分布式词表示
Distributed Word Representation

车万翔
社会计算与信息检索研究中心
2017年春季学期

Typical Approaches for Word Representation

- 1-hot representation: basis of bag-of-word 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 ☹

动机

– A bottle of *tezgüino* is on the table
– Everybody likes *tezgüino*
– *Tezgüino* makes you drunk
– We make *tezgüino* out of corn.

- Intuition
  – 人们从词的上下文中就能推测出*tezgüino*的意义
  – Basic idea: two words are similar if they appear in similar contexts

上下文向量 (Context Vector)

- 目标词为w
- 对于词表（包含N个词）中每个词v_i都对应一个二值特征f_i
  - 表示词 v_i 是否在w的附近出现
- $w=(f_1,f_2,f_3,\ldots,f_N)$
  - If w= tezgüino, v_1 = bottle, v_2 = drunk, v_3 = matrix
  - w = (1,1,0,…)

动机

- 用稀疏特征向量定义词语
- 基于向量距离/相似度公式进行计算
- 两个词语向量相似，那么这两个词相似

	arts	boil	data	function	large	sugar	summarized	water
apricot	0	1	0	0	1	1	0	1
pineapple	0	1	0	0	1	1	0	1
digital	0	0	1	1	1	0	1	0
information	0	0	1	1	1	0	1	0

分布式相似度 (Distributional Similarity)

- 四个问题
  1. 如何定义“共同出现”？
  2. 词语权重如何度量？
     - frequency? Logs? Mutual information?
  3. 如何降维？
  4. 如何选择向量距离/相似度计算公式？
     - Cosine? Euclidean distance?

上下文(共现)向量定义

- 基于窗口中的词
- 基于句法结构
  - 进行句法分析，抽取依存关系

I discovered dried tangerines:

discover (subject I)
tangerine (obj-of discover)
dried (adj-mod-of tangerine)
I (subj-of discover)
tangerine (adj-mod dried)

基于依存关系的共现向量

	subj-of, absorb	subj-of, adapt	subj-of, behave	...	pobj-of, inside	pobj-of, into	...	nmod-of, abnormality	nmod-of, anemia	nmod-of, architecture	...	obj-of, attack	obj-of, call	obj-of, come from	obj-of, decorate	...	nmod, bacteria	nmod, body	nmod, bone marrow
cell	1	1	1		16	30		3	8	1		6	11	3	2		3	2	2

向量中词语特征权重计算

- 频率及其变形
- 考虑如下特征
  - f= (obj-of,attack)
  - $P(f|w)=count(f,w)/count(w)$
  - $Assoc_{prob}(w,f)=p(f|w)$

词语权重计算之互信息 (Mutual Information)

- Pointwise mutual information (PMI): measure of how often two events x and y occur, compared with what we would expect if they were independent:

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- PMI between a target word w and a feature f :

$$\text{assoc}_{\text{PMI}}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(f)}$$

互信息举例

- Objects of the verb *drink*

Object	Count	PMI assoc	Object	Count	PMI assoc
bunch beer	2	12.34	wine	2	9.34
tea	2	11.75	water	7	7.65
Pepsi	2	11.75	anything	3	5.15
champagne	4	11.75	much	3	5.15
liquid	2	10.53	it	3	1.25
beer	5	10.20	<SOME AMOUNT>	2	1.22

Latent Semantic Analysis

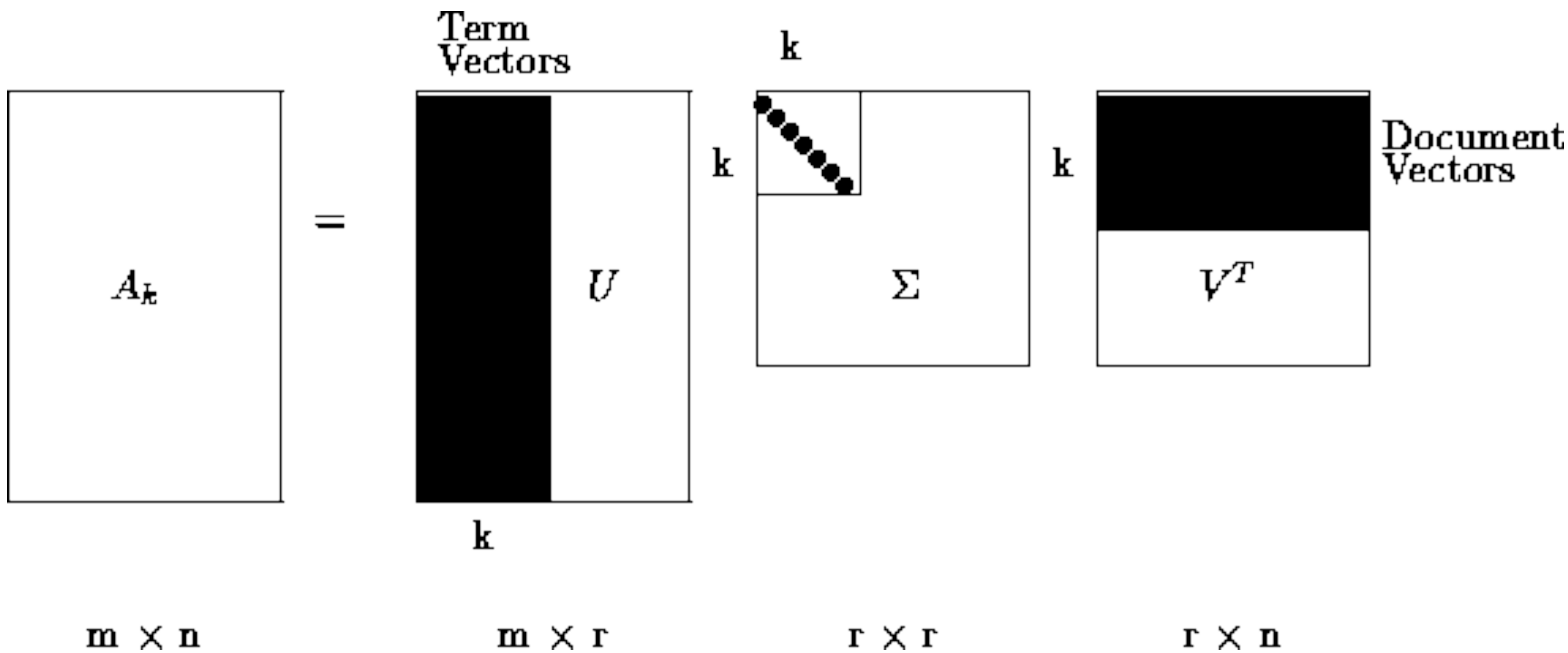
- Start with a Term-by-Document matrix (A)
- Optionally weight cells
- Apply Singular Value Decomposition:
  - m = # of terms
  - n= # of documents
  - r = min(m, n)

$$A_{m\times n} = U_{m\times r} \times \sum_{r\times r} \times (V_{n\times r})^T$$

- Approximate using k (semantic) dimensions:

$$\hat{A}_{m\times n} = U_{m\times k} \times \sum_{k\times k} \times (V_{n\times k})^T$$

Latent Semantic Analysis



Latent Semantic Analysis

- 矩阵U中的每一行表示相应词语与隐含语义空间中语义维度之间的关联，Likewise for V
- 文档相似度: vector comparison of V
- 词语相似度: vector comparison of U

Simple SVD in Python

- Example corpus:
  - *I like deep learning.*
  - *I like NLP.*
  - *I enjoy flying.*

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Simple SVD in Python

- Example corpus:
  - *I like deep learning.*
  - *I like NLP.*
  - *I enjoy flying.*

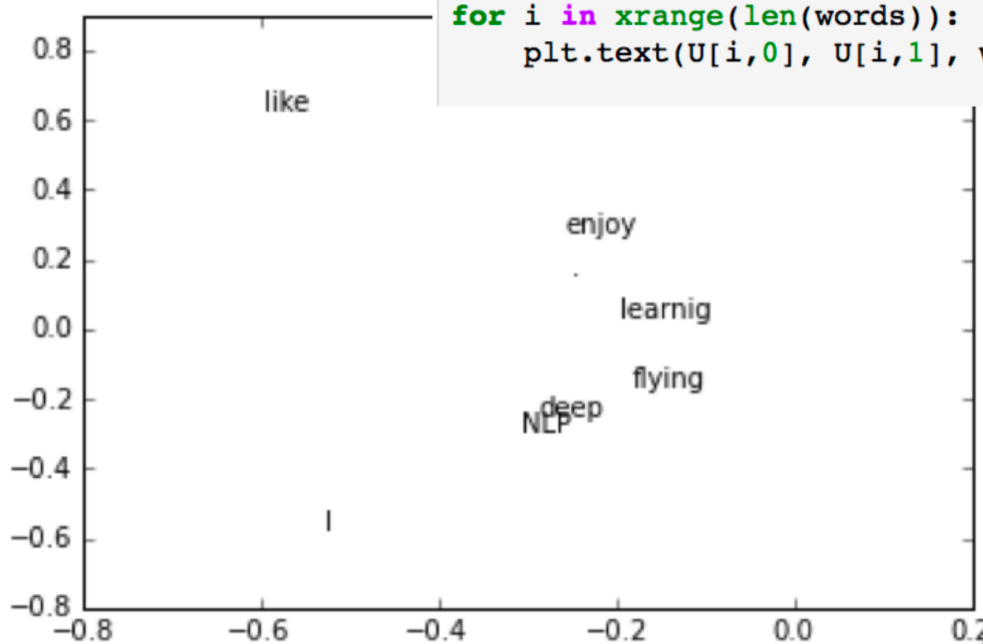
```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep","learnig","NLP","flying","."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

U, s, Vh = la.svd(X, full_matrices=False)
```

Simple SVD in Python

- Example corpus:
  - *I like deep learning.*
  - *I like NLP.*
  - *I enjoy flying.*

```python
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```



相似度计算公式

$$\mathrm{sim}_{\mathrm{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\mathrm{sim}_{\mathrm{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)}$$

$$\mathrm{sim}_{\mathrm{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} (v_i+w_i)}$$

$$\mathrm{sim}_{\mathrm{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2})$$

$$D(p_1(V)\,\|\,p_2(V)) = \sum_v p_1(v)\log\frac{p_1(v)}{p_2(v)}\,.$$

Summary

- Has proved to be a valuable tool in many areas of NLP as well as IR
  - summarization
  - cross-language IR
  - topics segmentation
  - text classification
  - question answering
  - more

Summary

- Some Issues
  - SVD Algorithm complexity O(n^2k^3)
    - n = number of terms
    - k = number of dimensions in semantic space (typically small ~50 to 350)
    - for stable document collection, only have to run once
    - dynamic document collections: might need to rerun SVD, but can also “fold in” new documents

Summary

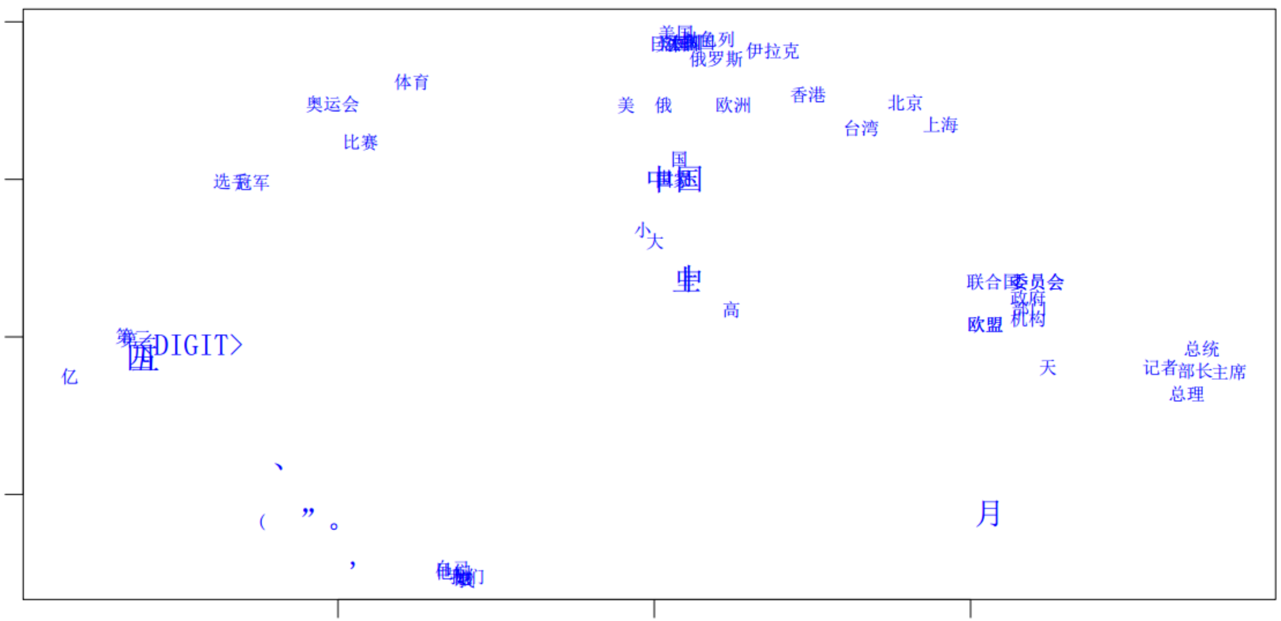
- Some issues
  - Finding optimal dimension for semantic space
    - precision-recall improve as dimension is increased until hits optimal, then slowly decreases until it hits standard vector model
    - run SVD once with big dimension, say k = 1000
      - then can test dimensions <= k
    - in many tasks 150-350 works well, still room for research

Summary

- Ongoing research and extensions include
  - Probabilistic LSA (Hofmann)
  - Iterative Scaling (Ando and Lee)
  - Psychology
    - model of semantic knowledge representation
    - model of semantic word learning

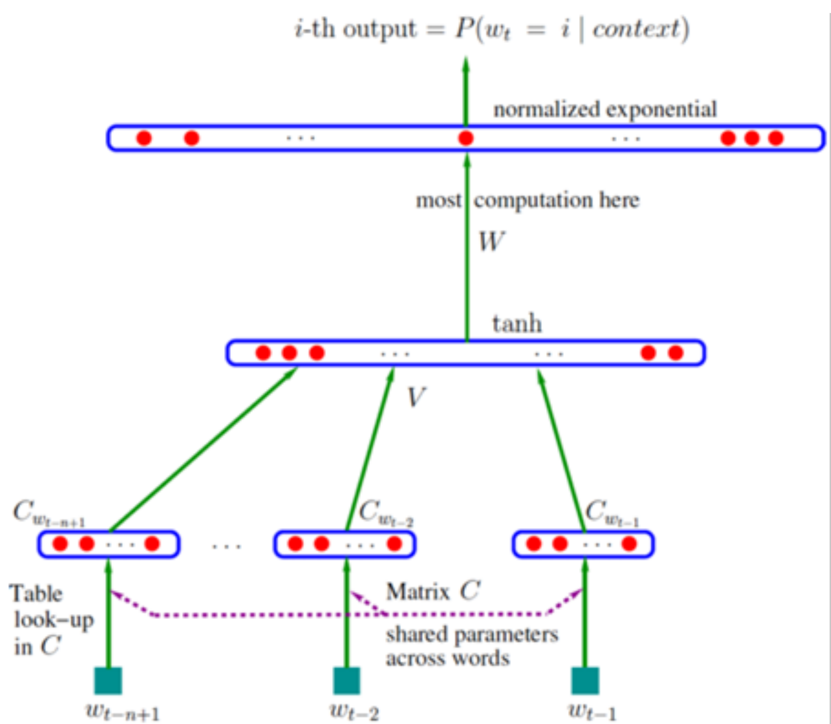
Distributed Word Representation

- 将词映射为低维、稠密、实数向量，又称 "词嵌入"（word embedding）
  - 特指采用有指导学习的方法获得的词向量
- 性质
  - 越相似的词距离越近



Neural Network Language Models

- Neural Network Language Models (NNLM)
  - Feed Forward (Bengio et al. 2003)



- **Maximum-Likelihood Estimation**
- Back-propagation
- Input: $(n-1)$ embeddings

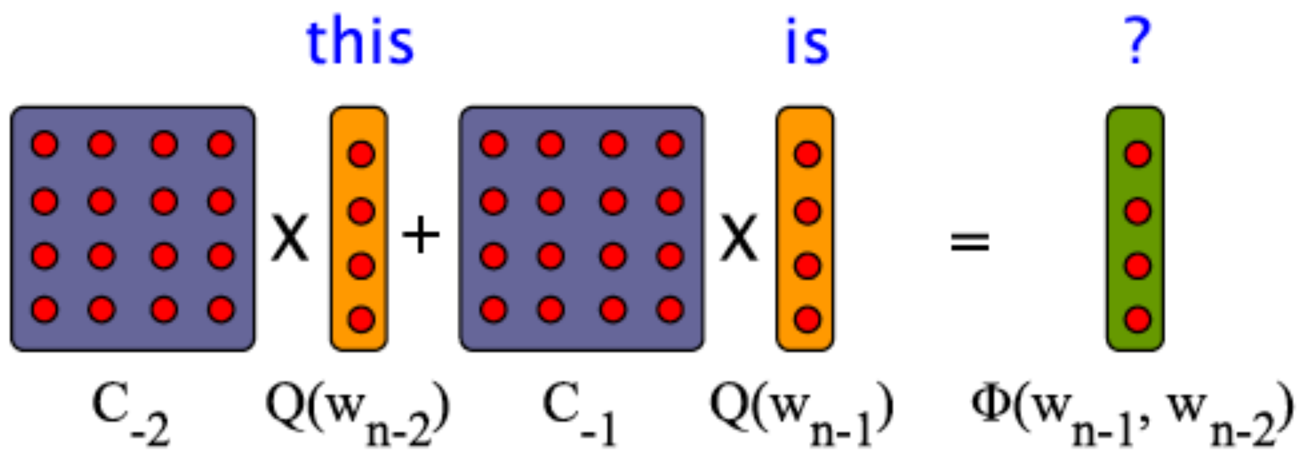
$$P(w_t = k | w_{t-n+1}, \ldots w_{t-1}) = \frac{e^{a_k}}{\sum_{l=1}^{N} e^{a_l}}$$

$$a_k = b_k + \sum_{i=1}^{h} W_{ki} \tanh(c_i + \sum_{j=1}^{(n-1)d} V_{ij} x_j)$$

$$L(\theta) = \sum_t \log P(w_t | w_{t-n+1}, \ldots w_{t-1})$$

Log-Bilinear Language Models

- Log-Bilinear Language Models (Mnih and Hinton 2008)



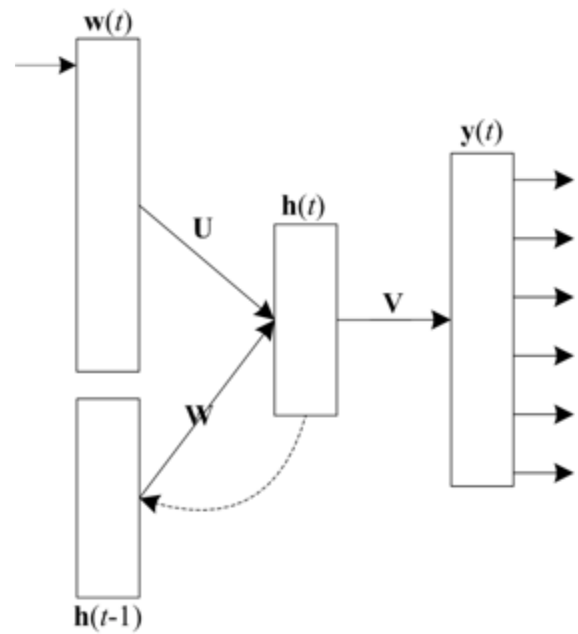
Recurrent Neural Network Language Model (RNNLM)

- Modeling the conditional probability (Mikolov et al. 2010-2013)
  - Compute:

$$\mathbf{h}(t) = f(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{h}(t-1))$$

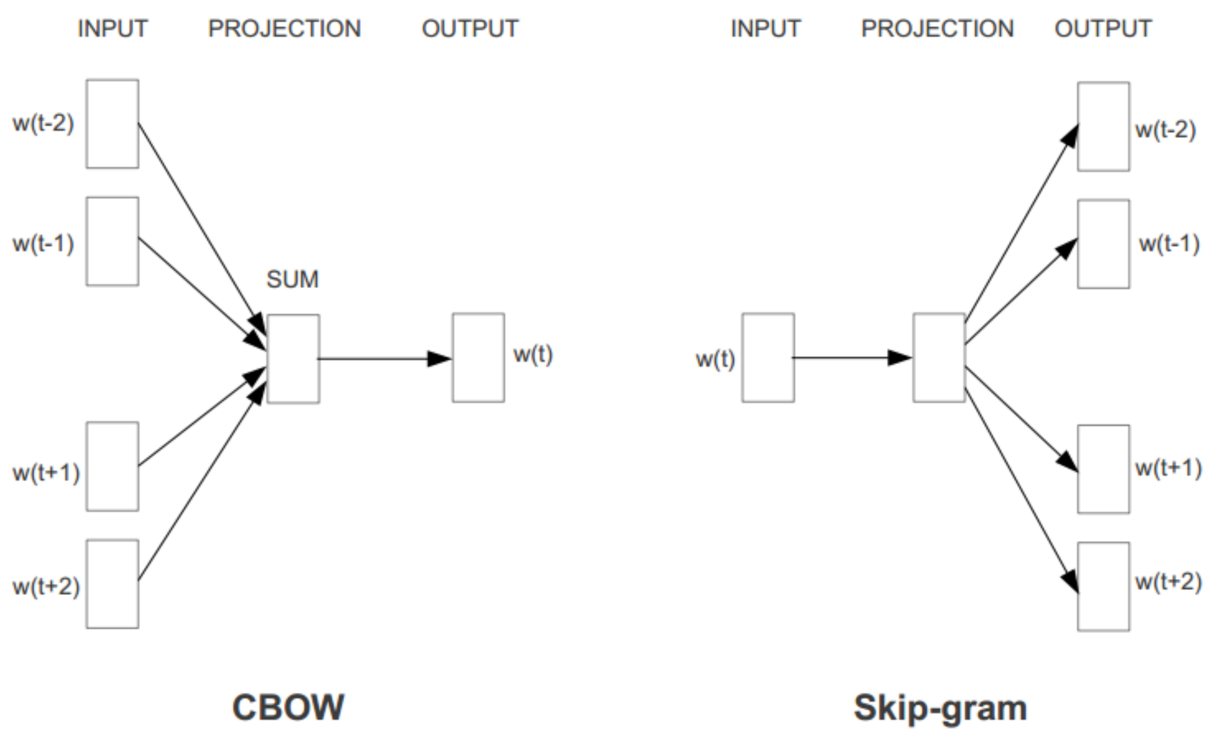
$$\mathbf{y}(t) = g(\mathbf{V}\mathbf{h}(t))$$

  - is the Embedding Matrix



Word2vec

- CBOW and Skip-Gram (Mikolov et al. 2013)



Details of Word2vec

- Predict surrounding words in a window of length m of every word.
- Objective function: Maximize the log probability of any context word given the current center word:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$

Details of Word2vec

$$p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$

- where *o* is the outside (or output) word id, *c* is the center word id, *u* and *v* are "center" and "outside" vectors of *o* and *c*

效率问题

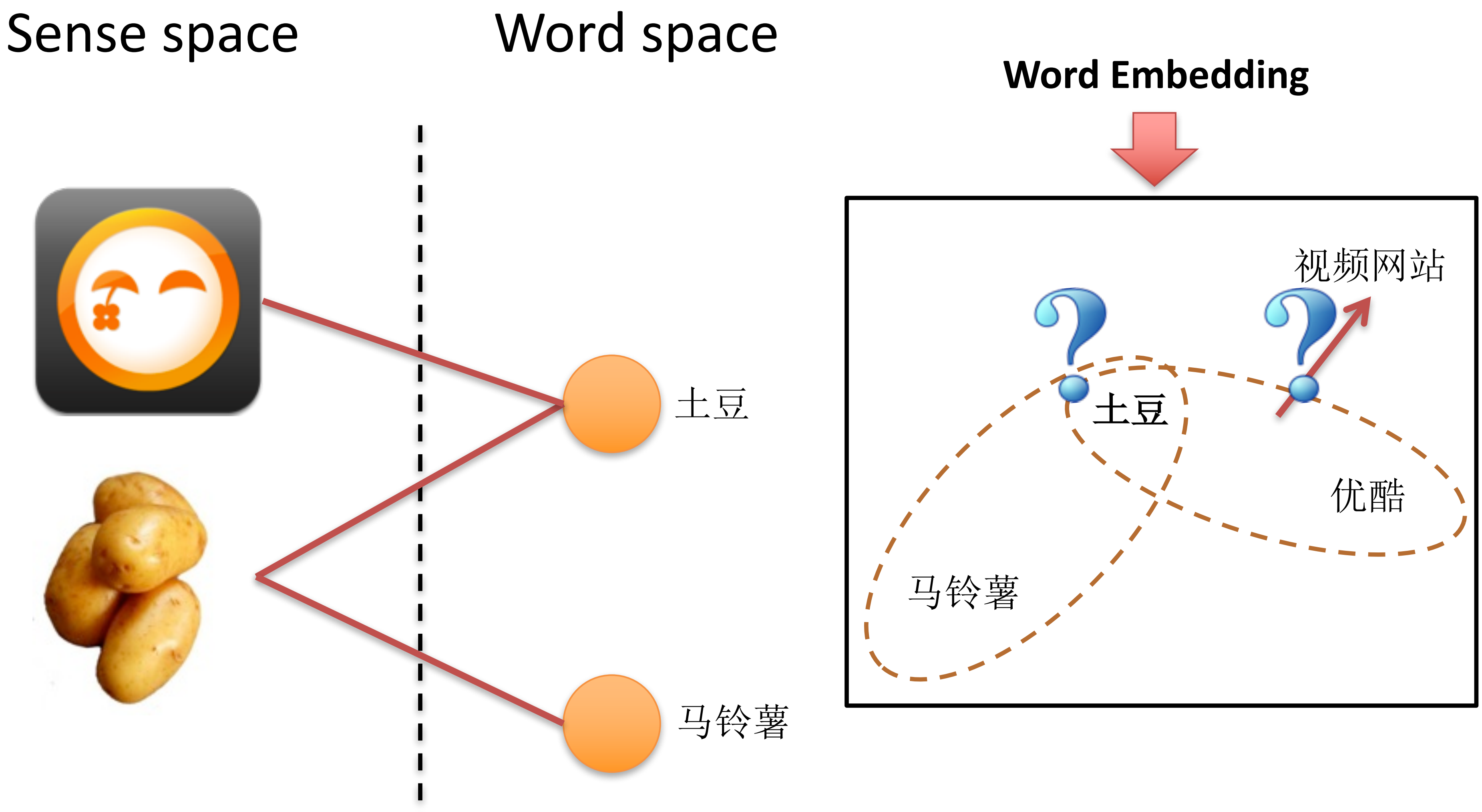
- 由于词表巨大，效率较低
- Training accelerating
  - Noise Contrastive Estimation
    - Schwenk et al. 2013. Mnih et al. 2013
  - Negative Sampling
    - Mikolov et al. 2013
  - Hierarchical Decomposition
    - Morin and Bengio 2005. Mnih and Hinton 2008. Mikolov et al. 2013
  - Graph Processing Unit（GPU）

相关资源

- word2vec
  - https://code.google.com/p/word2vec/
- 多种 Word Embedding 资源及性能比较
  - http://wordvectors.org
- t-SNE
  - Word Embedding 可视化
  - http://lvdmaaten.github.io/tsne/

WORD EMBEDDING 应用

Word and Sense Mapping



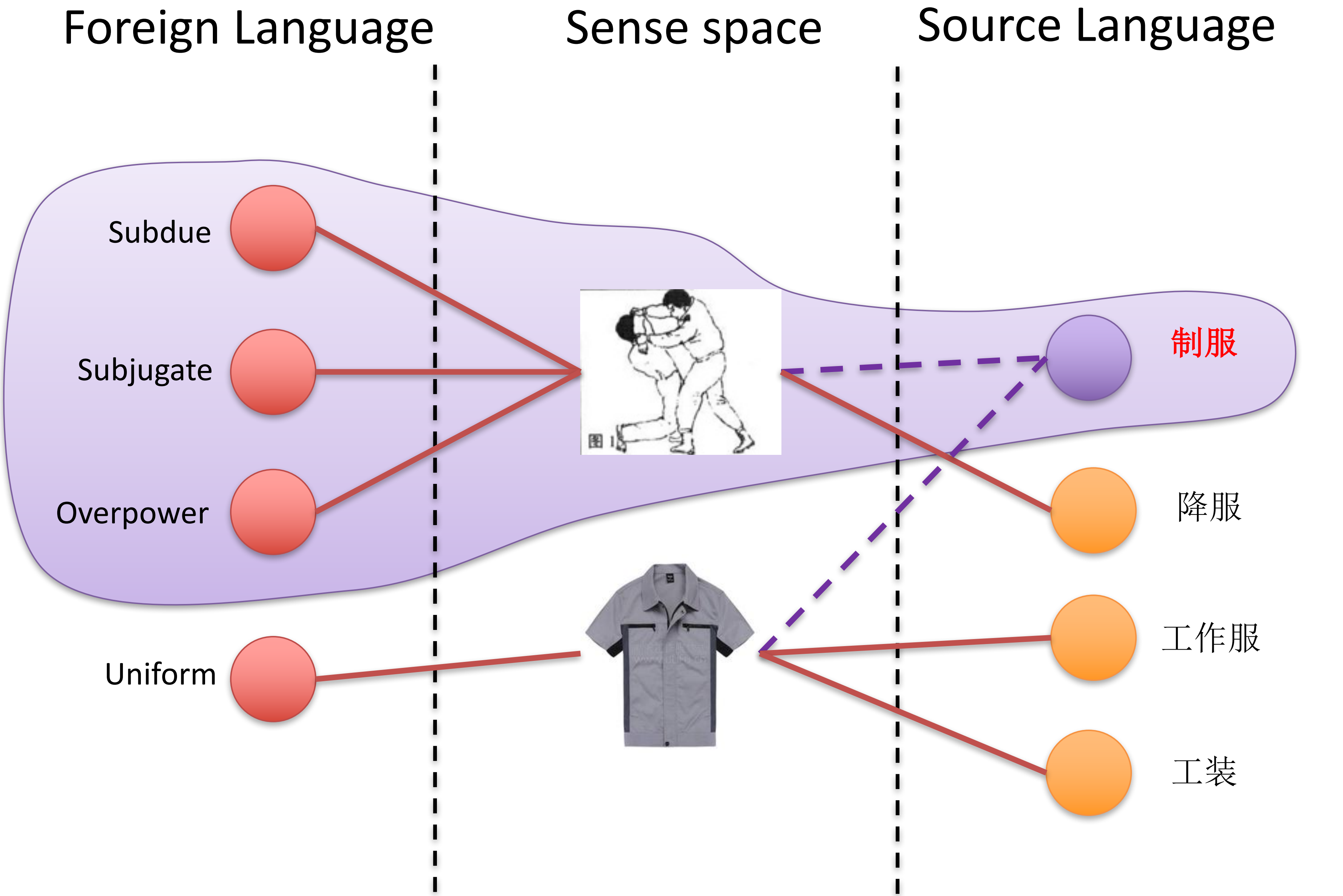
Learning Sense-specific Word Embedding by Exploiting Bilingual Resources

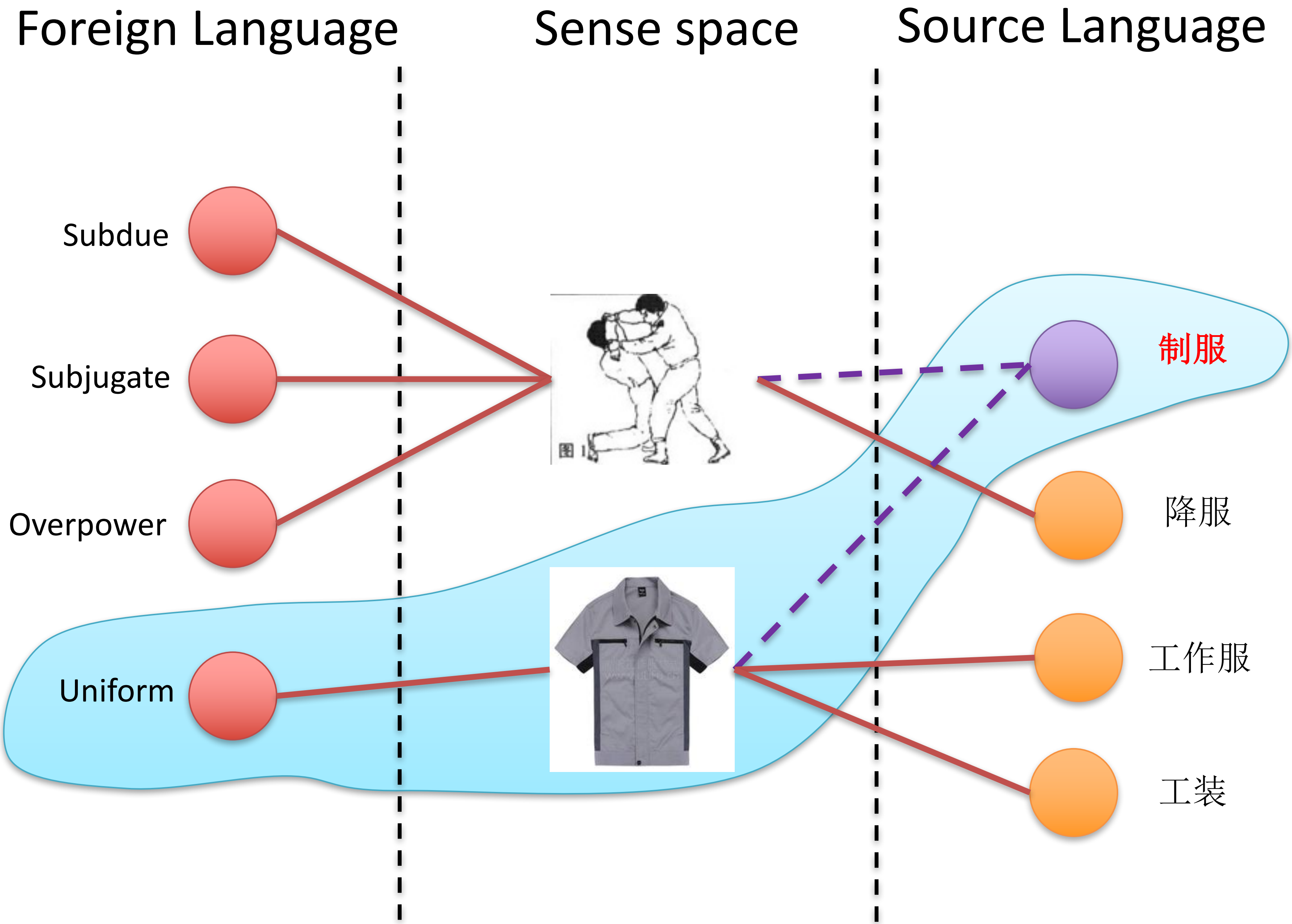
Coling 2014

Approach

- Represent words with sense-specific embeddings
  - Word sense induction using a bilingual approach
  - Train embeddings on the sense-tagged corpus

- Applications
  - Word similarity evaluation of polysemous words
  - Incorporating sense-specific embeddings to sequence labeling tasks (e.g. Named Entity Recognition)

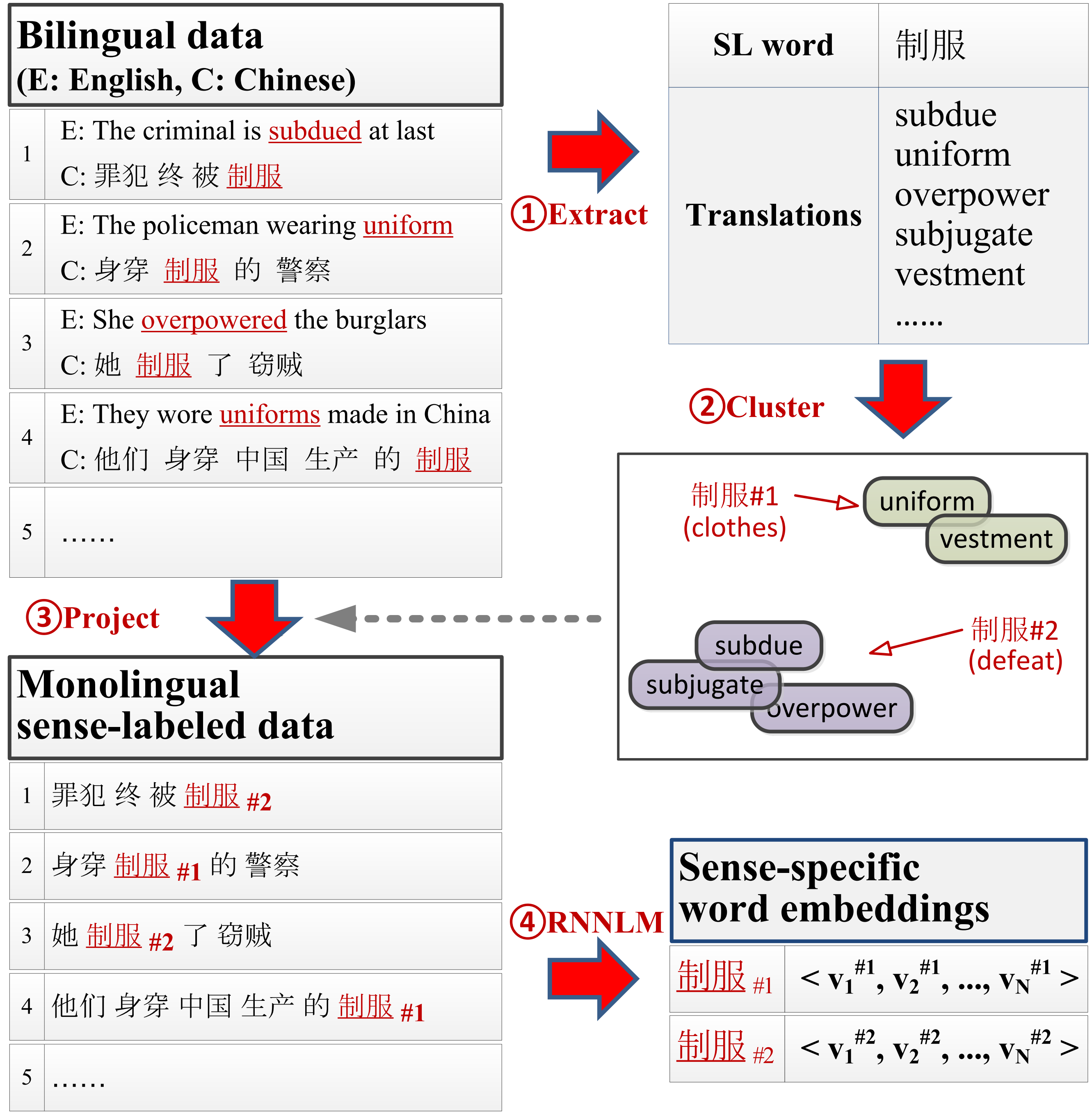

Foreign Language
Sense space
Source Language
Subdue
Subjugate
Overpower
Uniform
图 1
制服
降服
工作服
工装



Foreign Language
Sense space
Source Language
Subdue
Subjugate
Overpower
Uniform
图 1
制服
降服
工作服
工装

Method --- Learning

- Word Alignment
- Extract the translation words via bidirectional translation probability
- Cluster the translation words using their embeddings
  - Affinity Propagation (AP) clustering is used, for automatically determine the cluster number
- Cross-lingual Sense Projection
  - Tag the words in source language with its sense cluster
- Train embeddings using RNNLM (recurrent NNLM)



Bilingual data
(E: English, C: Chinese)
1 E: The criminal is subdued at last
C: 罪犯 终 被 制服
2 E: The policeman wearing uniform
C: 身穿 制服 的 警察
3 E: She overpowered the burglars
C: 她 制服 了 窃贼
4 E: They wore uniforms made in China
C: 他们 身穿 中国 生产 的 制服
5 ……
①Extract
SL word 制服
Translations subdue uniform overpower subjugate vestment ……
②Cluster
制服#1 (clothes)
uniform
vestment
subdue
subjugate
overpower
制服#2 (defeat)
③Project
Monolingual sense-labeled data
1 罪犯 终 被 制服 #2
2 身穿 制服 #1 的 警察
3 她 制服 #2 了 窃贼
4 他们 身穿 中国 生产 的 制服 #1
5 ……
④RNNLM
Sense-specific word embeddings
制服 #1 < v1#1, v2#1, ..., vN#1 >
制服 #2 < v1#2, v2#2, ..., vN#2 >

Related Work

- Huang et al. (2012), Reisinger and Mooney (2010)
  - Learning Multiple-prototype Word Embeddings
    - K embeddings for each word
  - Problem
    - Ignoring the fact that the number of senses of different words are varied.

Word Similarity Evaluation

- A manually constructed Polysemous Word Similarity Dataset
- Measurement
  - Spearman Correlation
  - Kendall Correlation
- Quantitative Evaluation

System	MaxSim		AvgSim	
	$\rho \times 100$	$\tau \times 100$	$\rho \times 100$	$\tau \times 100$
Ours	**55.4**	**40.9**	49.3	35.2
SingleEmb	42.8	30.6	42.8	30.6
Multi-prototype	40.7	29.1	38.3	27.4

Word Similarity Evaluation

- Qualitative Evaluation: K-nearest neighbors

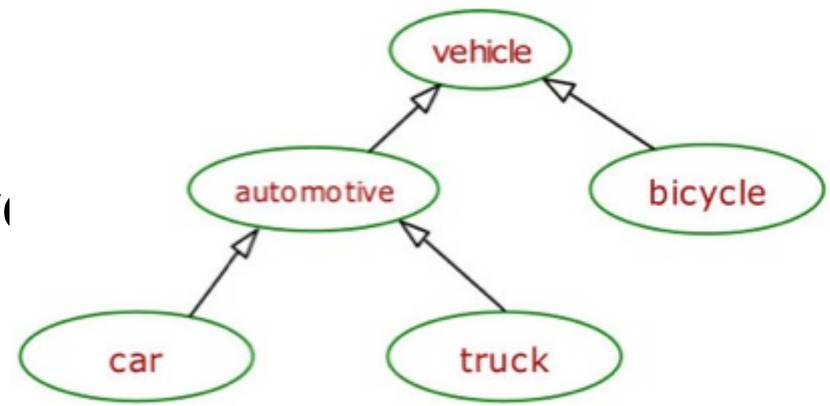
Word	Nearest Neighboors
制服$_{uniform}$	穿着$_{dress}$, 警服$_{policeman\ uniform}$
制服$_{subdue}$	打败$_{defeat}$, 击败$_{beat}$, 征服$_{conquer}$
花$_{spend}$	花费$_{cost}$, 节省$_{save}$, 剩下$_{rest}$
花$_{flower}$	菜$_{greens}$, 叶$_{leaf}$, 果实$_{fruit}$
法$_{law}$	法令$_{ordinance}$, 法案$_{bill}$, 法规$_{rule}$
法$_{method}$	概念$_{concept}$, 方案$_{scheme}$
法$_{French}$	德$_{Germany}$, 俄$_{Russia}$, 英$_{Britain}$
领导$_{lead}$	监督$_{supervise}$, 决策$_{decision}$
领导$_{leader}$	主管$_{chief}$, 上司$_{boss}$

Learning Semantic Hierarchies via Word Embeddings

ACL 2014

Semantic Hierarchies

- Learning Semantic Hierarchies via Word Embeddings
  - car → automotive
    - hypernym: automotiv
    - hyponym: car



  - manually-built semantic hierarchies
    - WordNet
    - HowNet
    - CilinE (Tongyi Cilin - Extended version)

Previous Work

- Pattern-based method
  - e.g. “such NP1 as NP2”
    - Hearst (1992) ；Snow et al. (2005)

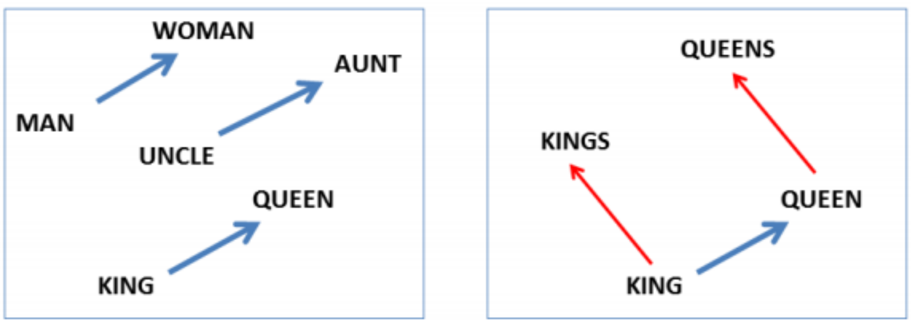
Pattern	Translation
w 是[一个|一种] h	w is a [a kind of] h
w [、] 等 h	w[,] and other h
h [，] 叫[做] w	h[,] called w
h [，] [像]如 w	h[,] such as w
h [，] 特别是 w	h[,] especially w

- Methods based on web mining
  - assuming that the hypernyms of an entity co-occur with it frequently
  - extracting hypernym candidates from multiple sources and learning to rank
    - Fu et al. (2013)

Word Embeddings

- Learning Semantic Hierarchies via Word Embeddings

$$v(\mathtt{king}) - v(\mathtt{queen}) \approx v(\mathtt{man}) - v(\mathtt{woman})$$



Mikolov et al. (2013a)

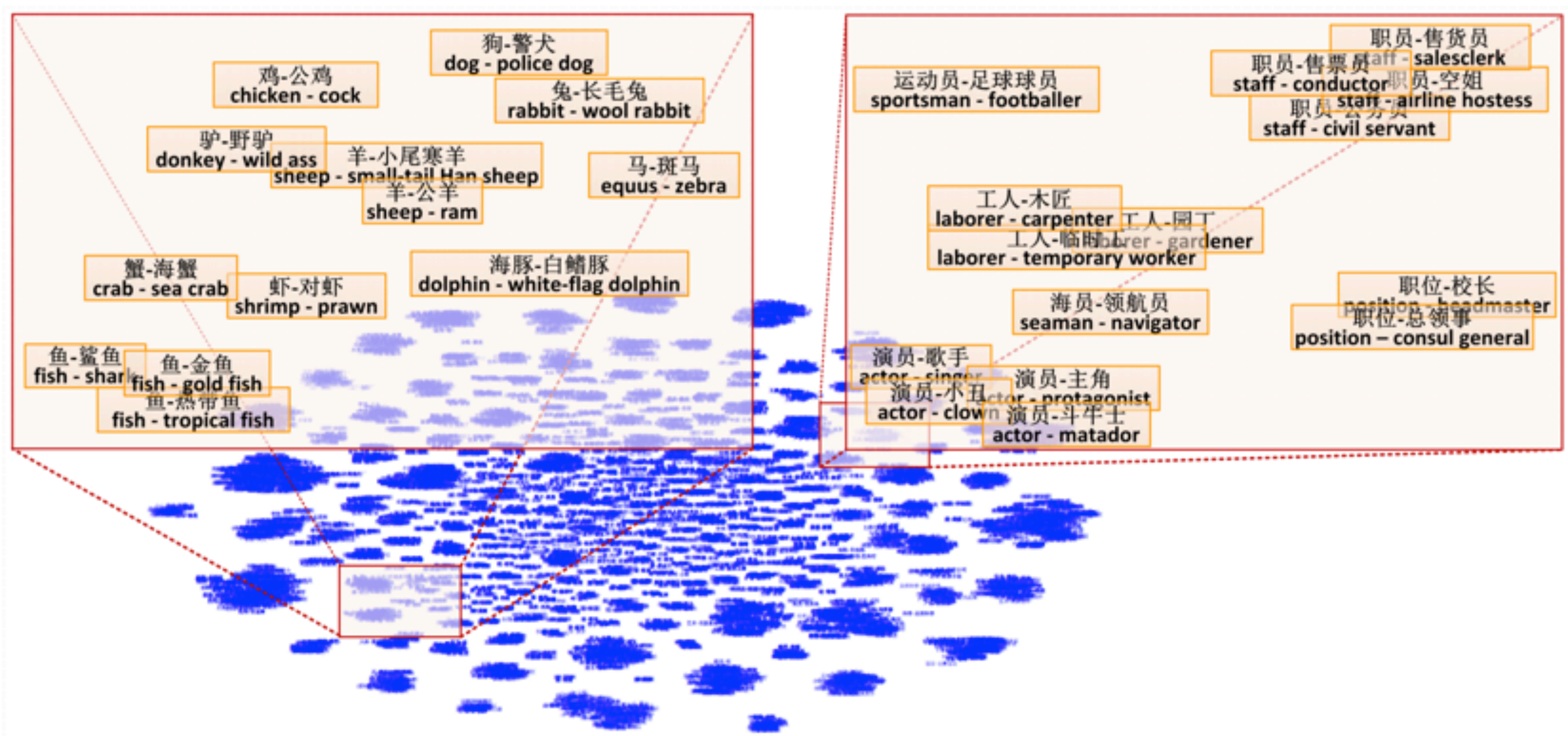
Motivation

- Does the embedding offset work well in hypernym–hyponym relations ?

No.	Examples
1	$v(虾) - v(对虾) \approx v(鱼) - v(金鱼)$ $v(\text{shrimp}) - v(\text{prawn}) \approx v(\text{fish}) - v(\text{gold fish})$
2	$v(工人) - v(木匠) \approx v(演员) - v(小丑)$ $v(\text{laborer}) - v(\text{carpenter}) \approx v(\text{actor}) - v(\text{clown})$
3	$v(工人) - v(木匠) \not\approx v(鱼) - v(金鱼)$ $v(\text{laborer}) - v(\text{carpenter}) \not\approx v(\text{fish}) - v(\text{gold fish})$

Motivation

- Clusters of the vector offsets in training data

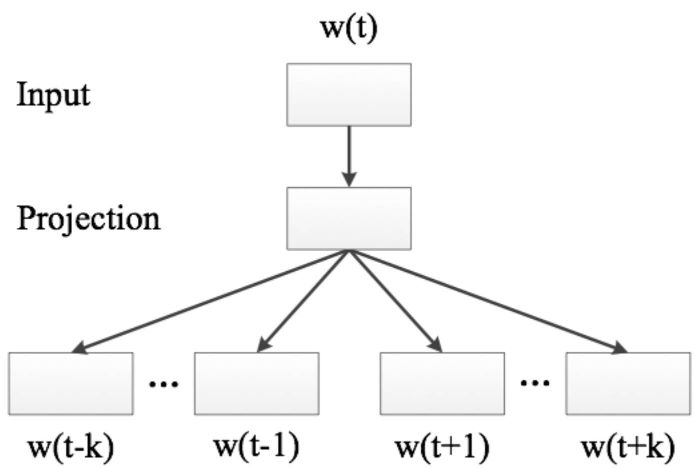


Method

- Word Embedding Training
- Projection Learning
- is-a Relation Identification

Word Embedding Training

- Skip-gram



Mikolov et al. (2013b)

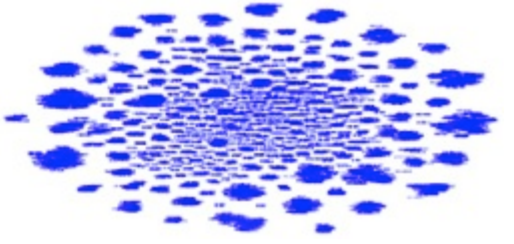
Projection Learning

- A uniform Linear Projection
  - Given a word $\boldsymbol{x}$ and its hypernym $\boldsymbol{y}$, there exists a matrix $\boldsymbol{\Phi}$ so that $\boldsymbol{y} = \boldsymbol{\Phi x}$.

$$\Phi^* = \arg\min_{\Phi} \frac{1}{N} \sum_{(x,y)} \| \Phi x - y \|^2$$

Projection Learning

- Piecewise Linear Projections
  - clustering y - x

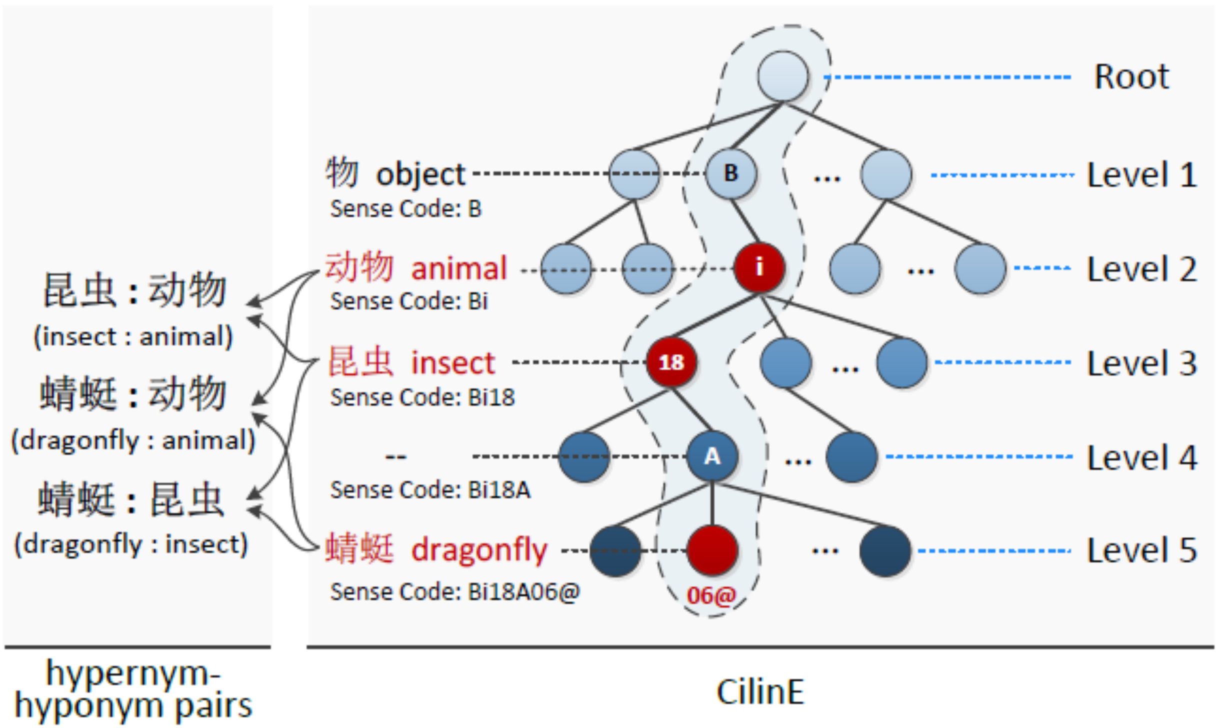


  - learning a separate projection for each cluster

$$\Phi_k^* = \arg\min_{\Phi_k} \frac{1}{N_k} \sum_{(x,y)\in C_k} \| \Phi_k x - y \|^2$$

Projection Learning

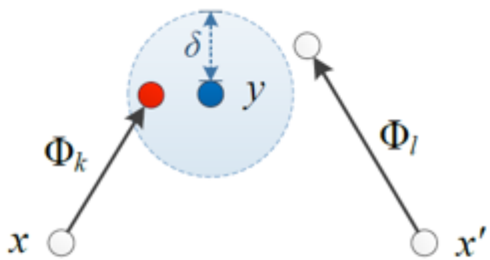
- Training data



is-a Relation Identification

- Given two words x and y
  - If y is determined as a hypernym of x, either of the two conditions must be satisfied

Condition 1:



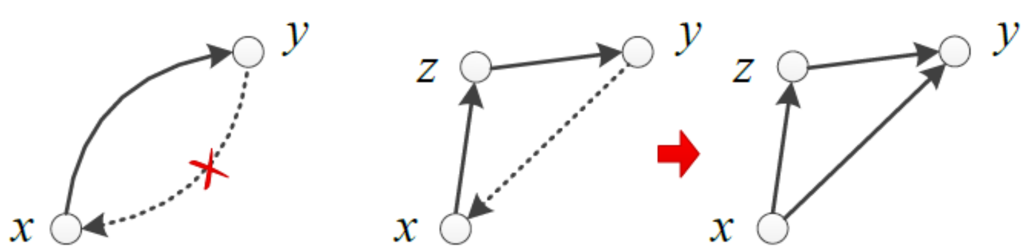
$$d(\Phi_k x, y) = \| \Phi_k x - y \|^2 < \delta$$

Condition 2:

$$x \xrightarrow{H} z \text{ and } z \xrightarrow{H} y$$

is-a Relation Identification

- Hierarchy Condition (DAG)
  - $\forall x, y \in L : x \xrightarrow{H} y \Rightarrow \neg(y \xrightarrow{H} x)$
  - $\forall x, y, z \in L : (x \xrightarrow{H} z \wedge z \xrightarrow{H} y) \Rightarrow x \xrightarrow{H} y$



(a) (b)

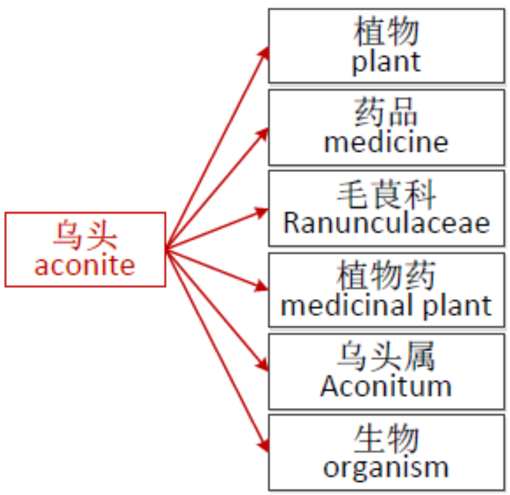
Experimental Data

- Word embedding training
  - corpus from Baidubaike
    - ~30 million sentences (~780 million words)

- Projection learning
  - CilinE
    - 15,247 is-a pairs

Experimental Data

- For evaluation



Relation	# of word pairs	
	Dev.	Test
hypernym–hyponym	312	1,079
hyponym–hypernym*	312	1,079
unrelated	1,044	3,250
Total	1,668	5,408

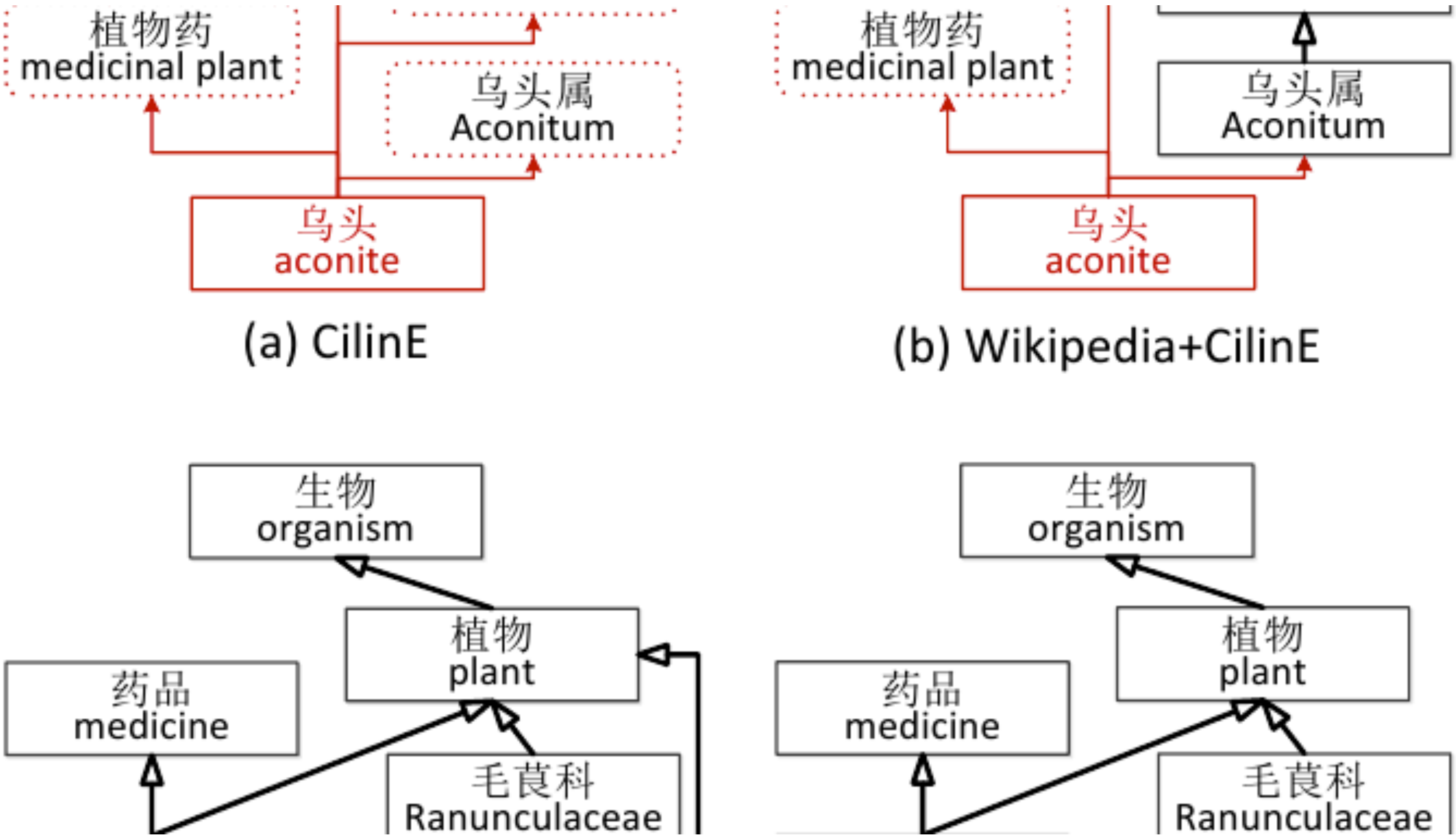
Fu et al. (2013)

Results and Analysis

- Comparison with existing methods

	P(%)	R(%)	F(%)	
M_{CilinE}	98.21	50.88	67.03	
$\mathrm{M}_{Wiki+CilinE}$	92.41	60.61	73.20	Suchanek et al. (2008)
$\mathrm{M}_{Pattern}$	97.47	21.41	35.11	Hearst (1992)
M_{Snow}	60.88	25.67	36.11	Snow et al. (2005)
$\mathrm{M}_{balApinc}$	54.96	53.38	54.16	Kotlerman et al. (2010)
M_{invCL}	49.63	62.84	55.46	Lenci and Benotto (2012)
M_{Fu}	71.64	52.92	60.87	Fu et al. (2013)
M_{offset}	59.26	63.19	61.16	
M_{Emb}	80.54	67.99	**73.74**	
$\mathrm{M}_{Emb+CilinE}$	80.59	72.42	76.29	
$\mathrm{M}_{Emb+Wiki+CilinE}$	79.78	80.81	**80.29**	

Results and Analysis

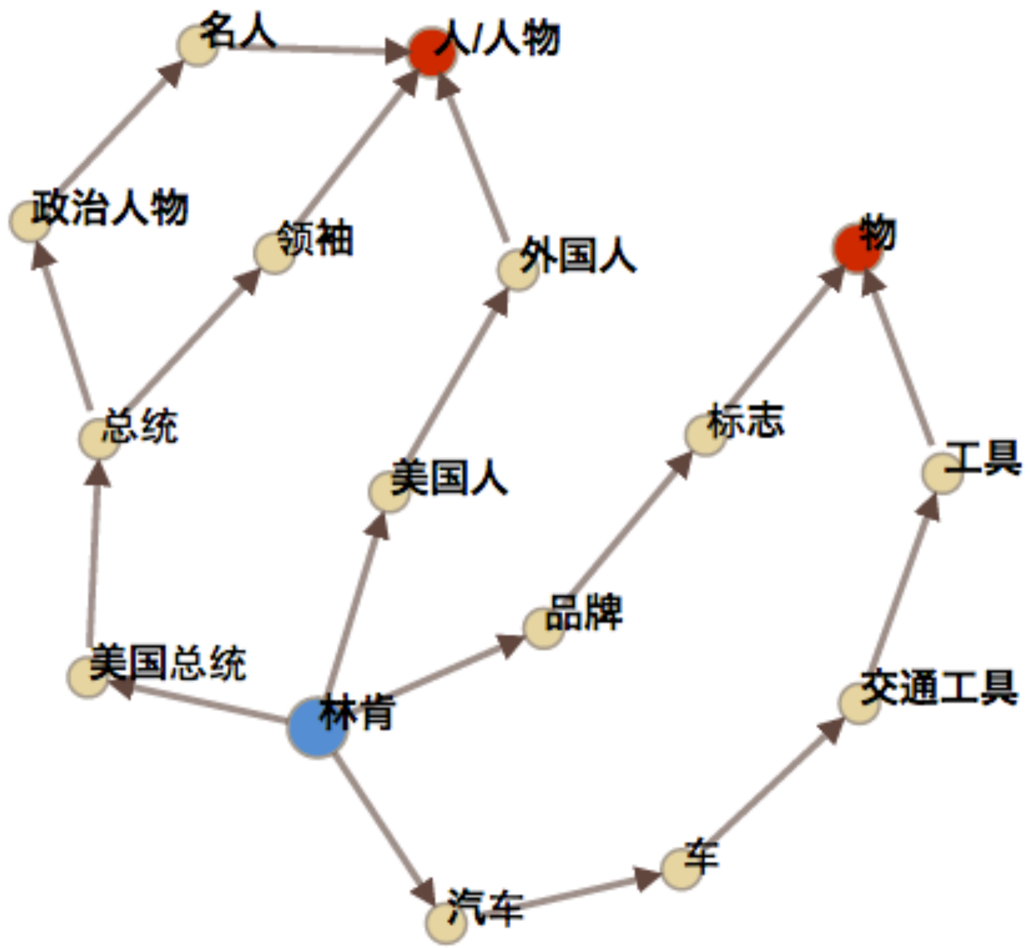


(a) CilinE

(b) Wikipedia+CilinE

Demo

- http://www.bigcilin.com



Revisiting Semi-supervised Learning with Embedding Features

EMNLP 2014

Motivation

- Learning generalized word representation is promising way for handling data sparsity.
  - Caused by the high-dimensional lexical features in NLP
  - Brown clusters
    - Liang, 2005 (Master Thesis)
    - Koo et al., 2008 (ACL)
    - Owoputi et al., 2013 (NAACL)

Motivation

- Learning generalized word representation is promising way for handling data sparsity.
  - Caused by the high-dimensional lexical features in NLP
  - Distributional word representations
    - Huang et al. 2009 (ACL), 2013 (CL)
      - Graphical models (HMM, MRF, LVM)

Motivation

- Learning generalized word representation is promising way for handling data sparsity.
  - Caused by the high-dimensional lexical features in NLP
  - Word embeddings (our focus)
    - Turian et al. 2010 (ACL)
    - Yu et al. 2013 (NAACL)

Problem

- Are the continuous embedding features fit for the generalized linear models (e.g., CRF) which are most widely adopted in NLP?
- How can the generalized linear models better utilize the embedding features?

Main related studies

- Turian et al. 2010
  – The embedding features brought significant less improvement than brown clusters
- Wang et al. 2012
  – Non-linear models benefit more with low-dimensional continuous feature.
  – Linear models are more effective in high-dimensional discrete space.
- Yu et al. 2013
  – Introduce the compound cluster feature.

Semi-supervised Learning with Word Embedding

- Data sparsity
  - Lack of the labeled training data
  - Natural language words follows Zipf law
- Semi-supervised learning
  - Effectively take the advantage of large amount of the unlabeled data.
  - The representation of words are effective in linking different words.

Approach

1. Direct usage of the dense continuous embedding feature

2. Binarized embedding feature

3. Compound cluster feature

4. Distributional prototype feature

Binarized embedding

- Conversion function

$$M_{ij} = \phi(C_{ij}) = \begin{cases} \text{+U}, & if\ C_{ij} \geq mean(C_{i+}) \\ \text{–B}, & if\ C_{ij} \leq mean(C_{i-}) \\ 0, & otherwise \end{cases}$$

- Words that have strong opinions (positive or negative) are considered.
- Omit the values that are close to zero

Compound cluster feature

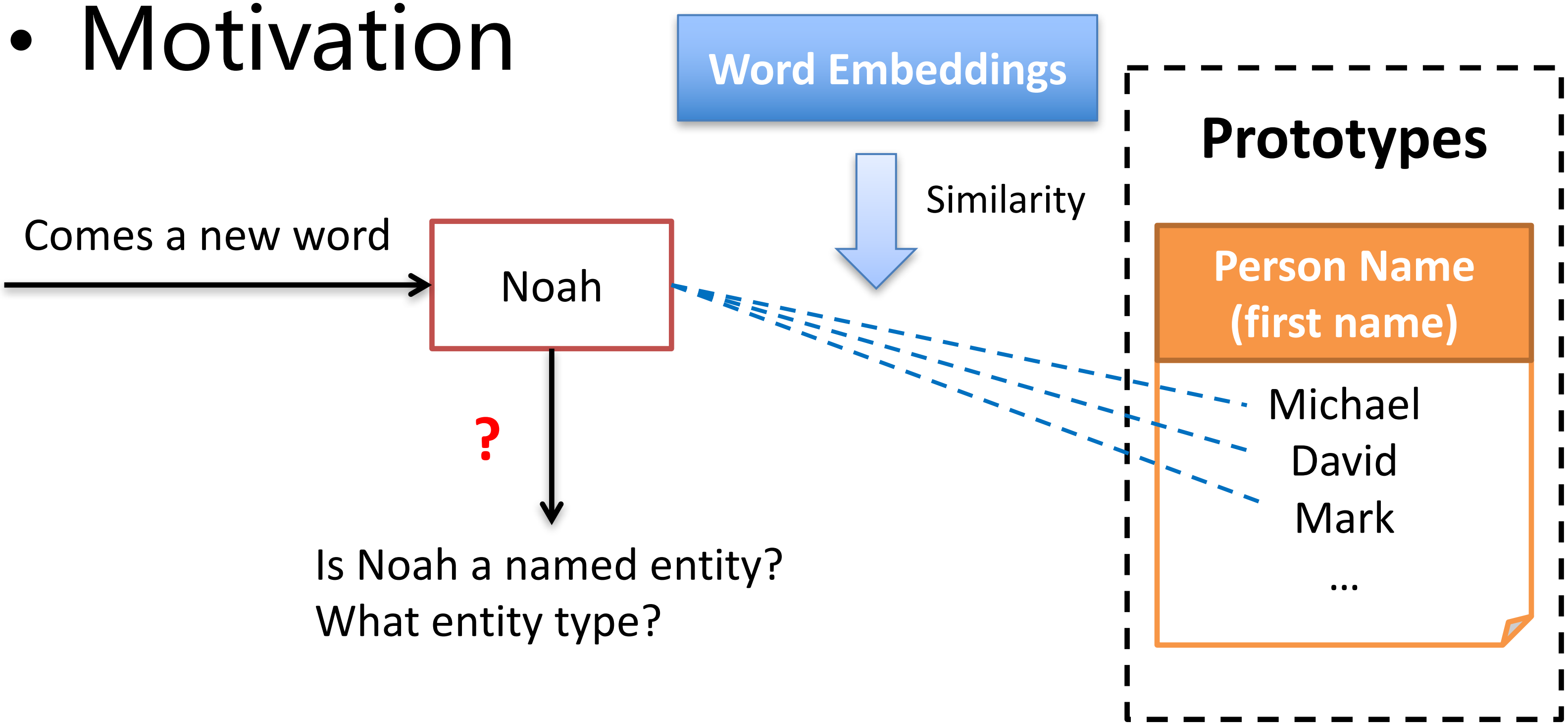
- Word embeddings are clustered to form discrete cluster features.
  - K-means clustering algorithm

- Cluster features can be combined to form compound features.

- Different k indicates different granularity, which are useful, so we combine different k.

Prototype-driven

- Motivation



We could add such a **link feature** indicating that Noah is potentially an NE of PER.

Question: How to obtain the Prototypes?

Normalized PMI

- Standard PMI:

$$\lambda(label, word) = \ln \frac{p(label, word)}{p(label)p(word)}$$

- Normalized (smoothing):

$$\lambda_n(label, word) = \frac{\lambda(label, word)}{-\ln p(label, word)}$$

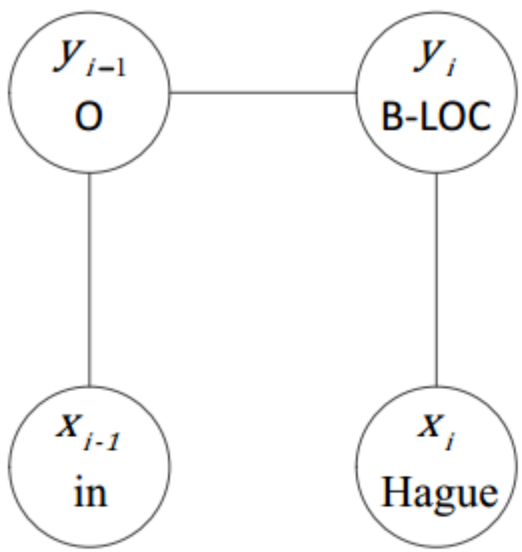
Prototypes extracted from CoNLL03 NER dataset

- Example of the top-4 prototypes for each target NER label

NE Type	Prototypes
B-PER	Mark, Michael, David, Paul
I-PER	Akram, Ahmed, Khan, Younis
B-ORG	Reuters, U.N., Ajax, PSV
I-ORG	Newsroom, Inc, Corp, Party
B-LOC	U.S., Germany, Britain, Australia
I-LOC	States, Republic, Africa, Lanka
B-MISC	Russian, German, French, British
I-MISC	Cup, Open, League, OPEN
O	., ,, the, to

How to add the feature?

- Sequence Labeling (Take NER as an example)



$$f(y_{i-1}, y_i) = \text{O} \wedge \text{B-LOC}$$

$$f(x_i, y_i) = \left\{ \begin{array}{l} \text{word} = \text{Hague} \\ \text{pos} = \text{NNP} \\ \text{proto} = \text{Britain} \\ \text{proto} = \text{England} \\ \ldots \end{array} \right\} \wedge \text{B-LOC}$$

Experiment results

Setting	F1
Baseline	83.43
+DenseEmb†	86.21
+BinarizedEmb	86.75
+ClusterEmb	86.90
+DistPrototype	**87.44**
+BinarizedEmb+ClusterEmb	87.56
+BinarizedEmb+DistPrototype	87.46
+ClusterEmb+DistPrototype	**88.11**
+Brown	87.49
+Brown+DistPrototype	88.04
+Brown+ClusterEmb+DistPrototype	**88.58**
Finkel et al. (2005)	86.86
Krishnan and Manning (2006)	87.24
Ando and Zhang (2005)	89.31
Collobert et al. (2011)	88.67

Results

- **BinarizedEmb** is already a big help! (Surprising!)
- Combination offers additive improvements
- **DistPrototype** features performs the best
  - Task-specific features