The Challenge of Composition in Distributional and Formal Semantics

IJCNLP2017 Tutorial

Ran Tian, Koji Mineshima and
Pascual Martinez-Gomez

Self-introduction

✦Ran Tian (田 然)

– https://tianran.github.io ←Find Tutorial Slides Here!

– Research Assistant Professor at Tohoku University, Japan

– I'm interested in natural language understanding, have worked on recognizing textual entailment, distributional representations, and recently published a theory of additive composition (method of composing meaning by simply adding word vectors)

- *Logical Inference on Dependency-based Compositional Semantics*; ACL 2014
- *Learning Semantically and Additively Compositional Distributional Representations*; ACL 2016
- *The Mechanism of Additive Composition*; Machine Learning Journal 2017

My colleagues

- Koji Mineshima (峯島 宏次)
  - https://abelard.flet.keio.ac.jp/person/minesima/
  - Project Associate Professor (Ochanomizu University, Tokyo, Japan)
  - Research Area: formal semantics, semantic parsing, and natural language inference (recognizing textual entailment)

- Pascual Martínez-Gómez
  - https://researchmap.jp/pascual
  - Research Scientist at the Artificial Intelligence Research Center, AIST (Tokyo, Japan)
  - Current main interests are in Question Answering over large Knowledge Bases, Semantic Parsing, Natural Language Inferences and multi-modality.

Principle of Compositionality

The meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them. [Frege]

It is the idea that the complicated meaning of a whole sentence can be built from simpler, more basic units.

Different Layers of Meaning

basic, shallow:

Is the word "wine" more similar to "beer" than "house"?

Individual Words

Predicate-argument Structures

Who did what to whom?

Logic

Does sentence A contradicts sentence B?

complicated, deep:

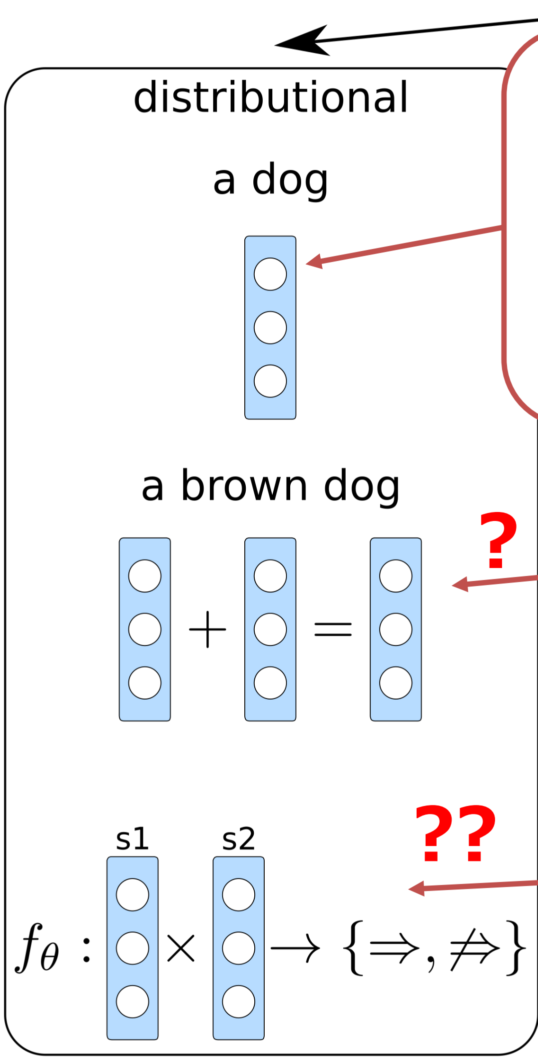
Modality, Intention, etc.

What's the intention of the speaker?

⋮

Two Approaches to Semantics

Two Approaches



Distributional semantics represents words as vectors; so word similarity can be calculated as vector similarity

Meaning Representation

Phrases? Predicate-argument structures?

Composionality

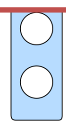
How to compose meanings from word vectors, so that inference can be conveyed?

Inference Reasoning

Two Approaches to Semantics

Two Approaches

Formal semantics represents words as symbolic predicates



The focus is on rules of combining the symbols into logical formulas

Those logical formulas can be directly used for inference and reasoning

formal

a dog

$$\exists x.\mathrm{dog}(x)$$

a brown dog

$$\exists x.\mathrm{dog}(x) \wedge \mathrm{brown}(x)$$

$$\exists xv.\mathrm{dog}(x) \wedge \mathrm{run}(v, x) \wedge \mathrm{slowly}(v)$$

$$\downarrow$$

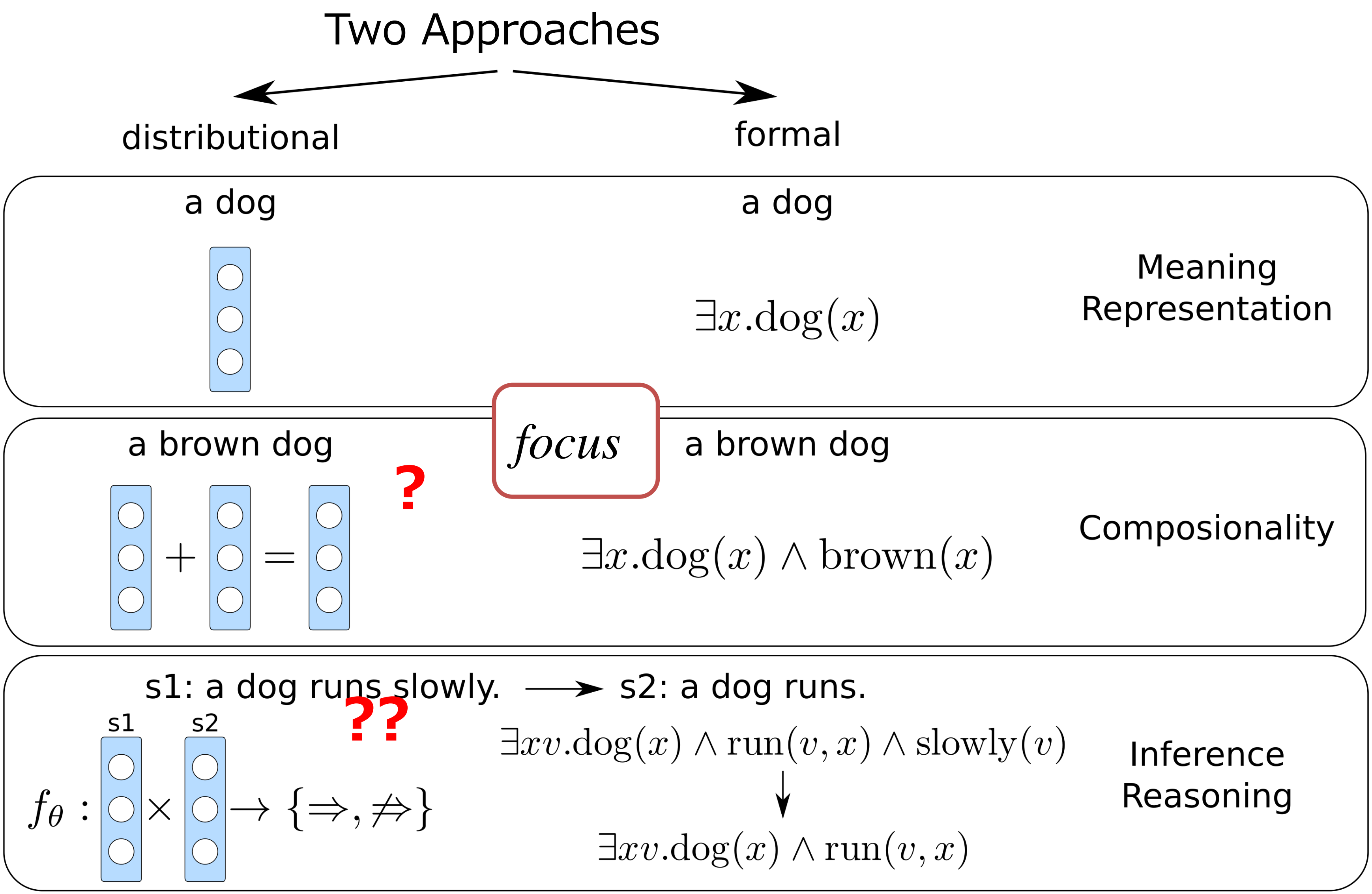
$$\exists xv.\mathrm{dog}(x) \wedge \mathrm{run}(v, x)$$

Meaning Representation

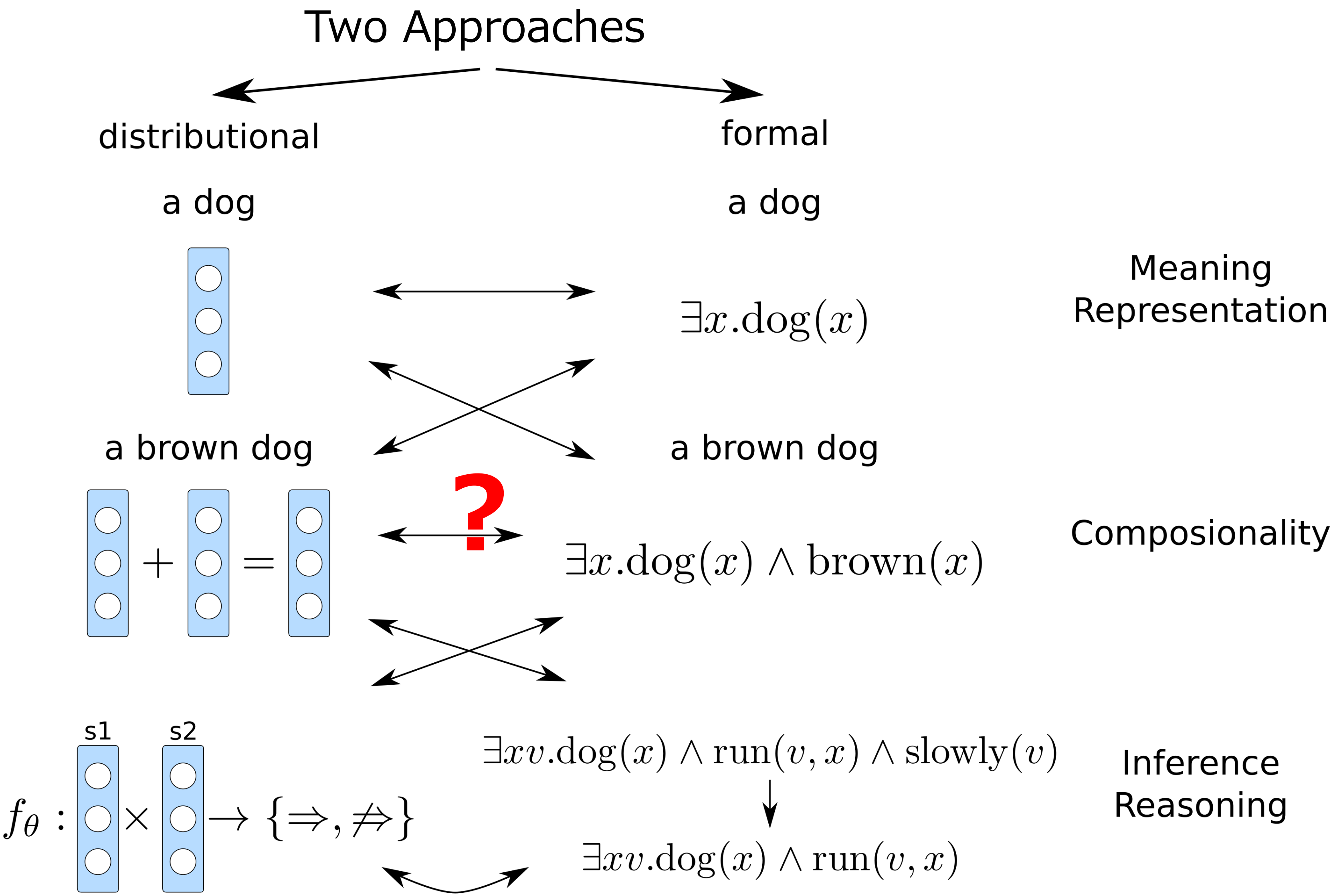
Composionality

Inference Reasoning

Composition is the Challenge



Composition is the Challenge



In this Tutorial:

We will cover the two approaches, first separately, then show some case studies bringing them together

✦ Distributional Semantics:

- How to make word embeddings? (with some detailed studies)
- How to combine word vector to represent phrases and sentences? (an overview)
- Can we just add the vectors? (with recent theoretical results)
- Case studies of combining vector-based composition and logical reasoning

✦ Formal Semantics:

- Introduction to the CCG grammar and semantic composition
- Datasets and challenging composition phenomena
- ccg2lambda: a general framework for formal semantic composition
- Hybrid two approaches for the task of RTE

RTE: an Ultimate Test

Recognizing Textual Entailment (RTE)

T: *Smoking in public spaces is prohibited in most cities in Japan.*

H: *Some cities in Japan do not allow smoking in restaurants.*

[Does the **T**ext entail the **H**ypothesis?]

One expects RTE systems to answer many questions at different levels of meaning:

Is "prohibited" similar to "not allow"?

Are restaurants public spaces?

"X is prohibited" means "does not allow X"?

Does "most" imply "some"?

So one needs to combine two approaches to semantics!

Part I: Composition in Distributional Semantics

1. Word Embeddings
(with some detailed studies)

Highlights for detailed studies

- ✦ Distributional Hypothesis
- ✦ Making co-occurrence table and applying a function
  - What function to apply, and why?
- ✦ Dimension reduction
  - Truncated SVD
    - How to choose the number of dimensions?
  - Noise Contrastive Estimation and *word2vec*

Distributional Hypothesis

"You shall know a word by the company it keeps"

[Harris 1954; Firth 1957]

```
into alcoholic drinks such as beer or hard liquor and derive …
… in miles per hour, pints of beer, and inches for clothes. M…
…ns and for pints for draught beer, cider, and milk sales. The
 carbonated beverages such as beer and soft drinks in non-ref…
…g of a few young people to a beer blast or fancy formal part…
…c and alcoholic drinks, like beer and mead, contributed to a…
 People are depicted drinking beer, listening to music, flirt…
… and for the pint of draught beer sold in pubs (see Metricat…

… ith people drinking beer or wine. Many restaurants can be f…
…gan to drink regularly, host wine parties and consume prepar…
 principal grapes for the red wines are the grenache, mourved…
… four or more glasses of red wine per week had a 50 percent …
…e would drink two bottles of wine in an evening. According t…
…. Teran is the principal red wine grape in these regions. In…
…a beneficial compound in red wine that other types of alcohol
… Colorino and even the white wine grapes like Trebbiano and …
```

Co-occurrence Table

Context: words before/after target

Target Words:	have	new	drink	bottle	ride	speed	read
beer	36	14	72	57	3	0	1
wine	108	14	92	86	0	1	2
car	578	284	3	2	37	44	3
train	291	94	3	0	72	43	2
book	841	201	0	0	2	1	338

f_{ij}=the frequency of "train" co-occurring with "drink"

a vector related to the meaning of "beer"

Additional Steps

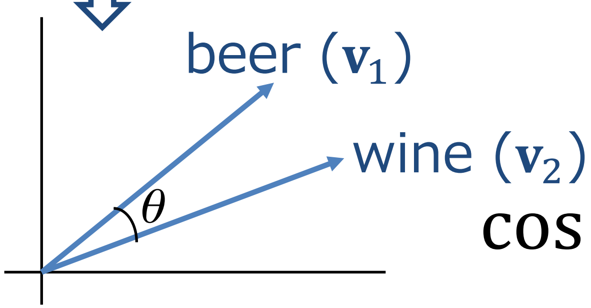
- Applying a function to the co-occurrence frequencies (More on this later)
- Dimension reduction (More on this later)
- Cosine similarity

Target Words: \ Context	have	new	drink	bottle	ride	speed	read
beer	36	14	72	57	3	0	1
wine	108	14	92	86	0	1	2
car	578	284	3	2	37	44	3
train	291	94	3	0	72	43	2
book	841	201	0	0	2	1	338

apply a function

(2.3 1.1 …)
(5.6 1.1 …)

project to lower dimensions



$$\cos\theta = \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} \cdot \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}$$

What Function to Apply?

- Conditional Probability: $p_{ij} = {}^{f_{ij}}\!/_{\sum_j f_{ij}}$
- Square root [Rohde+'06; Lebret+'14; Stratos+'15]: $\sqrt{p_{ij}}$
  - so the L2-norm of a vector is always 1
- Point-wise Mutual Information (PMI) [Church+ 1990; Dagan+ 1994; Turney'01]: $\mathrm{PMI}_{ij} = \ln p_{ij} - \ln p_j$
  - Positive PMI $= \max(\mathrm{PMI}, 0)$ to avoid $\ln 0 = -\infty$ [Bullinaria+'07]
  - $\ln(p_{ij} + \varepsilon) - \ln(p_j + \varepsilon)$ also works [Tian+'17]
  - More generally, $\ln p_{ij} - a_i - b_j$ where a_i and b_j are learned from data [Pennington+'14]

Rationale

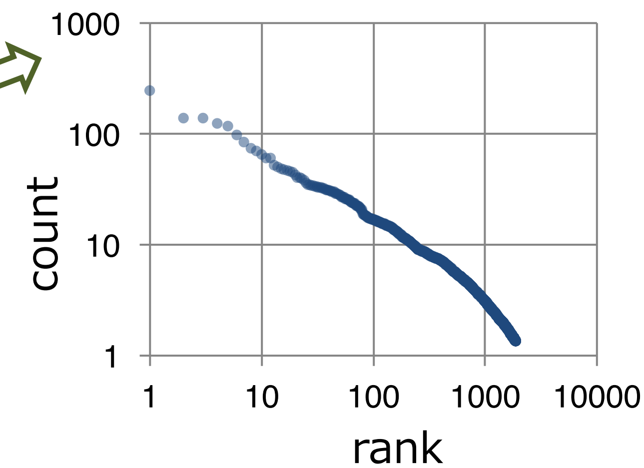
- In practice, square root or PMI perform better than bare conditional probability
- [Pennington+'14] attributes the superiority to a property of the log function:
  - probability values are naturally multiplied
  - vectors are naturally added
  - log is a homomorphism from multiplicative groups to additive groups, i.e. $\ln(xy) = \ln(x) + \ln(y)$
  - but this is not likely the only reason, because square root also works

(Generalized) Zipf's Law

✦Another explanation [Tian+'17]:

– co-occurrence frequencies obey a Zipf-like law:

Target Words: \ Context	have	new	drink	bottle	ride	speed	read
beer	36	14	72	57	3	0	1
wine	108	14	92	86	0	1	2
car	578	284	3	2	37	44	3
train	291	94	3	0	72	43	2
book	841	201	0	0	2	1	338

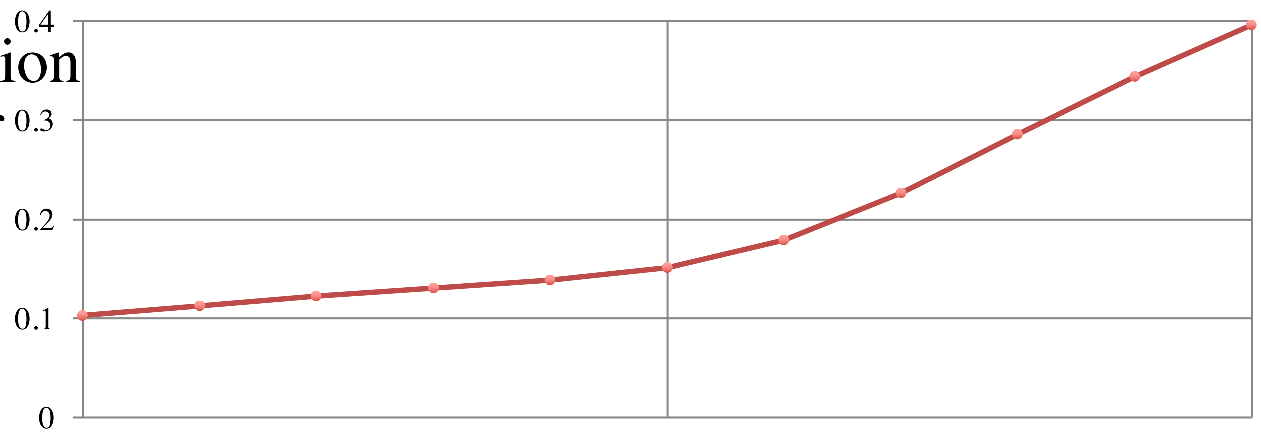


– the expected L2-norm of a vector = the 2^{nd} moment of the Zipf-Law distribution is predictable

Therefore…

- Since a Zipf-Law distribution has heavy tail, the expected L2-norm of a word vector will be ∞ unless one applies some function to reduce large entries

standard deviation of L2-norms of word vectors:

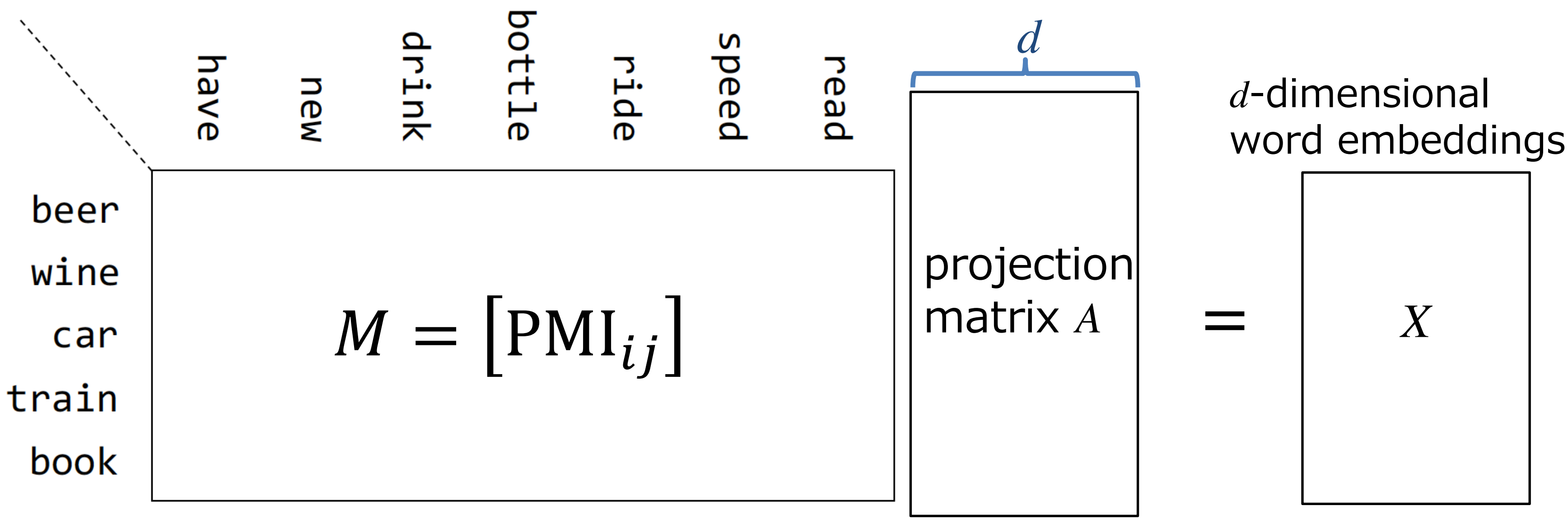


various functions: $\ln p$ $p^{0.1}$ $p^{0.2}$ … $\sqrt{p}$ $p^{0.6}$ $p^{0.7}$ … $p^{0.9}$ p

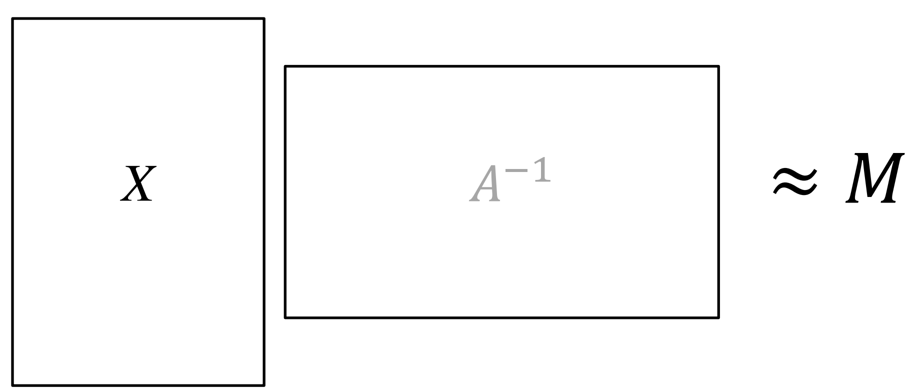
- Behavior of word vectors drastically change when there is/not an expected L2-norm

Dimension Reduction

- project word vectors to lower dimensionality:



- can be learned as factorization to approximate M:



Truncated SVD

- Singular Value Decomposition (SVD) of M:

$$M = U \Sigma V^T$$

where U and V are orthogonal matrices, Σ is diagonal of non-negative entries

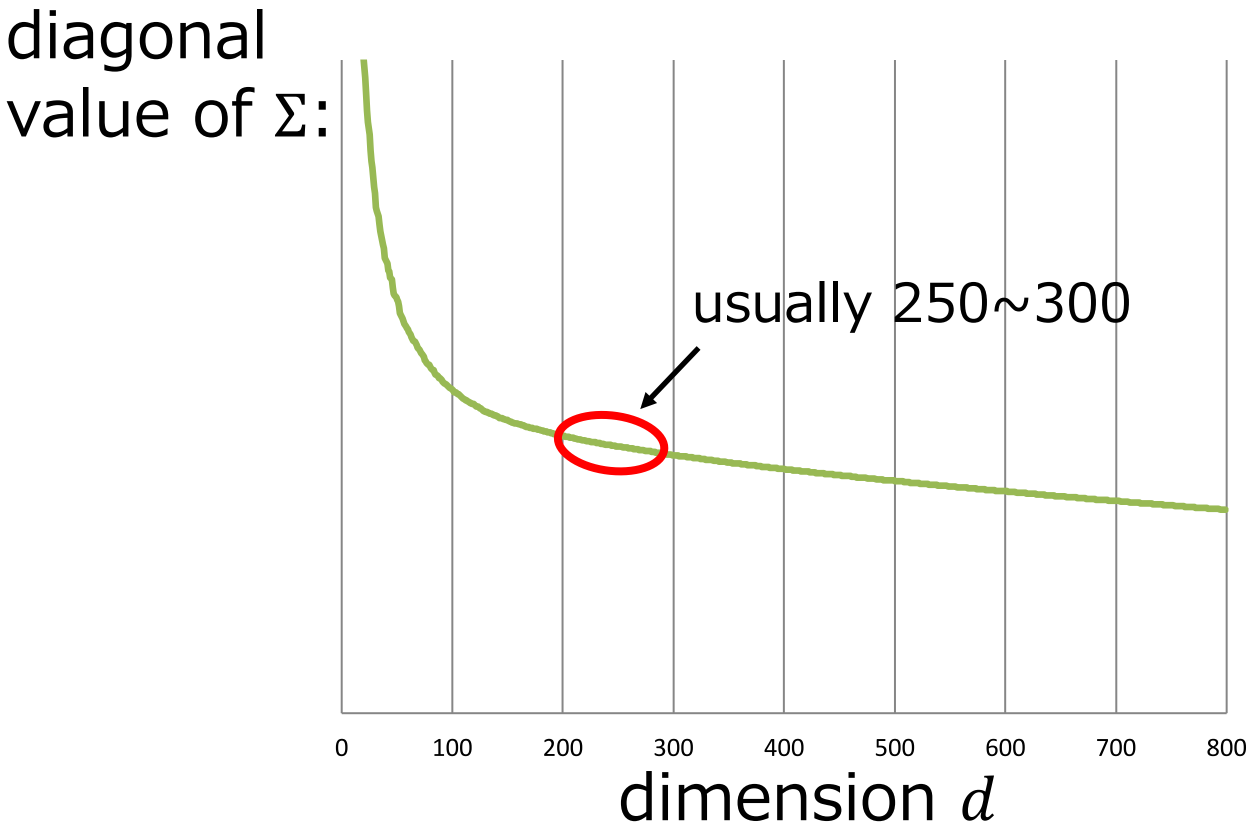
Σ_d takes the top-d diagonal entries of Σ

$\left(U\sqrt{\Sigma_d}\right)\left(\sqrt{\Sigma_d}V^T\right) \approx M$ is a factorization approximating M

$\left(U\sqrt{\Sigma_d}\right)$ only has d columns non-zero

Set $X := \left(U\sqrt{\Sigma_d}\right)$ as the d-dimensional word embeddings

How many dimensions?



known as the “knee finding” or “elbow finding” technique in machine learning

Calculating Truncated SVD

- Fast random algorithm for truncated SVD [Halko+'11]:

  1. calculate $Q = MB$, where B is a random matrix with $2d$ columns (i.e. randomly project M into $2d$ dimensions)
  2. Gram-Schmidt process making Q orthogonal
  3. calculate SVD for the smaller matrix $Q^T M$, then recover the d-dimensional truncated SVD for M

- Implemented in sklearn.decomposition.TruncatedSVD

- Step 2,3 almost take no time. Main cost is the matrix multiplication in Step 1, which can be accelerated by GPU

Online Estimation

For training word embeddings, a training example is a co-occurrence pair (i, j).

Instead of counting the frequency f_{ij} and explicitly calculating $\mathrm{PMI}_{ij} = \ln p_{ij} - \ln p_j$ and doing dimension reduction, one can also train embeddings in an online fashion, implicitly optimizing the log-likelihood $\ln p_{ij}$

This is implemented in the popular toolkit *word2vec* [Mikolov+'13], which uses Noise Contrastive Estimation (NCE) [Gutmann+'12]

NCE

- Problem setting: a model θ estimates probability $p_\theta(x)$ for each data point x. How to optimize θ?
  - Maximum likelihood: $\operatorname*{argmax}_\theta \sum_x \ln p_\theta(x)$
  - but simply optimizing this leads to $p_\theta(x) \to \infty$
  - NCE: mix data x with noise y.
    Model $p_\theta(x) = \mathrm{Prob}(x \text{ is data}|x)$, maximize the likelihood of x being data **and y being noise:**

$$\operatorname*{argmax}_\theta \left(\sum_x \ln p_\theta(x) + \sum_y \ln\bigl(1 - p_\theta(y)\bigr) \right)$$

NCE and *word2vec*

- In *word2vec* [Mikolov+'13], real co-occurrence data (i, j) is mixed with noise (i, k), where k is randomly generated from unigram distribution

- d-dimensional vector $\mathbf{v}_i$ for target word, $\mathbf{u}_j$ for context

- Model the probability of pair (i, j) being co-occurrence data as $\sigma(\mathbf{v}_i \cdot \mathbf{u}_j)$
  - σ is the sigmoid function

- Objective:

$$\operatorname*{argmax}_{\theta} \left(\sum_{i,j} \ln \sigma(\mathbf{v}_i \cdot \mathbf{u}_j) + \sum_{i,k} \ln(1 - \sigma(\mathbf{v}_i \cdot \mathbf{u}_k)) \right)$$

Part I: Composition in Distributional Semantics

2. Vector-based Composition (an overview)

Composition

- Word embeddings can be used to calculate semantic similarities between words; but words can compose more complicated meanings into phrases and sentences.

good lawyer

Mary loves John.

- How would vectors deal with the composition?
- We need composition models for word vectors [Mitchell+'10]

Evaluation Datasets

- Phrase similarities obtained from crowd-sourcing [Mitchell+'10; Kartsaklis+'14; Takase+'17]
  - controlled phrase pairs of the same combination of POS
  - similarity scores by crowd-sourcing

phrase1	phrase2	score1	score2	...
win battle	*fight war*	5	6	...
pay price	*cut cost*	2	3	...
...				

- phrase-word synonymy compiled from *WordNet* [Turney'12]
  - *WordNet* contains multi-word expressions
  - some of them synonyms to single words

phrase	word
red salmon	*sockeye*
dance hall	*ballroom*
...	

Composition Models

✦ Four influencing ideas for modeling composition:

– tree/recursive structure [Socher+'13]

– lexicalization [Baroni+'10; Paperno+'14; Bride+'15]

– syntactic types corresponding to tensor types [Baroni+'10; Coecke+'10]

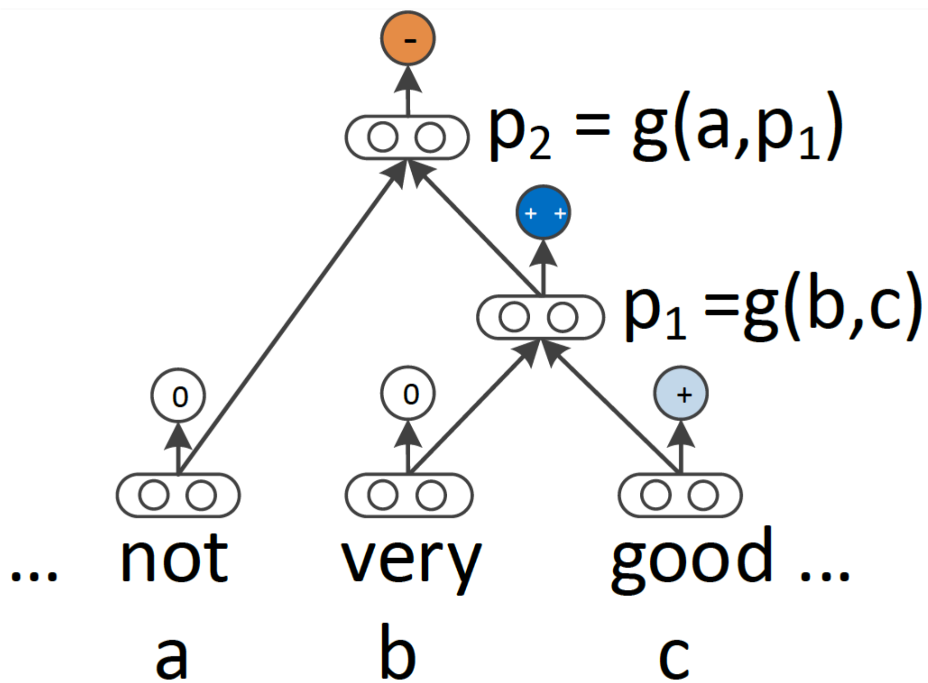
– training word embeddings and composition operators jointly [Hashimoto+'14; Pham+'15]

Recursive Composition

- Natural language has recursive structure

Dorothy thinks that Toto suspects that Tin Man said that….

- Recursive Neural Networks [Socher+'13]

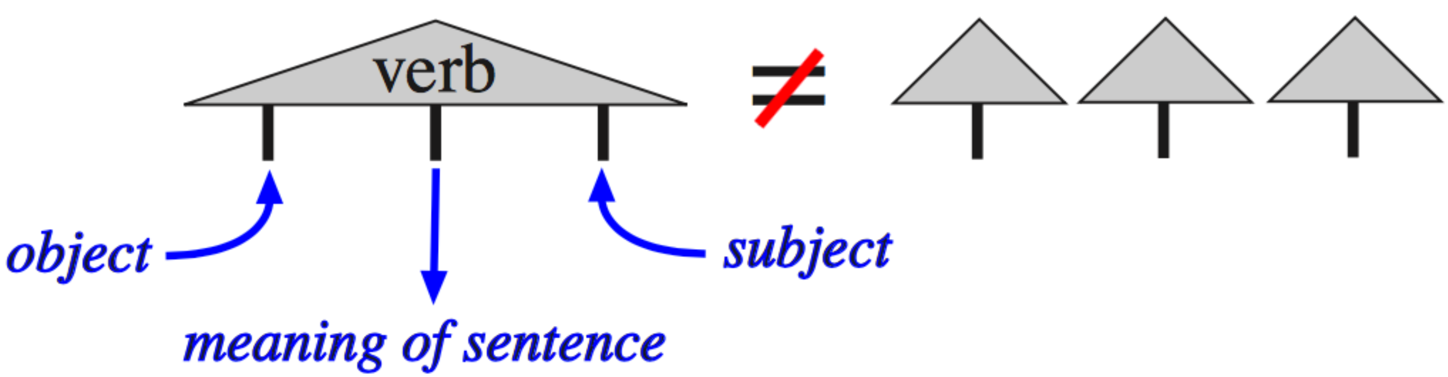


Lexicalization

- Early studies explore simple functions as composition models, such as addition and multiplication [Mitchell+'10]
- Intuition of lexicalization: composition depends on the lexicon being composed [Baroni+'10]
  - *"red car"* is a car, *"fake car"* is not a car
  - should they be composed by the same function?

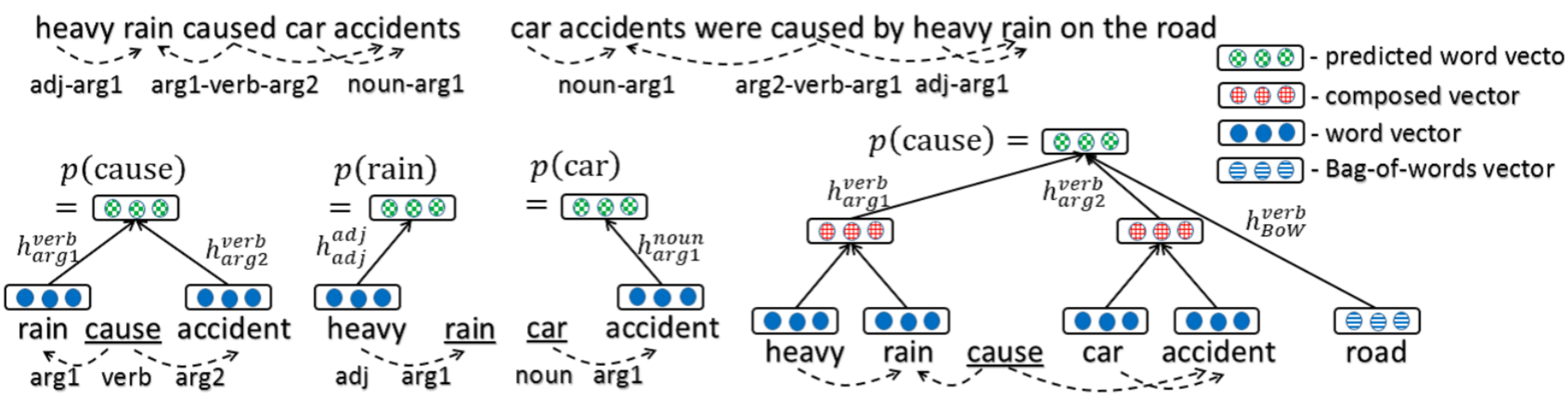
Syntactic↔Tensor Types

- Nouns are vectors, adjectives are matrices [Baroni+'10]:
  - adjectives are functions from the meaning of a noun onto the meaning of a modified noun
- More generally, every syntactic type corresponds to a tensor type [Coecke+'10]



Joint Training

- ✦Use syntactic structures to jointly learn both stand-alone word embeddings and their compositions [Hashimoto+'14]:
  - – composed vectors can provide additional contexts for training word embeddings



(a) PAS-LBLM (b) PAS-CLBLM

Part I: Composition in Distributional Semantics

3. Theory of Additive Composition (with recent theoretical results)

Additive Composition

- In order to represent the meaning of a phrase, simply take the average of word embeddings
  - classical and widely used; works well in practice

dance	***hall***	***dance + hall***
dancing	*halls*	*ballroom*
bhangra	*faneuil*	*halls*
dances	*fame*	*xrco*
breakdancing	*auditorium*	*DDDD-seat*
breakdance	*DDDD-seat*	*rockettes*
music	*inductee*	*banqueting*
striptease	*banqueting*	*barbershop*
hip-hop	*xrco*	*burlesque*
hi-nrg	*now-demolished*	*dancing*
cha-cha-cha	*wigmore*	*wigmore*
disco	*natatorium*	*auditorium*

Theoretical Analysis

- In order to theoretically analyze a composition method, investigate two types of vectors:
  - regard a two-word phrase "$s\ t$" as a single target, construct a vector of co-occurrence; the natural vector $\mathbf{w}^{\{st\}}$
  - take the average of word vectors $\mathbf{w}^s$ and $\mathbf{w}^t$; the composed vector $(\mathbf{w}^s+\mathbf{w}^t)/2$

apply a function, e.g. PMI

more on this later

	Target \ Context	have	new	drink	bottle	ride	speed	read
$\mathbf{w}^s$	beer	36	14	72	57	3	0	1
$\mathbf{w}^t$	glass	53	27	60	43	2	4	34
$\mathbf{w}^{\{st\}}$	beer glass	17	9	24	14	0	0	5

- Estimate distance between the two

Intuition

- when two words occur next to each other and form a phrase (*"beer glass"*), they have almost the same contexts; in contrast, when they separately appear elsewhere, their contexts are independent

...the strength of modern beer is usually around 4%...

...you can find the perfect beer glass from the new website...

...I wish I can drink a glass of wine in the evening...

- when word vectors are added, independence cancels out [Tian+'17]

Formalization

- In order to measure how rare two words occur next to each other:
  - set $\pi_s := 1 - f_{\{st\}}/f_t$ and $\pi_t := 1 - f_{\{st\}}/f_s$, where f_s, f_t and $f_{\{st\}}$ are counts of s, t and "s t" respectively

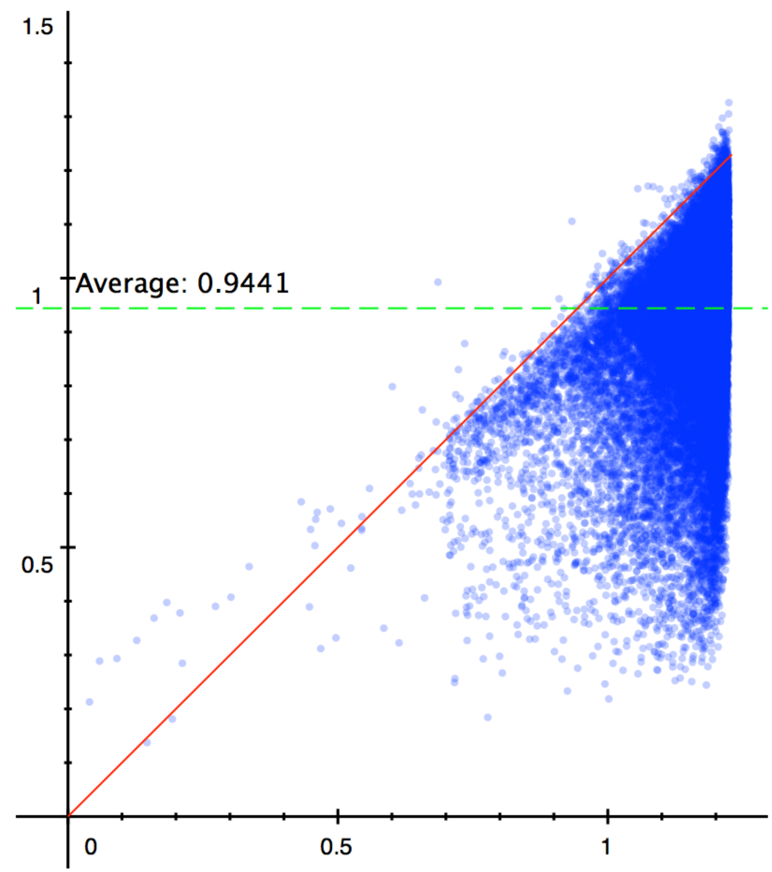
- Then, under certain assumptions, one has [Tian+'17]:

$$\left\|\mathbf{w}^{\{st\}} - (\mathbf{w}^s + \mathbf{w}^t)/2\right\| \leq \sqrt{(\pi_s^2 + \pi_t^2 + \pi_s\pi_t)/2}$$

  - if s and t frequently occur next to each other, π_s and π_t decrease, the bound of distance gets stricter

Verified in Real Corpus

- In the British National Corpus:
  - take each bigram "s t" occurring more than 200 times
  - plot $\left\|\mathbf{w}^{\{st\}} - (\mathbf{w}^s + \mathbf{w}^t)/2\right\|$ on y
  - plot $\sqrt{(\pi_s^2 + \pi_t^2 + \pi_s \pi_t)/2}$ on x
  - theoretically $y \le x$ (under red line)

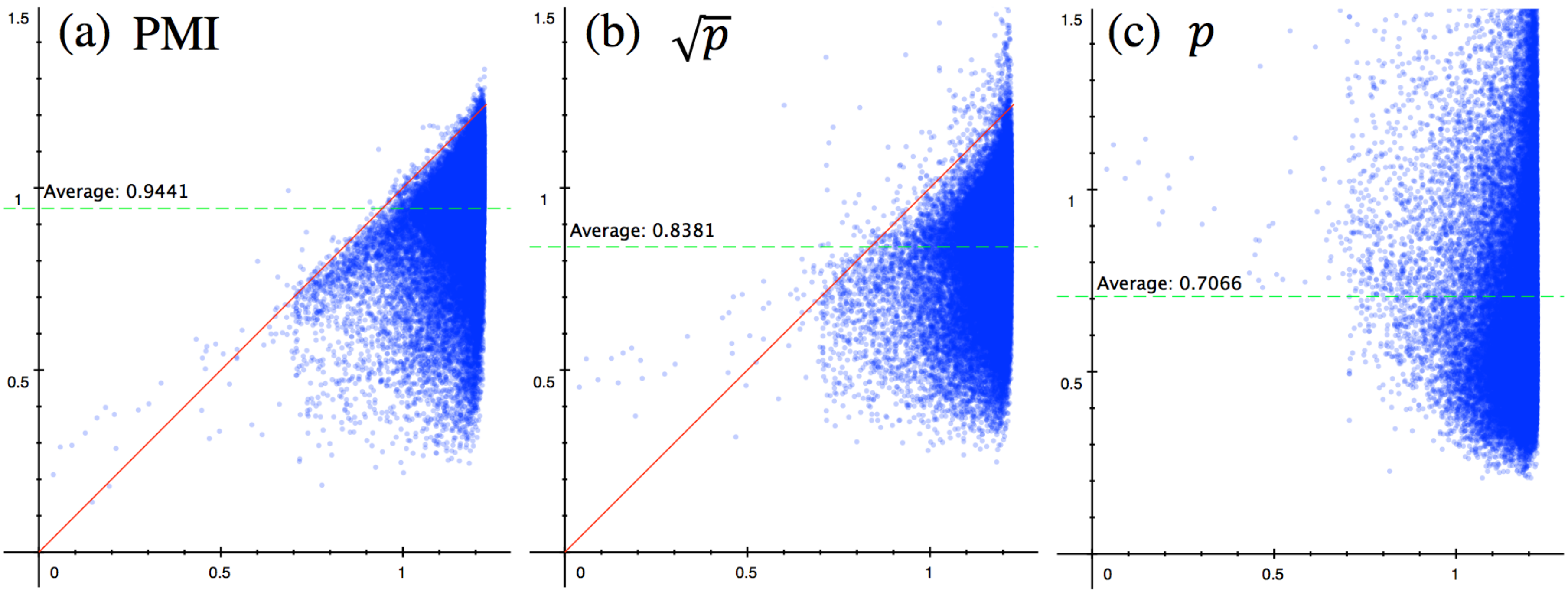


And what Function to Apply...

- Conditional Probability: $p_{ij} = {}^{f_{ij}}/_{\sum_j f_{ij}}$
- Square root [Rohde+'06; Lebret+'14; Stratos+'15]: $\sqrt{p_{ij}}$
  - so the L2-norm of a vector is always 1
- Point-wise Mutual Information (PMI) [Church+ 1990; Dagan+ 1994; Turney'01]: $\mathrm{PMI}_{ij} = \ln p_{ij} - \ln p_j$
  - Positive PMI = $\max(\mathrm{PMI}, 0)$ to avoid $\ln 0 = -\infty$ [Bullinaria+'07]
  - $\ln(p_{ij} + \varepsilon) - \ln(p_j + \varepsilon)$ also works [Tian+'17]
  - More generally, $\ln p_{ij} - a_i - b_j$ where a_i and b_j are learned from data [Pennington+'14]

...for Additive Composition?

✦ PMI and $\sqrt{p}$ work well; bare p does not.



✦ The same tendency observed in phrase similarity tasks [Tian+'17]

Bias and Variance

- In machine learning, generalization error is decomposed into bias and variance
- From this point of view, the estimation of a natural vector has high variance, because **phrases are sparse**
- In contrast, composed vector can be estimated with lower variance because **words are abundant**; word vectors are easy to obtain
- For additive composition, the result shows that bias is bounded. So, *with lower variance and bounded bias, additive composition is a reasonable method for estimating phrase vector*. [Tian+'17]

To conclude:

Additive composition is justifiable by a machine learning theory, as long as you choose the right function to apply.

Word Order

- So far, additive composition is not aware of word order:

$$beer + glass = glass + beer$$

$$John + loves + Mary = Mary + loves + John$$

- A proof-of-concept application: improved additive composition for two-word phrases, with order awareness [Tian+'17]

$$beer_L + glass_R \neq glass_L + beer_R$$

Idea

- two sets of word vectors, one for "left side" and one for "right side"

$$beer_L + glass_R \text{ for } \text{“}beer\ glass\text{”}$$

$$glass_L + beer_R \text{ for } \text{“}glass\ beer\text{”}$$

- How to make sure that the additive composition indeed approximates the intended vector?
  - addition cancels independent contexts, meanwhile it reinforces those shared by two words occurring next to each other

Near-far Context

- Put N-F labels on context words such that:
  - when s_L and t_R occur in the order "$s\ t$", they share the same context:

$$\mathrm{a}\quad \mathrm{b}\quad \mathrm{c}\quad \mathrm{d}\quad \mathrm{e}\quad s\quad t\quad \mathrm{u}\quad \mathrm{v}\quad \mathrm{w}\quad \mathrm{x}\quad \mathrm{y}$$

$$\mathrm{a}\quad \boxed{\mathrm{b}^F\quad \mathrm{c}^F\quad \mathrm{d}^N\quad \mathrm{e}^N\quad s_L\quad t\quad \mathrm{u}^N\quad \mathrm{v}^N\quad \mathrm{w}^F\quad \mathrm{x}^F}\quad \mathrm{y}$$

$$\mathrm{a}\quad \boxed{\mathrm{b}^F\quad \mathrm{c}^F\quad \mathrm{d}^N\quad \mathrm{e}^N\quad s\quad t_R\quad \mathrm{u}^N\quad \mathrm{v}^N\quad \mathrm{w}^F\quad \mathrm{x}^F}\quad \mathrm{y}$$

  - when s_L and t_R occur in the order "$t\ s$", they **do not** share context because of different N-F labels:

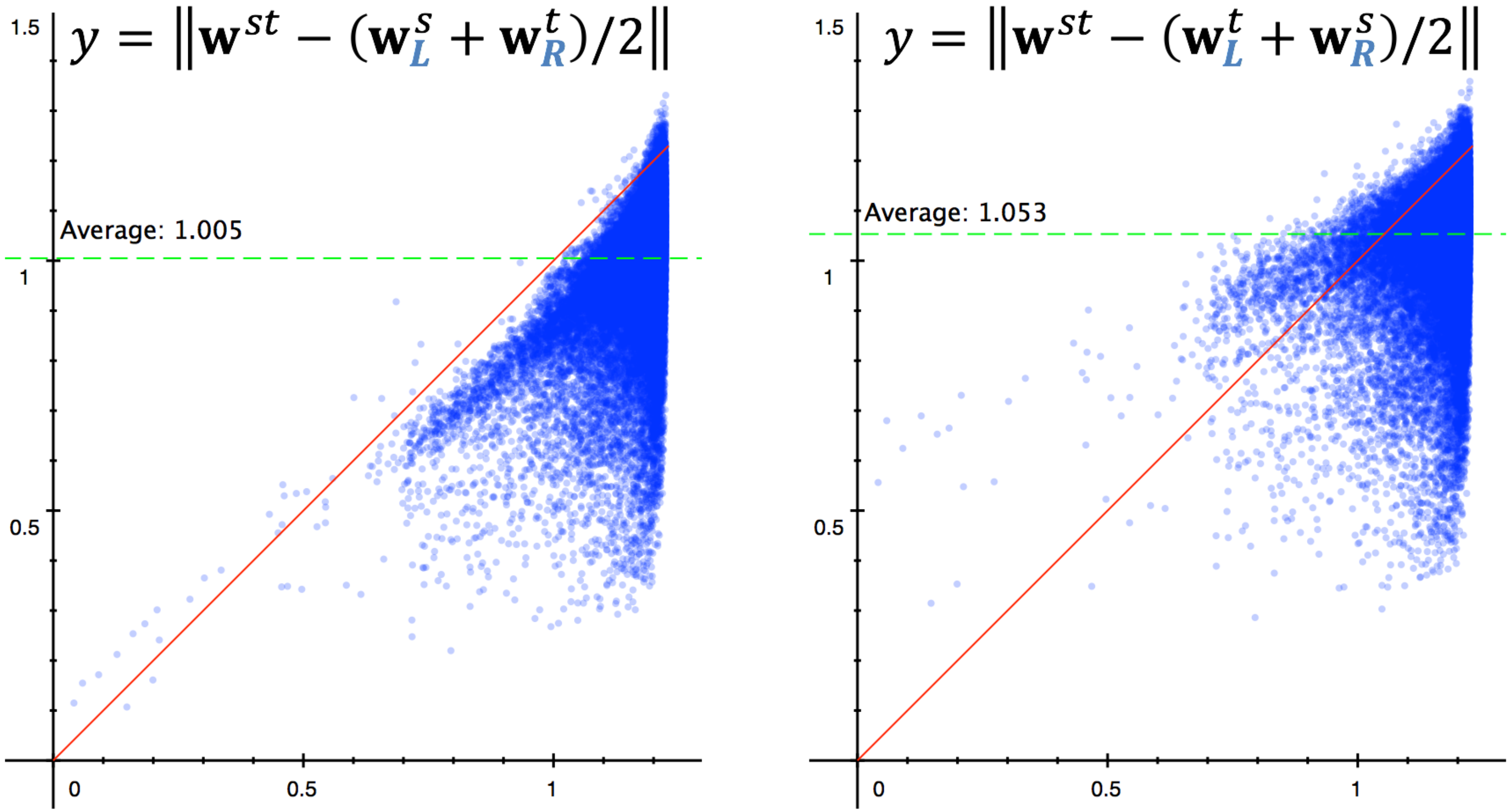
$$\mathrm{a}\quad \mathrm{b}\quad \mathrm{c}\quad \mathrm{d}\quad \mathrm{e}\quad t\quad s\quad \mathrm{u}\quad \mathrm{v}\quad \mathrm{w}\quad \mathrm{x}\quad \mathrm{y}$$

$$\mathrm{a}\quad \mathrm{b}\quad \boxed{\mathrm{c}^F\quad \mathrm{d}^F\quad \mathrm{e}^N\quad t^N\quad s_L\quad \mathrm{u}\quad \mathrm{v}^N\quad \mathrm{w}^N\quad \mathrm{x}^F\quad \mathrm{y}^F}$$

$$\boxed{\mathrm{a}^F\quad \mathrm{b}^F\quad \mathrm{c}^N\quad \mathrm{d}^N\quad \mathrm{e}\quad t_R\quad s^N\quad \mathrm{u}^N\quad \mathrm{v}^F\quad \mathrm{w}^F}\quad \mathrm{x}\quad \mathrm{y}$$

Error Plot

$beer_L + glass_R$ is closer to *"beer glass"* than $glass_L + beer_R$



Demo: most similar word pairs

pose problem	***problem pose***	***tax rate***	***rate tax***
solve dilemma	*difficulty solve*	*income price*	income inflation
arise dilemma	*difficulty cause*	*income inflation*	*premium taxation*
solve difficulty	*difficulty tackle*	*taxation premium*	*premium inflation*
solve concern	*tendency solve*	*income premium*	*price income*
cause dilemma	*solution cause*	inflation income	taxation premium
tackle difficulty	*dilemma cause*	*taxation price*	*inflation income*
dilemma serious	*shortage solve*	premium taxation	*earnings taxation*
confront difficulty	*consequence solve*	*inflation premium*	*premium income*

high price	***price high***	***not enough***	***enough not***
low rate	*rate low*	*really sufficient*	*too never*
low premium	*level low*	*insufficient bother*	really never
low output	*value low*	*still bother*	*too really*
low value	*cost low*	*always want*	ought too
low cost	*premium low*	*always bother*	*too actually*
low wage	*output low*	*really prepared*	*too always*
low level	*inflation low*	*really unwilling*	*sufficient never*
low margin	*market low*	*really obliged*	*quite never*

To conclude:

With theoretical insights, word order is no longer an issue for additive composition, at least for two-word phrases.

Composition and Analogy

✦ Composition is closely related to analogy [Turney'12; Tian+'17; Gittens+'17]:

– if one can construct a relation between words by composition, one might also reverse the process to subtract that relation from words

– a hypothesized illustration [Tian+'17]:

$$\begin{aligned} man &\approx male + human \\ king &\approx royal + male + human \\ woman &\approx female + human \\ queen &\approx royal + female + human \end{aligned} \quad \Rightarrow \quad \begin{aligned} & king - man + woman \\ \approx\ & royal + female + human \\ \approx\ & queen \end{aligned}$$

Is it real?

- Having theoretical support is nice. But to what extent can we count on the additive composition?
- additive composition is...
  - far from perfect
  - no syntax
  - occasionally inspiring, mostly trivial
  - but no chaos

Part I: Composition in Distributional Semantics

4. Toward Vector-based Reasoning: (Case Studies)

Composition and Reasoning

- Composition is closely related to reasoning:

"A and B" implies *A*

"not B" contradicts *B*

"Tad Lincoln's farther is Abraham Lincoln. Abraham Lincoln was born in Kentucky." implies *"Birthplace of Tad Lincoln's farther is Kentucky"*.

- For vector-based composition, reasoning is an ultimate test of whether composition is properly modeled

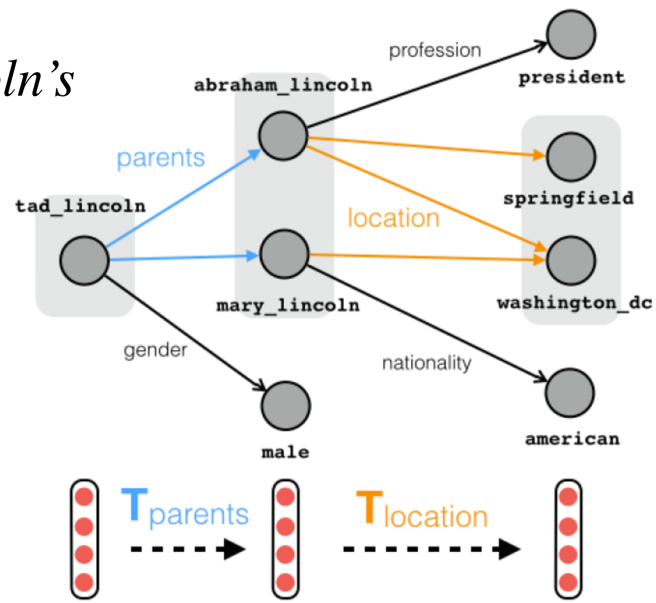
- We discuss three case studies in which vector composition is linked to reasoning
  - Compositional training for knowledge base completion [Guu+'15]
  - A composition model implementing formal semantics [Tian+'16]
  - An embedding model for first order logic [Rocktaschel+'15]

- Compositional training for knowledge base completion [Guu+'15]

Path Query on Knowledge Graph

- Knowledge graph can be used to answer complicated, compositional questions [Guu+'15]:

Where are Tad Lincoln's parents located?



- Modeling path queries in a low-dimensional vector space forces generalization, and can recover some missing facts in knowledge base

Compositional Training

- Train on not only single edges in a knowledge graph, but also longer paths [Guu+'15]

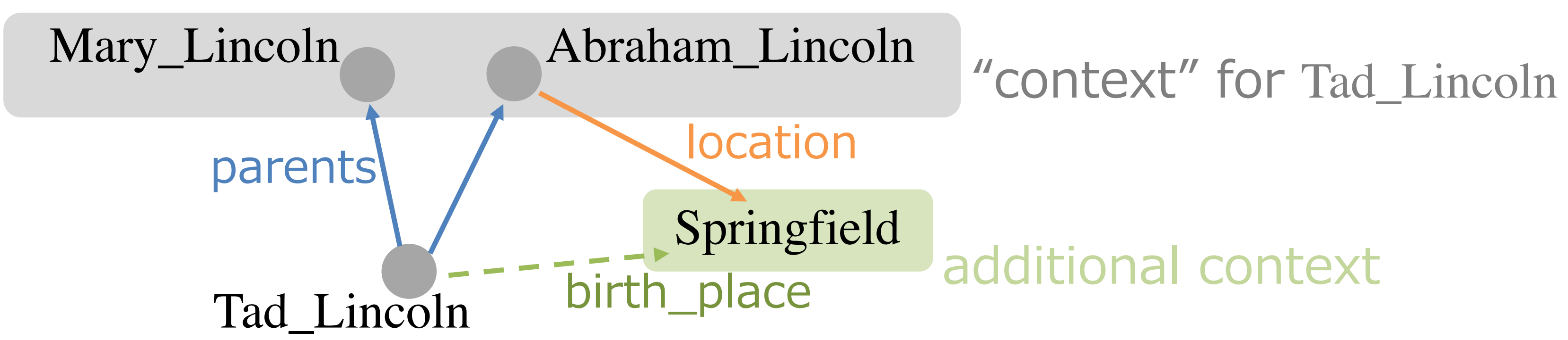
- e.g. a bilinear model:
  - entities are vectors ($\mathbf{x}_s$), relations are matrices (W_r)
  - query vector: $\mathbf{x}_{\text{Tad\_Lincoln}} W_{\text{parents}} W_{\text{location}}$
  - answer score:

$$\mathbf{x}_{\text{Tad\_Lincoln}} W_{\text{parents}} W_{\text{location}} \cdot \mathbf{x}_{\text{Washington\_DC}}$$

Findings

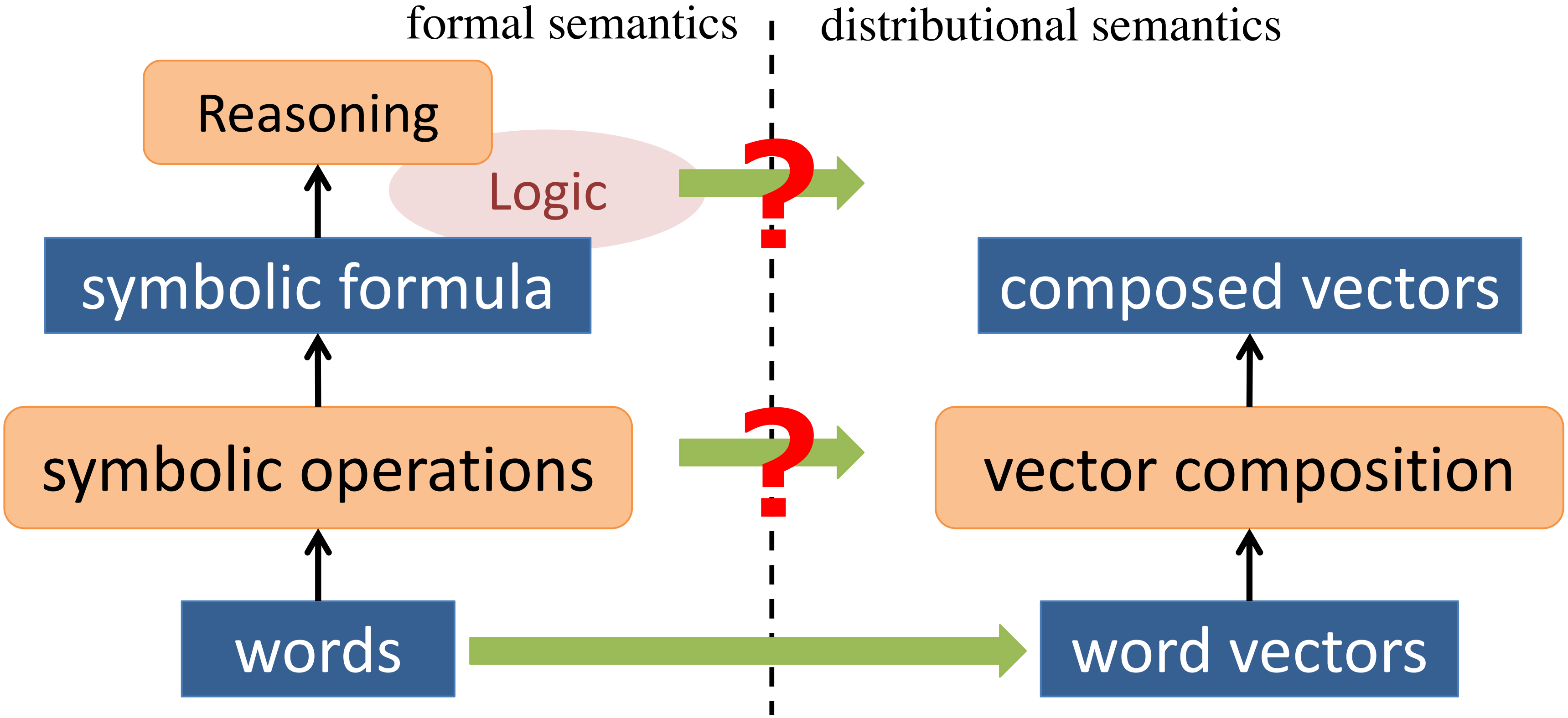
- ✦ Compositional training reduces cascading errors and improves path query answering

- ✦ Compositional training improves knowledge base completion as well:
  - viewed as a “distributional semantics” for database entities, compositional training provides additional contexts



- A composition model implementing formal semantics [Tian+'16]

Formal↔Distributional Semantics



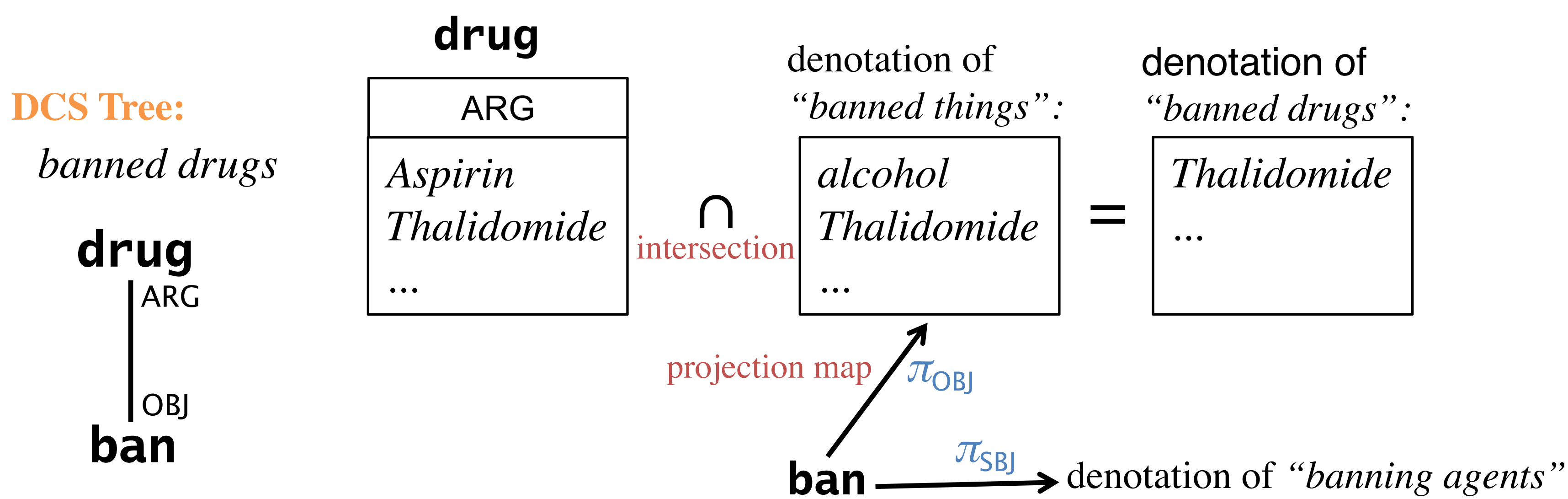
✦A composition model which is [Tian+’16]:

– based on the theory of additive composition

– modeling semantic roles in a logically consisted fashion

The Formal Semantics

- Dependency-based Compositional Semantics (DCS) [Liang+’11; Tian+’14]:
  - content words represent concepts
  - projection π maps concept to denotation (set of things), according to some semantic role
  - compose by set calculation, according to dependency-like trees



Proposal

Represent set operators as vector calculations

[Tian+'16]

X, Y : denotation $\cap$: intersection π : projection π^{-1} : inverse image	corr. $\longrightarrow$	$\mathbf{x}, \mathbf{y}$: vector $+$: addition $\leftarrow$ principle of additive composition M : matrix $\leftarrow$ logical consistency M^{-1}: inverse matrix
$\pi(X \cap Y) \subseteq \pi(X) \cap \pi(Y)$ $\pi^{-1}(X \cap Y) = \pi^{-1}(X) \cap \pi^{-1}(Y)$ $X \subseteq \pi^{-1} \circ \pi(X)$ $X = \pi \circ \pi^{-1}(X)$		$M(\mathbf{x}+\mathbf{y}) = M(\mathbf{x}) + M(\mathbf{y})$ $M^{-1}(\mathbf{x}+\mathbf{y}) = M^{-1}(\mathbf{x}) + M^{-1}(\mathbf{y})$ $\mathbf{x} = M^{-1} \cdot M(\mathbf{x})$ $\mathbf{x} = M \cdot M^{-1}(\mathbf{x})$

query vector:

$\mathbf{v}_{drug} + \mathbf{v}_{ban} M_{\mathrm{OBJ}}$

intersection

projection

$\mathbf{v}_{ban}$

dot product $\longrightarrow \bullet \longrightarrow$

answer vectors:

$\mathbf{u}_{food}$

$\mathbf{u}_{thalidomide}$

$\cdots$

coarse-grained list for "*banned drugs*"

drug
marijuana
cannabis
trafficking
thalidomide
…

Training

- sample paths from DCS trees
- mix real paths with random noise (Noise-contrastive training as in *word2vec* [Mikolov+'13])
- a path connects two words through several semantic roles, e.g. *John*-ARG-SBJ-OBJ-ARG-*Mary*
- probability of it being real path is modeled as:

$$\sigma\left(\mathbf{v}_{John} M_{\mathrm{ARG}} M_{\mathrm{SBJ}}^{-1} M_{\mathrm{OBJ}} M_{\mathrm{ARG}}^{-1} \cdot \mathbf{u}_{Mary}\right)$$

- joint training as in [Hashimoto+'14], and compositional training as in [Guu+'15]

Demo: most similar words

An example from a recent model showing additive composition aware of semantic roles:

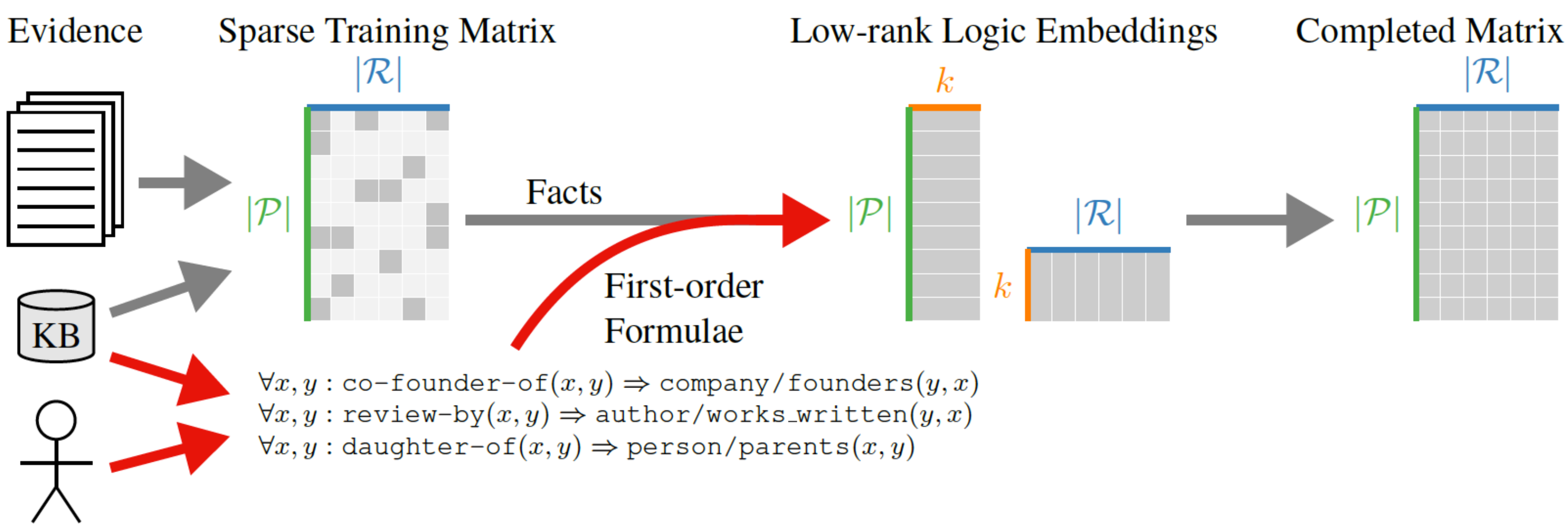
officer who arrests:	*officer who is arrested:*	
$\mathbf{v}_{arrest} M_{\mathrm{SBJ}} + \mathbf{v}_{officer}$	$\mathbf{v}_{arrest} M_{\mathrm{OBJ}} + \mathbf{v}_{officer}$	*officer + arrested*
police	*suspect*	*court-martialed*
gestapo	*policeman*	*cashiered*
FBI	*prisoner*	*inspector-general*
policeman	*inmate*	*reservist*
officer	*accomplice*	*paymaster*

Implementation: https://github.com/tianran/vecdcs

- An embedding model for first order logic [Rocktaschel+’15]

Injecting Logic into Factorization

Instead of learning embeddings by naïve matrix factorization, one can make them conform to logical background knowledge [Rocktaschel+’15]



Training

- Map logical formulas to continuous functions of embeddings, so that they have gradients and can be trained by backpropagation
  - Ground atom:

    parent(Tad_Lincoln, Mary_Lincoln) $\rightarrow$ $\sigma\left(\mathbf{x}_{\text{Tad\_Lincoln}} W_{\text{parent}} \cdot \mathbf{x}_{\text{Mary\_Lincoln}}\right)$

  - Logical conjunction:

    parent(Tad_Lincoln, Mary_Lincoln) ∧ parent(Tad_Lincoln, Abraham_Lincoln)

    $$\sigma\left(\mathbf{x}_{\text{Tad\_Lincoln}} W_{\text{parent}} \cdot \mathbf{x}_{\text{Mary\_Lincoln}}\right) \cdot \sigma\left(\mathbf{x}_{\text{Tad\_Lincoln}} W_{\text{parent}} \cdot \mathbf{x}_{\text{Abraham\_Lincoln}}\right)$$

  - Logical negation:

    ¬parent(Tad_Lincoln, Robert_Lincoln)

    $$1 - \sigma\left(\mathbf{x}_{\text{Tad\_Lincoln}} W_{\text{parent}} \cdot \mathbf{x}_{\text{Robert\_Lincoln}}\right)$$

  - Other logical operators can be deduced accordingly

Thank you!

Questions?