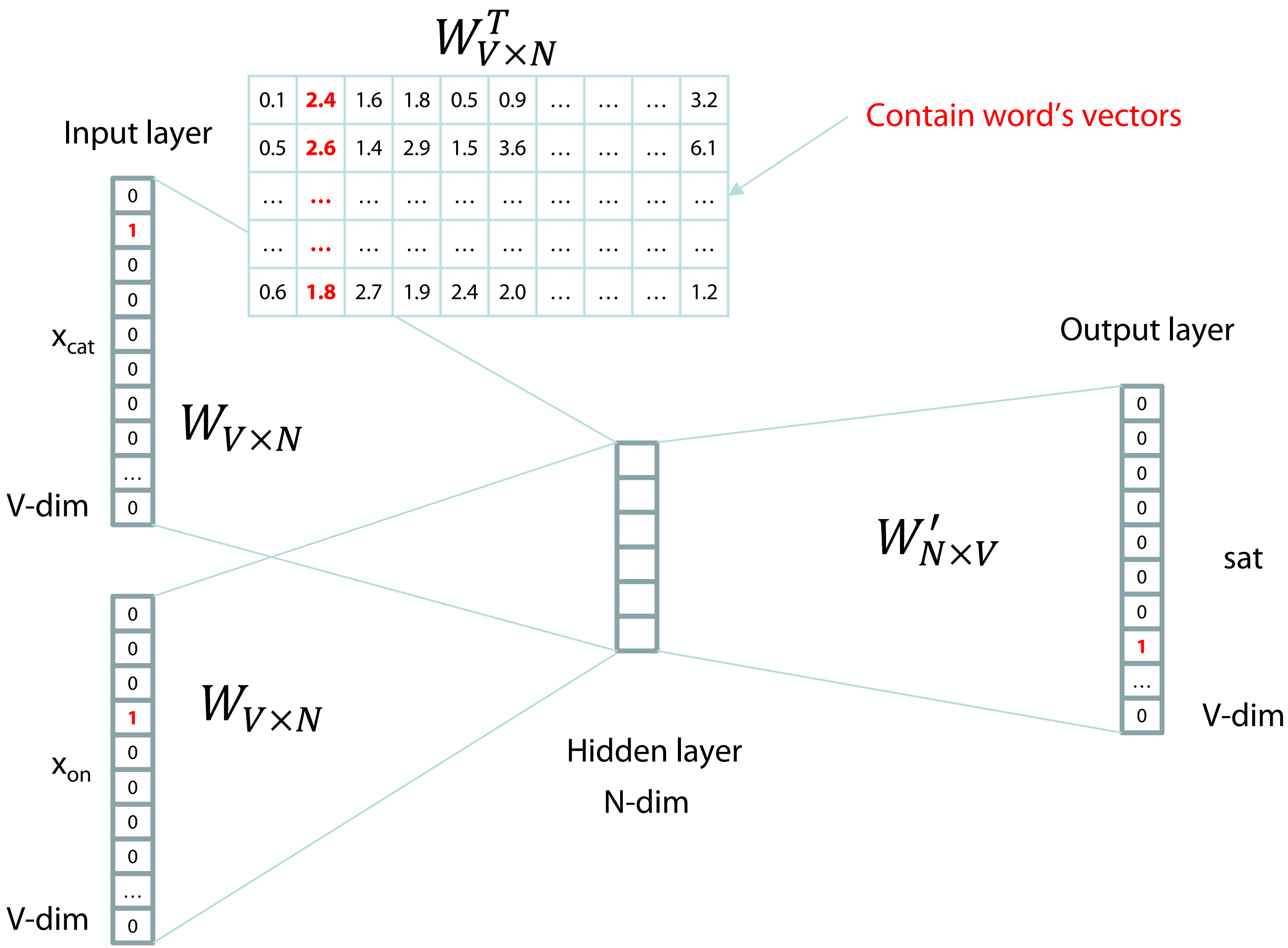
The **softmax function**, or **normalized exponential function**, is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range [0, 1] that add up to 1. The function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


Input layer
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
cat
$W_{V\times N}$
Hidden layer
Output layer
V-dim
$W'_{N\times V}\times\hat{v} = z$
$\hat{y} = softmax(z)$
$\hat{v}$
on
$W_{V\times N}$
N-dim
$\hat{y}_{\text{sat}}$
V-dim
V-dim
N will be the size of word vector
$\hat{y}$
0.01
0.02
0.00
0.02
0.01
0.02
0.01
0.7
…
0.00



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

Some interesting results

Word Analogies

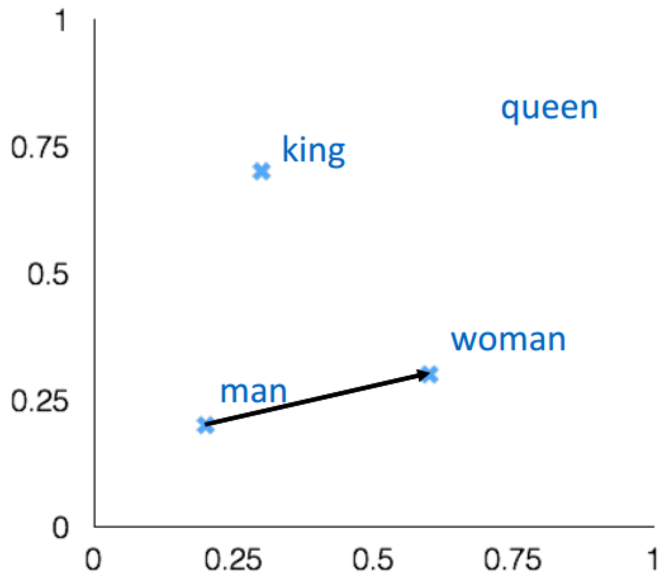
Test for linear relationships, examined by Mikolov et al. (2014)

a:b :: c:?

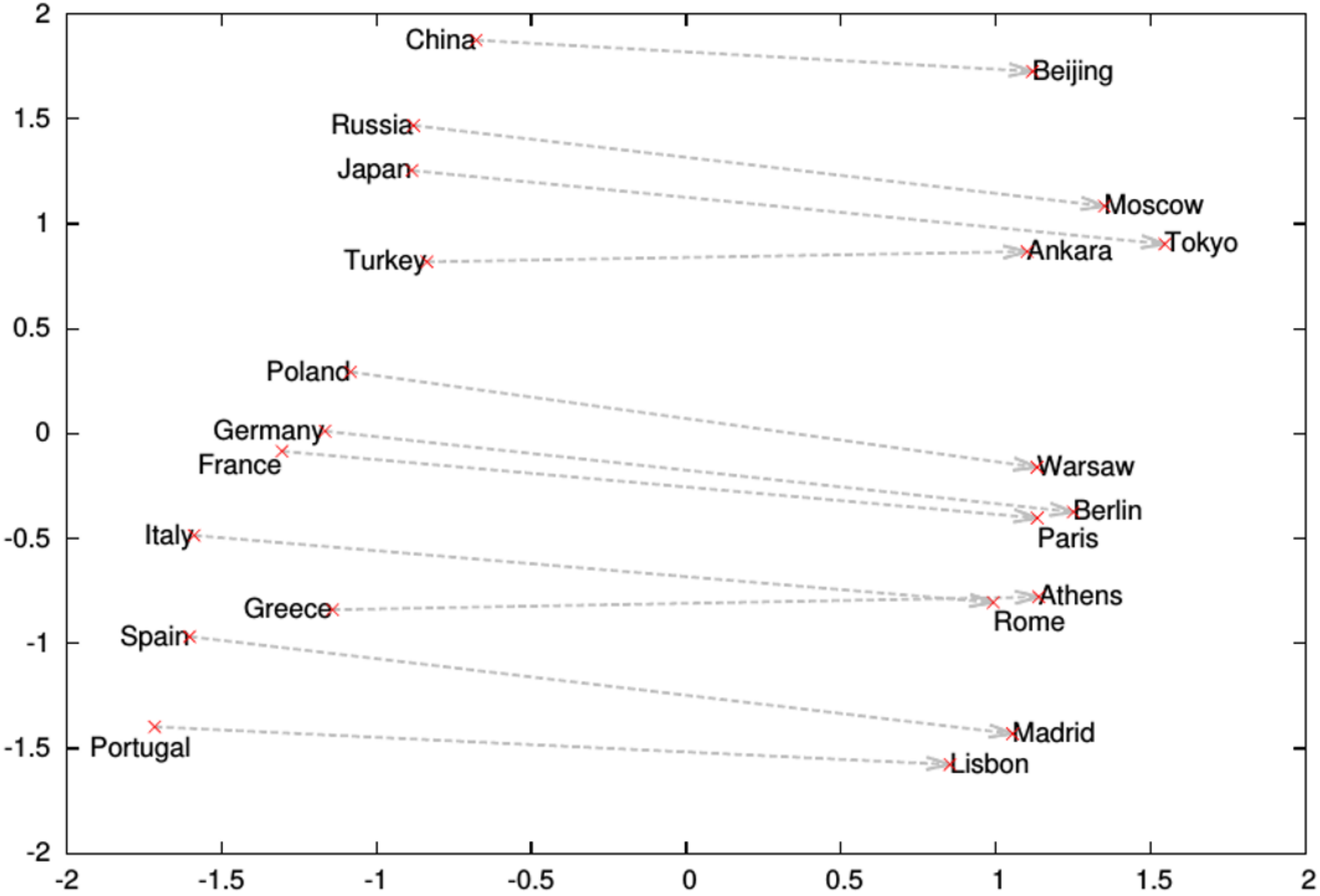
$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

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



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.

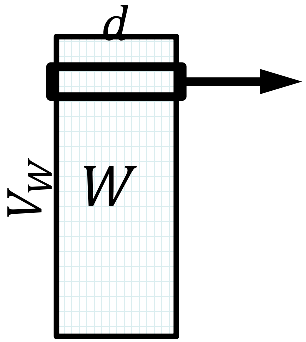
words	**contexts**
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in
…	…

D (data)

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

Skip-Grams with Negative Sampling (SGNS)

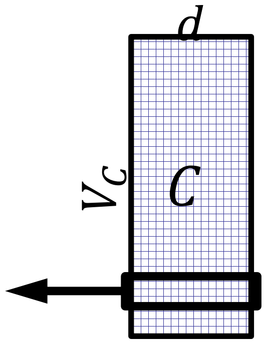
- SGNS finds a vector $\vec{w}$ for each word w in our vocabulary V_W
- Each such vector has d latent dimensions (e.g. $d = 100$)
- Effectively, it learns a matrix W whose rows represent V_W
- **Key point:** it also learns a similar auxiliary matrix C of context vectors
- In fact, each word has two embeddings



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

$\neq$

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



"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)

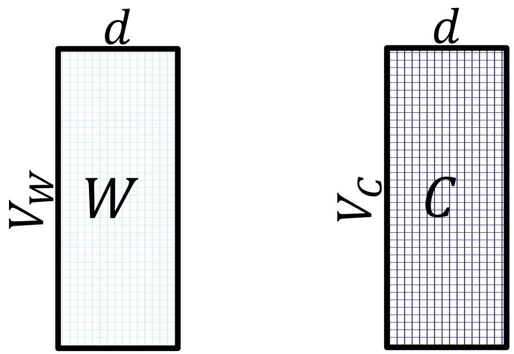
- “Negative Sampling”
- SGNS samples k contexts c' **at random** as **negative examples**
- “Random” = unigram distribution

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

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

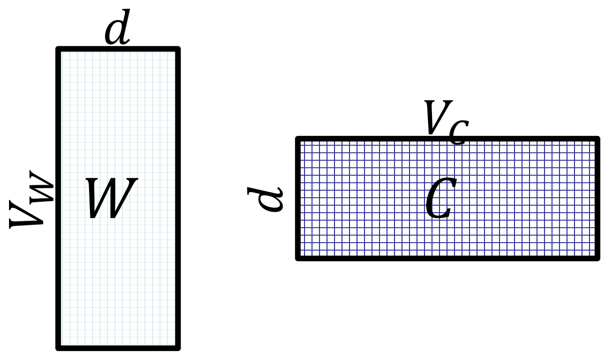
What is SGNS learning?

- 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?

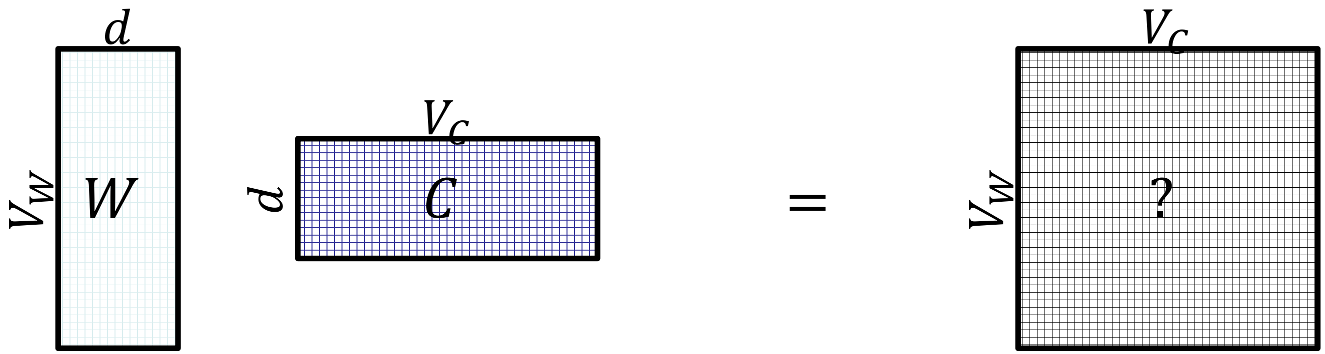


“Neural Word Embeddings as Implicit Matrix Factorization”
Levy & Goldberg, NIPS 2014

What is SGNS learning?

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

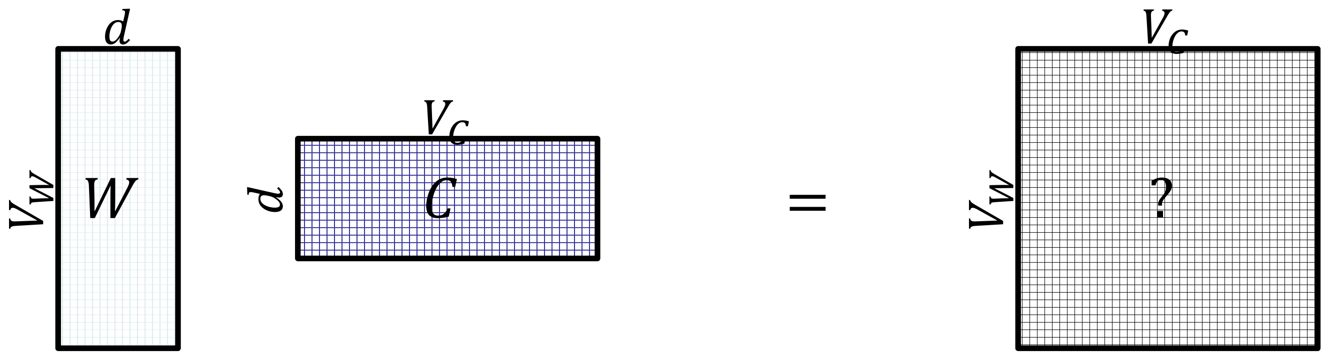
$$\vec{w} \cdot \vec{c} = ?$$



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

What is SGNS learning?

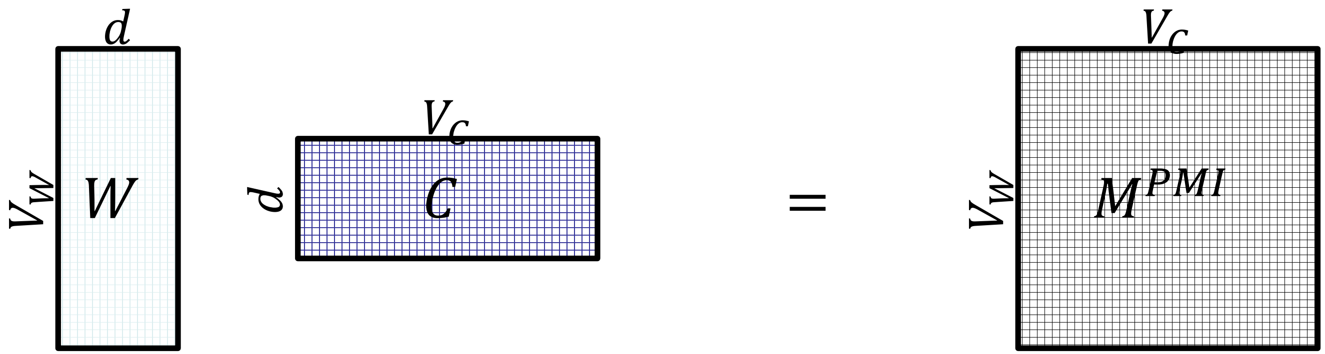
- We **proved** that for large enough d and enough iterations



"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

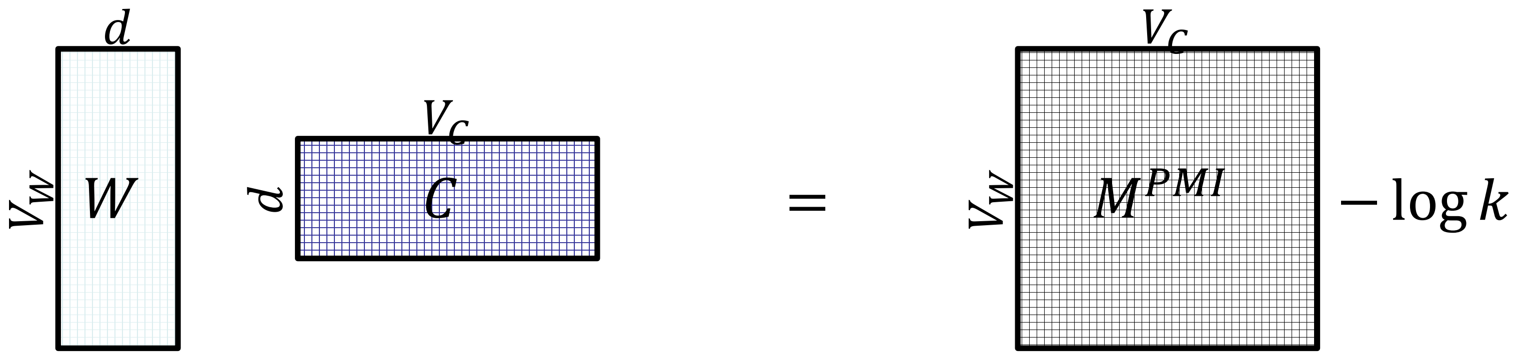


"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

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

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

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

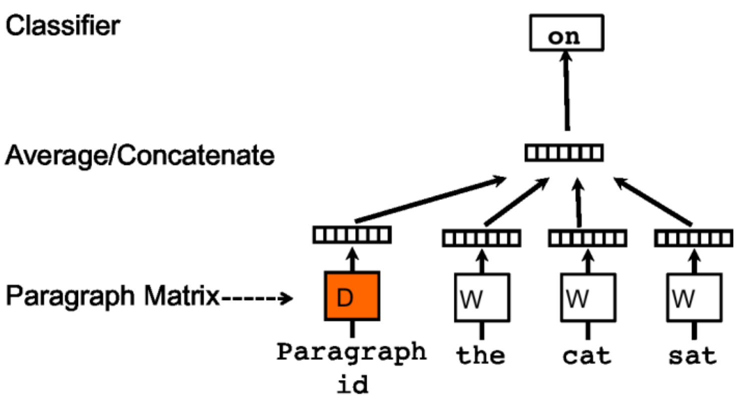
- We can **adapt** context distribution smoothing to PMI!
- Replace $P(c)$ with $P^{0.75}(c)$:

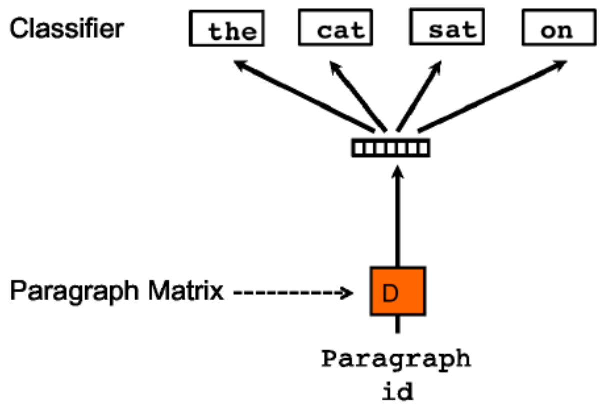
$$PMI^{0.75}(w,c) = \log \frac{P(w,c)}{P(w) \cdot \boldsymbol{P^{0.75}(c)}}$$

- 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

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

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

A	bomb [A1]	killer [A0]
car		
bomb		
that	bomb (Reference) [R-A1]	
exploded	V: explode	
outside	location [AM-LOC]	
the		
U.S.		
military	temporal [AM-TMP]	
base		
in	location [AM-LOC]	
Beniji		
killed		V: kill
11		corpse [A1]
Iraqi		
citizens		
.		

- 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: [$_{\text{PATIENT}}$ Napoleon] [$_{\text{PRED}}$ **defeat-synset**] [$_{\text{ARGM-TMP}}$ *ANS*]
- Machine Translation

English (SVO)	Farsi (SOV)	
[$_{\text{AGENT}}$ The little boy]	[$_{\text{AGENT}}$ pesar koocholo]	boy-little
[$_{\text{PRED}}$ **kicked**]	[$_{\text{THEME}}$ toop germezi]	ball-red
[$_{\text{THEME}}$ the red ball]	[$_{\text{ARGM-MNR}}$ moqtam]	hard-adverb
[$_{\text{ARGM-MNR}}$ hard]	[$_{\text{PRED}}$ **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

FrameNet [Fillmore et al. 01]

Frame: Hit_target
(hit, pick off, shoot)

Core	Non-Core
Agent	Means
Target	Place
Instrument	Purpose
Manner	Subregion
	Time

Lexical units (LUs):
Words that evoke the frame
(usually verbs)

Frame elements (FEs):
The involved semantic roles

[$_{\textbf{Agent}}$ *Kristina*] ***hit*** [$_{\textbf{Target}}$ *Scott*] [$_{\textbf{Instrument}}$ *with a baseball*] [$_{\textbf{Time}}$ *yesterday*].

Proposition Bank (PropBank) [Palmer et al. 05]

- 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] …

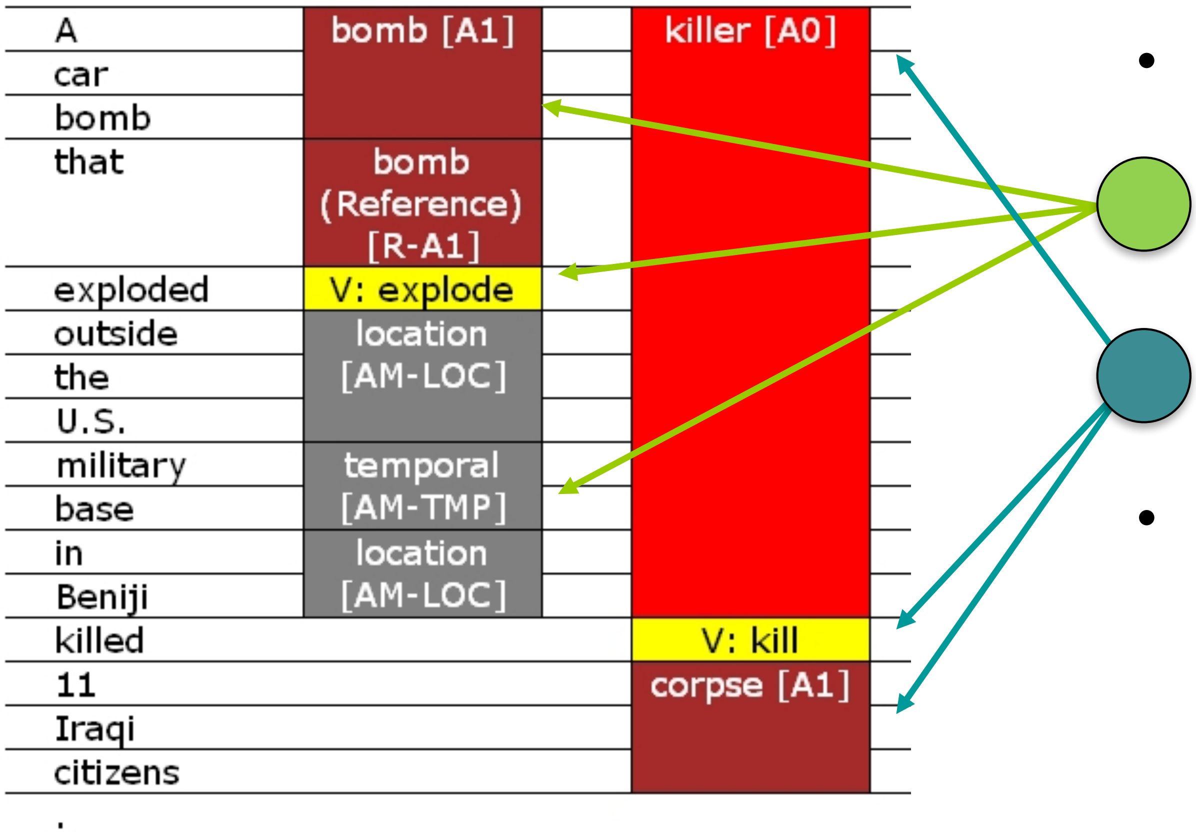
Latent Relational Structures

Input Text:

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

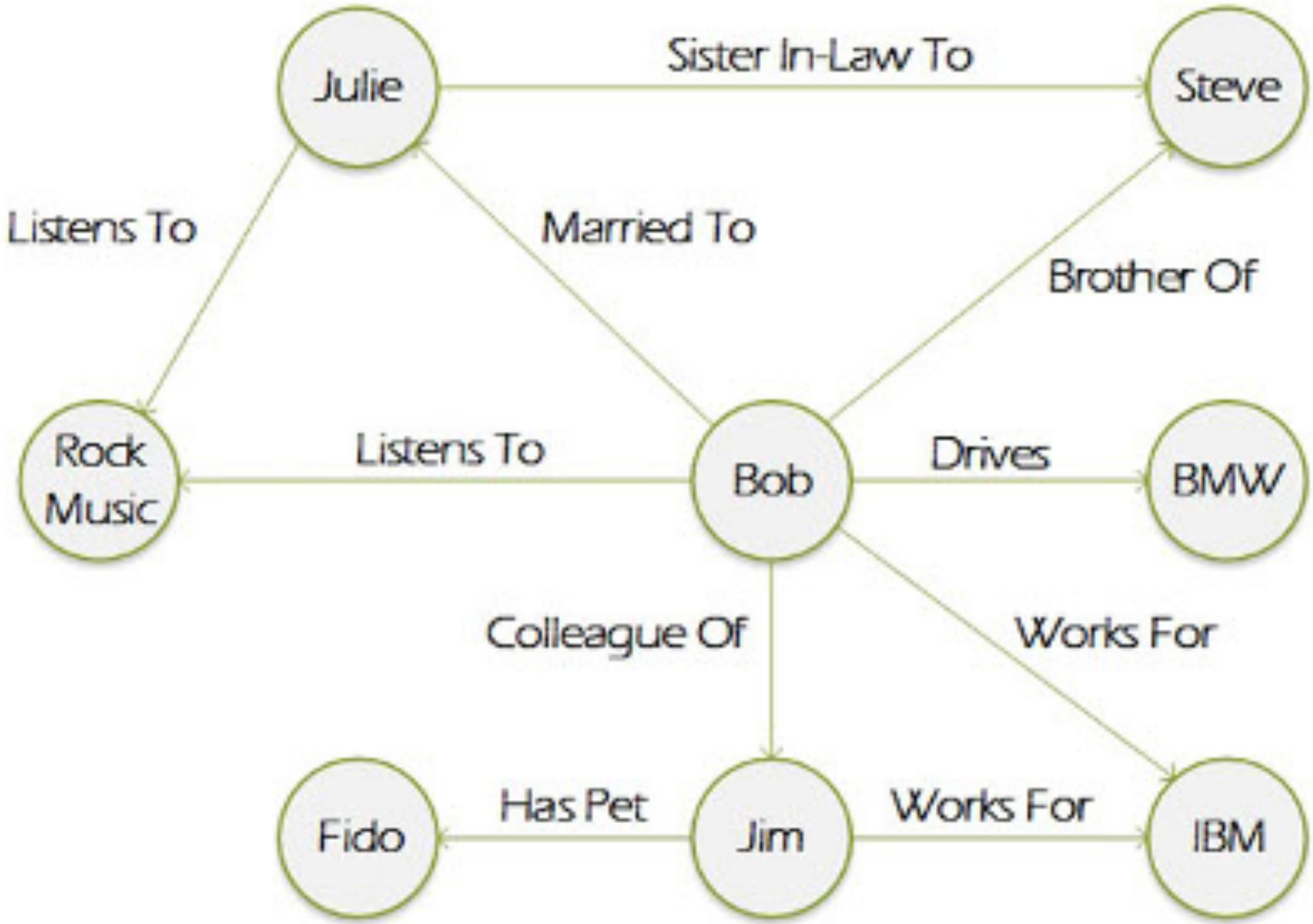
Result: Complete!

⊞ General Explanation of Argument Labels



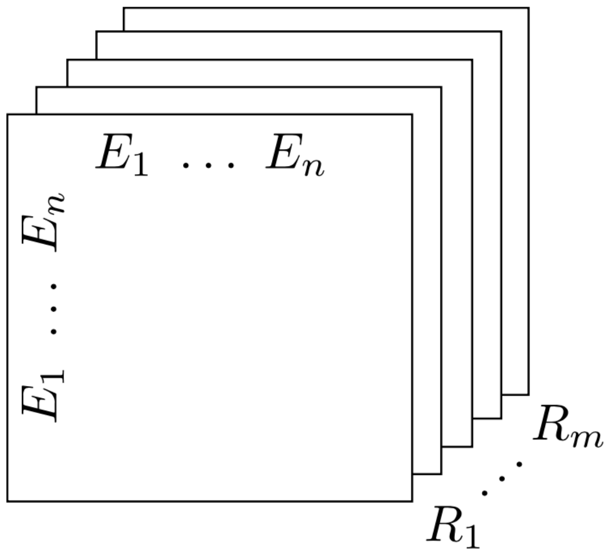
- SRL reveals relations and arguments in the sentence (where relations are expressed as verbs)
- Cannot abstract over variability of expressing the relations – e.g. kill vs. murder vs. slay…

Collective Learning on Multi-Relational Data



Towards Latent Relational Structures

- Modelling binary relations as a tensor: Two modes of a tensor refer to the entities, one mode to the relations.
- The entries of the tensor are 1 when a relation between two entities exists and 0 otherwise
- We use the RDF formalism to model relations as (`subject`, `predicate`, `object`) triples



Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel
A Three-Way Model for Collective Learning on Multi-Relational Data
In Proc. 28th International Conference on Machine Learning, **2011**

Motivation

Why Tensors?

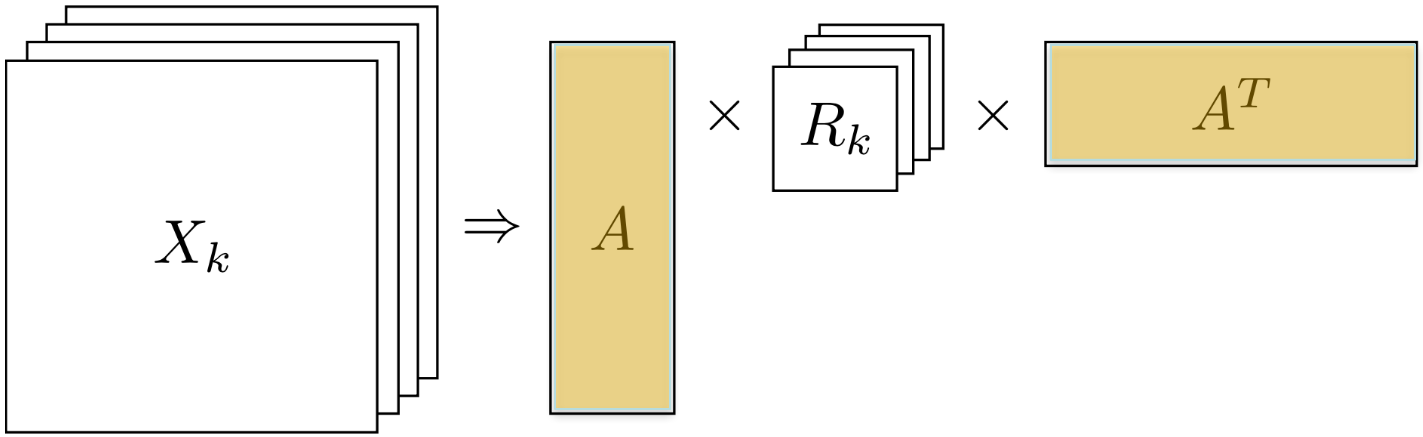
- **Modelling simplicity**: Multiple binary relations can be expressed straightforwardly as a three-way tensor
- **No structure learning**: Not necessary to have information about independent variables, knowledge bases, etc. or to infer it from data
- **Expected performance**: Relational domains are high-dimensional and sparse, a setting where factorization methods have shown very good results

Tensor Factorization with Rescal

- RESCAL takes the inherent structure of dyadic relational data into account, by employing the tensor factorization

$$X_k \approx A R_k A^T$$

- A is a $n \times r$ matrix, representing the *global* entity-latent-component space
- R_k is an *asymmetric* $r \times r$ matrix that specifies the interaction of the latent components per predicate



Solving Relational Learning Tasks

- **Link Prediction**: To predict the existence of a relation between two entities, it is sufficient to look at the rank-reduced reconstruction of the appropriate slice AR_kA^T
- **Collective Classification**: Can be cast as a link prediction problem by including the classes as entities and adding a `classOf` relation. Alternatively, standard classification algorithms could be applied to the entites' latent-component representation A
- **Link-based Clustering**: Since the entities' latent-component representation is computed considering all relations, Link-based clustering can be done by clustering the entities in the latent-component space A

Compute the Factorization

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}.$$

- To compute the factorization, we solve the optimization problem

$$\arg\min_{A,R_k} loss(A, R_k) + reg(A, R_k)$$

where $loss$ is the loss function

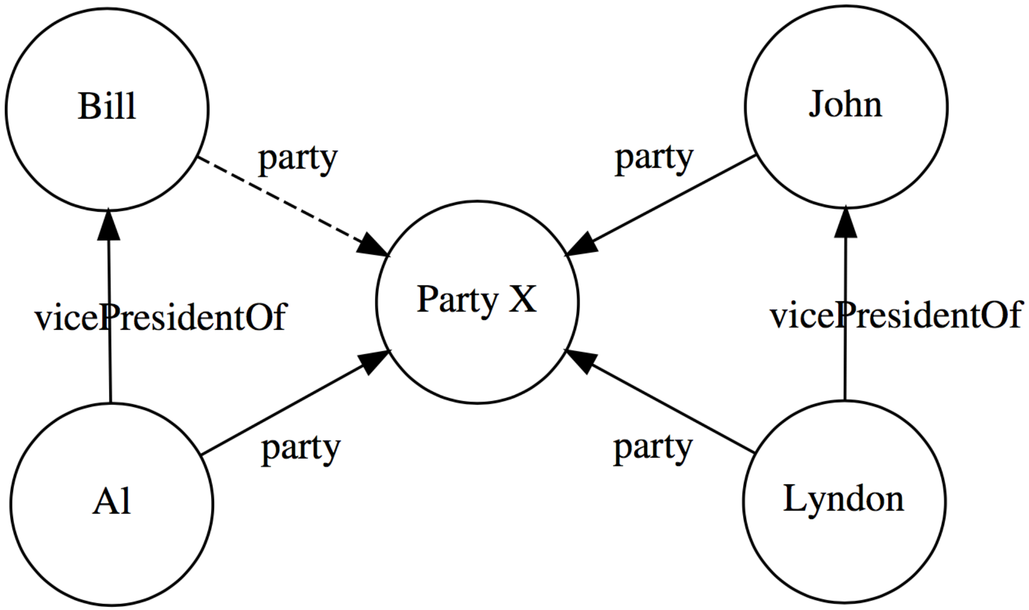
$$loss(A, R_k) = \frac{1}{2}\sum_k \|\mathcal{X}_k - AR_kA^T\|_F^2$$

and reg is the regularization term

$$reg(A, R_k) = \frac{1}{2}\lambda\left(\|A\|_F^2 + \sum_k \|R_k\|_F^2\right)$$

Collective Learning

- Predict party membership of US (vice) presidents

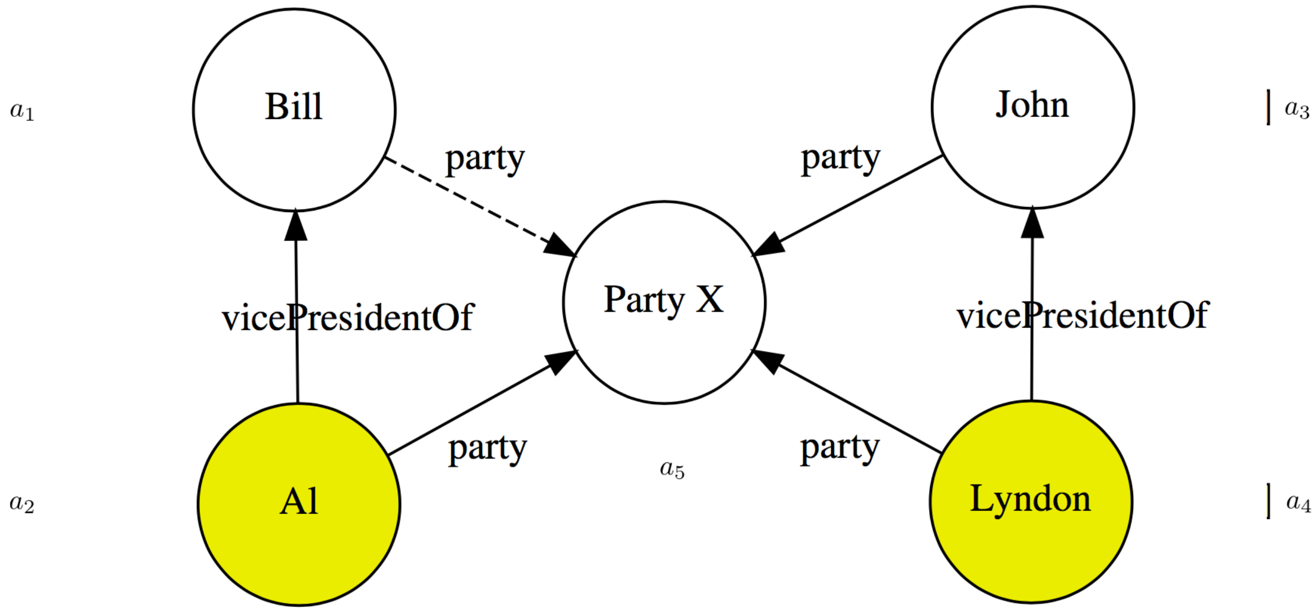


- Helpful to consider element-wise version of the loss function f

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left(\mathcal{X}_{ijk} - \boldsymbol{a}_i^T R_k \boldsymbol{a}_j \right)^2$$

Collective Learning

- Predict party membership of US (vice) presidents

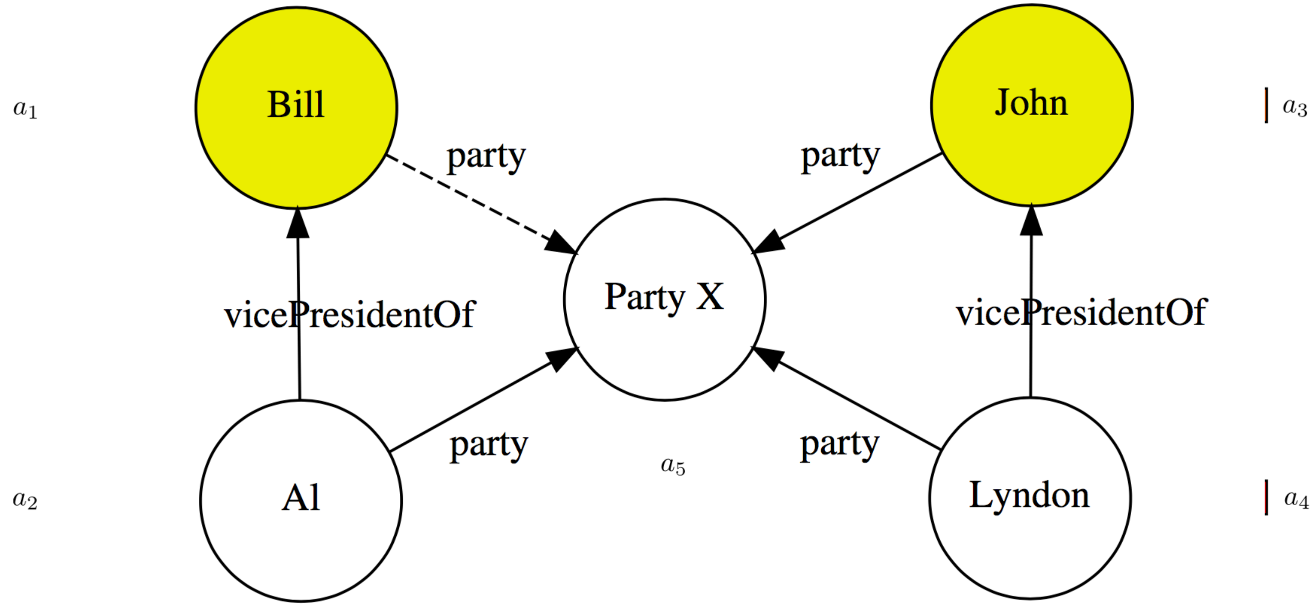


- Helpful to consider element-wise version of the loss function f

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left(\mathcal{X}_{ijk} - \boldsymbol{a}_i^T R_k \boldsymbol{a}_j \right)^2$$

Collective Learning

- Predict party membership of US (vice) presidents

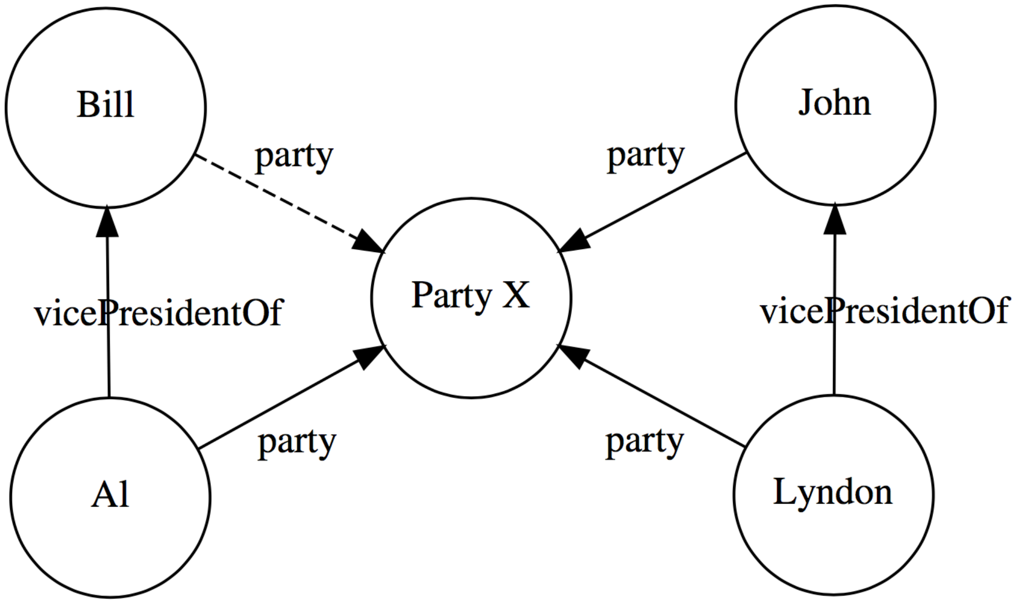


- Helpful to consider element-wise version of the loss function f

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left(\mathcal{X}_{ijk} - \boldsymbol{a}_i^T R_k \boldsymbol{a}_j \right)^2$$

Collective Learning

- Predict party membership of US (vice) presidents

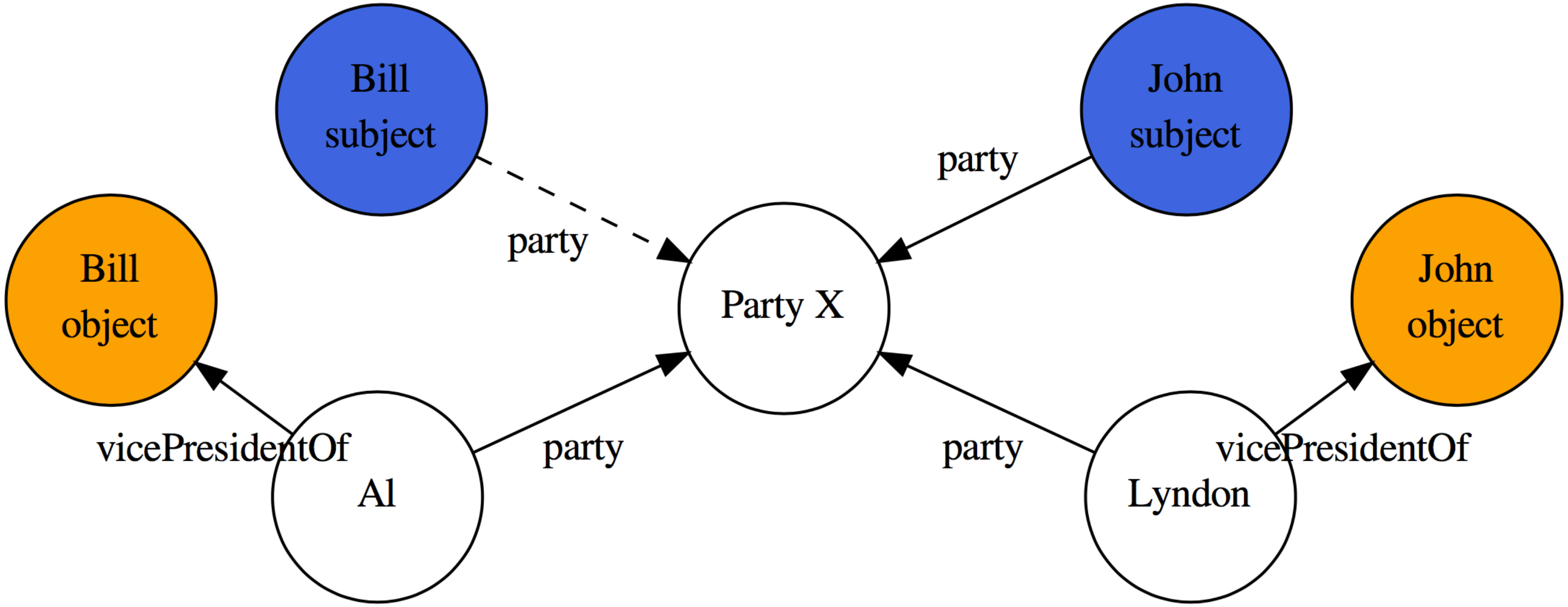


- Helpful to consider element-wise version of the loss function f

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left(\mathcal{X}_{ijk} - \boldsymbol{a}_i^T R_k \boldsymbol{b}_j \right)^2$$

Collective Learning

- Predict party membership of US (vice) presidents



- Helpful to consider element-wise version of the loss function f

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left(\mathcal{X}_{ijk} - \boldsymbol{a}_i^T R_k \boldsymbol{b}_j \right)^2$$

Collective Learning

- Collective learning is performed via the entities' latent-component representation
- Important aspect of the model: Entities have a unique latent-component representation, regardless of their occurrence as subjects or objects

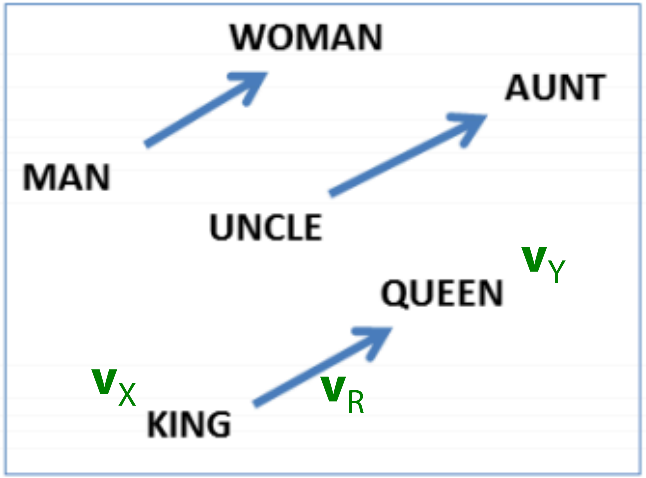
Using Learned Relational Networks for IR

- Query answering: indirect queries requiring chains of reasoning
- KB Completion: exploits redundancy in the KB + chains to infer missing facts

Freebase 15k benchmark

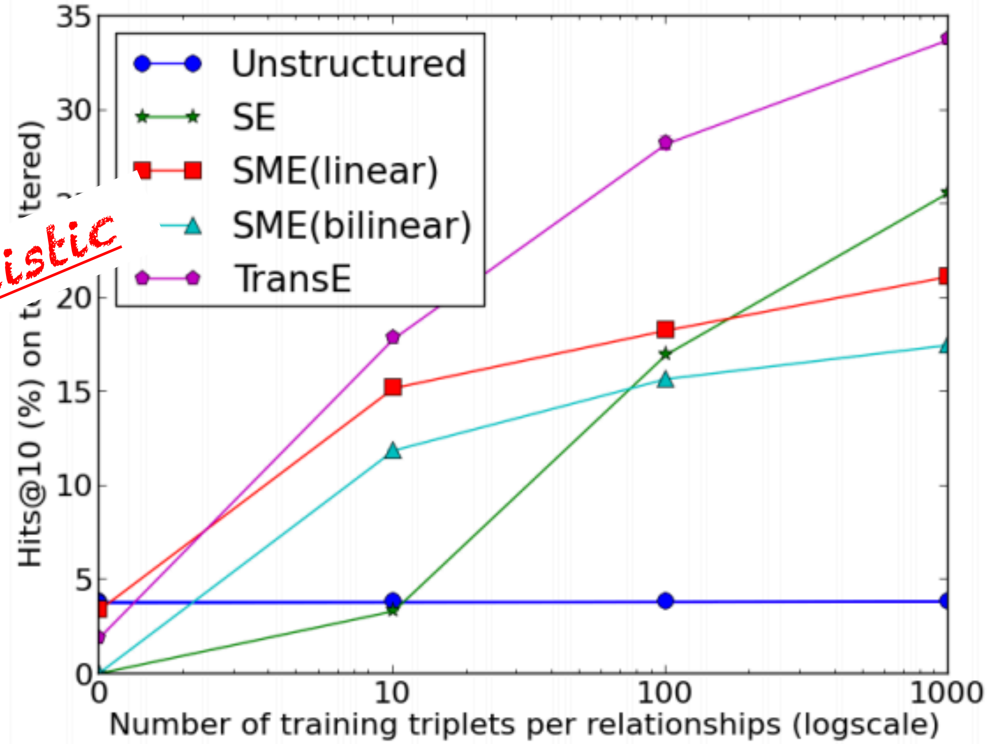
Methods	Hits@10	
Unstructured [Bordes *et al.*, 2014]	4.5	baseline method
RESCAL [Nickel *et al.*, 2011]	28.4	tensor factorization
SE [Bordes *et al.*, 2011]	28.8	deep NN embedding
SME [Bordes *et al.*, 2014]	31.3	
LFM [Jenatton *et al.*, 2012]	26.0	
TransE [Bordes *et al.*, 2013]	34.9	
ConvNets [Shi and Zhu, 2015]	37.7	
TransH [Wang *et al.*, 2014b]	45.7	
TransR [Lin *et al.*, 2015b]	48.4	
PTransE [Lin *et al.*, 2015a]	51.8	

Using Learned Relational Networks for IR



TransE: find an embedding for entitities and relations so that R(X,Y) iff $\mathbf{v}_Y$-$\mathbf{v}_X$ ~= $\mathbf{v}_R$

TransE [Bordes *et al.*, 2013]



learned probabilistic

Alternative is explicit inference rules:

uncle(X,Y) :- aunt(X,Z), husband(Z,Y).