Representation Learning

Jie Tang

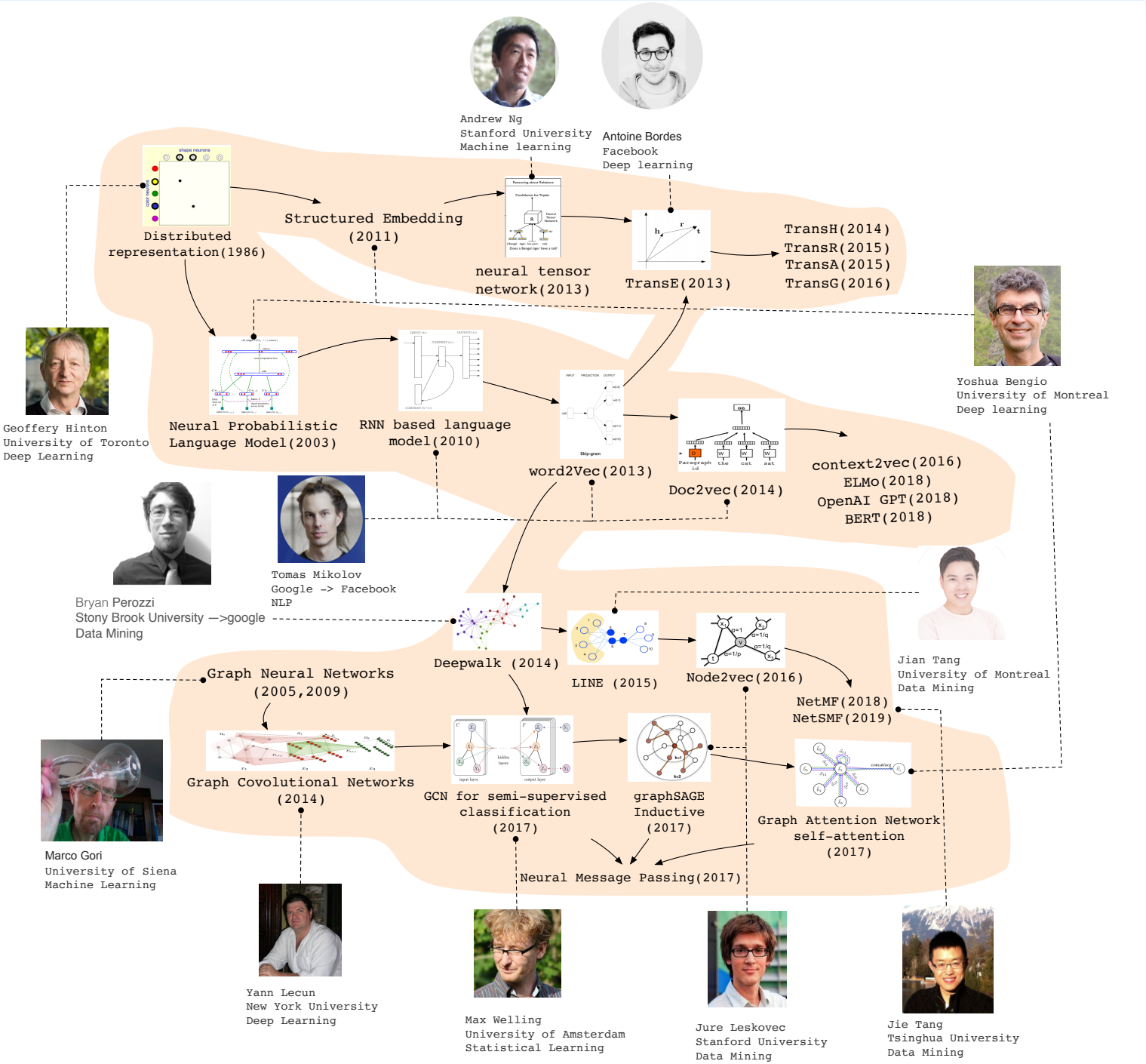
Tsinghua University

April 8, 2020

Overview

1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary

2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec


Andrew Ng
Stanford University
Machine learning
Antoine Bordes
Facebook
Deep learning
Distributed representation(1986)
Structured Embedding (2011)
neural tensor network(2013)
TransE(2013)
TransH(2014)
TransR(2015)
TransA(2015)
TransG(2016)
Yoshua Bengio
University of Montreal
Deep learning
Geoffery Hinton
University of Toronto
Deep Learning
Neural Probabilistic Language Model(2003)
RNN based language model(2010)
word2Vec(2013)
Doc2vec(2014)
context2vec(2016)
ELMo(2018)
OpenAI GPT(2018)
BERT(2018)
Tomas Mikolov
Google -> Facebook
NLP
Bryan Perozzi
Stony Brook University —>google
Data Mining
Deepwalk (2014)
LINE (2015)
Node2vec(2016)
Jian Tang
University of Montreal
Data Mining
NetMF(2018)
NetSMF(2019)
Graph Neural Networks (2005,2009)
Graph Covolutional Networks (2014)
GCN for semi-supervised classification (2017)
graphSAGE Inductive (2017)
Graph Attention Network self-attention (2017)
Neural Message Passing(2017)
Marco Gori
University of Siena
Machine Learning
Yann Lecun
New York University
Deep Learning
Max Welling
University of Amsterdam
Statistical Learning
Jure Leskovec
Stanford University
Data Mining
Jie Tang
Tsinghua University
Data Mining

What is representation learning?

- Feature learning or representation learning is a set of techniques that learn a feature: a transformation from raw input (data) into a representation that can be effectively exploited in machine learning tasks
- This excludes manual feature engineering—feature learning is very different from feature engineering
- This needs a machine to learn the features themselves, or learn the ML task and features simultaneously

What makes a representation good?

- Multiple explanatory factors $\rightarrow$ Distributed Representation
- A hierarchical organization of explanatory factors $\rightarrow$ Deep Architecture
- Shared factors across tasks $\rightarrow$ Multi-task and Multi-modal Learning
- Disentangling the underlying factors of variation $\rightarrow$ Semi-supervised Learning

Overview

1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary
2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec

Distributed Representation

- Good representations are expressive, meaning that a reasonably-sized learned representation can capture a huge number of possible input configurations.
- Distributed representation: Each element of the representation can be set separately from each other:
  - one concept is represented by more than one neuron activation.
  - one neuron represents more than one concept.

Distributed Representation vs One-hot Representation

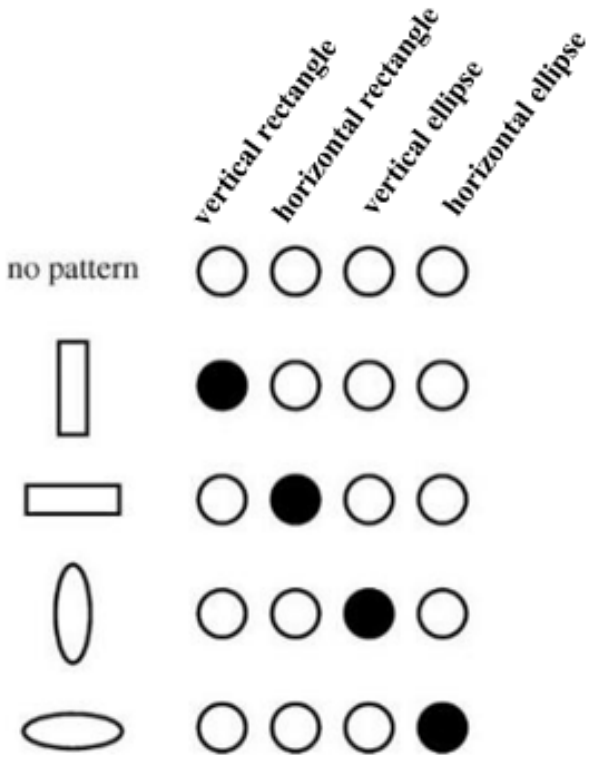


Figure 1: One-hot representation

- If we wanted to represent a new shape with a sparse representation, we would have to increase the dimensionality.

Distributed Representation vs One-hot Representation

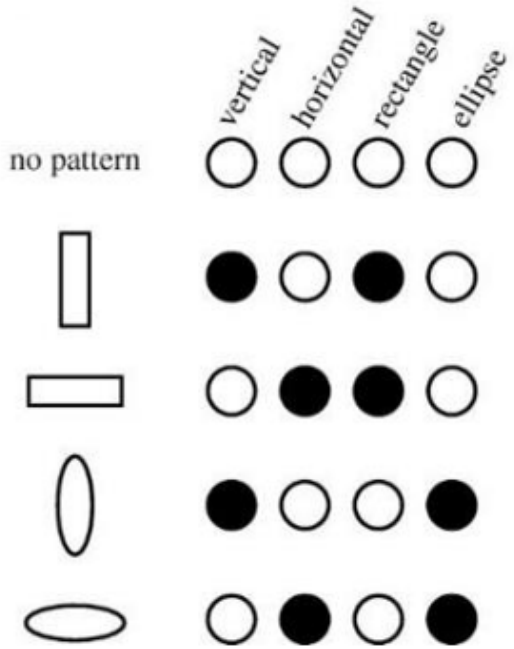
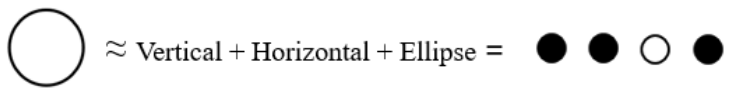
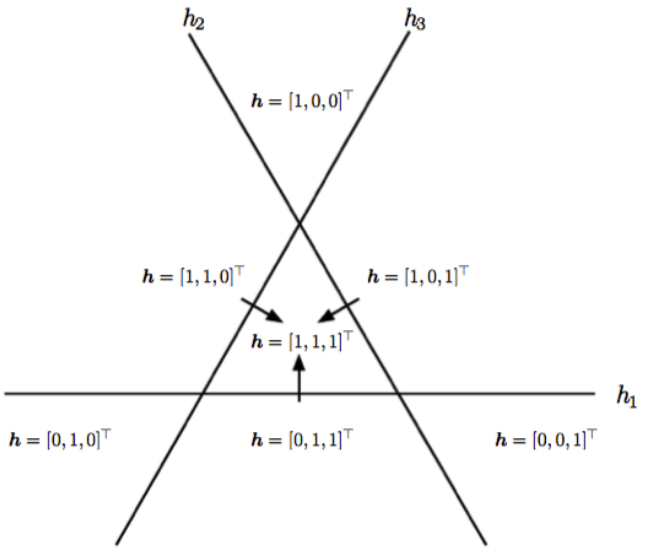


Figure 2: Distributed representation

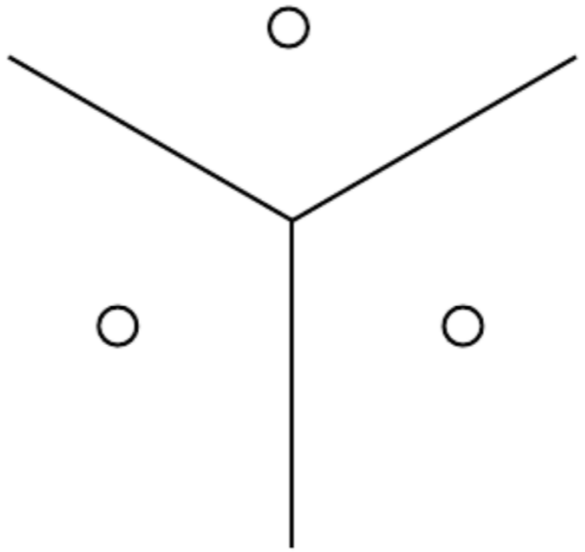
- In distributed representation, we may be able to represent a new shape with the existing dimensionality.



Distributed Representation vs One-hot Representation



(a) Distributed representation



(b) One-hot representation

- The distributed representation with n binary features assigns unique codes to $O(n^d)$ different regions in $\mathbb{R}^d$.
- The one-hot representation with n examples assigns unique codes to only $O(n)$ regions.

Overview

1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary
2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec

Deep Architecture

Advantages of deep architectures:

- Deep architectures are more expressive.
- Deep architectures can lead to abstract representations.

Expressiveness of Deep Networks

- Expressiveness of deep networks with piecewise linear activation functions: exponential advantage for depth. (Montufar et al, NIPS 2014)

Expressiveness of Deep Networks

- Expressiveness of deep networks with piecewise linear activation functions: exponential advantage for depth. (Montufar et al, NIPS 2014)
- Number of pieces (regions) distinguished by a network with depth L and n_i units per layer is at least

$$\left(\prod_{i=1}^{L-1} \left\lfloor \frac{n_i}{n_0} \right\rfloor^{n_0}\right) \sum_{j=0}^{n_0} \binom{n_L}{j} \tag{1}$$

Expressiveness of Deep Networks

- Expressiveness of deep networks with piecewise linear activation functions: exponential advantage for depth. (Montufar et al, NIPS 2014)
- Number of pieces (regions) distinguished by a network with depth L and n_i units per layer is at least

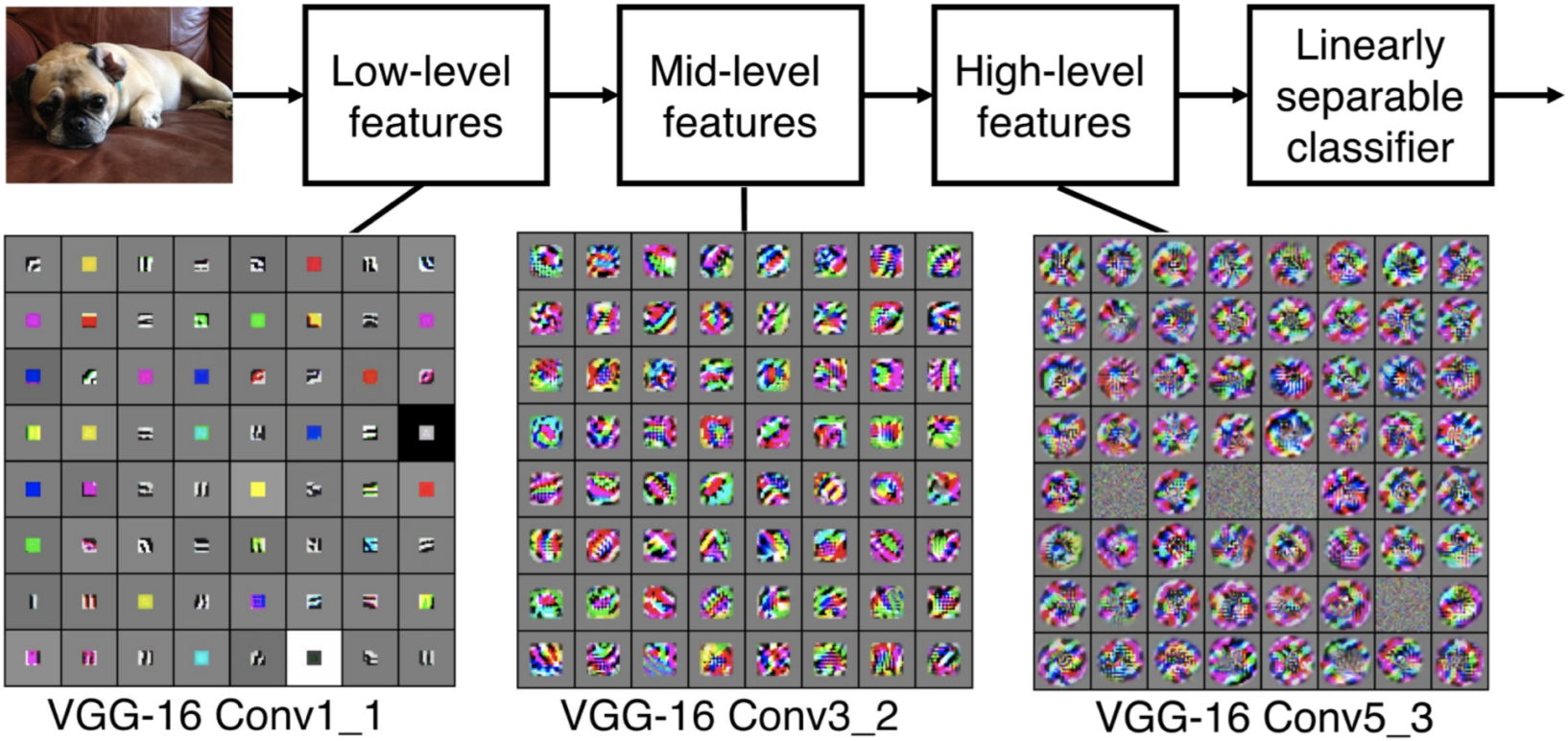
$$\left(\prod_{i=1}^{L-1} \left\lfloor \frac{n_i}{n_0} \right\rfloor^{n_0}\right) \sum_{j=0}^{n_0} \binom{n_L}{j} \tag{1}$$

- or, if hidden layers have width n and input has size n_0:

$$\Omega\left((n/n_0)^{(L-1)n_0} n^{n_0}\right) \tag{2}$$

Abstraction of Deep Networks

- Feature representations in different CNN layers[1]:



[1]Example from Fei-fei's slides

Overview

1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary
2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec

How do humans generalize from very few examples?

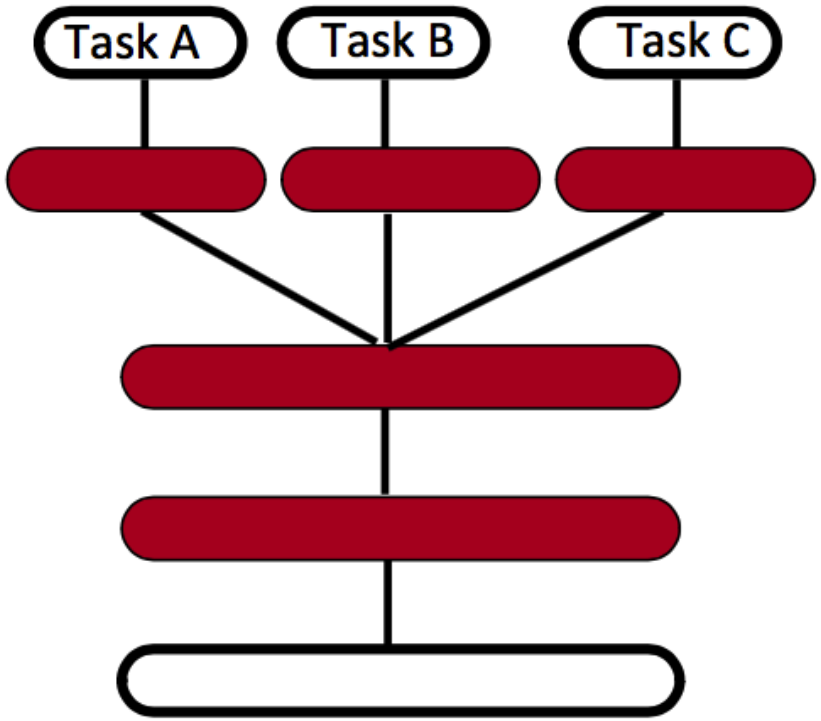
They transfer knowledge from previous learning:

- Representations
- Explanatory factors

Previous learning from unlabeled data + labels for other tasks.
Prior: shared underlying explanatory factors, in particular between $P(x)$ and $P(y|x)$

Multi-Task Learning

Example: speech recognition, sharing across multiple languages



Good representations that disentangle underlying factors of variation make sense for many tasks because each task concerns a subset of the factors

Google Image Search: Joint Embedding

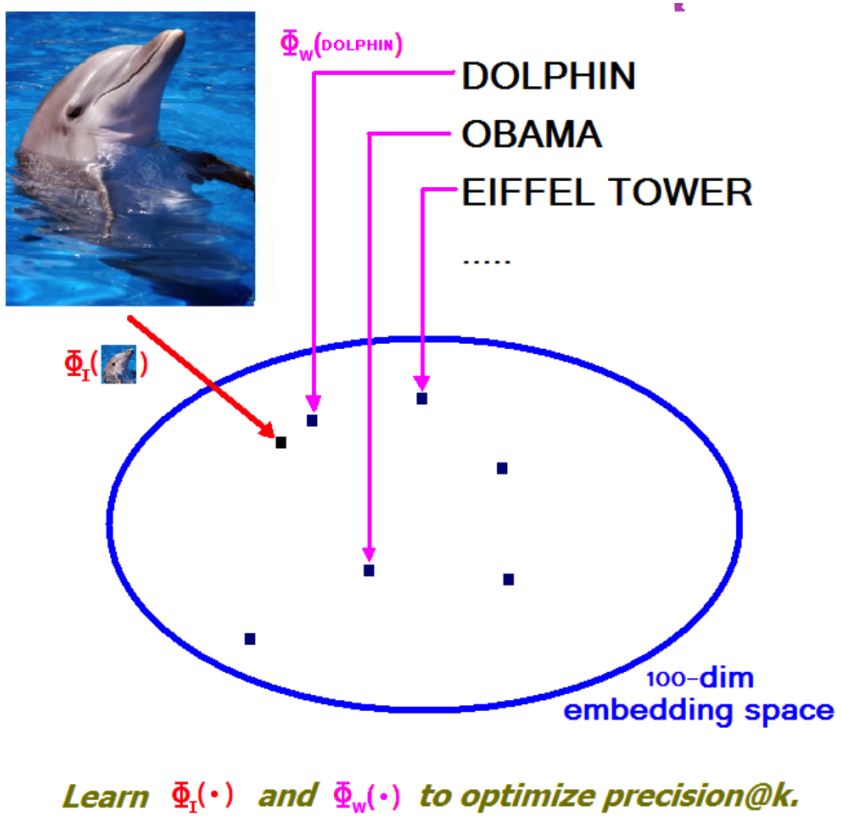


Figure 3: Wsabie: Scaling Up To Large Vocabulary Image Annotation, IJCAI'2011

Multi-Task / Multimodal Learning with Different Inputs for Different Tasks

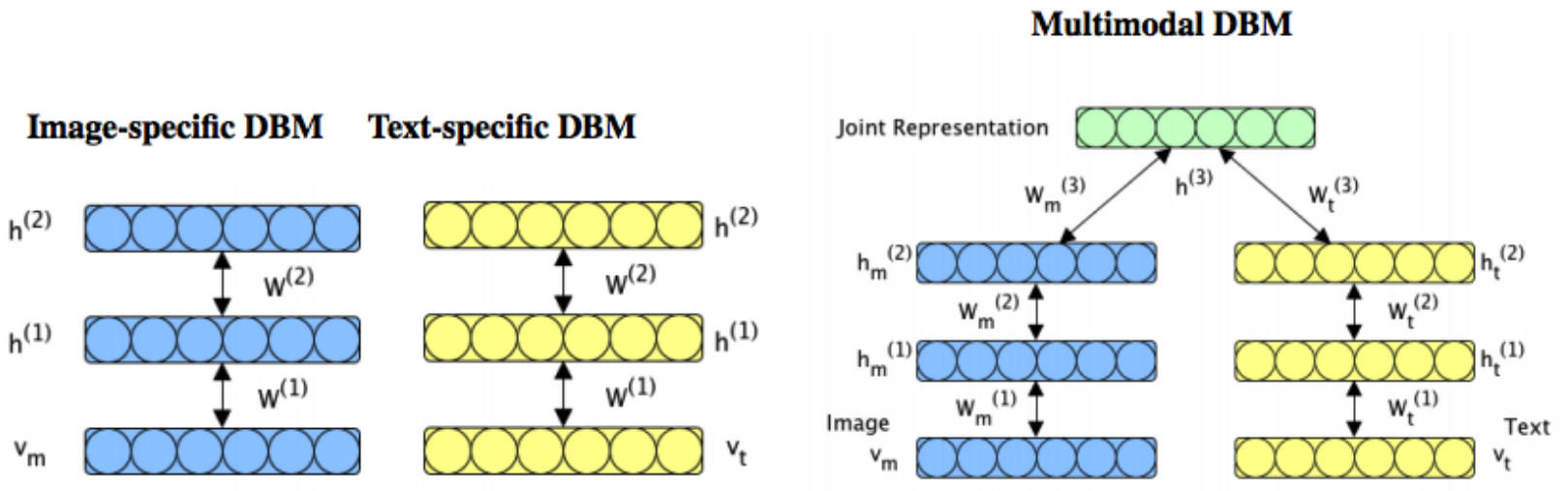
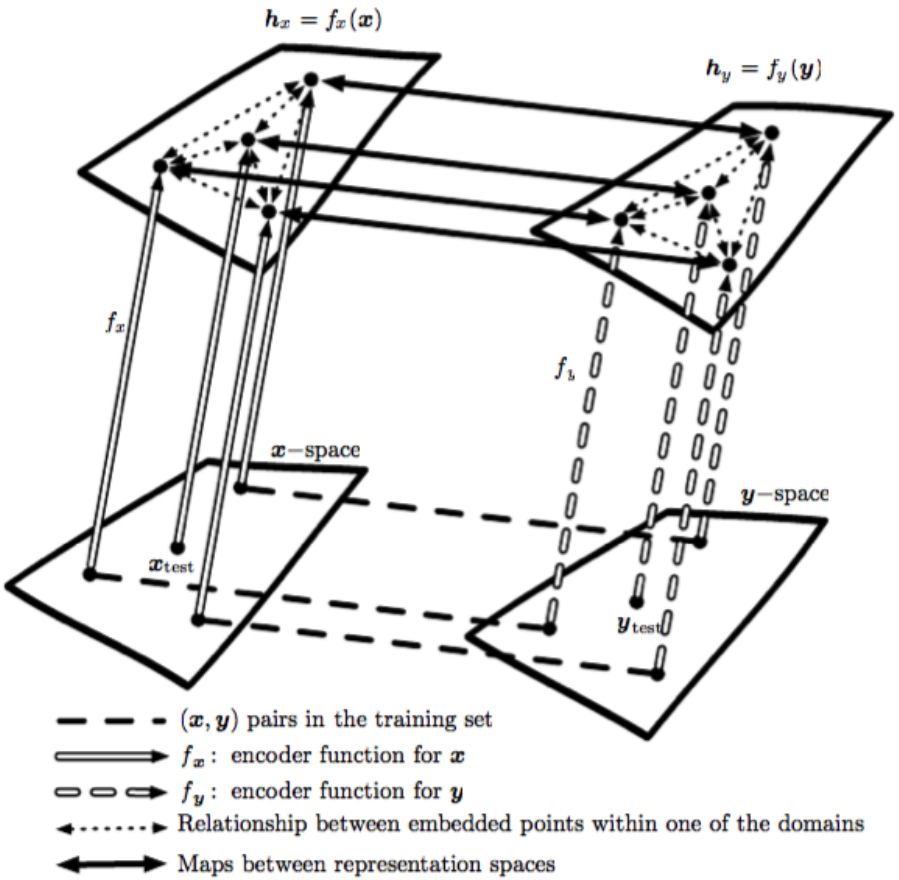


Figure 4: Multimodal Learning, NIPS'2012

Maps Between Representations



- Labeled pairs (x, y) allow us to learn mapping between f_x and f_y.
- Zero-shot Learning: associate an image x_{test} to a word y_{test}, even if no image of that word was ever presented.

Overview

1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary
2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec

Semi-supervised Learning

- An ideal representation is one in which the features within the representation correspond to the underlying causes of the observed data.
- For a good representation of $p(x)$, such a representation for computing $p(y|x)$ if y is among the most salient causes of x (semi-supervised learning).

When semi-supervised learning can fail or succeed?

Failure case:

- $p(x)$ is uniformly distributed, we want to learn $f(x) = \mathbb{E}(y|x)$.
- Observing a training set of x values alone gives us no information about $p(y|x)$.

Success case:

- x arises from a mixture of several components, with one mixture componennt per value of y.
- Modeling $p(x)$ reveals precisely where each component is, a single labeled example of each class will then be enough to perfectly learn $p(y)$.

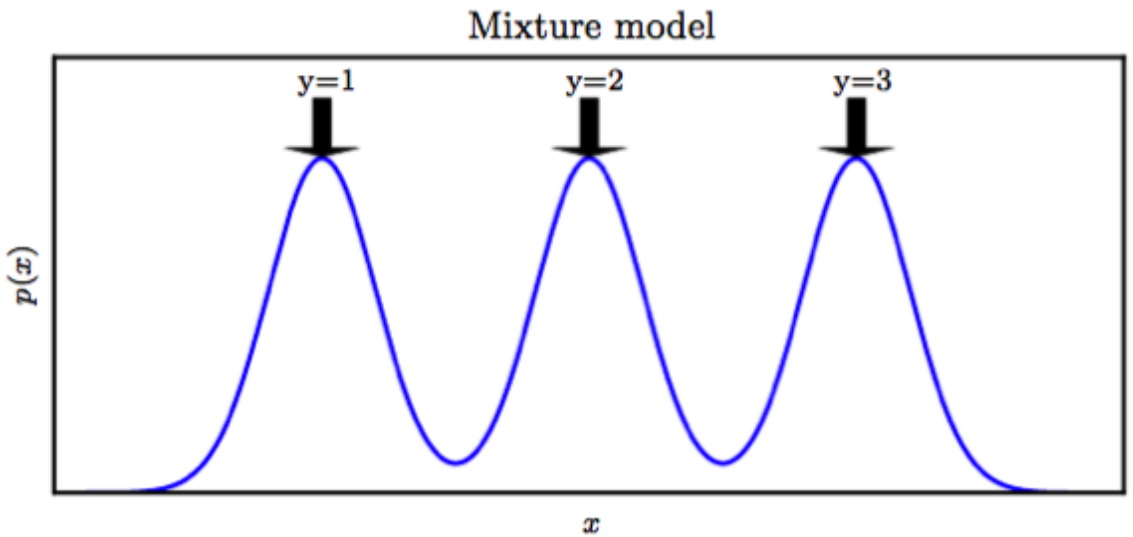
Why semi-supervised learning works? Because of causality

- Think about a generative process $P(x|h)$, where h are all the causal factors.
- If $y = h_i$ is one of the causal factors of x.
- The best possible model of x must involve y as a latent factor.
- Representation learning seeks the latent variable h that explain the variation of x, making it likely to also uncover y.

Let's have an example!

Why semi-supervised learning works? Because of causality

- Just observing the x-density reveals the causes y (cluster ID).
- After learning $P(x)$ as a mixture, a single labeled example per class suffices to learn $P(y|x)$.



Fact: Most Observations are Formed by a Large Number of Underlying Causes

Brute Force Strategy: Try to learn a representation that captures all the reasonably salient generative factors $h_i \in h$.

- Not feasible because it is not possible to capture all or most of the factors.
- Example: smallest objects in visual scene.

Fact: Most Observations are Formed by a Large Number of Underlying Causes

Brute Force Strategy: Try to learn a representation that captures all the reasonably salient generative factors $h_i \in h$.

- Not feasible because it is not possible to capture all or most of the factors.
- Example: smallest objects in visual scene.

Other strategies:

- Use a supervised learning signal at the same time. so that the model will capture the most relevant factors.
- Use much larger representations.
- Modify the definition of which underlying causes are most salient.

Revisit the Definition of Which Underlying Causes are Most Salient

Mean squared error applied to the pixels of an image implicitly specifies that an underlying cause is only salient if it significantly changes the brightness of a large number of pixels.

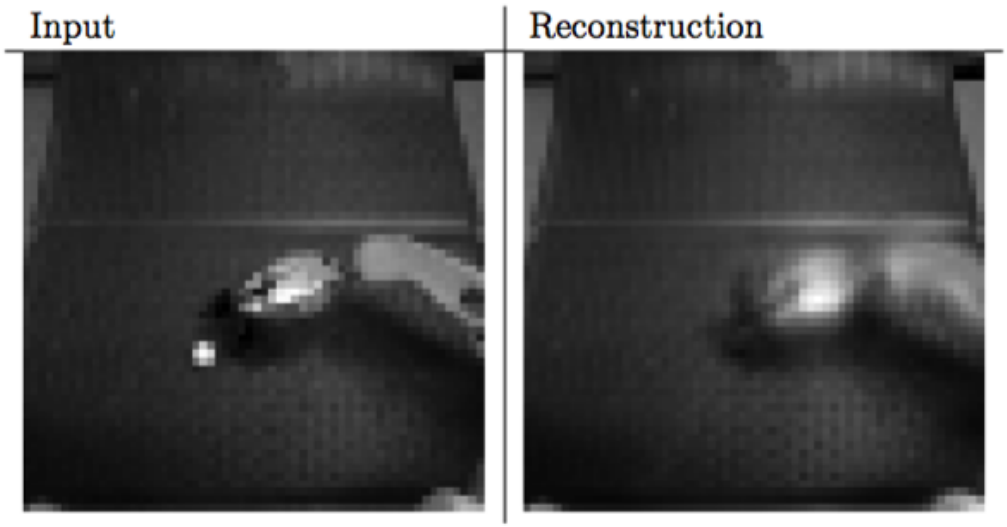


Figure 5: An autoencoder trained with mean squared error for a robotics task has failed to reconstruct a ping pong ball.

Other Definition of Salience — Generative Adversarial Networks

- A generative model is trained to fool a feedforward classifier.
- The feedforward classifier attempts to recognize all samples from the generative model as being fake, and all samples from the training set as being real.
- Be described in more detail later.

Summary

- Computational Scaling
- Optimization and Underfitting
- Approximate Inference and Sampling
- Disentangling Factors of Variation
- Reasoning and One-Shot Learning of Facts

Further Reading: Representation Learning: A Review and New Perspectives. Bengio et al.

Overview

1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary

2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec

Typical Approaches for Word Representation

- 1-hot Representation
- Foundation of Bag-of-Words Model

star [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ···]

sun [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ···]

sim(star, sun) = 0

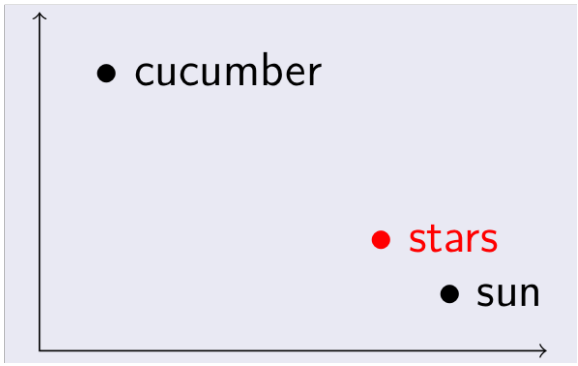


Typical Approaches for Word Representation

- Count-based distributional representation

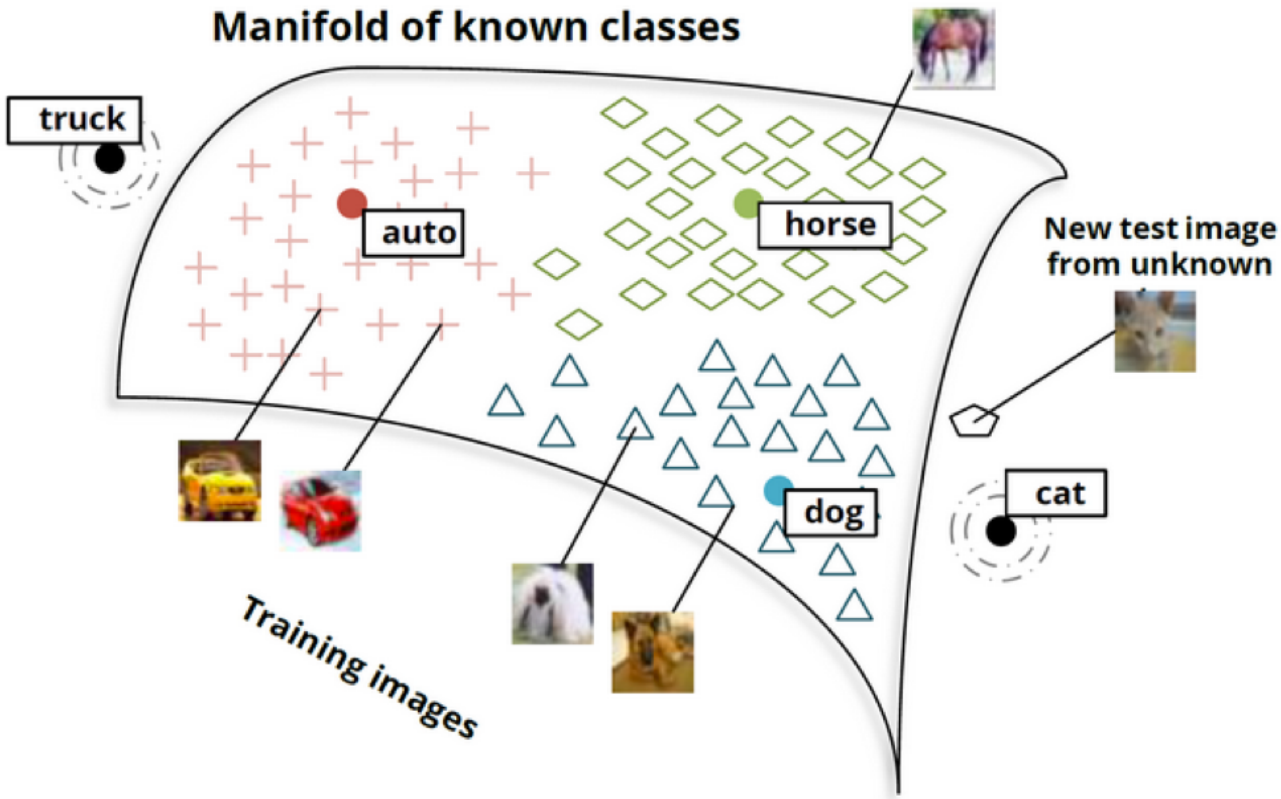
```
he curtains open and the stars shining in on the barely
ars and the cold , close stars " . And neither of the w
rough the night with the stars shining so brightly , it
made in the light of the stars . It all boils down , wr
 surely under the bright stars , thrilled by ice-white
sun , the seasons of the stars ? Home , alone , Jay pla
m is dazzling snow , the stars have risen full and cold
```

	shining	bright	trees	dark	look
stars	38	45	2	27	12

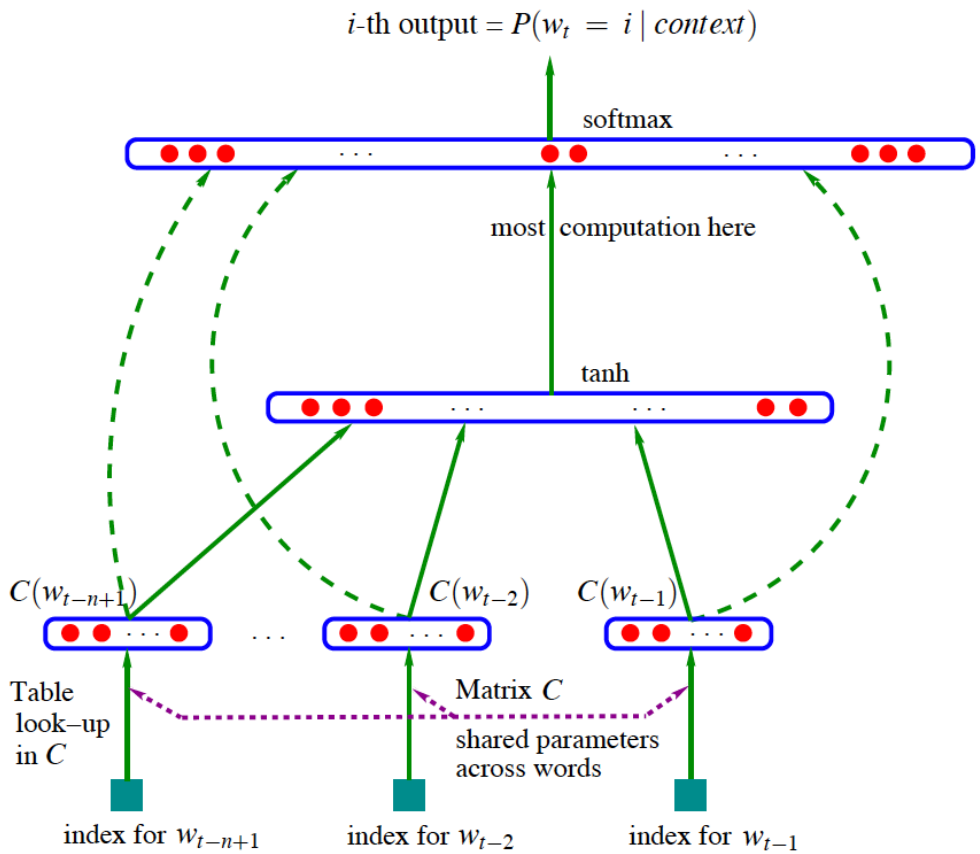


Distributed Word Representation

- Each word are represented as dense, real-valued, and low-dimensional vector



Neural Language Model



Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. J. Mach. Learn. Res. 3 (March 2003), 1137-1155.

Objective of Neural Language Model

The training set is a sequence $w_1, ..., w_T$ of words $w_t \in V$, and the objective is to learn a good model

$$L(w_t, ..., w_{t-n+1}) = P(w_t | w_1^{t-1})$$

where $w_i^j = (w_i, w_{i+1}, \cdots, w_{j-1}, w_j)$.

In general, there are two mappings:

- A mapping C from any element i of V to a real vector $C(i) \in R^m$.
  - A distributed feature vector.
  - C is a matrix $|V| \times m$.
- A mapping function g maps an input vectors, $(C(w_{t-n+1}), \cdots, C(w_{t-1}))$, to a conditional probability distribution over words in V for the next word w_t.
  - The output of g is a vector whose i-th element estimates the probability $P(w_t = i | w_q^{t-1})$.

Solving Neural Language Model

The overall parameter set is $\theta = (C, \omega)$, training is achieved by looking for θ that maximizes the training corpus penalized log-likelihood:

- $L = \frac{1}{T} \sum_t \log f(w_t, w_{t-1}, \cdots, w_{t-n+1;\theta}) + R(\theta)$.
- **Forward propagation**
  - Input: $x = (C(w_{t-n+1}), \cdots, C(w_{t-1}))$
  - Output: $y = b + Wx + U \tanh(d + Hx)$
  - Apply softmax on output layer: $P(w_t | w_{t-1}, \cdots, w_{t-n+1}) = \frac{e^{y w_t}}{\sum_i e^{y_i}}$
- The parameter is $\theta = (b, d, W, U, H, C)$

Complexity of Neural Language Model

- $n \times m \times h + h \times |V| + n \times m \times |V|$
  - The dominating term is the last two terms
  - n is context size, m is word vector size, h is hidden layer size, $|V|$ is the vocabulary size.
- Several methods, such as binary tree and Huffman binary tree has been proposed for reducing $|V|$ to $\log |V|$—e.g., Hierarchical softmax (we will talk later).

Outline

1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary
2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec

Word2Vec

Motivation

- For NNLM, most of the complexity is caused by the non-linear hidden layer in the model.
  - While this is what makes neural networks so attractive
  - It is necessary to explore simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.
- It was found that neural network language model can be successfully trained in two steps
  - Continuous word vectors are learned using simple model
  - The N-gram NNLM is trained on top of these distributed representations of words.
- Word2Vec Model focuses on learn the word representations efficiently.

Word2Vec

There are two models[2] [3] [4]

- Continuous Bag-of-Words Model
- Continuous Skip-gram Model

For each model, there are two learning methods:

- Hierarchical Softmax
- Negative Sampling

---

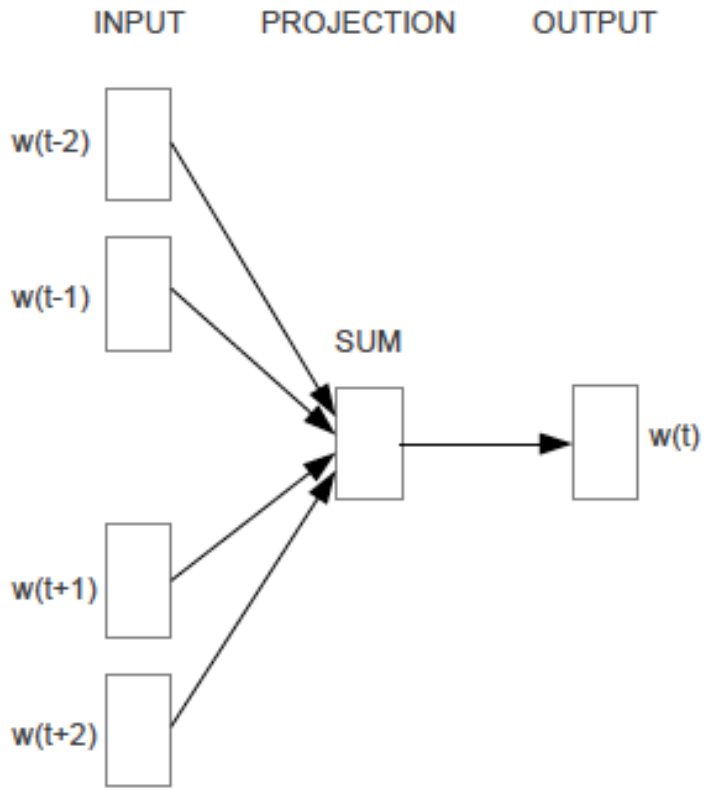
[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeff Dean, Efficient Estimation of Word Representations in Vector Space. In Proceedings of CoRR. 2013.

[3] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In AISTATS 2005.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In proceedings of NIPS. 2013.

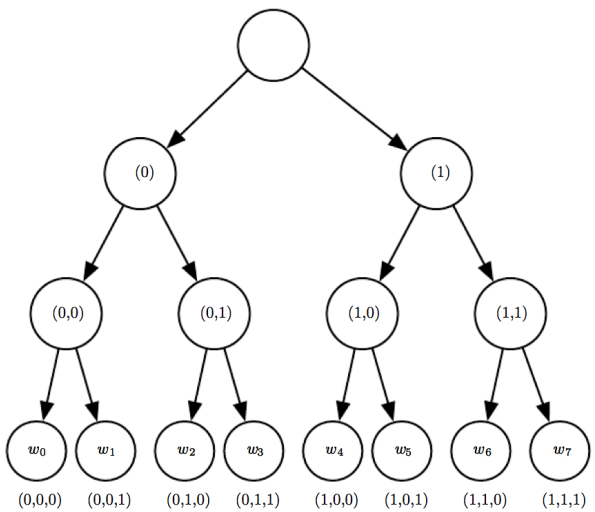
Continuous Bag-of-Words Model

- The non-linear hidden layer is removed.
- All the words get projected into the same position (their vectors are averaged).
- The model is called bag-of words model as the order of words in the history does not influence the projection.
- The words in the future are also considered.
- The complexity is $n \times m + m \times \log_2 |V|$

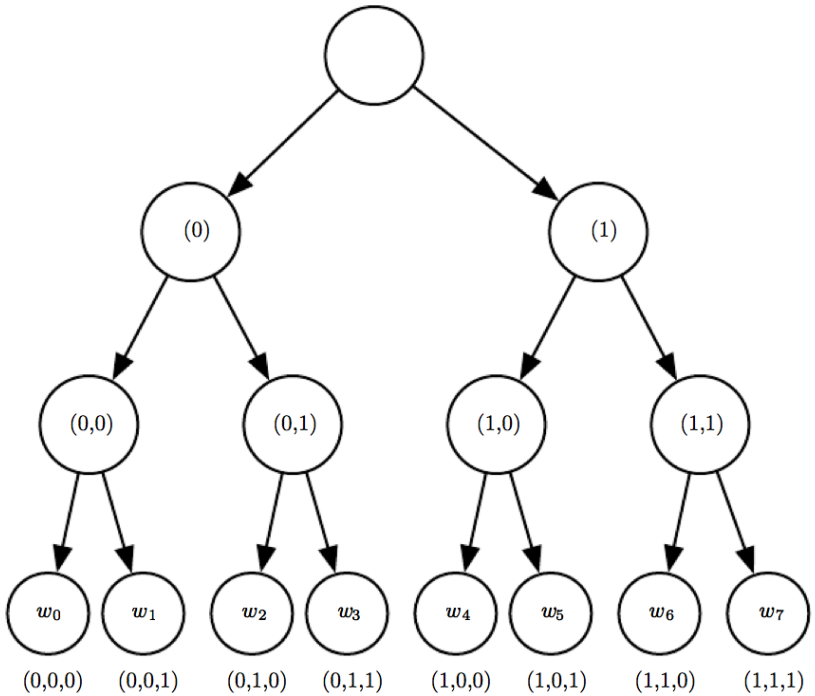


Hierarchical Softmax for CBOW

- The goal is maximize $\prod_{i=1}^{T} p(w_i|C_i)$ (C_i is the projection of the contextual word vectors of word w_i), however
  - If use energy function and softmax function, $p(w_i|C_i) = \frac{\exp(w_i \cdot C_i)}{\sum_{v=1}^{|V|} exp(w_v \cdot C_i)}$.
  - It need to scan all the words in the vocabulary to get the probability.
- An efficient way is to
  - Build a binary tree for all words in the vocabulary, with words as leaves, the left child being encoded as 1 and the right child being encoded as 0;
  - Each word can be assigned a unique code from the root to the leaf.



Hierarchical Softmax for CBOW



- Then

$$p(w_i|C_i) = \prod_{k=1}^{K} p(d_k|q_k, C_i) = \prod_{k=1}^{K} \left(\sigma(q_k \cdot C_i)^{1-d_k} (1 - \sigma(q_k \cdot C_i))^{d_k} \right)$$

where σ is the sigmoid function, q_k is the vector of non-leaf node on the path from root to word leaf, d_k is the corresponding code of q_k.

Hierarchical Softmax for CBOW

- The negative log-likelihood of one word given its context is

$$L_i = \sum_{k=1}^{K} ((1 - d_k) \log \sigma(q_k \cdot C_i) + d_k \log(1 - \sigma(q_k \cdot C_i)))$$

- Then each non-leaf node vector can be updated by

$$q_k^{(n+1)} = q_k^{(n)} - \eta \frac{\partial L_i}{\partial q_k}$$

$$\frac{\partial L_i}{\partial q_k} = -(1 - d_k - \sigma(q_k \cdot C_i)) \times C_i$$

- For an input word, its weights can be updated using context gradient:

$$w_i^{(n+1)} = w_i^{(n)} - \eta \frac{\partial L_i}{\partial C_i}$$

$$\frac{\partial L_i}{\partial C_i} = -\sum_{k=1}^{K} (1 - d_k - \sigma(q_k \cdot C_i)) \times q_k$$

Negative Sampling for CBOW

- Another efficient way to calculate $p(w_i|C_i)$ is to sample a few words in the vocabulary instead of scanning all the words in the vocabulary.

$$\begin{aligned}\frac{\partial L_i}{\partial \theta} = \frac{\partial \log p(w_i|C)}{\partial \theta} &= \frac{\partial \log \text{softmax}_{w_i}(a)}{\partial \theta} \\ &= \frac{\partial}{\partial \theta} \log \frac{e^{\alpha_{w_i}}}{\sum_k e^{\alpha_k}} \\ &= \frac{\partial \alpha_{w_i}}{\partial \theta} - \sum_k p(w_k|C) \frac{\partial \alpha_{w_k}}{\partial \theta}\end{aligned}$$

- For a context C_i of word w_i, the objective is to maximize the likelihood of $p(w_i|C_i)$ (positive instance), while minimize the likelihood of $p(w_k|C_i)$ for other sampled word w_k (negative instance).
- The intuition is to distinguish the input word w_i from the draws from the noise distribution $p_V(w)$, which can be represented by word frequent distribution.

Negative Sampling for CBOW

- Assume the label of a positive instance as 1, and the label of a negative instance as 0, then the negative log-likelihood of each instance is represented uniformly as:

$$L_w = -label \times \log \sigma(w \cdot C_i) - (1 - label) \times \log(1 - \sigma(w \cdot C_i))$$

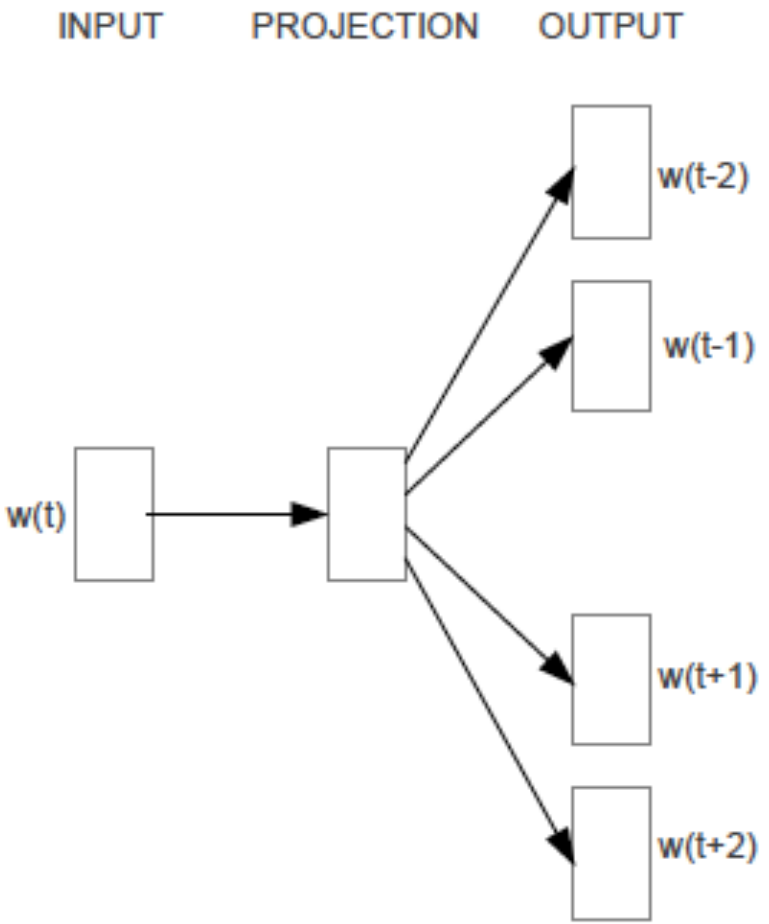
$$\frac{\partial L_w}{\partial w} = -(label - \sigma(w \cdot C_i)) \times C_i$$

$$\frac{\partial L_w}{\partial C_i} = -(label - \sigma(w \cdot C_i)) \times w$$

- Then the input word vector can be updated by using the gradient of context vector:
$w_i^{(n+1)} = w_i^{(n)} - \eta(\frac{\partial L_{w_i}}{\partial C_i} + \sum_{k=1}^{K} \frac{\partial L_{w_k}}{\partial C_i})$ (1 positive and K negative gradients)

Skip-gram Model

- The second model predicts the context based on the current word.
- Maximize classification of a word based on another word in the same sentence, more precisely,
  - Use each current word as an input to a log-linear classifier with continuous projection layer
  - Predict words within a certain range before and after the current word
- The complexity is also $n \times m + m \times \log_2 |V|$



Hierarchical Softmax for Skip-gram

- The log-likelihood of generating context word w_o given input word w_i is defined as:

$$L_{io} = \sum_{k=1}^{K} ((1 - d_k)\sigma(q_k \cdot w_i) + d_k(1 - \sigma(q_k \cdot w_i)))$$

- Then each non-leaf node vector can be updated by

$$q_k^{(n+1)} = q_k^{(n)} - \eta \frac{\partial L_{io}}{\partial q_k}$$

$$\frac{\partial L_{io}}{\partial q_k} = -(1 - d_k - \sigma(q_k \cdot w_i)) \times w_i$$

- Then the input word vector can be updated by :

$$w_i^{(n+1)} = w_i^{(n)} - \eta \frac{\partial L_i}{\partial w_i}$$

$$\frac{\partial L_i}{\partial w_i} = -\sum_{k=1}^{K} (1 - d_k - \sigma(q_k \cdot w_i)) \times q_k$$

Negative Sampling for Skip-gram

- The loglikelihood of generating context word w_o given input word w_i is defined as:

$$L_{io} = \log \sigma(w_i \cdot w_o) + \sum_{k=1}^{K} E_{w_k \ p_V(w)} \log(1 - \sigma(w_k \cdot w_i))$$

- Assume the label of a positive instance as 1, and the label of a negative instance as 0, then the negative log-likelihood of each instance is represented uniformly as:

$$L_w = -label \times \log \sigma(w \cdot w_i) - (1 - label) \times \log(1 - \sigma(w \cdot w_i))$$

$$\frac{\partial L_w}{\partial w} = -(label - \sigma(w \cdot w_i)) \times w_i$$

$$\frac{\partial L_w}{\partial w_i} = -(label - \sigma(w \cdot w_i)) \times w$$

- Then the input word vector can be updated using the gradient of context vector: $w_i^{(n+1)} = w_i^{(n)} - \eta(\frac{\partial L_{w_i}}{\partial w_i} + \sum_{k=1}^{K} \frac{\partial L_{w_k}}{\partial w_i})$

SGNS as Implicit Matrix Factorization

- Skip-gram with negative sampling (SGNS) embeds both words and their contexts into a low-dimensional space $\mathbb{R}^d$, resulting in the word and context matrices W and C.
- The rows of matrix W are typically used in NLP tasks (such as computing word similarities) while C is ignored. It is nonetheless instructive to consider the product $W \times C^T = M$.
- Viewed this way, SGNS can be described as factorizing an implicit matrix M of dimensions $|V_W| \times |V_C|$ into two smaller matrices.

SGNS as Implicit Matrix Factorization (cont.)

The global objective sums over the observed (w, c) pairs in the corpus:

$$\ell = \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) \left(\log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \mathbb{E}_{c_N \sim P_D} \left[\log \sigma \left(-\vec{w} \cdot \vec{c}_N \right) \right] \right)$$

We begin by rewriting the global objective:

$$\begin{aligned} \ell &= \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) (\log \sigma(\vec{w} \cdot \vec{c})) + \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) \left(k \cdot \mathbb{E}_{c_N \sim P_D} \left[\log \sigma \left(-\vec{w} \cdot \vec{c}_N \right) \right] \right) \\ &= \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) (\log \sigma(\vec{w} \cdot \vec{c})) + \sum_{w \in V_W} \#(w) \left(k \cdot \mathbb{E}_{c_N \sim P_D} \left[\log \sigma \left(-\vec{w} \cdot \vec{c}_N \right) \right] \right) \end{aligned}$$

and explicitly expressing the expectation term:

$$\begin{aligned} &\mathbb{E}_{c_N \sim P_D} \left[\log \sigma \left(-\vec{w} \cdot \vec{c}_N \right) \right] = \sum_{c_N \in V_C} \frac{\#(c_N)}{|D|} \log \sigma \left(-\vec{w} \cdot \vec{c}_N \right) \\ &= \frac{\#(c)}{|D|} \log \sigma(-\vec{w} \cdot \vec{c}) + \sum_{c_N \in V_C \setminus \{c\}} \frac{\#(c_N)}{|D|} \log \sigma \left(-\vec{w} \cdot \vec{c}_N \right) \end{aligned}$$

SGNS as Implicit Matrix Factorization (cont.)

Combining the equations reveals the local objective for a specific (w, c) pair:

$$\ell(w, c) = \#(w, c) \log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \#(w) \cdot \frac{\#(c)}{|D|} \log \sigma(-\vec{w} \cdot \vec{c})$$

To optimize the objective, we define $x = \vec{w} \cdot \vec{c}$ and find its partial derivative with respect to x:

$$\frac{\partial \ell}{\partial x} = \#(w, c) \cdot \sigma(-x) - k \cdot \#(w) \cdot \frac{\#(c)}{|D|} \cdot \sigma(x)$$

We compare the derivative to zero, and arrive at:

$$e^{2x} - \left(\frac{\#(w, c)}{k \cdot \#(w) \cdot \frac{\#(c)}{|D|}} - 1 \right) e^{x} - \frac{\#(w, c)}{k \cdot \#(w) \cdot \frac{\#(c)}{|D|}} = 0$$

SGNS as Implicit Matrix Factorization (cont.)

If we define $y = e^x$, this equation becomes a quadratic equation of y, which has two solutions, $y = -1$ (which is invalid), and

$$y = \frac{\#(w,c)}{k \cdot \#(w) \cdot \frac{\#(c)}{|D|}} = \frac{\#(w,c) \cdot |D|}{\#w \cdot \#(c)} \cdot \frac{1}{k}$$

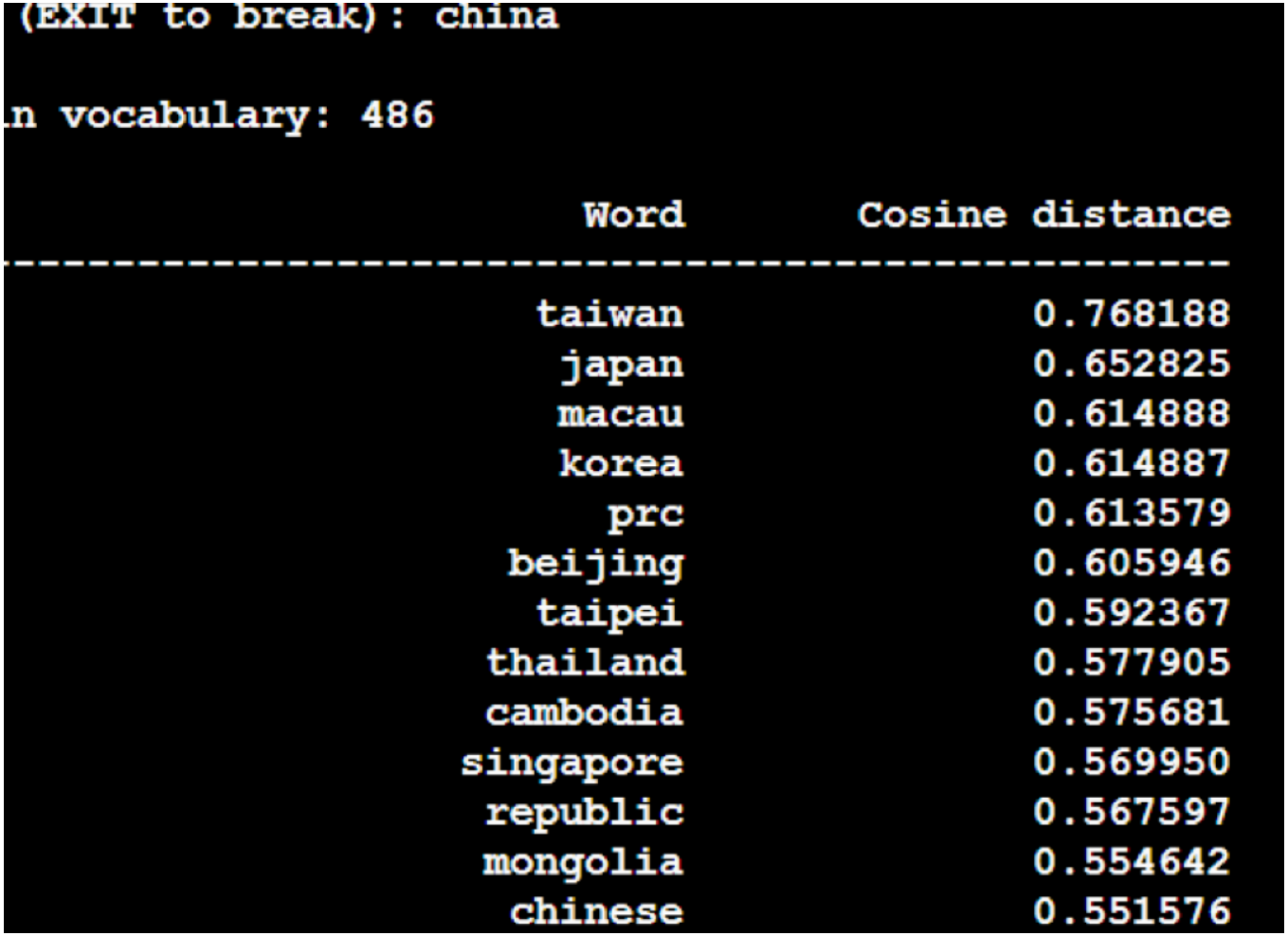
Substituting y with e^x and x with $\vec{w} \cdot \vec{c}$ reveals:

$$\vec{w} \cdot \vec{c} = \log\left(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)} \cdot \frac{1}{k}\right) = \log\left(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)}\right) - \log k$$

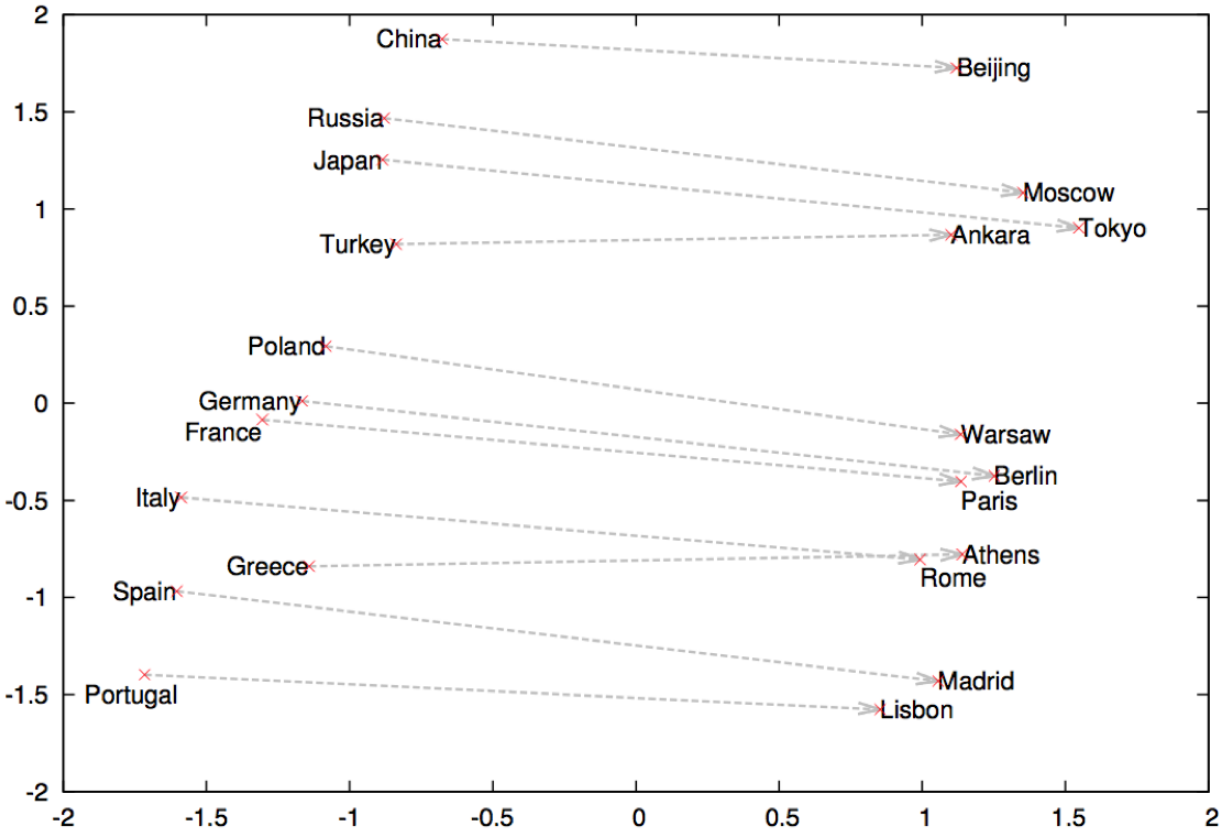
Interestingly, the expression $\log\left(\frac{\#(w,c)\cdot|D|}{\#(w)\cdot\#(c)}\right)$ is the well-known pointwise mutual information (PMI) of (w,c). Finally, the matrix M that SGNS is factorizing is

$$M_{ij}^{\mathrm{SGNS}} = W_i \cdot C_j = \vec{w}_i \cdot \vec{c}_j = PMI\left(w_i, c_j\right) - \log k$$

Word Relatedness

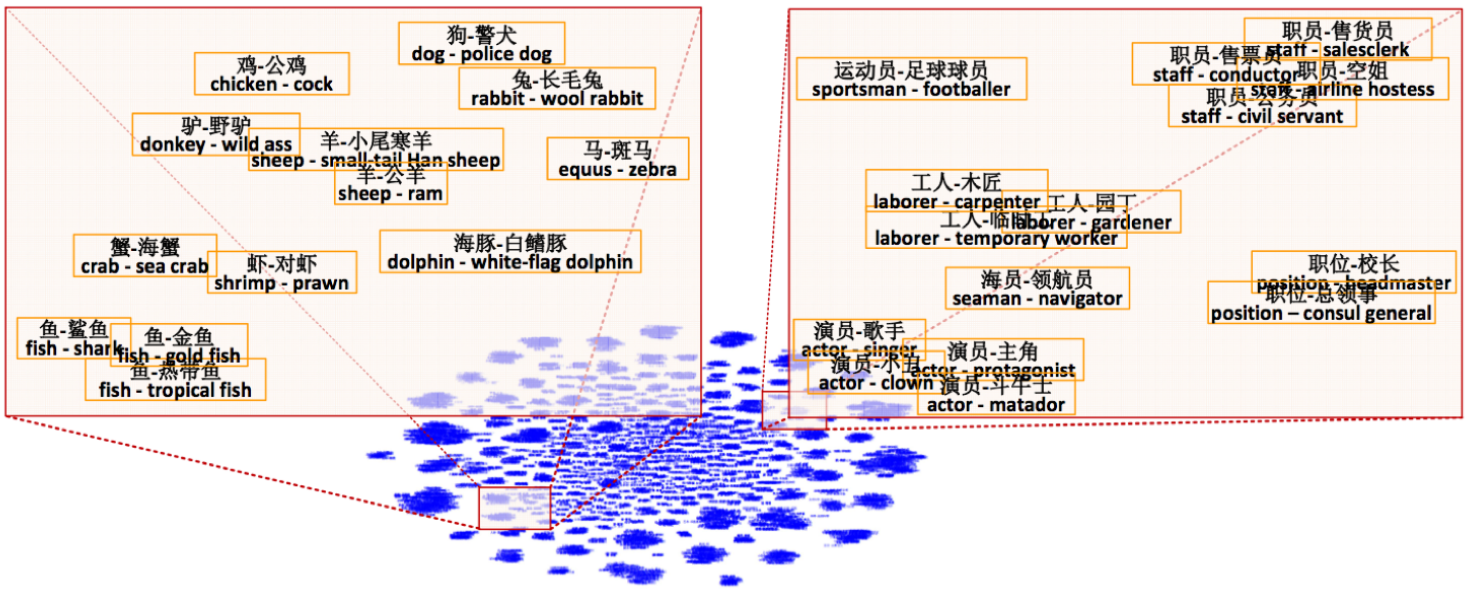


Semantic Space Encode Implicit Relationships between Words



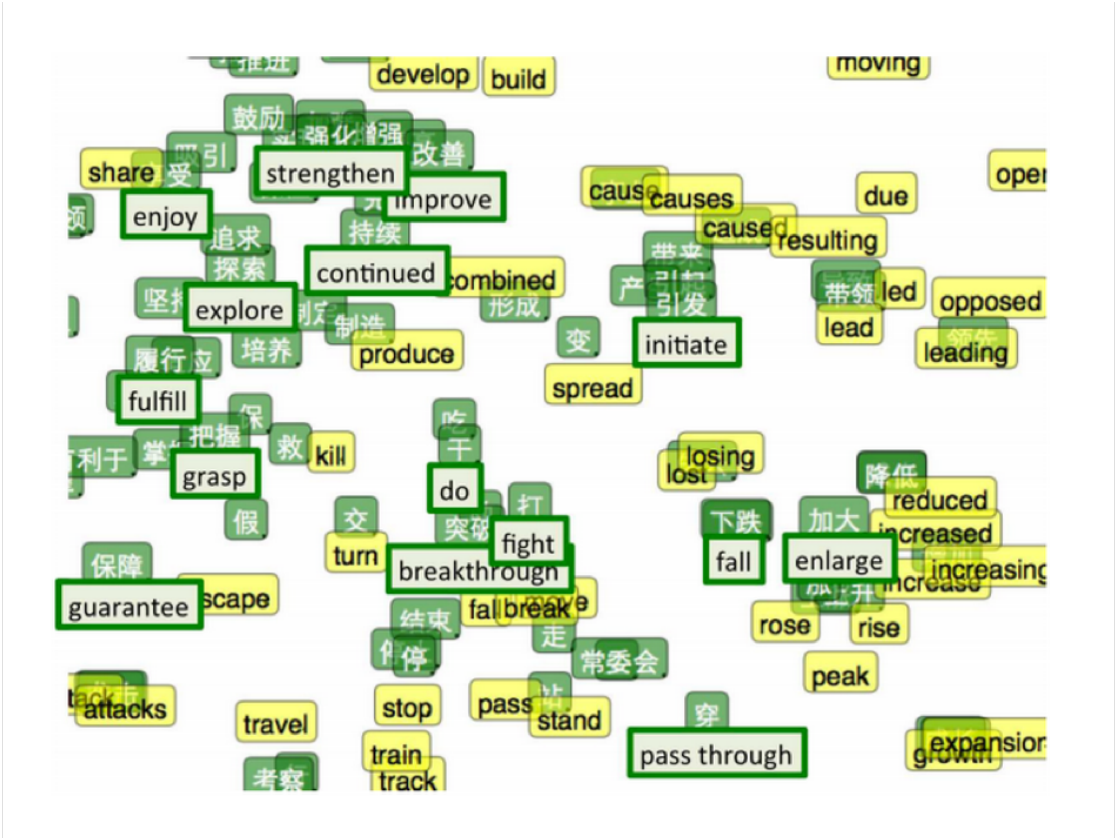
$$W(\text{"}China\text{"}) - W(\text{"}Beijing\text{"}) \approx W(\text{"}Japan\text{"}) - W(\text{"}Tokyo\text{"})$$

Applications: Semantic Hierarchy Extraction[5]



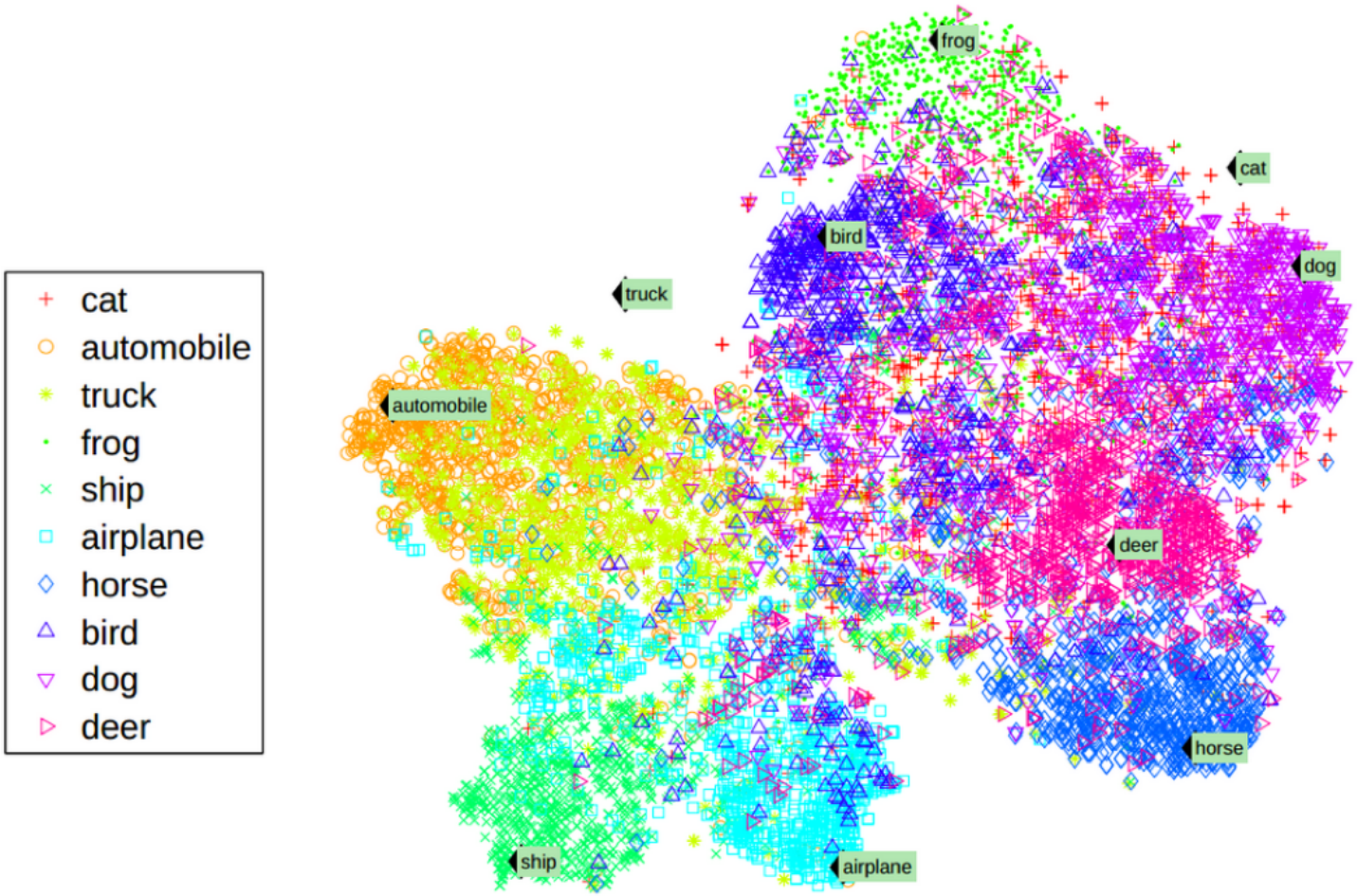
[5] Fu, Ruiji, et al. Learning semantic hierarchies via word embeddings. ACL 2014.

Applications: Cross-lingual Joint Representation



Zou, Will Y., et al. Bilingual word embeddings for phrase-based machine translation. EMNLP 2013.

Applications: Visual-Text Joint Representation



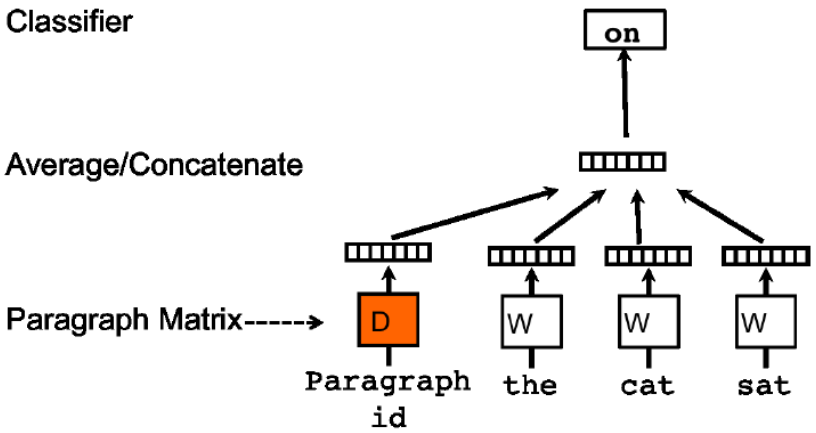
Richard Socher, et al. Zero-Shot Learning Through Cross-Modal Transfer. ICLR 2013

Outline

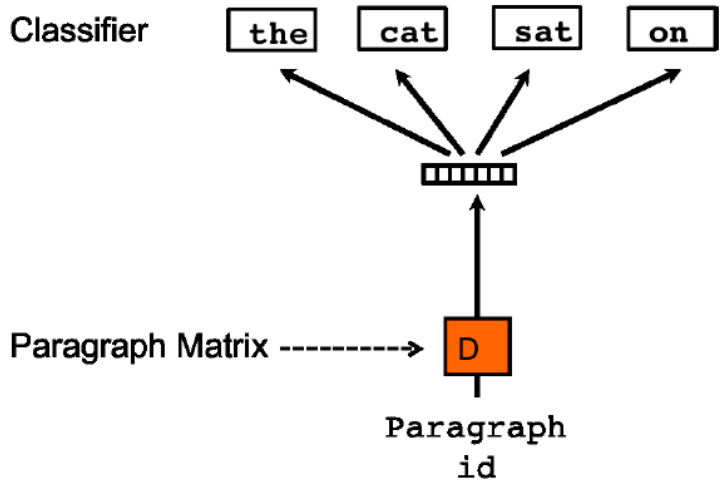
1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary
2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec

Word2Vec for Documents

The objective is to calculate both the word vector and document vector[6].



The document vector is shared across all contexts generated from the same document but not across documents.



Sample a context word from both the input word vector and the document vector.

[6] Quoc V. Le, Tomas Mikolov. Distributed Representations of Sentences and Documents. In Proceedings of ICML. 2014.

Document Representation Models

- Replicated Softmax: an Undirected Topic Model (NIPS 2010)
- A Deep Architecture for Matching Short Texts (NIPS 2013)
- Modeling Documents with a Deep Boltzmann Machine (UAI 2013)
- A Convolutional Neural Network for Modeling Sentences (ACL 2014)
- Convolutional Neural Network Architectures for Matching Natural Language Sentences (NIPS 2014)

Outline

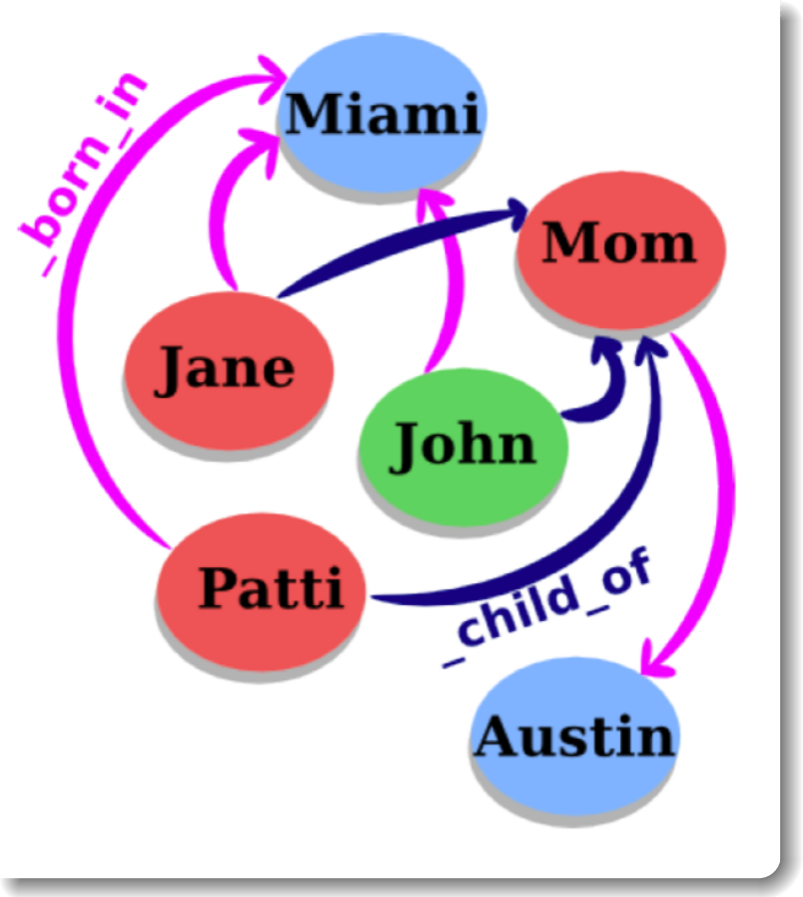
1. Representation Learning
   - Distributed Representation
   - Deep Architecture
   - Multi-task Learning and Multi-modal Learning
   - Semi-supervised Learning
   - Summary
2. Representation Learning in NLP
   - Neural Language Model
   - Word2Vec
   - Doc2Vec
   - KnowledgeBase2Vec

Knowledge Bases and Knowledge Graphs

Knowledge is structured as a graph

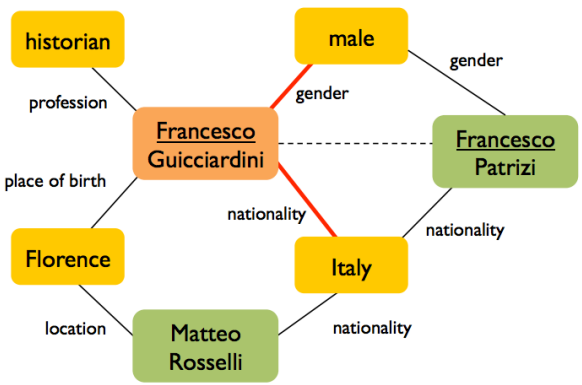
- Each node = an entity
- Each edge = a relation
- A relation = (head, relation, tail), where
  - head = subject entity
  - relation = relation type
  - tail = object entity

Typical knowledge bases include WordNet, Freebase, and so on.



Research Issues

- KG is far from complete, we need relation extraction
- Relation extraction from text: information extraction
- Relation extraction from KG: knowledge graph completion
- Issues: KGs are hard to manipulate
  - High dimensions: $10^5 \sim 10^8$ entities, $10^7 \sim 10^9$ relation types
  - Sparse: few valid links
  - Noisy and incomplete
- How to encode KGs into low-dimensional vector spaces

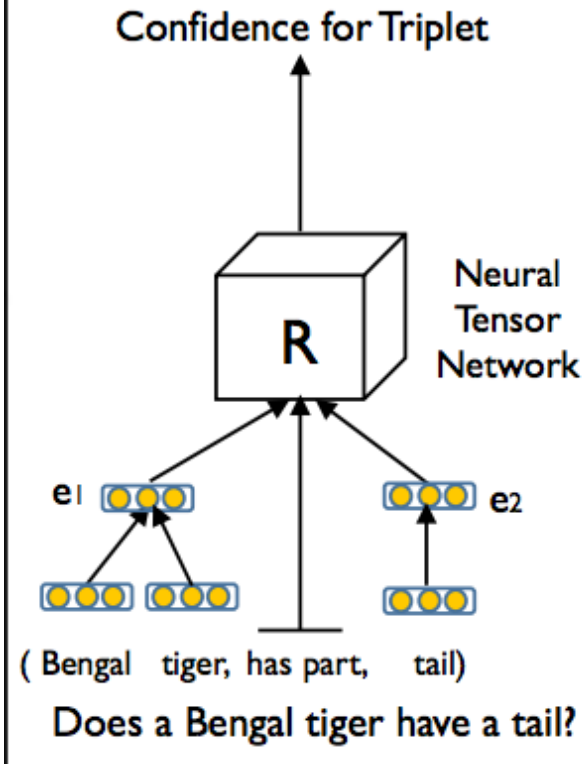


- How can we infer that Francesco Guicciardini is an Italian male ?

Overview of Neural Tensor Networks

General Idea of NTN[a]

- Each relation is described by a neural network and pairs of entities are given as input to the model.
- The model returns a high score if they are in that relationship and a low one otherwise. This allows any fact, whether implicitly or explicitly mentioned in the database to be answered with a certainty score.



[a] Richard Socher, Danqi Chen, Christopher Manning, Andrew Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In NIPS 2013.

Neural Tensor Networks

- The Neural Tensor Network (NTN) replaces a standard linear neural network layer with a bilinear tensor layer that directly relates the two entity vectors across multiple dimensions. The model computes a score of how likely it is that two entities are in a certain relationship by the following NTN-based function.

$$g(e_1, R, e_2) = u_R^T f(e_1^T W_R^{[1:k]} e_2 + V_r [e_1 e_2] + b_R)$$

  - $f = \tanh$ is a standard nonlinearity applied element-wise. $W_R^{[1:k]} \in \mathcal{R}^{d \times d \times k}$ is a tensor and the bilinear tensor product $e_1^T W_R^{[1:k]} e_2$ results in a vector $h \in \mathcal{R}^k$ with each entry computed by one slice the tensor. The other parameters for relation R are the standard form of a neural network: $V_R \in \mathcal{R}^{k \times 2d}$ and $U \in \mathcal{R}^k$, $b_R \in \mathcal{R}^k$.

NTN: Training Objective

- **Training objective:** $T_c^{(i)} = (e_1^{(i)}, R^{(i)}, e_c)$ is a triplet with a random entity corrupted from a correct triplet $T^{(i)} = (e_1^{(i)}, R^{(i)}, e_2)$,

$$J(\Omega) = \sum_{i=1}^{N} \sum_{c=1}^{C} \max\left(0, 1 - g(T^{(i)}) + g(T_c^{(i)})\right) + \lambda ||\Omega||_2^2$$

Example of Neural Tensor Networks: $k = 2$

Fig. 2 shows a visualization of this model. The main advantage is that it can relate the two inputs multiplicatively instead of only implicitly through the nonlinearity as with standard neural networks where the entity vectors are simply concatenated. Intuitively, we can see each slice of the tensor as being responsible for one type of entity pair or instantiation of a relation. For instance, the model could learn that both animals and mechanical entities such as cars can have parts (i.e., (car, has part, x)) from different parts of the semantic word vector space. In our experiments, we show that this results in improved performance. Another way to interpret each tensor slice is that it mediates the relationship between the two entity vectors differently.

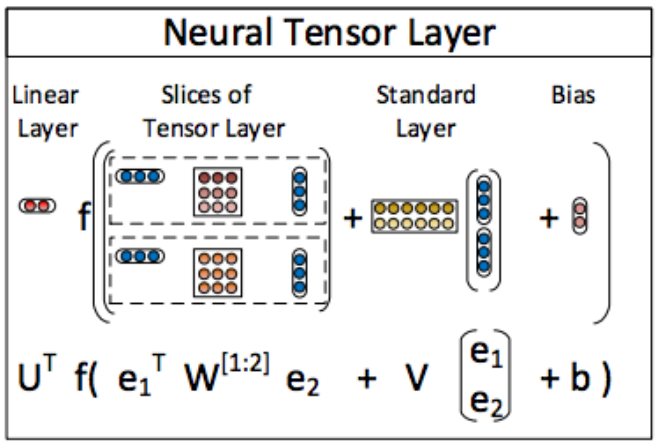
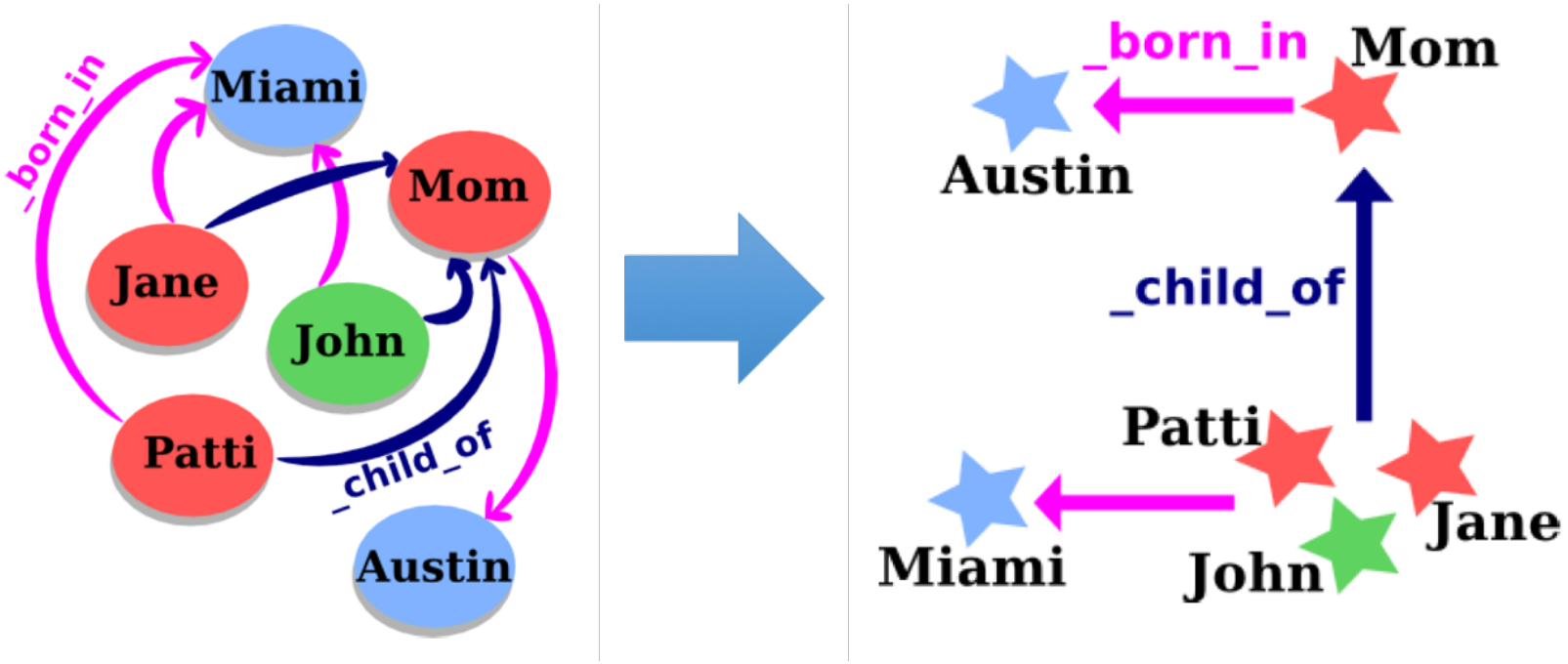


Figure 2: Visualization of the Neural Tensor Network. Each dashed box represents one slice of the tensor, in this case there are $k = 2$ slices.

Translation-based Model[7]

For each (head, relation, tail), relation works as a translation from head to tail.



[7] Bordes A, Usunier N, Garcia-Duran A, et al. Translating embeddings for modeling multi-relational data. Advances in Neural Information Processing Systems. 2013: 2787-2795.

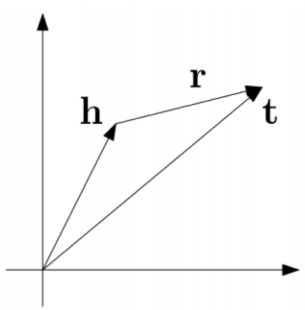
TransE

- **Basic idea**: The functional relation induced by the ℓ-labeled edges corresponds to a translation of the embeddings, i.e., we want $h + \ell \approx t$ when (h, ℓ, t) holds, while $h' + \ell$ should be far away from t otherwise.
- We minimize a margin-based ranking criterion over the training set:

$$\mathcal{L} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} [\gamma + d(h + \ell, t) - d(h' + \ell, t')]_+ \quad (3)$$

where $[x]_+$ denotes the positive part of x, γ is a margin hyperparameter, and

$$S'_{(h,\ell,t)} = \{(h', \ell, t)|h' \in E\} \cup \{(h, \ell, t')|t' \in E\} \quad (4)$$



Summary

Representation Learning in NLP

- Neural Language Model
- Word2Vec
- Doc2Vec
- KnowledgeBase2Vec

Thanks.

HP: http://keg.cs.tsinghua.edu.cn/jietang/
Email: jietang@tsinghua.edu.cn