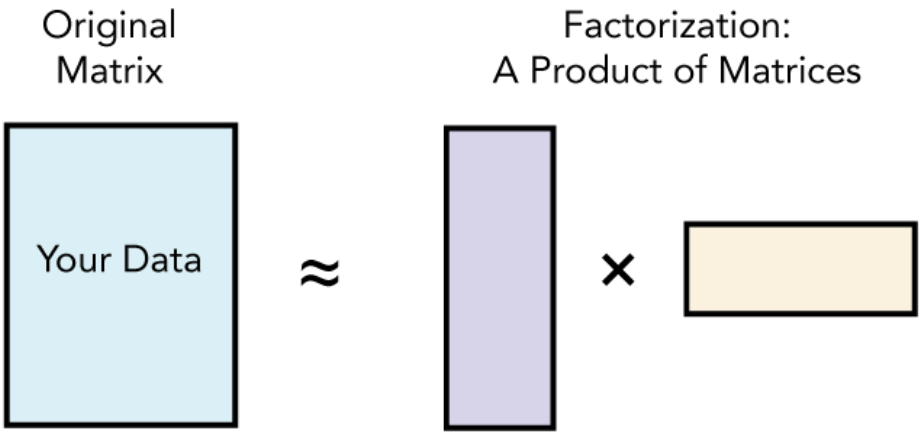
Matrix and Tensor Factorization Methods for Natural Language Processing

Guillaume Bouchard, Jason Naradowsky, Sebastian Riedel, Tim Rocktäschel, Andreas Vlachos

Computer Science Department
University College London

Matrix Factorization in a Nutshell



Matrix Factorization You Know

Matrix Factorization You Know

Matrix Factorization You Know

NLP and Matrix Factorization

We already use it!

- topic modelling: probabilistic LSA, LDA...
- word embedding learning: word2vec, GLOVE...
- knowledge base population

Factorization can improve structured prediction models:

- Smoothing of model parameters

Matrix factorization supports logical reasoning:

- Recent work unites factorization and first-order logic rules

Motivation

Matrix and tensor factorization

- encompass a variety of learning problems
- well-studied in ML/stats literature
- state-of-the-art performance in many tasks

This tutorial

Part 1

- Seeing the matrix
- Matrix factorization basics
- Non-negative matrix factorization
- Binary matrix factorization
- Inclusion of prior knowledge

This tutorial

Part 2

- Tensor factorization basics
- Factorization with a predictive model
- Collective matrix factorization
- Convexification

Outcomes

- Recognizing potential applications
- Understanding of the mathematical concepts
- Clarification of the connections among models
- Familiarization with existing NLP applications

Seeing the Matrix

Binary Classification

	label	features
TRAIN	**1**	**f1, f3, f4, f6**
	1	**f3, f6**
	0	**f1, f2, f5**
	0	**f1, f2**
TEST	???	f1, f3, f4
	???	f2

Matrix Completion

	task1	task2	feat1	feat2	feat3	feat4	feat5	feat6
TRAIN	1	1	1	0	1	1	0	1
	0	1	0	0	1	0	0	1
	1	0	1	1	0	0	1	0
	0	0	1	1	0	0	0	0
TEST	???	???	1	0	1	1	0	0
	???	???	0	1	0	0	0	0
UNLAB			1	0	0	1	0	0
			0	1	0	0	1	0

Multi-task learning (transductive) learning

Notation

- Scalars: lower-case letters v, m
- Vectors: boldface lower-case letters $\mathbf{v}$

$$\begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix}$$

- Matrices: boldface capital letters $\mathbf{M}$

$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,J} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ m_{I,1} & m_{I,2} & \cdots & m_{I,J} \end{pmatrix}$$

- ith row of $\mathbf{M}$: $\mathbf{m}_{i:}$
- jth column of $\mathbf{M}$:$\mathbf{m}_{:j}$

Vector Operations

- the **inner product** between vectors $\mathbf{u} \in \mathfrak{R}^L$ and $\mathbf{v} \in \mathfrak{R}^L$

$$\langle \mathbf{u}, \mathbf{v} \rangle := \sum_{\ell=1}^{L} u_\ell v_\ell$$

- the **outer product** between vectors $\mathbf{u} \in \mathfrak{R}^N$ and $\mathbf{v} \in \mathfrak{R}^M$

$$\mathbf{u} \otimes \mathbf{v} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} \otimes \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix} := \begin{pmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_M \\ u_2 v_1 & u_2 v_2 & \cdots & u_2 v_M \\ \vdots & \vdots & \ddots & \vdots \\ u_N v_1 & u_N v_2 & \cdots & u_N v_M \end{pmatrix}_{N \times M}$$

Matrix Operations

- the **inner product** for matrices $\mathbf{U} \in \mathfrak{R}^{N \times L}$ and $\mathbf{V} \in \mathfrak{R}^{L \times M}$

$$\mathbf{UV} := \begin{pmatrix} \sum_\ell u_{1,\ell} v_{\ell,1} & \sum_\ell u_{1,\ell} v_{\ell,2} & \cdots & \sum_\ell u_{1,\ell} v_{\ell,M} \\ \sum_\ell u_{2,\ell} v_{\ell,1} & \sum_\ell u_{2,\ell} v_{\ell,2} & \cdots & \sum_\ell u_{2,\ell} v_{\ell,M} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_\ell u_{N,\ell} v_{\ell,1} & \sum_\ell u_{N,\ell} v_{\ell,2} & \cdots & \sum_\ell u_{N,\ell} v_{\ell,M} \end{pmatrix}_{N,M}$$

Matrix Operations

- the **outer product** for matrices $\mathbf{U} \in \mathfrak{R}^{N \times L}$ and $\mathbf{V} \in \mathfrak{R}^{M \times L}$

$$\mathbf{U} \otimes \mathbf{V} := \sum_{\ell=1}^{L} \mathbf{u}_{:\ell} \otimes \mathbf{v}_{:\ell}$$

$$= \mathbf{u}_{:1} \otimes \mathbf{v}_{:1} + \mathbf{u}_{:2} \otimes \mathbf{v}_{:2} + \cdots \mathbf{u}_{:L} \otimes \mathbf{v}_{:L}$$

Matrix Operations

- the element-wise **Hadamard product** for $\mathbf{U}, \mathbf{V} \in \mathfrak{R}^{N \times L}$

$$\mathbf{U} \circ \mathbf{V} := \begin{pmatrix} u_{1,1}v_{1,1} & u_{1,2}v_{1,2} & \cdots & u_{1,L}v_{1,L} \\ u_{2,1}v_{2,1} & u_{2,2}v_{2,2} & \cdots & u_{2,L}v_{2,L} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N,1}v_{N,1} & u_{N,2}v_{N,2} & \cdots & u_{N,L}v_{N,L} \end{pmatrix}_{N,L}$$

Norms

Commonly used to define:

- loss functions (how far we are from our objective)
- regularizers (how large we allow our parameters to grow)

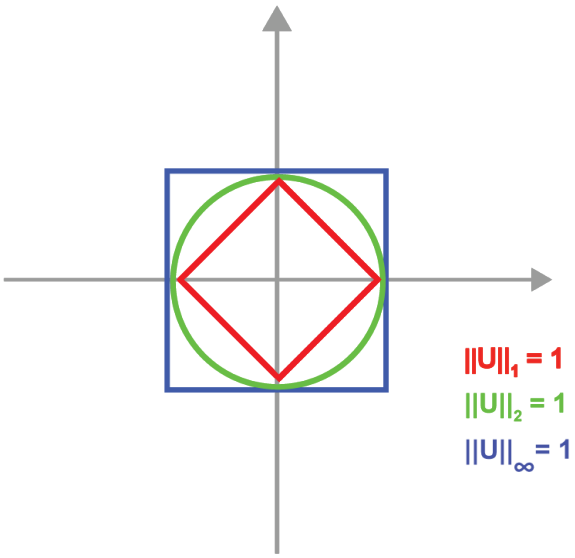
element-wise p-norms:

$$\|\mathbf{U}\|_p := \left(\sum_{m=1}^{M} \sum_{n=1}^{N} |u_{n,m}|^p \right)^{1/p}$$

Frobenius norm $\|\mathbf{U}\|_F := \|\mathbf{U}\|_2$

Norm example

$$\mathbf{U} = \begin{pmatrix} u_1 & u_2 \end{pmatrix}$$



Matrix rank

The maximum number of linearly independent columns/rows

For matrix $\mathbf{U} \in \Re^{N \times M}$:

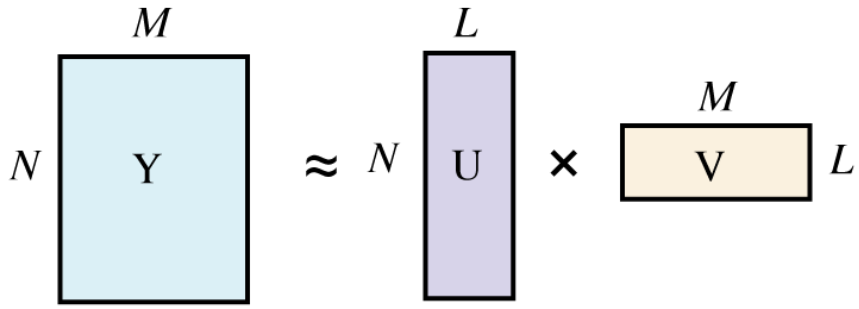
- if $N = M = 0$ then $rank(\mathbf{U}) = 0$
- else $max(rank(\mathbf{U})) = min(N, M)$ (**full rank**)

$$rank \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \\ 3 & 4 & 7 \\ 4 & 5 & 9 \end{pmatrix} = ??$$

hint: $\mathbf{u}_{:1} + \mathbf{u}_{:2} = \mathbf{u}_{:3}$

Vanilla Matrix Factorization

Matrix Completion via Low-Rank Factorization



Given $\mathbf{Y} \in \Re^{N \times M}$

find $\mathbf{U} \in \Re^{N \times L}$ and $\mathbf{V} \in \Re^{M \times L}$

so that $\mathbf{Y} \approx \mathbf{U}\mathbf{V}^T$

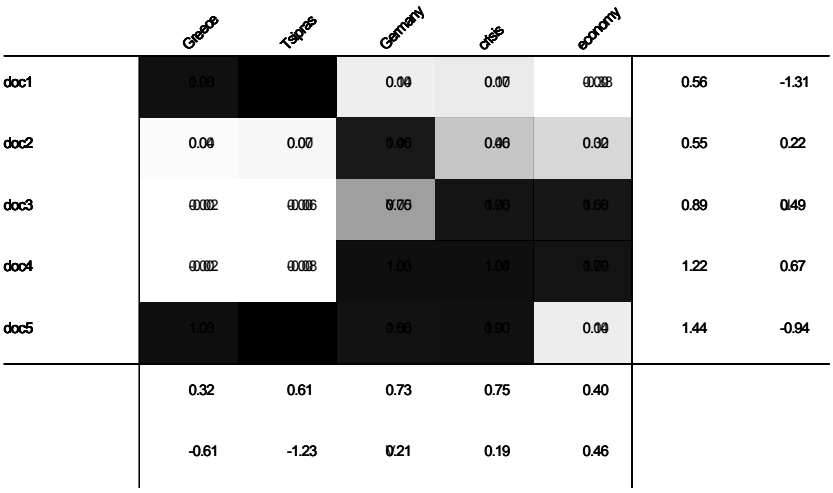
low rank assumption: $rank(\mathbf{Y}) = L << M, N$

Why Low-Rank

- low-rank assumption usually does not hold
- reconstruction unlikely to be perfect
- if full-rank then perfect reconstruction is trivial: $\mathbf{Y} = \mathbf{Y}\mathbf{I}$

Key insight:

*original matrix exhibits **redundancy** and **noise**, low-rank reconstruction exploits the former to remove the latter*



Document-Term matrix $\mathbf{Y}$

Document-Topic $\mathbf{U}$, Topic $\mathbf{S}$, Topic-Term $\mathbf{V}$

Why Low-Rank

Because we want a model on matrices with:

1. Invariance by permutation of rows and columns

   The model depends only on the singular values

2. A measure to control complexity

   A (pseudo-)norm on the singular values

3. Fast to compute and memory efficient

   A sparsity-inducing norm

Three ways to Factorize

Singular Value Decomposition (SVD)

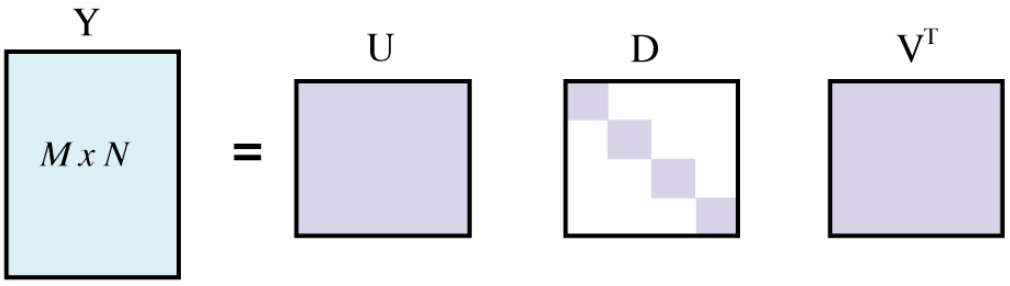
Given $\mathbf{Y} \in \Re^{M \times N}$ there exists

$$\mathbf{Y} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

where $\mathbf{U} \in \Re^{M \times M}$, $\mathbf{V} \in \Re^{N \times N}$ are orthogonal:

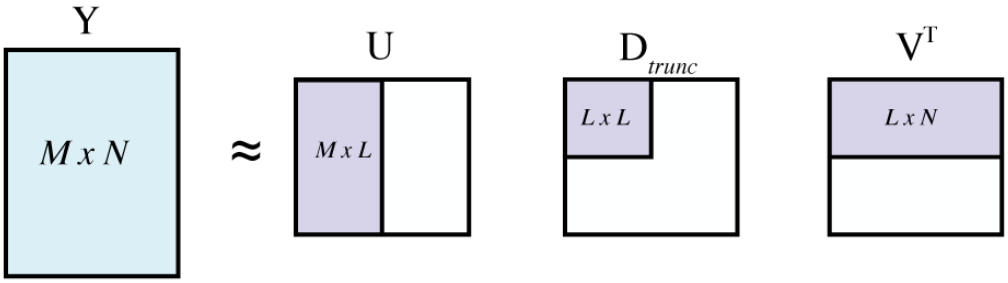
$$\mathbf{U}\mathbf{U}^T = \mathbf{V}\mathbf{V}^T = I$$

and $\mathbf{D} \in \Re^{M \times N} \geq 0$ is diagonal



Truncated SVD

If we truncate $\mathbf{D}$ to its L largest values



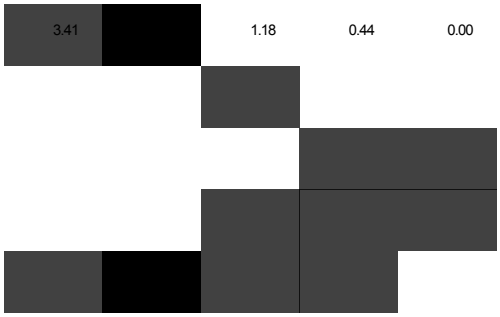
then $\hat{\mathbf{Y}} = \mathbf{U}\mathbf{D}_{\mathbf{trunc}}\mathbf{V} = (\mathbf{U}\sqrt{\mathbf{D}_{\mathbf{trunc}}})(\sqrt{\mathbf{D}_{\mathbf{trunc}}}\mathbf{V}^T)$

is the rank-L minimizer of $\|\mathbf{Y} - \hat{\mathbf{Y}}\|_F^2$

Eckart-Young theorem

SVD demo

```
val (u,s,v) = breezeSVD(M)
val eigen = numbers(rowVector(s))
val (ea,ev) = toEmbeddings(u,s,v,4)
val m1 = opacity(matrix(dots(ea,ev)),0.2)
render(Seq(opacity(matrix(M),0.2).numbers(rowVector(s)).m1)
```



MF with Stochastic Gradient Descent

Objective: $\gamma(U, V) := \sum_{(i,j)\in\Omega}(y_{ij} - \langle \mathbf{u}_i, \mathbf{v}_j \rangle)^2$

$(i, j) \notin \Omega$: **missing values**

Algorithm:

1. Pick $(i, j) \in \Omega$ uniformly at random
2. Gradient steps

$$\mathbf{u}_i \leftarrow \mathbf{u}_i - \eta(\langle \mathbf{u}_i, \mathbf{v}_j \rangle - y_{ij})\mathbf{v}_j$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j - \eta(\langle \mathbf{u}_i, \mathbf{v}_j \rangle - y_{ij})\mathbf{u}_i$$

where η is the gradient step size (can be adaptive)

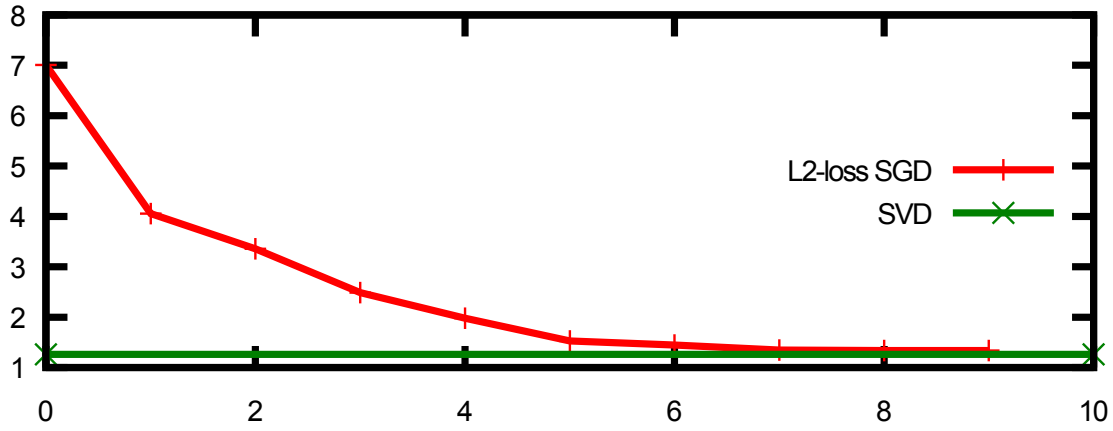
SGD code

```
def optimizeL2(M: Mat, K: Int, iters: Int, alpha:Double = 0.1,
               initScale:Double = 1.0): (Seq[Vect], Seq[Vect]) = {
  val rand = new scala.util.Random(0)
  val AV = initialAV(K, M.dim1, M.dim2, initScale)
  val A = AV._1; val V = AV._2
  def update(i: Int, j: Int) ={
    val a = A(i).copy
    val v = V(j).copy
    val y = a dot v
    A(i) += v * alpha * (M(i, j) - y)
    V(j) += a * alpha * (M(i, j) - y)
  }
  for (i <- Range(0,iters).toList) {
    update(rand.nextInt(M.dim1),
           rand.nextInt(M.dim2))
  }
  (A, V)
}
```

?

SGD converges to SVD

```
val K = 2 //rank
val x = Range(0,10).map(_.toDouble)
val models = for (i <- x.map(_.toInt)) yield optimizeL2(M, K, i * 10
0.1)
val ySGD = models.map(m => l2Loss(M,dots(m._1,m._2)))
val (u,s,v) = breezeSVD(M)
val (ea,ev) = toEmbeddings(u,s,v,K)
val ySVD = l2Loss(M,dots(ea,ev))
val c = plot(x,ySGD,"L2-loss SGD",Seq(0.0,10.0),Seq(ySVD,ySV
c.size = (500.0 -> 200.0)
c//ml.wolfe.ui.D3Plotter.lineplot(c)
```



Alternating Least Squares

Objective: $\gamma(U, V) := \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2$

block-coordinate descent:

$$\arg\min_{\mathbf{u}_{i:}} \gamma(U, V) = \arg\min_{\mathbf{u}_{i:}} \|\mathbf{y}_{i:} - \mathbf{V}\mathbf{u}_{i:}\|_F^2$$

$$\arg\min_{\mathbf{v}_{j:}} \gamma(U, V) = \arg\min_{\mathbf{v}_{j:}} \|\mathbf{y}_{:j} - \mathbf{U}\mathbf{v}_{j:}\|_F^2$$

Least square problems ⇛ Unique solutions!

$$\mathbf{u}_i \leftarrow (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\mathbf{y}_{i:}$$

$$\mathbf{v}_j \leftarrow (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{y}_{:j}$$

Each step can be parallelized efficiently

Regularization

Objective:

$$\gamma(U, V) := \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2 + \lambda_1 Reg1(\mathbf{U}) + \lambda_2 Reg2(\mathbf{V})$$

Common choices for $Reg1$ and $Reg2$:

- sparsity-inducing 1-norm: $\|\mathbf{v_j}\|_1$
- overfitting-avoiding squared 2-norm: $\|\mathbf{u}_i\|_F^2$

1-norm not continuously differentiable
Optimization harder but still possible

Regularized Latent Semantic Indexing

document-term matrix $\mathbf{Y} \approx \mathbf{U}\mathbf{V}^T$

document-topic matrix $\mathbf{U}$, term-topic matrix $\mathbf{V}$

$\mathbf{U}$	$\mathbf{V}$	**Readability**	**Compactness**	**Retrieval**
1-norm	2-norm	✓	✓	✓
2-norm	1-norm	✓	✗	✗
1-norm	1-norm	✓	✓	✗
2-norm	2-norm	✗	✗	✗

Wang et al. (2011)

Learning word embeddings

	Greece	Tsipras	Germany	crisis	economy		
context1	1.00	2.00	0.00	0.00	0.00	0.56	-1.31
context2	0.00	0.00	1.00	0.00	0.00	0.55	0.22
context3	0.00	0.00	0.00	1.00	1.00	0.89	0.49
context4	0.00	0.00	1.00	1.00	1.00	1.22	0.67
context5	1.00	2.00	1.00	1.00	0.00	1.44	-0.94
	0.32	0.61	0.73	0.75	0.40		
	-0.61	-1.23	0.21	0.19	0.46		

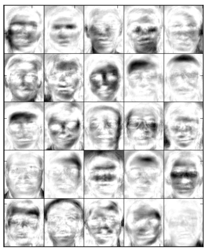
- implicit MF: SkipGram (Mikolov et al. 2013)
- explicit MF: GloVe (Pennington et al. 2014) and S-PPMI (Levy and Goldberg, 2014)

Non-Negative Matrix Factorization

Non-negativity

Low-rank factors are commonly interpreted as:

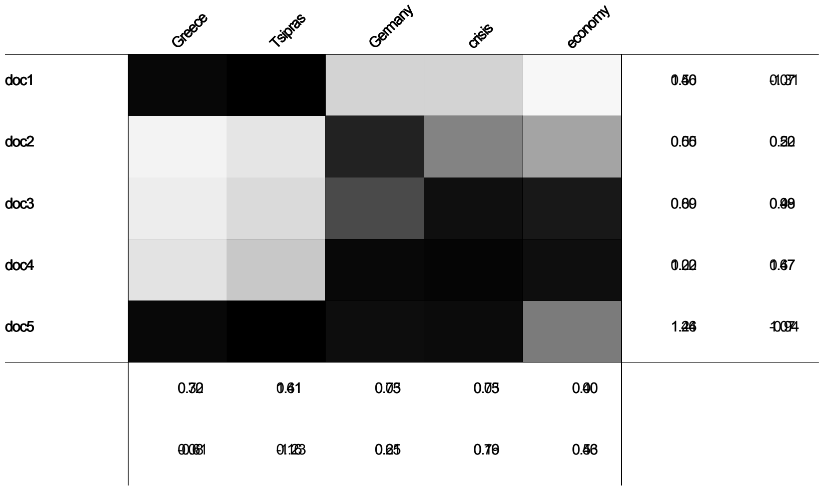
- topics
- image features



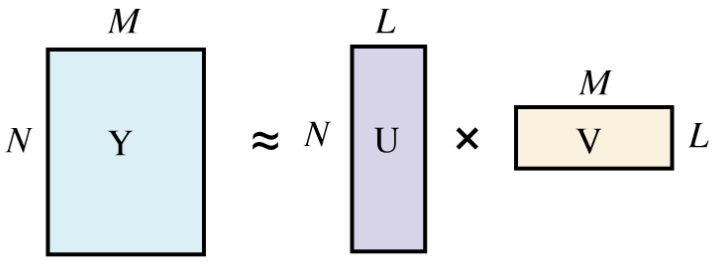
http://nimfa.biolab.si/nimfa.examples.orl_images.html

MF is commomly used to find how each instance is composed of its parts; Subtraction is often counter-intuitive

Non-negative MF vs MF



Definition



Given $\mathbf{Y} \in \Re^{N \times M} \geq 0$

find $\mathbf{U} \in \Re^{N \times L} \geq 0$ and $\mathbf{V} \in \Re^{M \times L} \geq 0$

so that $\mathbf{Y} \approx \mathbf{U}\mathbf{V}^T$

NMF is additive mixture/(soft) clustering

Basic NMF algorithm

Multiplicative update rules to minimize $\|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_2^2$:

$$u_{n,l} = u_{n,l} \frac{(\mathbf{Y}\mathbf{V})_{n,l}}{(\mathbf{U}\mathbf{V}^T\mathbf{V})_{n,l}}$$

$$v_{m,l} = v_{m,l} \frac{(\mathbf{Y}^T\mathbf{U})_{m,l}}{(\mathbf{V}\mathbf{U}^T\mathbf{U})_{m,l}}$$

Lee and Seung (2001)

Recent work have switched to constrained Alternating Least Squares optimization

Kim and Park (2008)

NMF with KL divergence

Assume that $\mathbf{Y}$ represents a probability distribution

$$\sum_{n,m} y_{n,m} = 1$$

Minimize the Kullback-Leibler divergence:

$$KL_{div}(\mathbf{Y} \| \mathbf{U}\mathbf{V}^T) = \sum_{n,m} (y_{n,m} \log \frac{y_{n,m}}{(\mathbf{U}\mathbf{V}^T)_{n,m}})$$

- results in similar multiplicative updates
- ensures that $\mathbf{U}$ and $\mathbf{V}$ also represent probability distributions

Probabilistic Latent Semantic Analysis

$$y_{n,m} = P(n, m)$$

$$= \sum_{l} P(l)P(m|l)P(n|l)$$

where n is a document, m a word and l a component.

$$\cdots = u'_{n,l} v'_{l,m}$$

where $u'_{n,l} = P(l)P(n|l)$ and $v'_{l,m} = P(m|l)$

Probabilistic Latent Semantic Analysis

$$\mathbf{Y} \approx \mathbf{U}\mathbf{V}^T = (\mathbf{U}\mathbf{A}^{-1}\mathbf{A})(\mathbf{V}\mathbf{B}^{-1}\mathbf{B})^T$$

$$= (\mathbf{U}\mathbf{A}^{-1})(\mathbf{A}\mathbf{B})(\mathbf{V}\mathbf{B}^{-1})^T$$

where $\mathbf{A} = diag(\sum_n v_{n,1}, \ldots, \sum_n v_{n,L})$

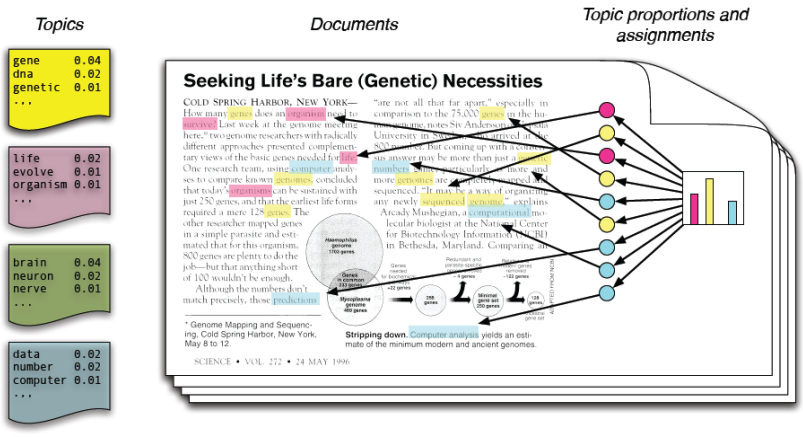
and $\mathbf{B} = diag(\sum_m u_{m,1}, \ldots, \sum_m u_{m,L})$

---

pLSA uses EM for parameter estimation and can converge to different local optima of the same objective as NMF-KL.

Gaussier and Goutte (2005)

Latent Dirichlet Allocation



Generative story

- For each topic l
  - draw multinomial over words $\mathbf{v}_{:l} \in \Re^M$ from $Dir(\beta)$
- For each document n
  - draw multinomial over topics $\mathbf{u}_{n:} \in \Re^L$ from $Dir(\alpha)$
  - for each word i
    - sample a topic assignment z_i from $\mathbf{u}_{n:}$
    - sample a word w_i from $\mathbf{v}_{:z_i}$

Given the row-normalized document-term matrix $\boldsymbol{Y}$, find the factors $\boldsymbol{U}$ and $\boldsymbol{V}$ that reconstruct it.

Arora et al (2012)

Summary

- Saw the matrix
- Refresher on matrix rank and norms
- Standard approaches to MF
- Non-negative MF and connections to pLSA and LDA

Binary Matrix Factorization

Binary Data

	worksFor	teacherAt	affiliatedWith	profAt
(Blunsom,Oxford)	0.56	1.39	0.91	0.38
(Clark,Cambridge)	0.88	0.34	0.12	2.06
(Riedel,UCL)	0.59	2.29	2.45	1.04
(Vlachos,Sheffield)	1.66	1.01	2.61	2.96

Logistic Loss

Objective

$$\sum_{(i,j)\in\Omega} \log(\text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle)^{y_{ij}} + \log(1 - \text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle)^{1-y_{ij}}$$

- Probability of $y_{ij} = 1$
- Probability of $y_{ij} = 0$

Gradient Steps (for $y_{ij} = 1$)

$$\mathbf{u}_i \leftarrow \mathbf{u}_i - \eta \text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle \mathbf{v}_j$$

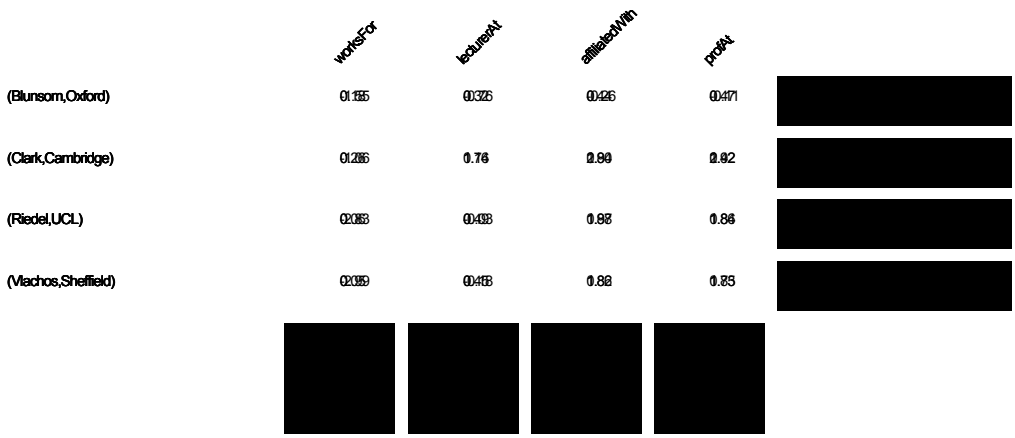
$$\mathbf{v}_j \leftarrow \mathbf{v}_j - \eta \text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle \mathbf{u}_i$$

Equivalent View: Generalized PCA

```
1
2
3
4
5
```

```
val k = 2
val m1 = opacity(matrix(M),0.1) + names
val (a,v) = optimizeL2(M, k, 1000)
val m2 = opacity(matrix(dots(a,v)),0.1) + names
render(Seq(m1,m2).layout)
```

?



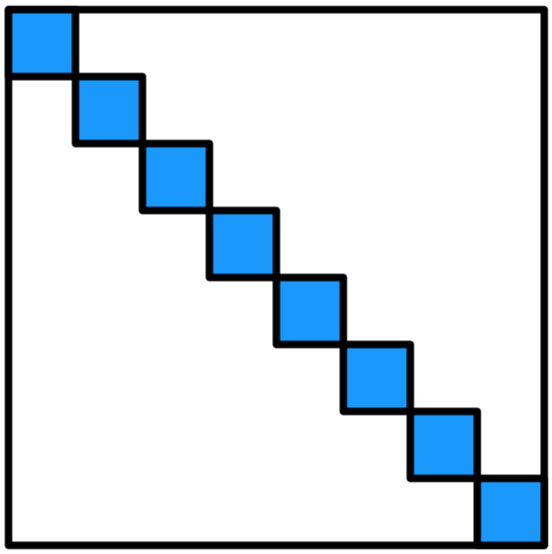
```
def optimizeLogistic(M: Mat, K: Int, iters: Int, alpha:Double = 0.
        initScale:Double = 1.0): (Seq[Vect], Seq[Vect]) = {
    val rand = new scala.util.Random(0)
  val AV = initialAV(K, M.dim1, M.dim2, initScale)
    val A = AV._1; val V = AV._2
     def update(i: Int, j: Int) ={
        val a = A(i).copy
        val v = V(j).copy
      val p = sigmoid(a dot v)
        A(i) += v * alpha * (M(i, j) - p)
        V(j) += a * alpha * (M(i, j) - p)
                }
  for (i <- Range(0,iters).toList) {
  update(rand.nextInt(M.dim1),
        rand.nextInt(M.dim2))
                }
                (A, V)
                }
                '''
```

?

#Slide3

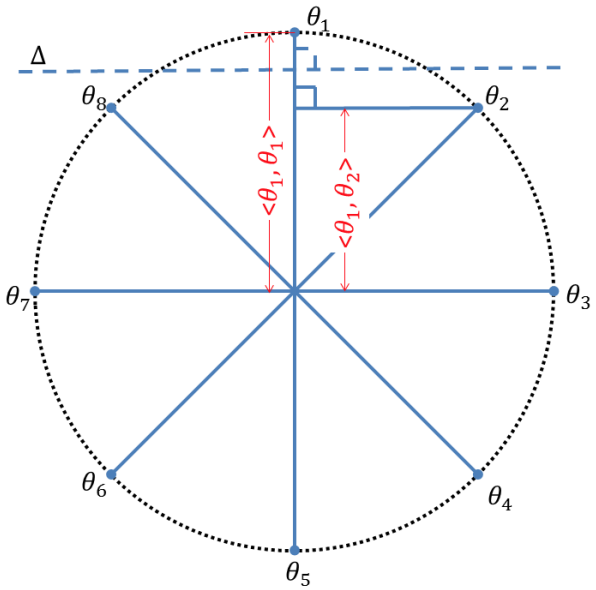
---

	worksFor	lectureAt	professorAt	studiedAt
(Anna,UCL)	0.0	1.0	1.0	0.0
(Bob,MSR)	0.0	0.0	1.0	1.0
(Andreas,UCL)	0.0	1.0	1.0	1.0
(James,Oxford)	0.0	0.0	0.0	0.0
(Bob,Sheffield)	0.0	1.0	0.0	0.0



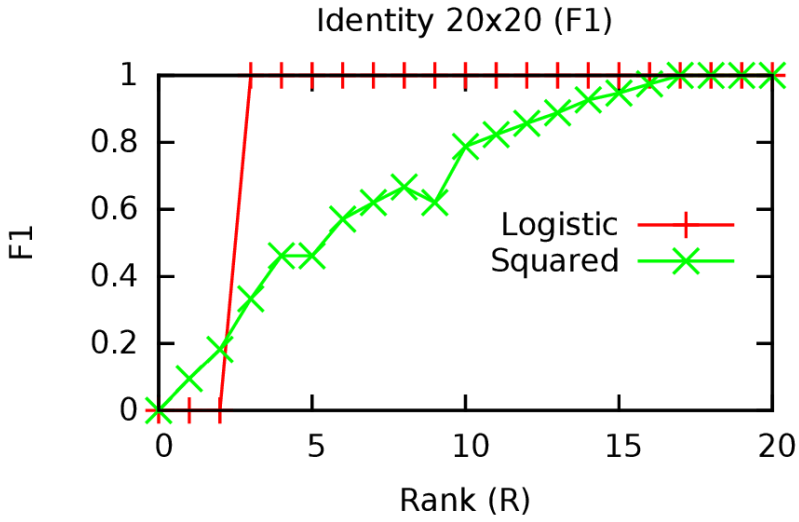
Sign-rank

(Linial et al., 2007; Bouchard et al., 2015)



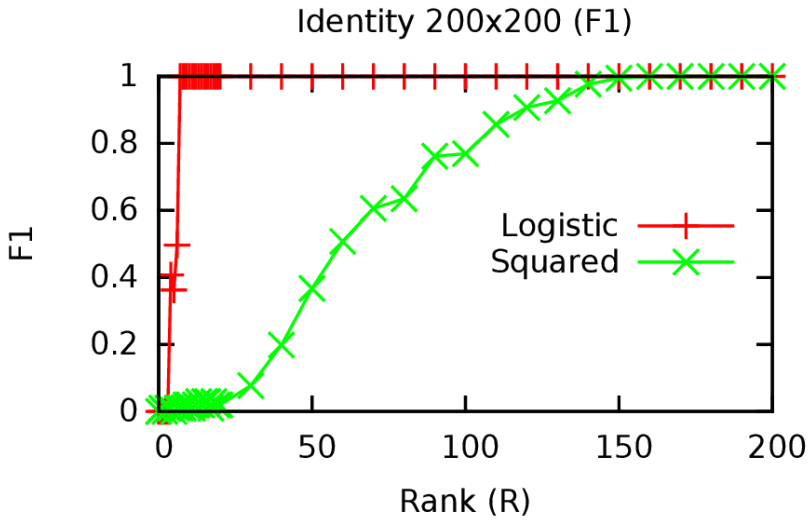
Sign-rank

(Linial et al., 2007; Bouchard et al., 2015)



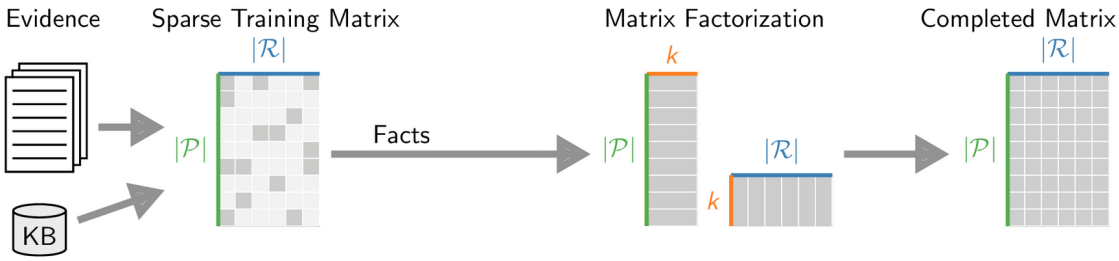
Sign-rank

(Linial et al., 2007; Bouchard et al., 2015)



Negative Data
Application: Relation Extraction

(Riedel et al, 2013)



- Often only positive data is observed
- Sample unobserved cells and treat as negative
- Leads to bias, requires many samples

Implicit Negative Training Data

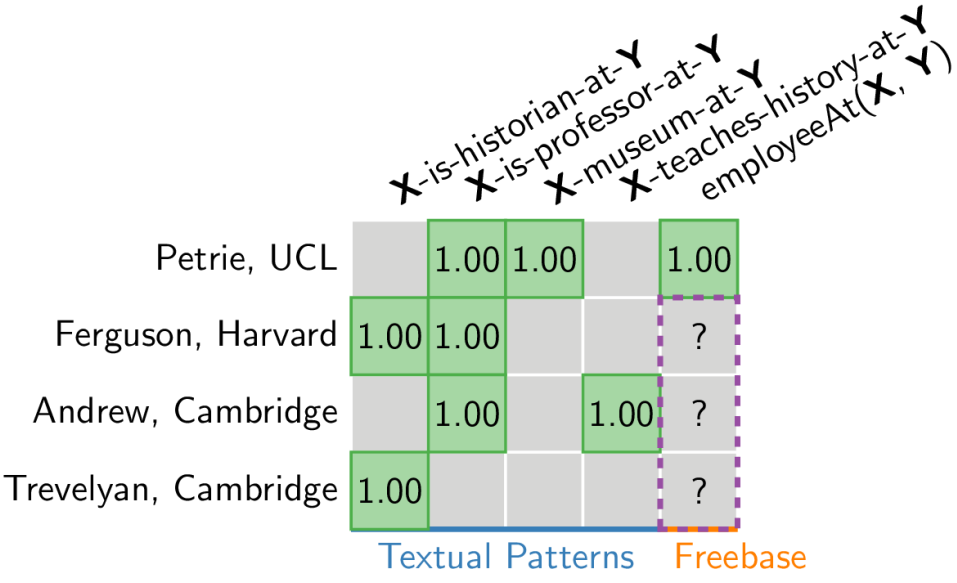
$$\sum_{(i,j)\in\Omega} \omega_{\text{pos}} \log(\text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle) + \sum_{(a,b)\notin\Omega} \omega_{\text{neg}} \log\left(1 - \text{sigm}\langle \mathbf{u}_a, \mathbf{v}_b \rangle\right)$$

Bayesian Personalized Ranking

(Rendle et al, 2009)

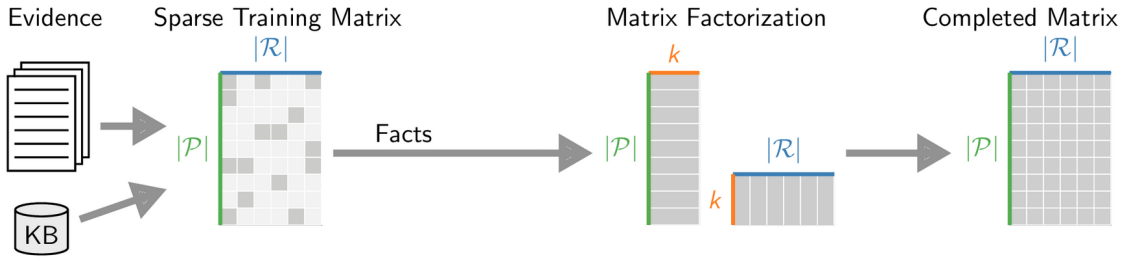
$$\sum_{(i,j)\in\Omega} \sum_{(a,b)\notin\Omega} \log\left(\mathrm{sigm}(\langle \mathbf{u}_i, \mathbf{v}_j\rangle - \langle \mathbf{u}_a, \mathbf{v}_b\rangle)\right)$$

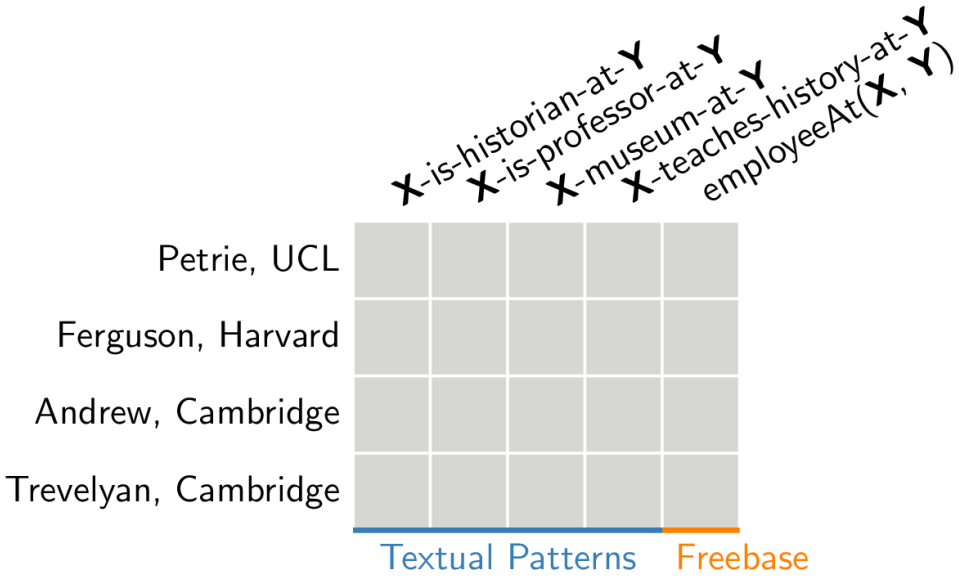
Sample During SGD

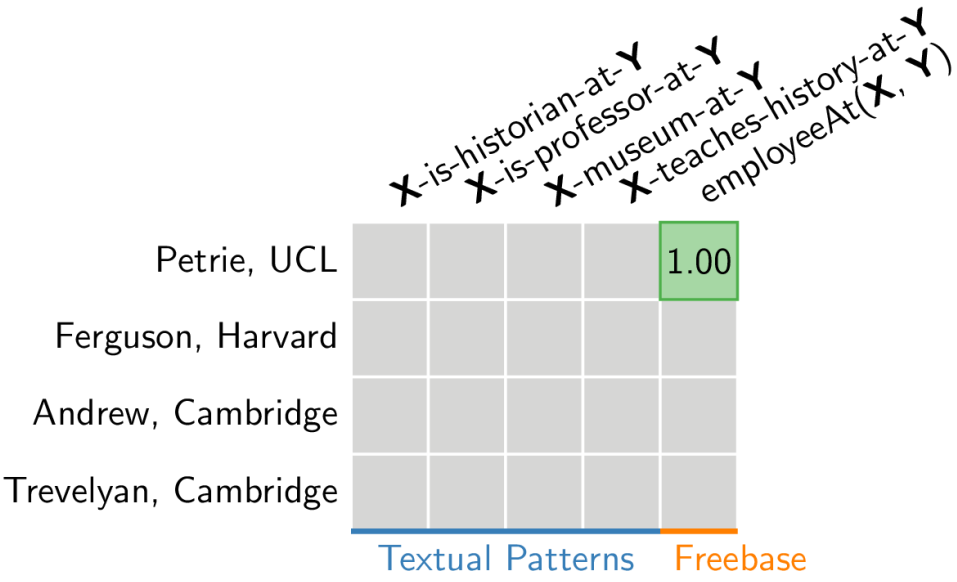


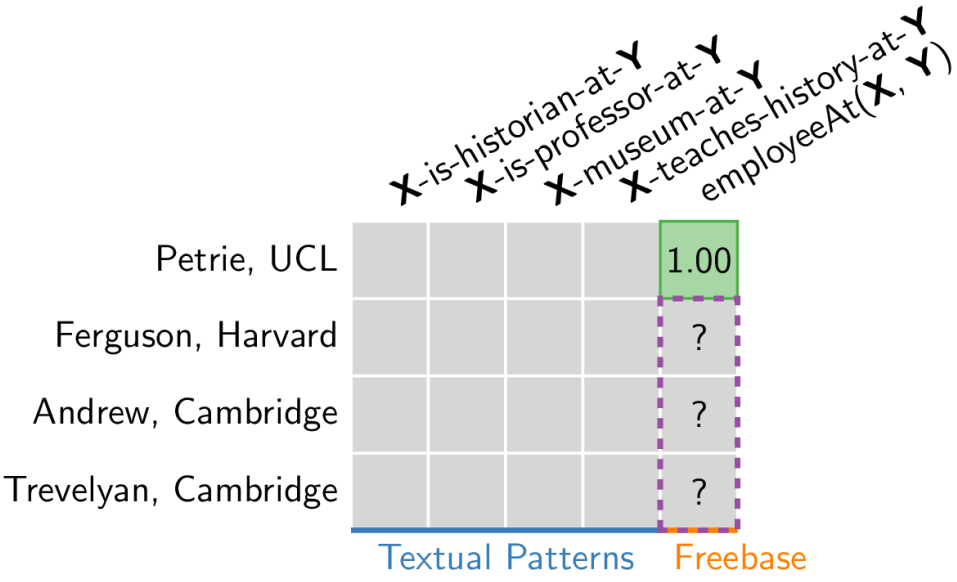
Application: Relation Extraction

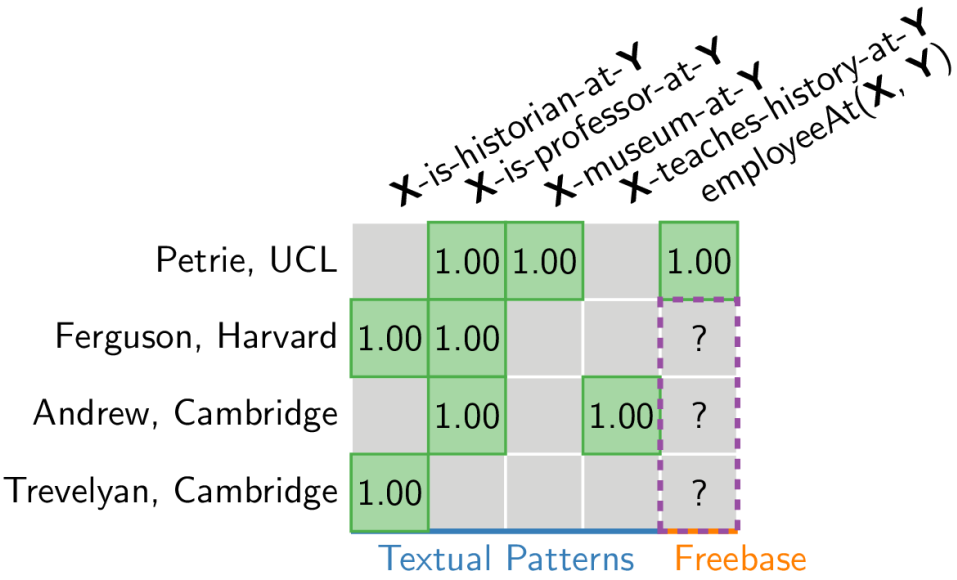
(Riedel et al, 2013)

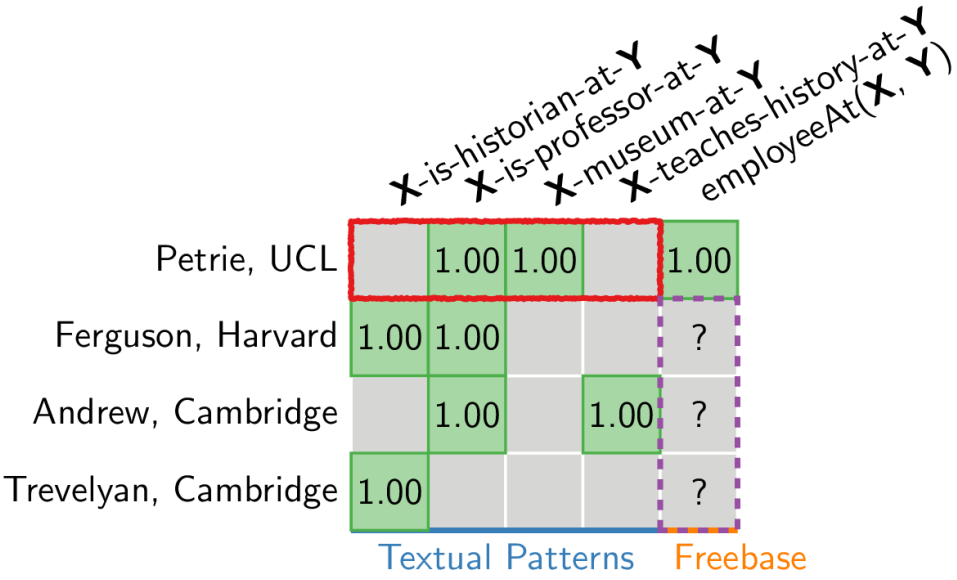


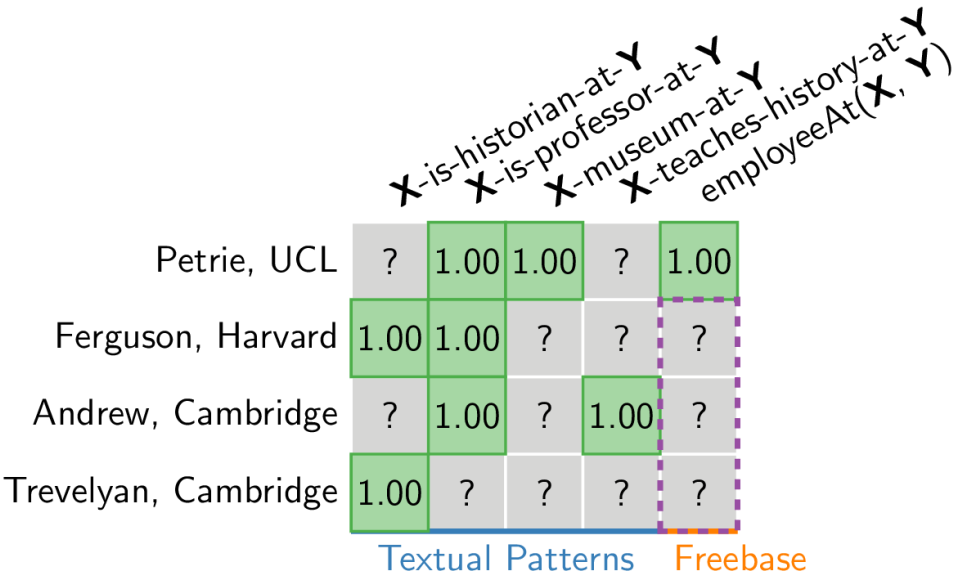


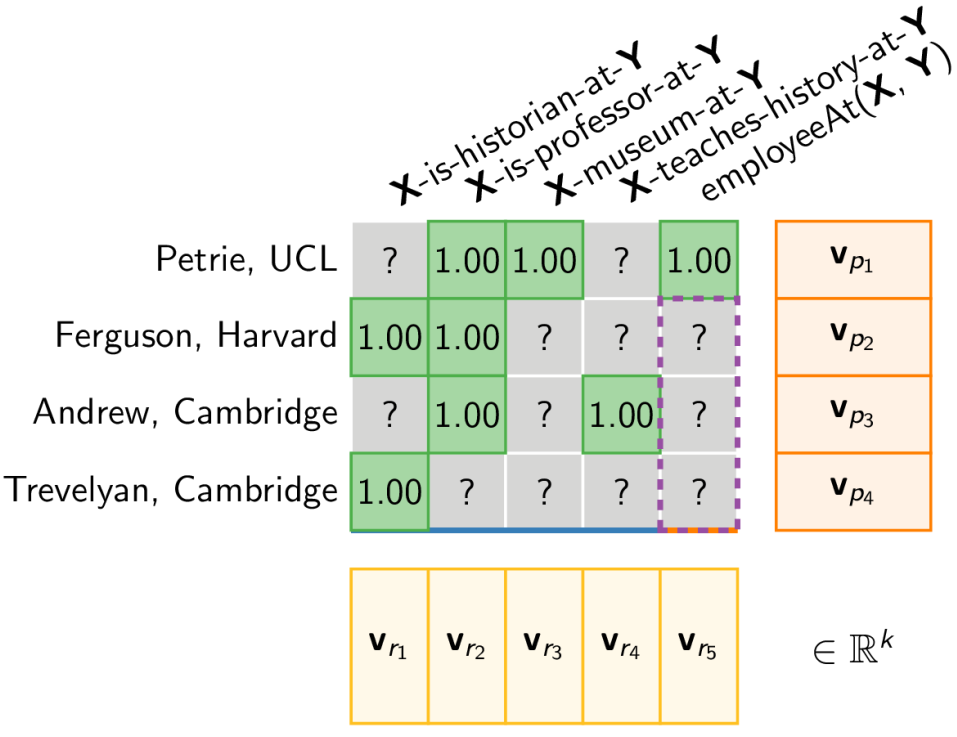

X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
1.00
Textual Patterns
Freebase

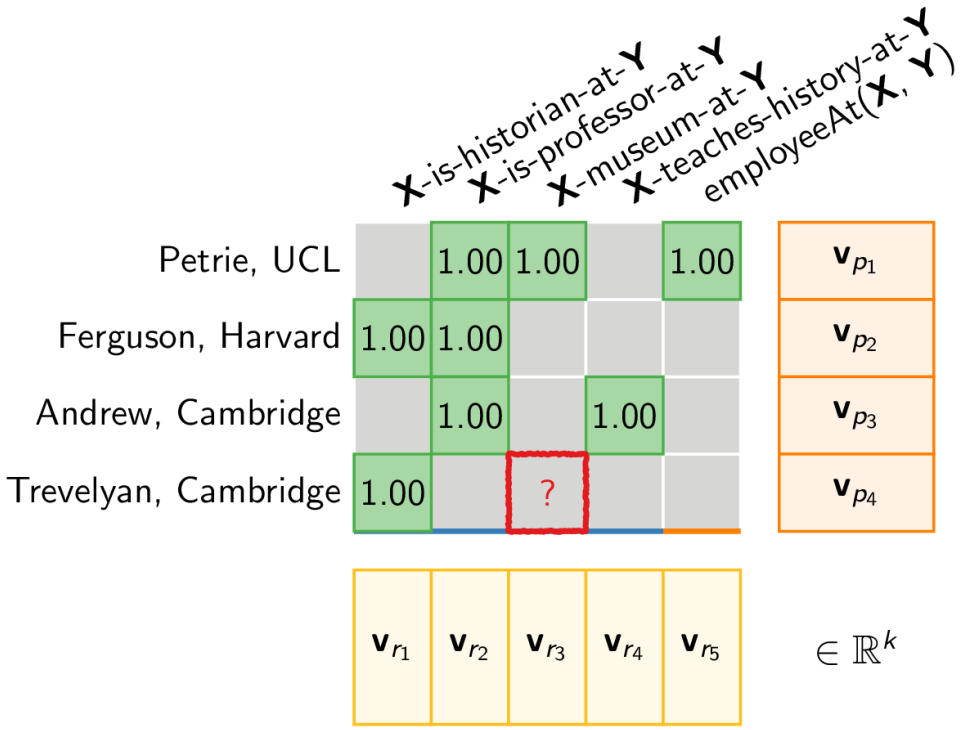

X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
1.00
?
?
?
Textual Patterns
Freebase

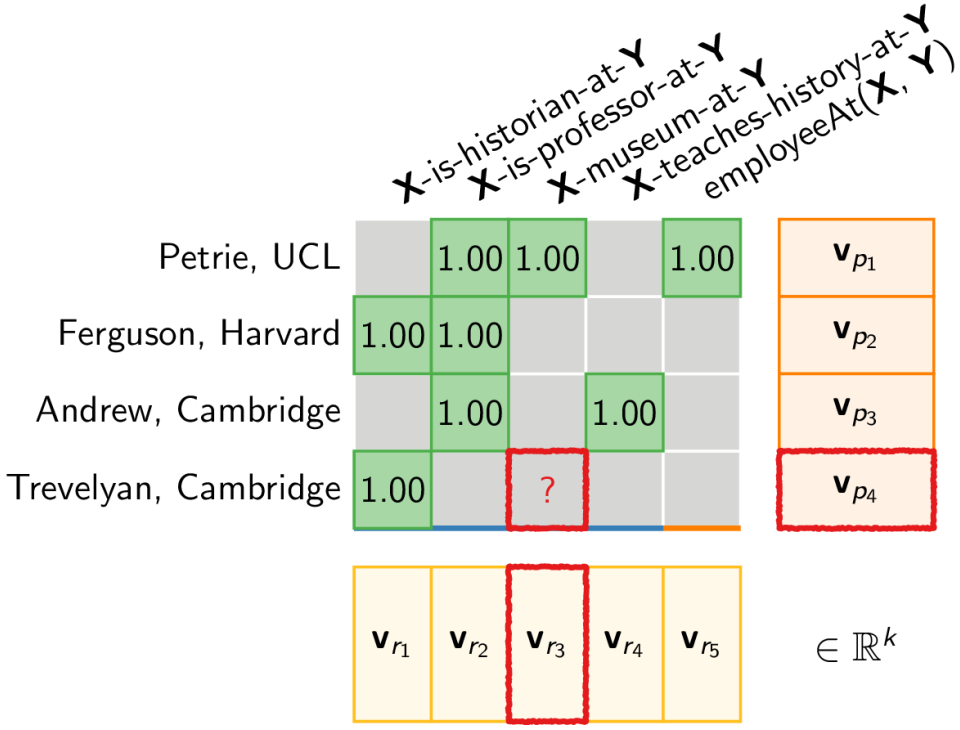

X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
1.00
?
Textual Patterns
Freebase

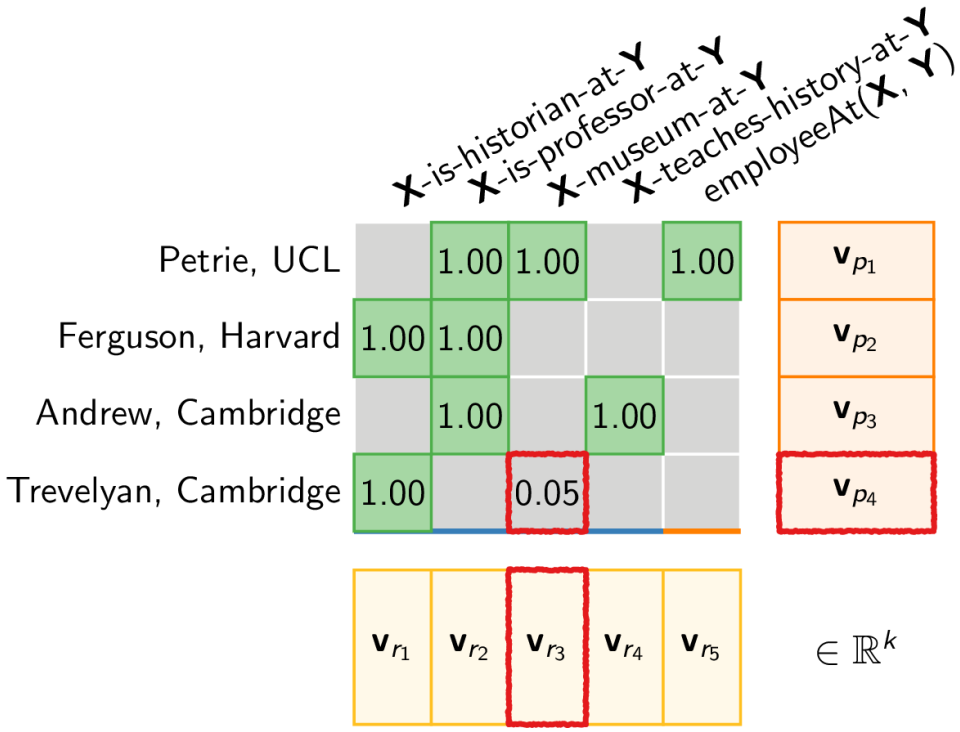

X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
1.00
?
Textual Patterns
Freebase

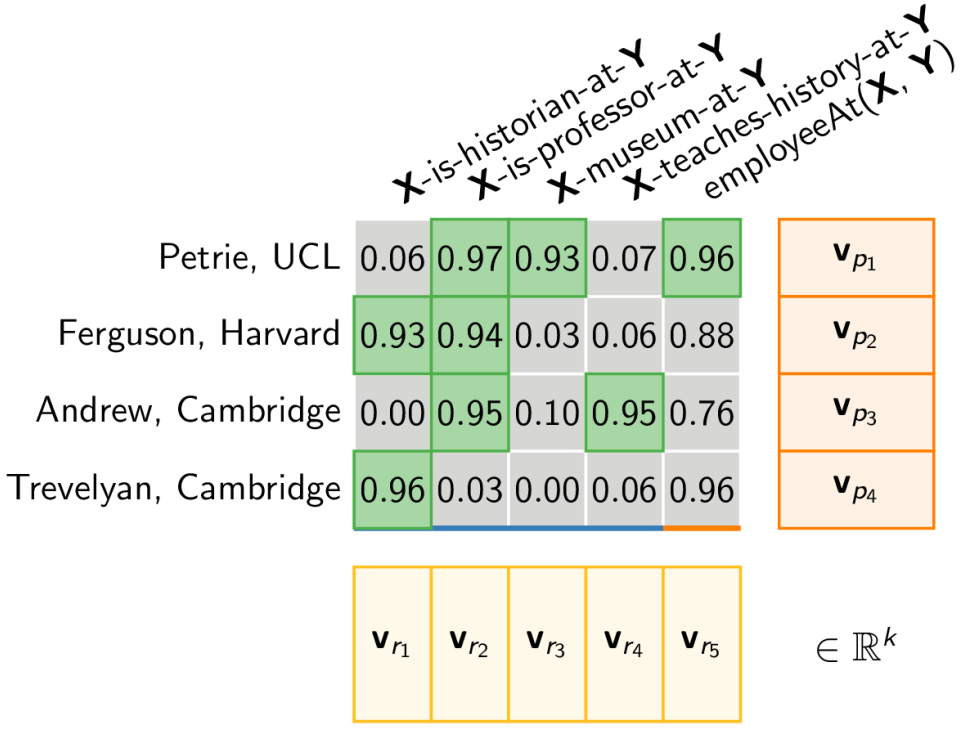

X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
? 1.00 1.00 ? 1.00
1.00 1.00 ? ? ?
? 1.00 ? 1.00 ?
1.00 ? ? ? ?
Textual Patterns
Freebase

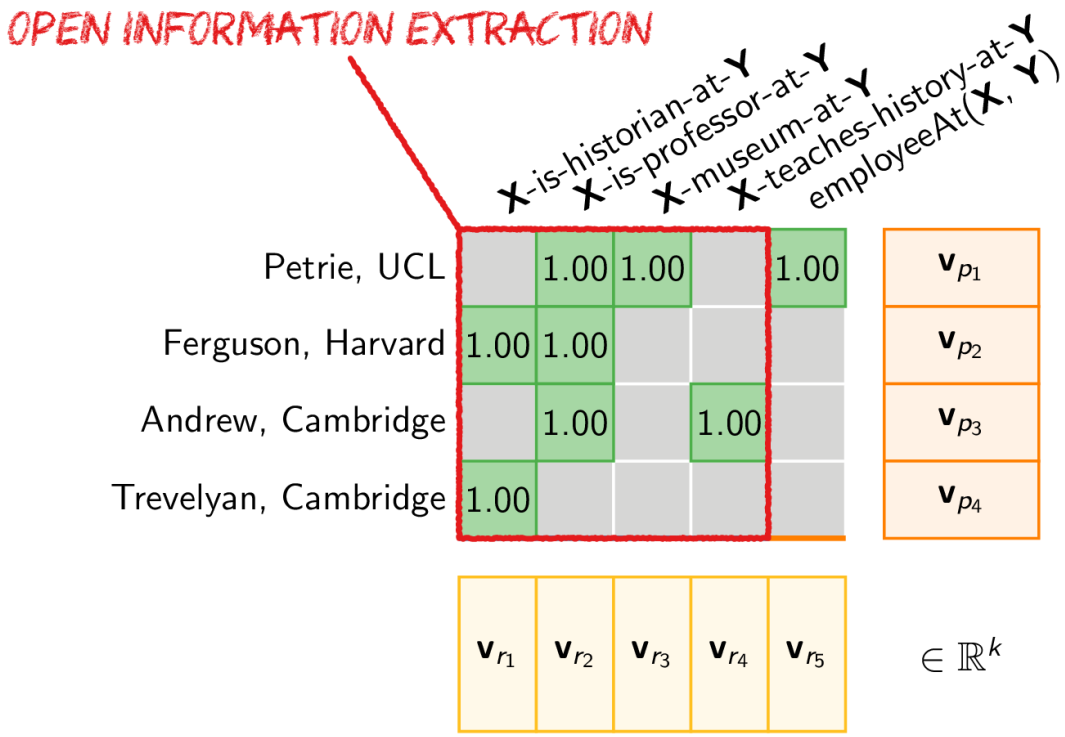

X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
? 1.00 1.00 ? 1.00
1.00 1.00 ? ? ?
? 1.00 ? 1.00 ?
1.00 ? ? ? ?
$\mathbf{v}_{p_1}$
$\mathbf{v}_{p_2}$
$\mathbf{v}_{p_3}$
$\mathbf{v}_{p_4}$
$\mathbf{v}_{r_1}$ $\mathbf{v}_{r_2}$ $\mathbf{v}_{r_3}$ $\mathbf{v}_{r_4}$ $\mathbf{v}_{r_5}$
$\in \mathbb{R}^k$

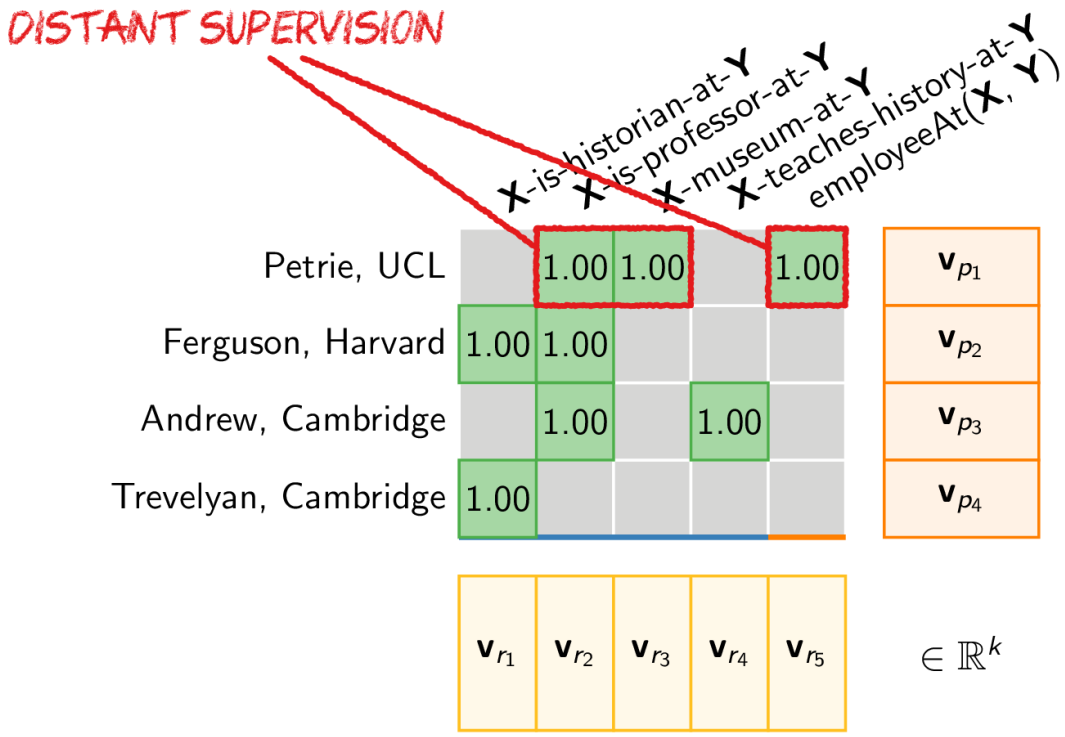

X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
1.00
?
$\mathbf{v}_{p_1}$
$\mathbf{v}_{p_2}$
$\mathbf{v}_{p_3}$
$\mathbf{v}_{p_4}$
$\mathbf{v}_{r_1}$ $\mathbf{v}_{r_2}$ $\mathbf{v}_{r_3}$ $\mathbf{v}_{r_4}$ $\mathbf{v}_{r_5}$
$\in \mathbb{R}^k$

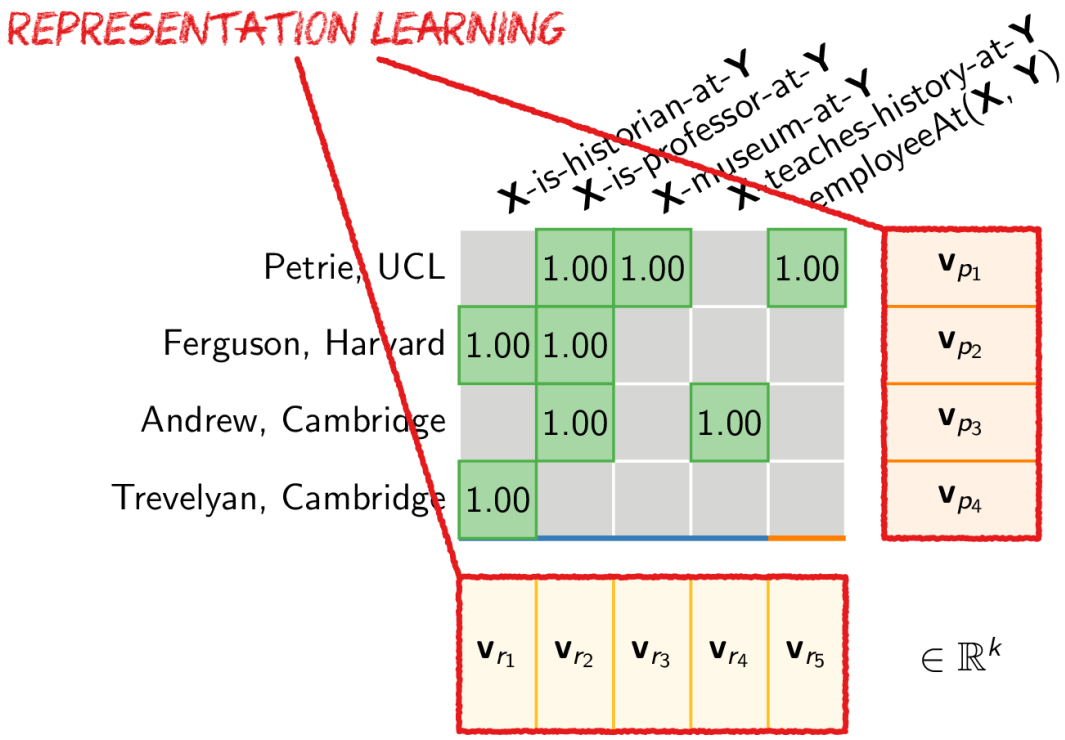

X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
1.00
?
$\mathbf{v}_{p_1}$
$\mathbf{v}_{p_2}$
$\mathbf{v}_{p_3}$
$\mathbf{v}_{p_4}$
$\mathbf{v}_{r_1}$ $\mathbf{v}_{r_2}$ $\mathbf{v}_{r_3}$ $\mathbf{v}_{r_4}$ $\mathbf{v}_{r_5}$
$\in \mathbb{R}^k$


X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
1.00 1.00 1.00
Ferguson, Harvard
1.00 1.00
Andrew, Cambridge
1.00 1.00
Trevelyan, Cambridge
1.00 0.05
$\mathbf{v}_{p_1}$
$\mathbf{v}_{p_2}$
$\mathbf{v}_{p_3}$
$\mathbf{v}_{p_4}$
$\mathbf{v}_{r_1}$ $\mathbf{v}_{r_2}$ $\mathbf{v}_{r_3}$ $\mathbf{v}_{r_4}$ $\mathbf{v}_{r_5}$
$\in \mathbb{R}^k$


X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
0.06 0.97 0.93 0.07 0.96
Ferguson, Harvard
0.93 0.94 0.03 0.06 0.88
Andrew, Cambridge
0.00 0.95 0.10 0.95 0.76
Trevelyan, Cambridge
0.96 0.03 0.00 0.06 0.96
$\mathbf{v}_{p_1}$
$\mathbf{v}_{p_2}$
$\mathbf{v}_{p_3}$
$\mathbf{v}_{p_4}$
$\mathbf{v}_{r_1}$ $\mathbf{v}_{r_2}$ $\mathbf{v}_{r_3}$ $\mathbf{v}_{r_4}$ $\mathbf{v}_{r_5}$
$\in \mathbb{R}^k$

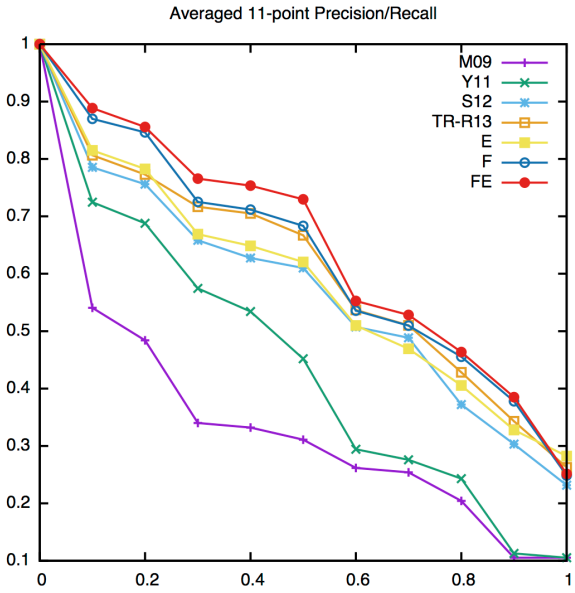

OPEN INFORMATION EXTRACTION
X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
1.00
$\mathbf{v}_{p_1}$
$\mathbf{v}_{p_2}$
$\mathbf{v}_{p_3}$
$\mathbf{v}_{p_4}$
$\mathbf{v}_{r_1}$ $\mathbf{v}_{r_2}$ $\mathbf{v}_{r_3}$ $\mathbf{v}_{r_4}$ $\mathbf{v}_{r_5}$
$\in \mathbb{R}^k$


DISTANT SUPERVISION
X-is-historian-at-Y
X-is-professor-at-Y
X-museum-at-Y
X-teaches-history-at-Y
employeeAt(X, Y)
Petrie, UCL
Ferguson, Harvard
Andrew, Cambridge
Trevelyan, Cambridge
1.00
$\mathbf{v}_{p_1}$
$\mathbf{v}_{p_2}$
$\mathbf{v}_{p_3}$
$\mathbf{v}_{p_4}$
$\mathbf{v}_{r_1}$ $\mathbf{v}_{r_2}$ $\mathbf{v}_{r_3}$ $\mathbf{v}_{r_4}$ $\mathbf{v}_{r_5}$
$\in \mathbb{R}^k$



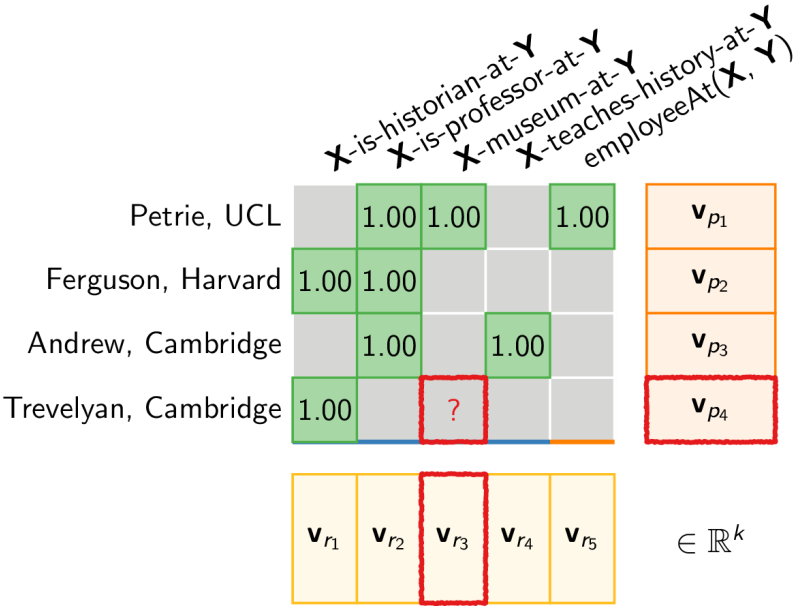
Results

(Riedel et al, 2013)

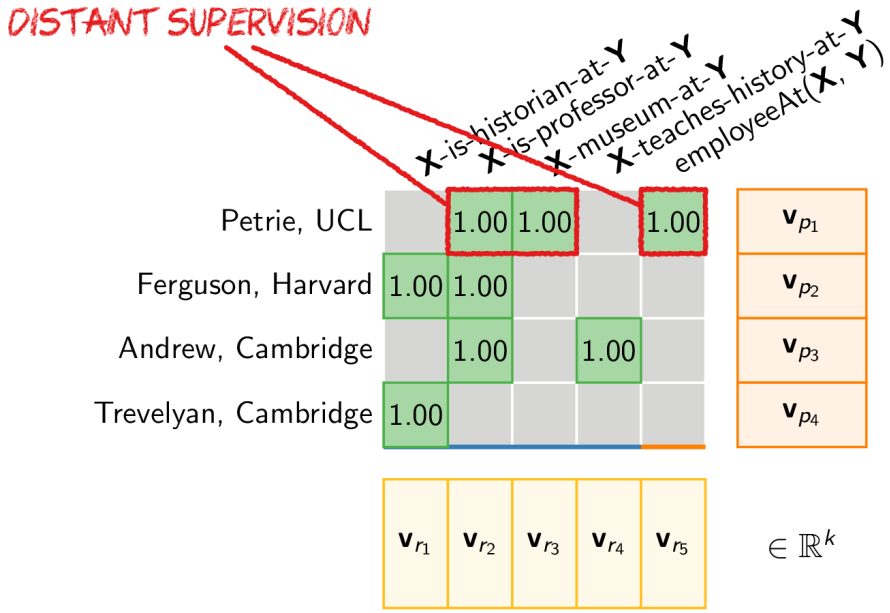


Inclusion of Prior Knowledge in Factorization Models

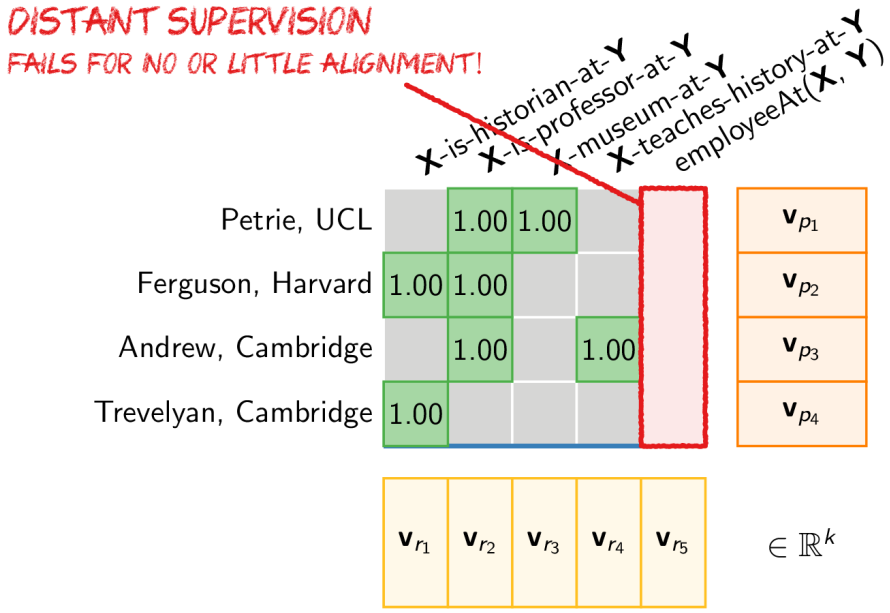
Drawbacks of OpenIE and Embeddings



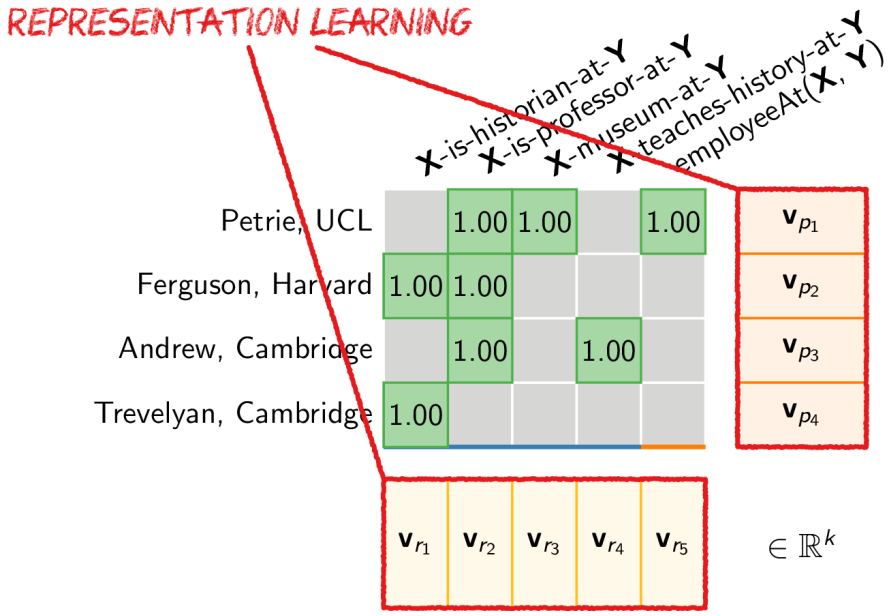
Drawbacks of OpenIE and Embeddings



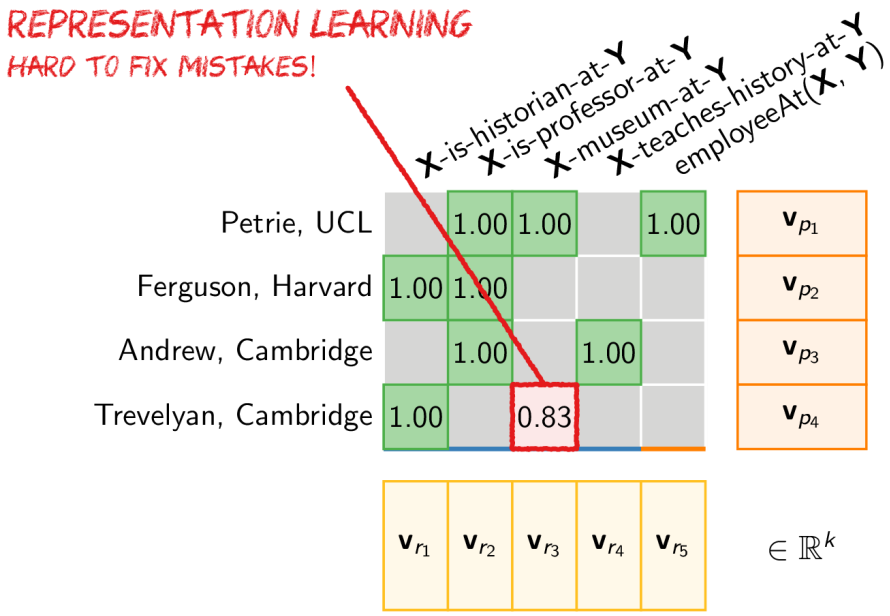
Drawbacks of OpenIE and Embeddings



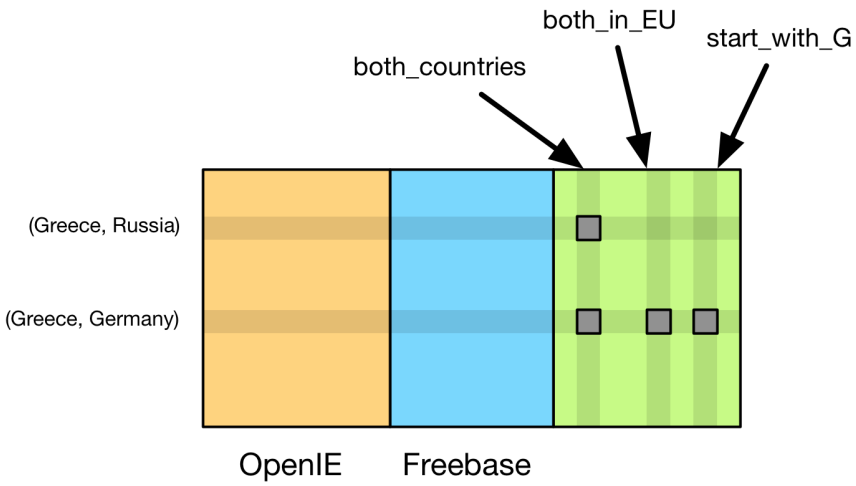
Drawbacks of OpenIE and Embeddings



Drawbacks of OpenIE and Embeddings



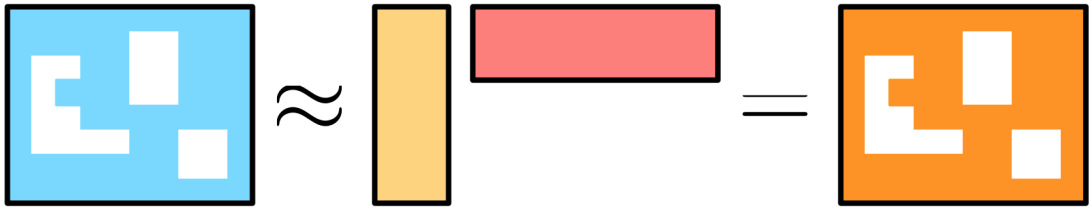
Factual Prior Knowledge



- Collective Matrix Factoriztion (Section 7)
- Discriminative Factorial Models (Section 6)

Hard Constraints

(e.g. Chang et al. 2014)



First-order Logic Prior Knowledge

- **Representation Learning**: hard to fix mistakes

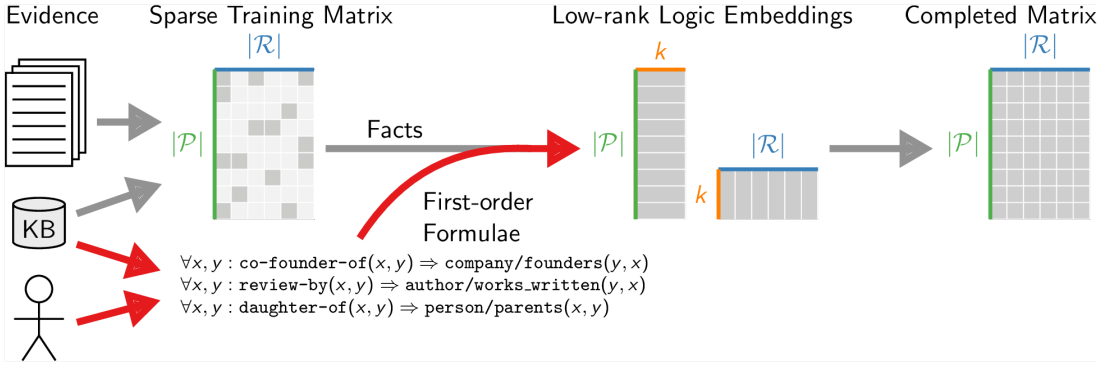
$$\neg \text{person}(x) \Rightarrow \neg \text{place of birth}(x, y)$$

- **Distant Supervision**: fails for no or little alignment

$$\#1\text{-is-a-student-in-}\#2\text{-'s-lab}(x, y) \Rightarrow \text{supervisedBy}(x, y)$$

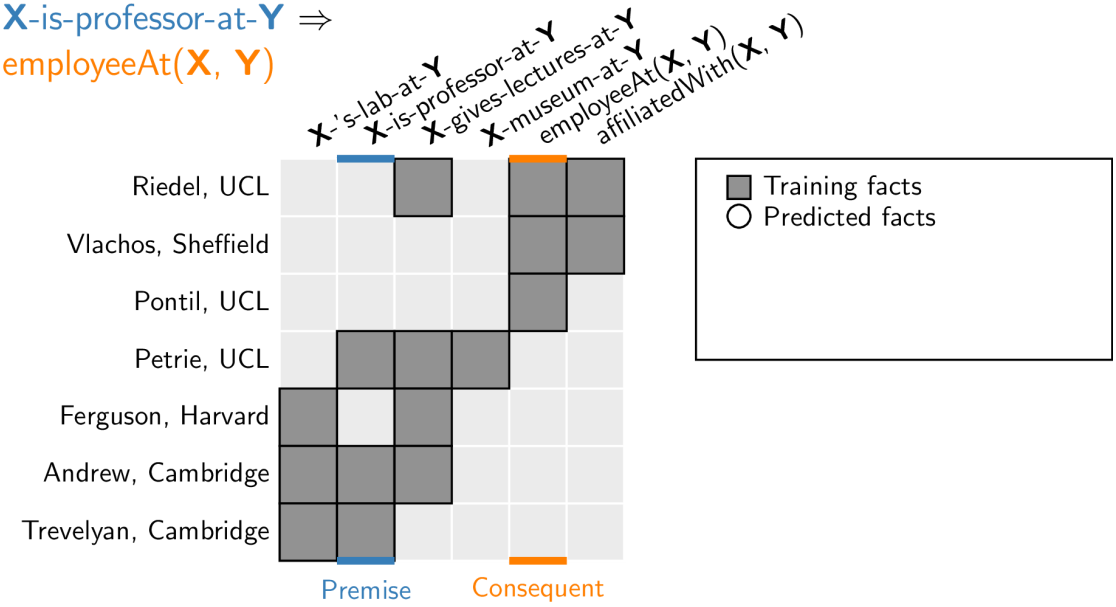
- **Pros**: Formulae are easy to modify and improve
- **Cons**: Brittle, no generalization and inference can become intractable
- *Markov Logic Networks*: often intractable in practice

Low-rank Logic Embeddings



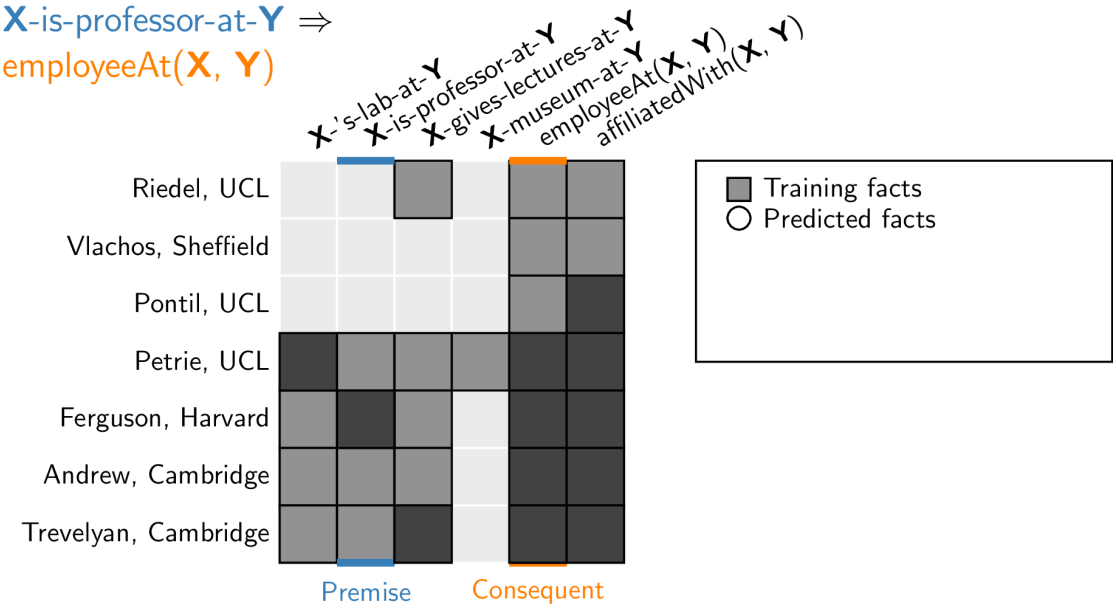
Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



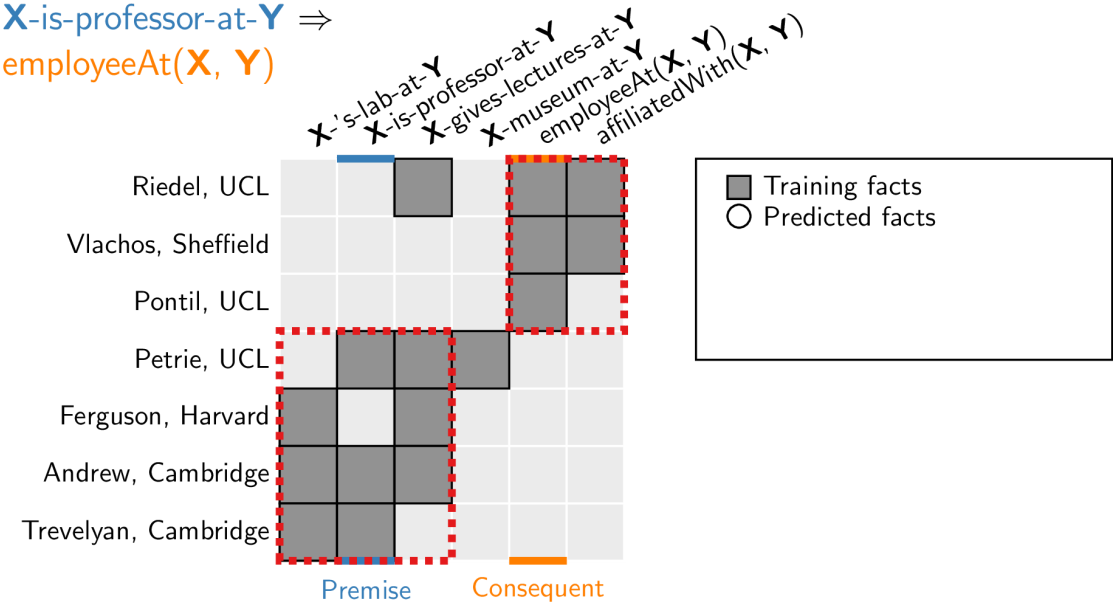
Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



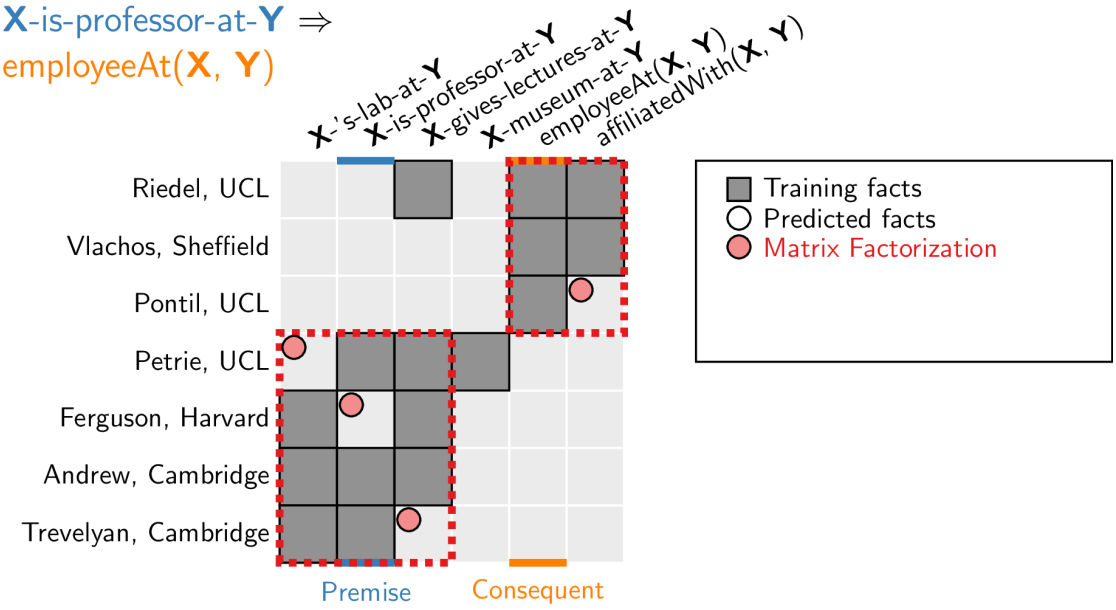
Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)

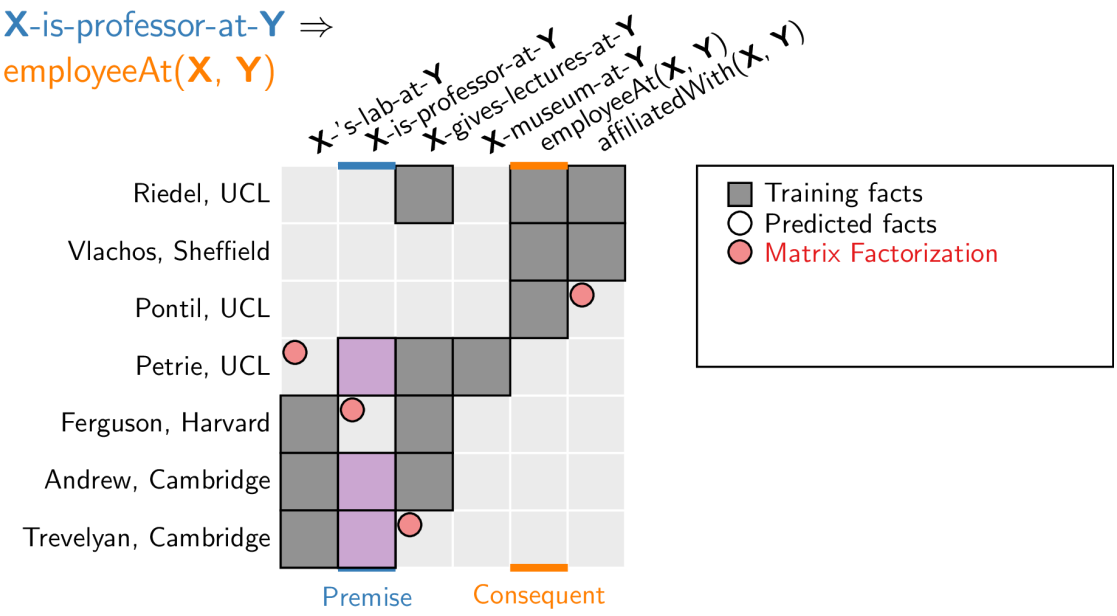


Combining Logic and Matrix Factorization

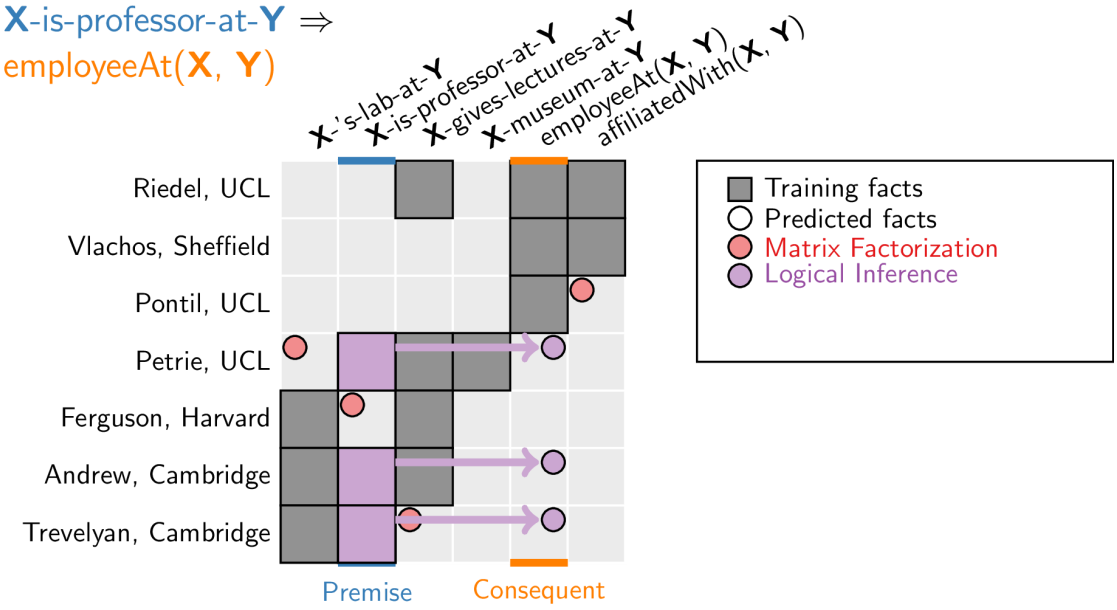
X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



Combining Logic and Matrix Factorization

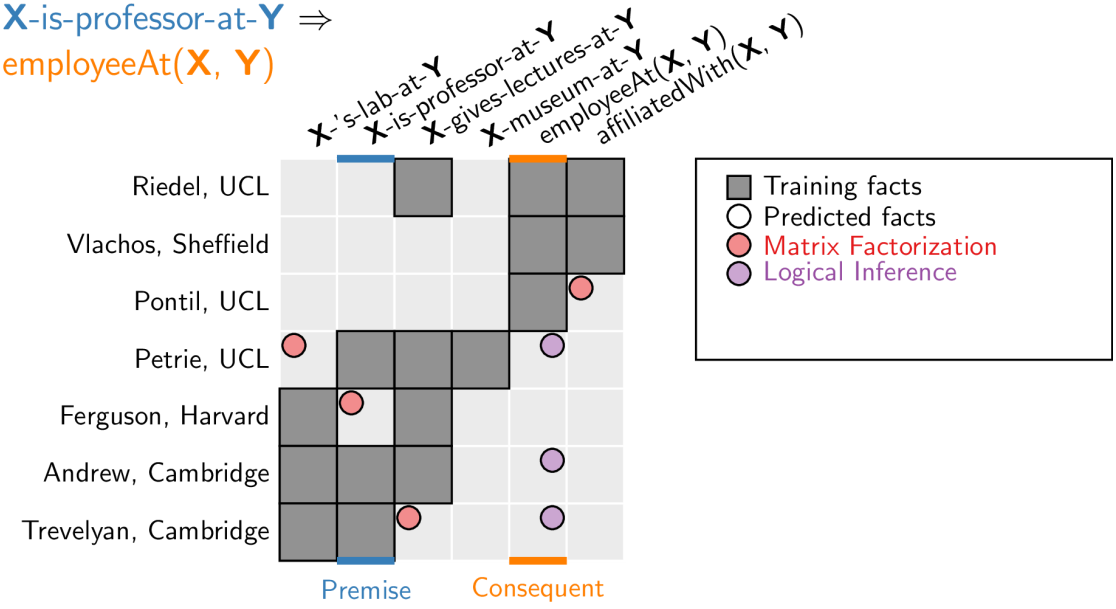


Combining Logic and Matrix Factorization



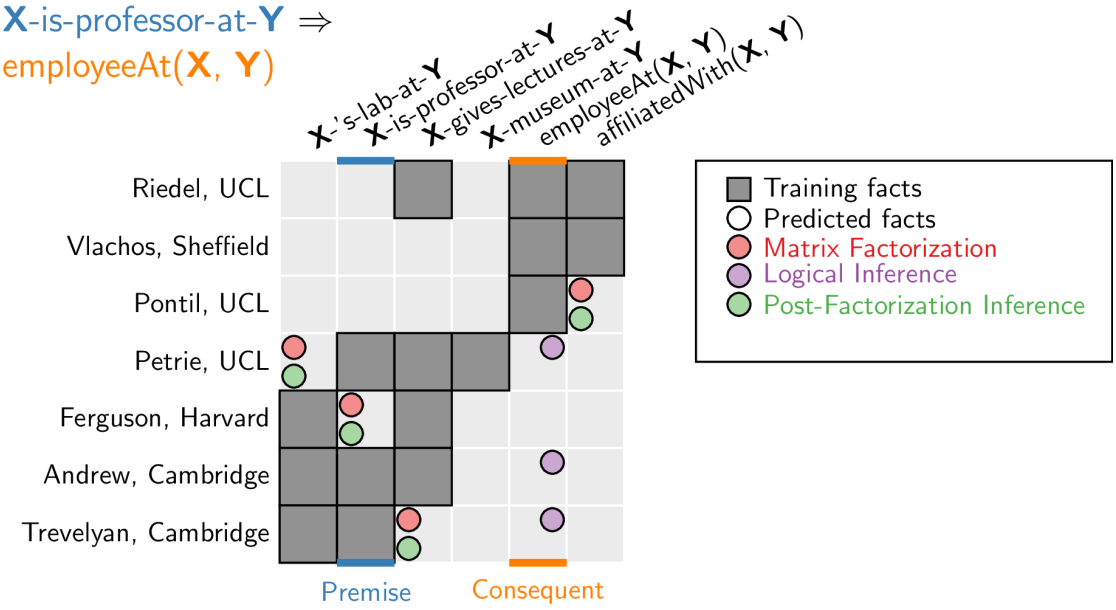
Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)

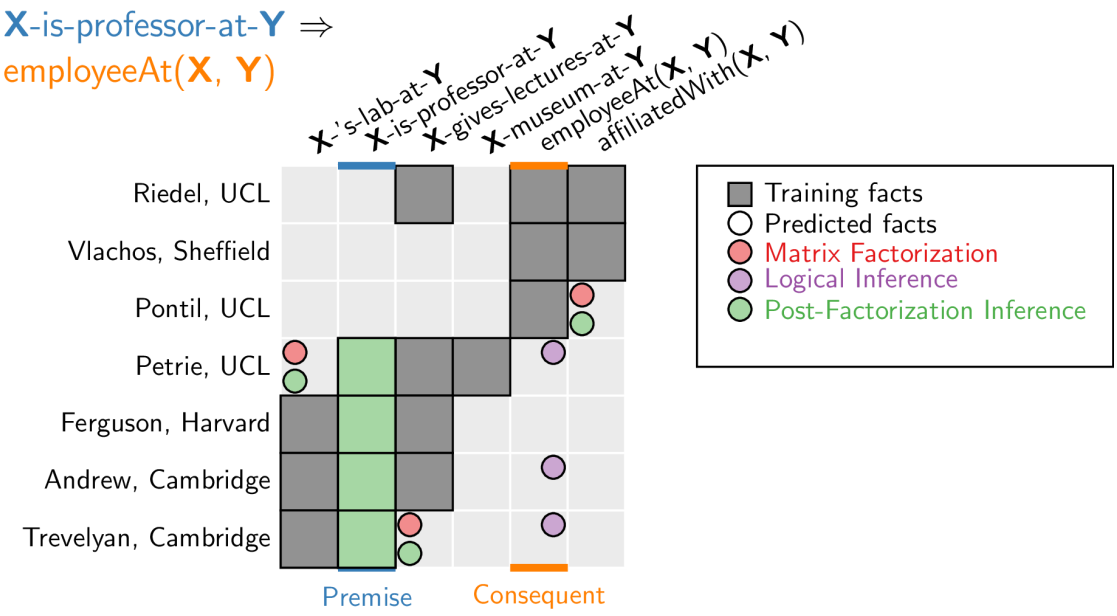


Combining Logic and Matrix Factorization

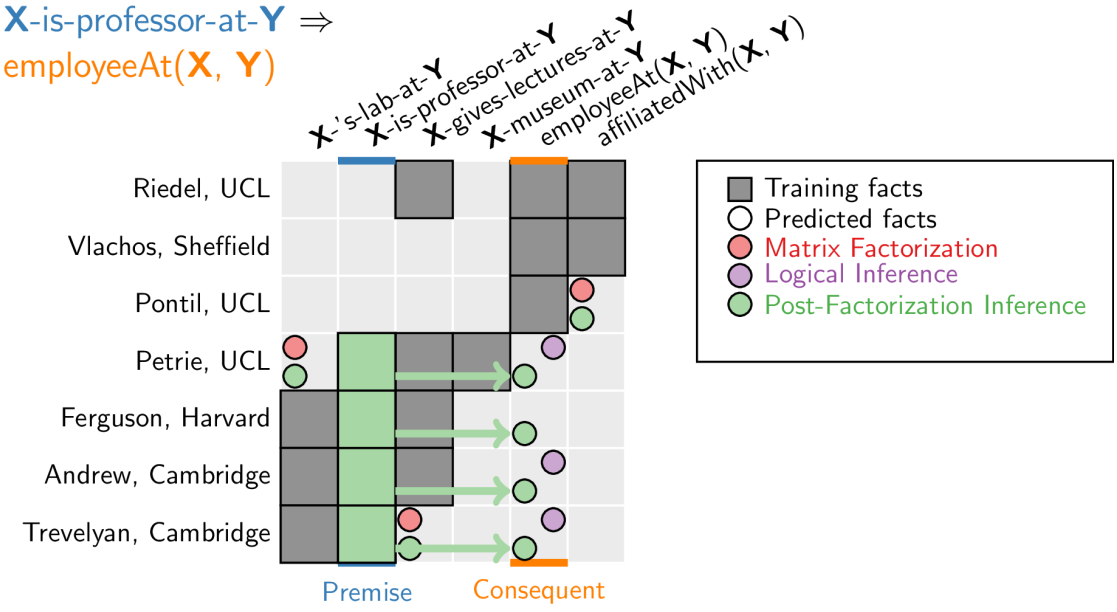
X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



Combining Logic and Matrix Factorization

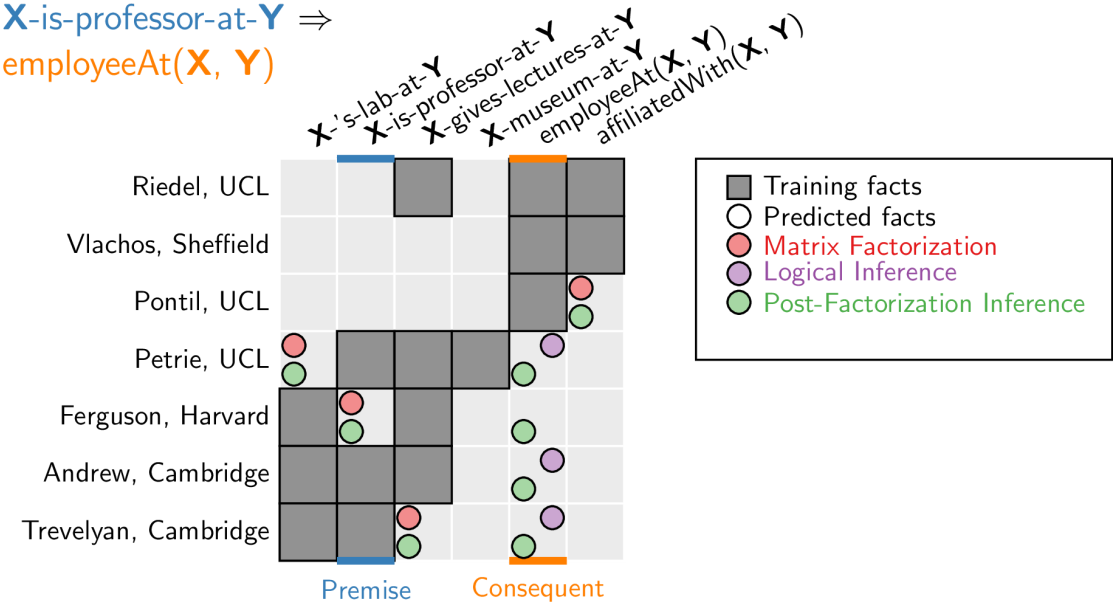


Combining Logic and Matrix Factorization



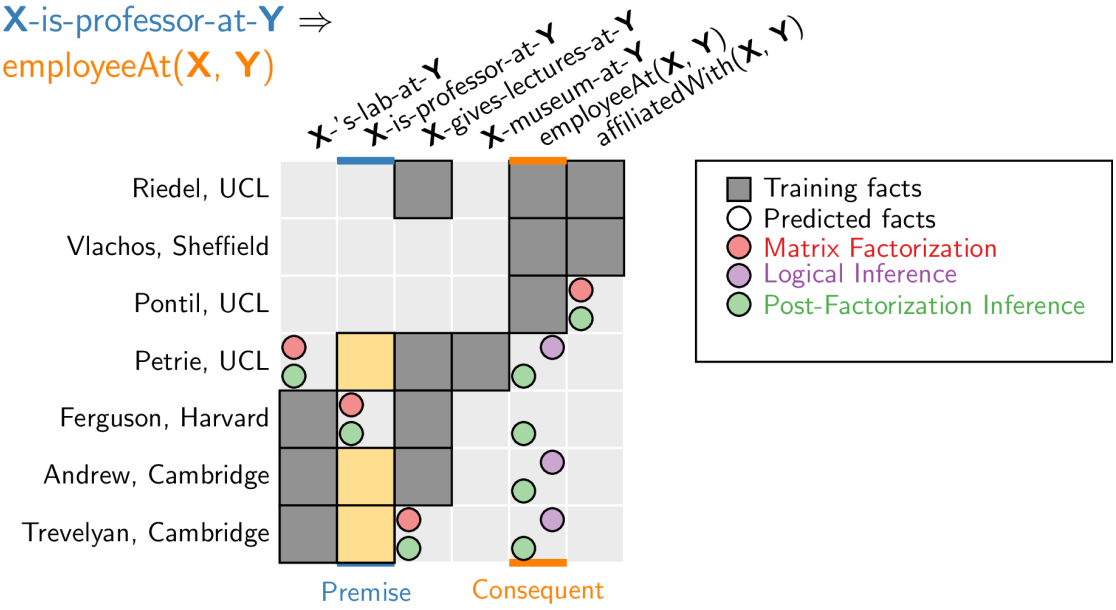
Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



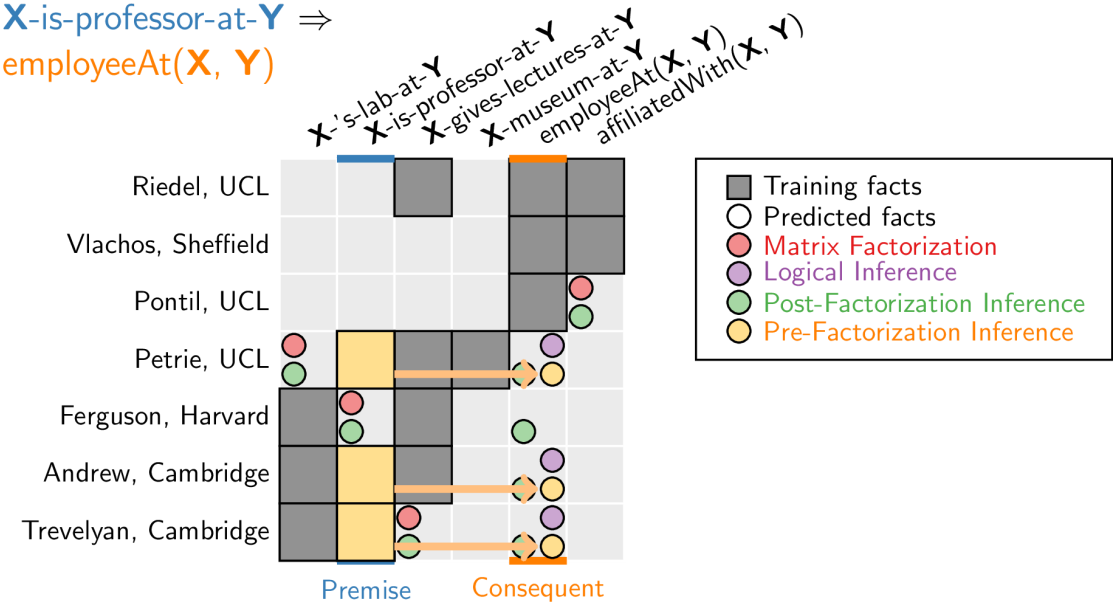
Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



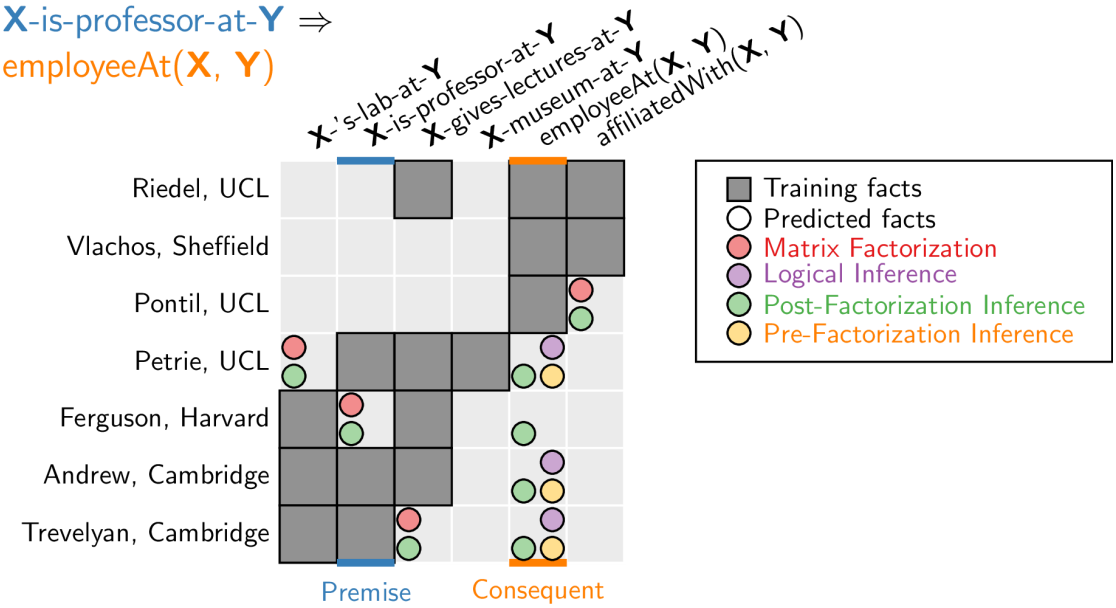
Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



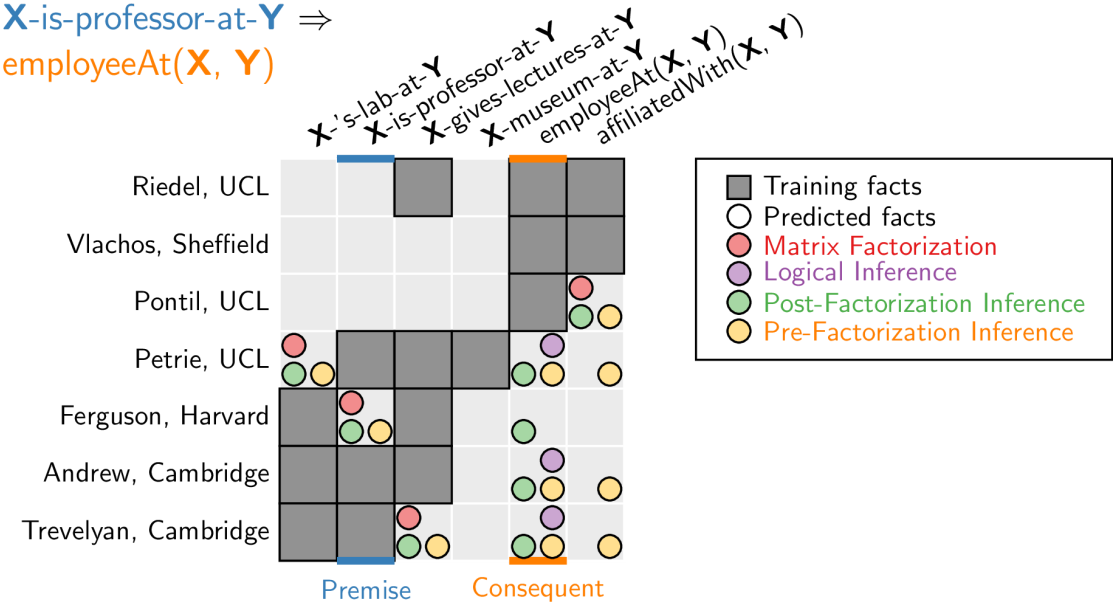
Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



Combining Logic and Matrix Factorization

X-is-professor-at-**Y** ⇒
employeeAt(**X**, **Y**)



Matrix Factorization is "Injecting" Atomic Formulae

- Training facts are ground atoms:

$$\mathcal{F} = r_s(e_i, e_j)$$

- Let $[\bullet]$ denote mapping from symbolic formulae to expectations

$$[r_s(e_i, e_j)] := \sigma(\mathbf{v}_s \cdot \mathbf{v}_{ij})$$

- Training objective:

$$\max_{\{\mathbf{v}_s\},\{\mathbf{v}_{ij}\}} \sum_{\mathcal{F} \in \mathfrak{F}} \log([\mathcal{F}])$$

- Can we do this for *any* propositional formulae?

$$[\mathcal{F}] = [r_1(e_i\ e_j) \wedge \neg r_2(e_i\ e_j) \Rightarrow r_3(e_i\ e_j)]$$

Differentiable Logic Formulae

$$[\mathcal{F}] = \begin{cases} \sigma(\mathbf{v}_s \cdot \mathbf{v}_{ij}) & \text{if } \mathcal{F} = r_s(e_i, e_j), \text{ i.e., facts} \\ 1 - [\mathcal{A}] & \text{if } \mathcal{F} = \neg\mathcal{A} \\ [\mathcal{A}] * [\mathcal{B}] & \text{if } \mathcal{F} = \mathcal{A} \wedge \mathcal{B} \end{cases}$$

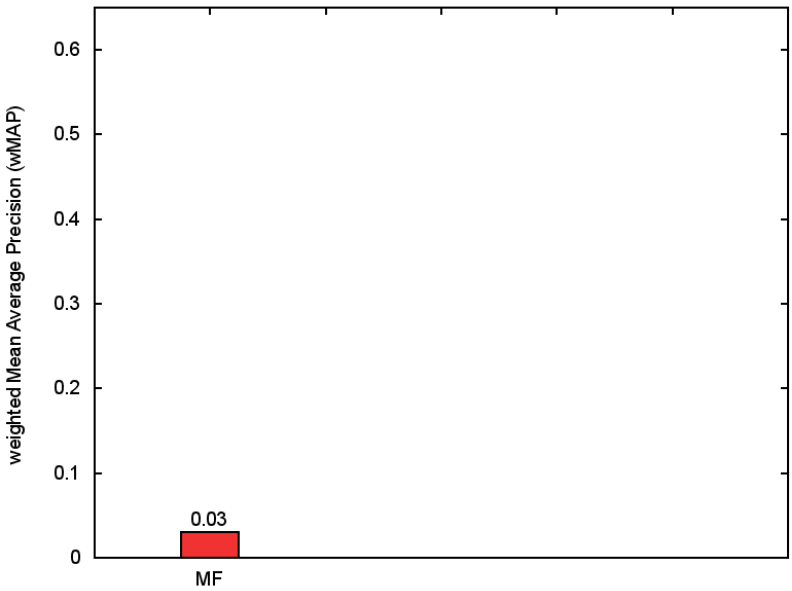
- From negation and conjunction we can build any propositional formula
  - Disjunction:
    $$[\mathcal{A} \vee \mathcal{B}] = 1 - (1 - [\mathcal{A}]) * (1 - [\mathcal{B}])$$
  - Implication:
    $$[\mathcal{A} \Rightarrow \mathcal{B}] = [\mathcal{A}]([\mathcal{B}] - 1) + 1$$
- Jointly maximize the log-likelihood of atomic and propositional formulae
  $$\max_{\{\mathbf{v}_s\},\{\mathbf{v}_{ij}\}} \sum_{\mathcal{F} \in \mathfrak{F}} \log([\mathcal{F}])$$

Grounding

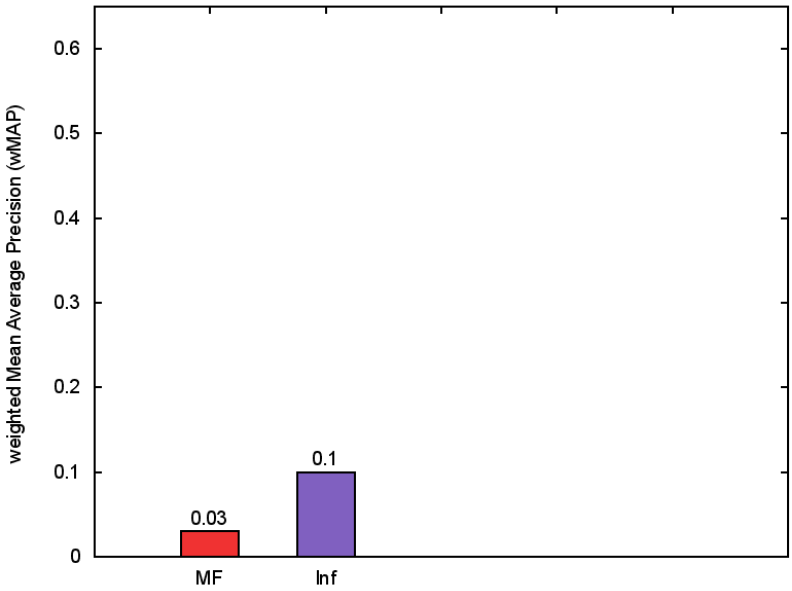
$$\forall x, y : r_s(x, y) \Rightarrow r_t(x, y)$$

- Grounding based on all observed facts for premise $r_s(e_i, e_j)$ and consequent $r_t(e_i, e_j)$
- Sample unobserved facts $r_s(e'_i, e'_j)$ and $r_t(e'_i, e'_j)$
- Add propositional formulae to the matrix factorization training objective
  - $[r_s(e_i, e_j) \Rightarrow r_t(e_i, e_j)]$
  - $[r_s(e'_i, e'_j) \Rightarrow r_t(e'_i, e'_j)]$

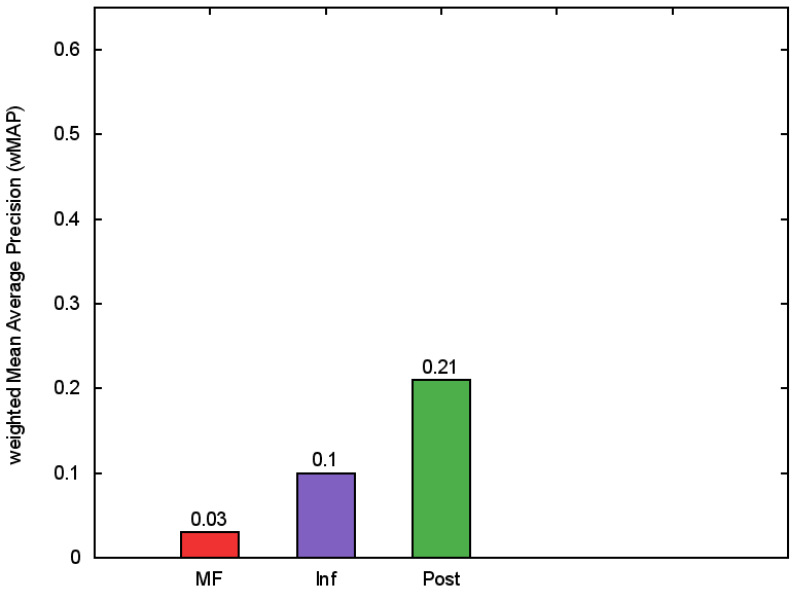
Zero-shot Relation Learning



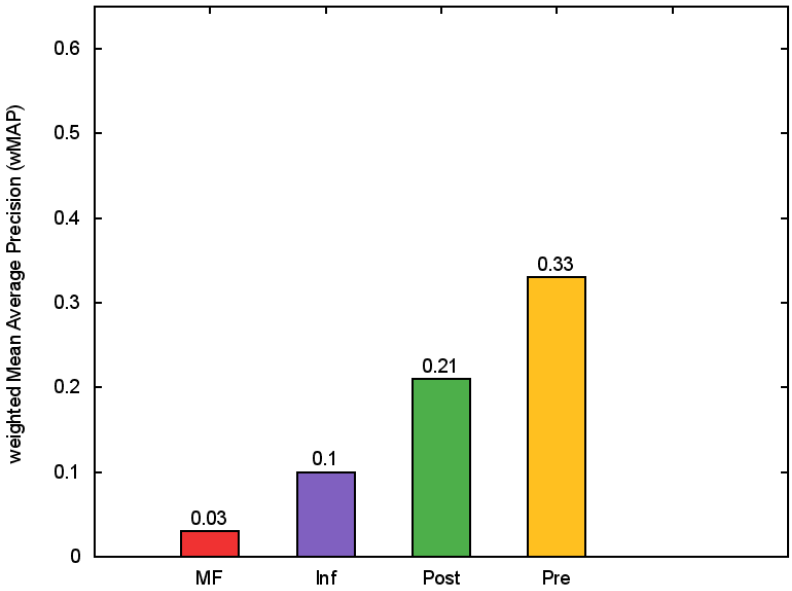
Zero-shot Relation Learning



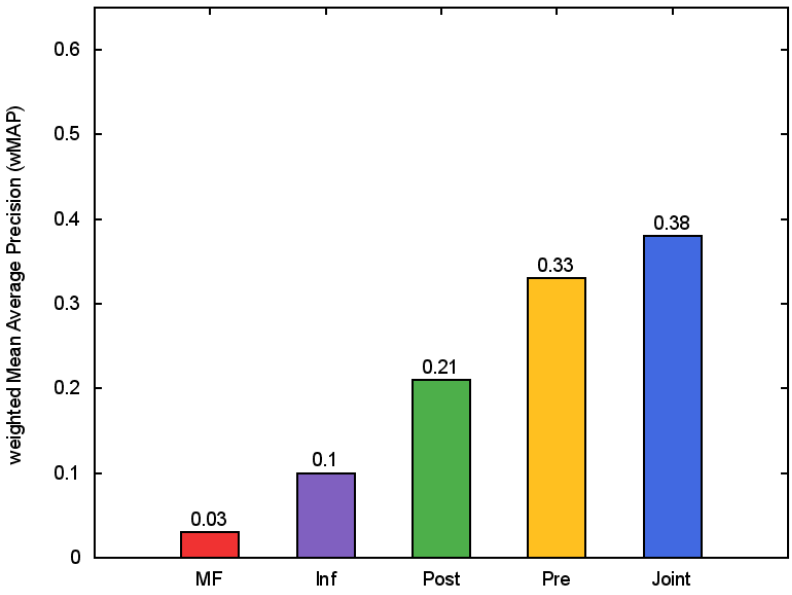
Zero-shot Relation Learning



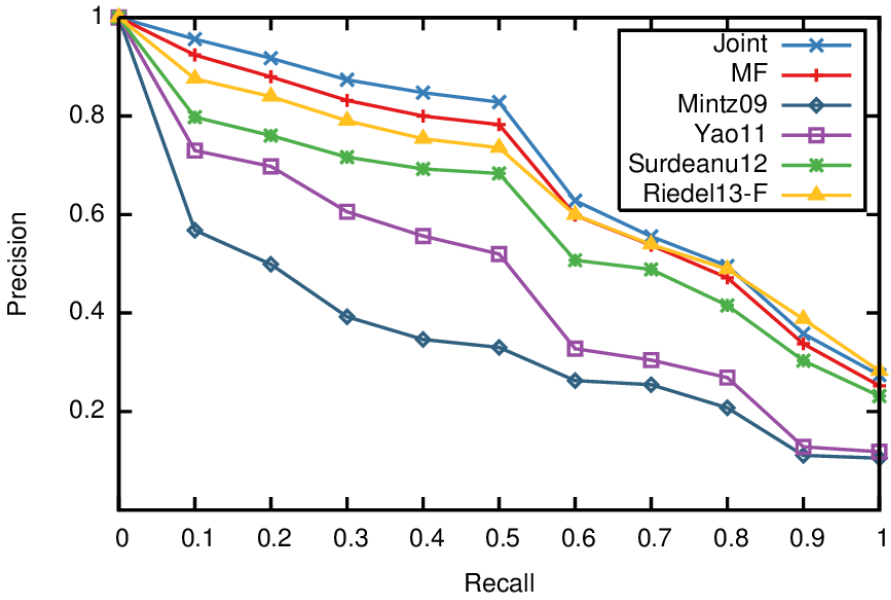
Zero-shot Relation Learning



Zero-shot Relation Learning



Full Dataset



Tensor Factorization

Tensors

- In many real-world data more then two variables are interacting
  - Subject, verb, object triplets (s, v, o)
  - Binary relation triplets (r_s, e_i, e_j)
  - Tag-recommendation: user, item, tag triplets (u, i, t)
  - User-movie-song-book recommendation: (u, m, s, b)
- 2D Matrix representation is an unnatural choice
- **Tensors** are the higher-order generalization of matrices
- Can be used to model N-way interactions

Notation

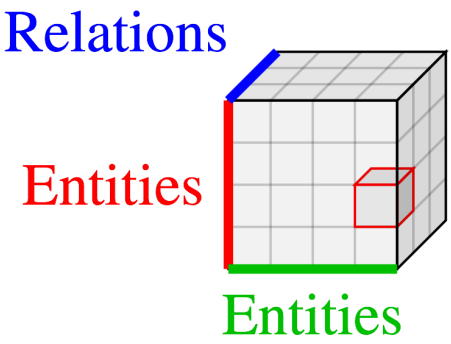
(Kolda and Bader, 2009)

- Vectors: boldface lower-case letters $\mathbf{v}$
- Matrices: boldface capital letters $\mathbf{M}$
- Tensors: Euler script letters $\mathcal{T}$
- Order-N tensor is a multidimensional array with N indices

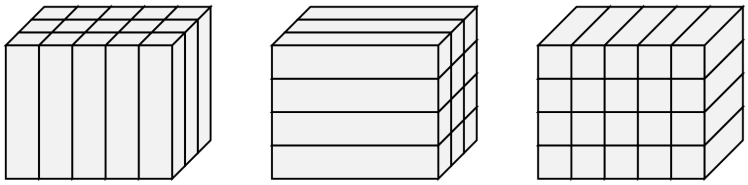
$$\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$$

Example Order-3 Tensor

- $\mathcal{T} \in \mathbb{R}^{|E| \times |E| \times |R|}$



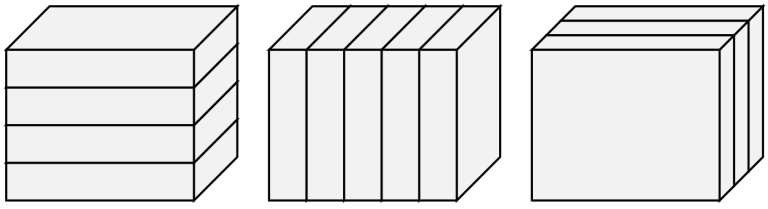
Order-3 Tensor Fibers



Mode-1 (column) fibers | Mode-2 (row) fibers | Mode-3 (tube) fibers

- $\mathbf{t}_{.jk}$ is used to denote the mode-1 **column fiber** when fixing mode 2 to j and mode 3 to k
- $\mathbf{t}_{i.k}$ mode-2 **row fiber**
- $\mathbf{t}_{ij.}$ mode-3 **tube fiber**

Order-3 Tensor Slices

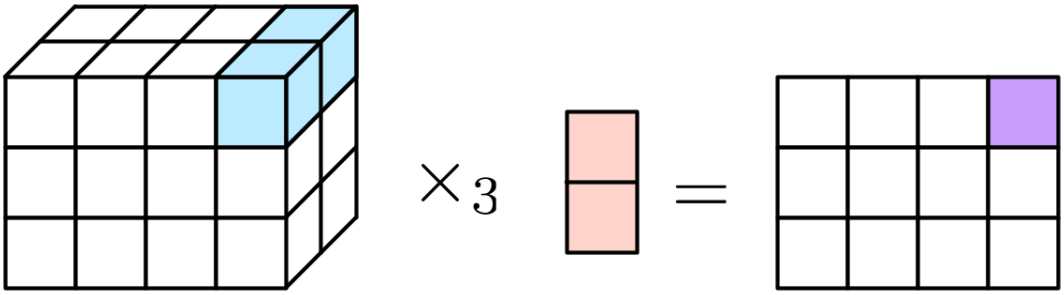


Horizontal slices | Vertical slices | Lateral slices

- $\mathbf{T}_{i..}$ denotes the **horizontal slice** when fixing the mode-1 index to i
- $\mathbf{T}_{.j.}$ **vertical slice**
- $\mathbf{T}_{..k}$ **lateral slice**

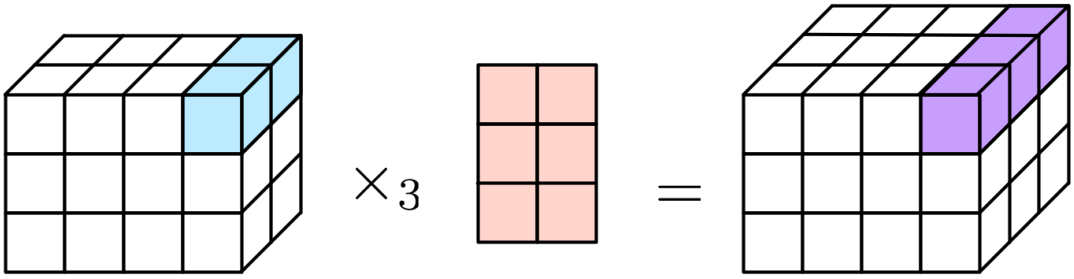
Tensor Operations

- Mode-N **tensor-vector product** $\mathcal{T} \times_N \mathbf{v}$
  - Dot product of each mode-N fiber in $\mathcal{T}$ with $\mathbf{v}$
  - $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times \cdots I_n}$
  - $\mathbf{v} \in \mathbb{R}^{I_N}$
  - $\mathcal{T} \times_N \mathbf{v} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times I_{N+1} \times \cdots I_n}$
  - Resulting tensor has $N - 1$ modes



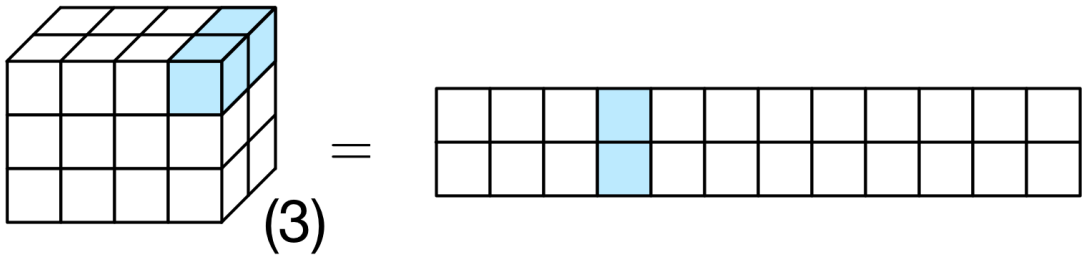
Tensor Operations

- Mode-N **tensor-matrix product** $\mathcal{T} \times_N \mathbf{M}$
  - Matrix-vector product of each mode-N fiber in $\mathcal{T}$ with $\mathbf{M}$
  - $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times \cdots I_n}$
  - $\mathbf{M} \in \mathbb{R}^{K \times I_N}$
  - $\mathcal{T} \times_N \mathbf{M} \in \mathbb{R}^{I_1 \times \cdots \times K \times \cdots I_n}$



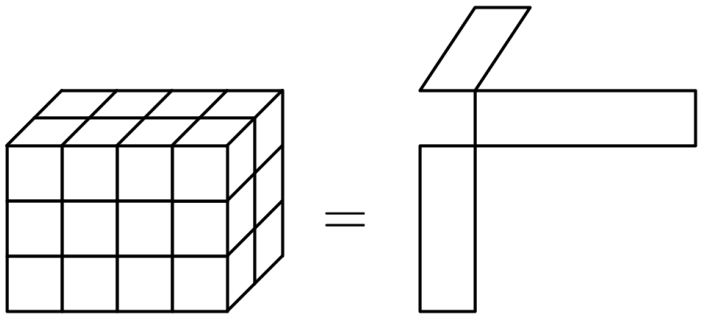
Tensor Operations

- Mode-N **matricization** $\mathbf{T}_{(N)}$
  - Arrange all mode-N fibers as columns



Tensor Rank

- **Rank-1 tensor**: outer product of three vectors:
  $\mathcal{X} = \mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}$



- Rank of a tensor $\mathcal{T}$: smallest number of rank-1 tensor that add up to $\mathcal{T}$

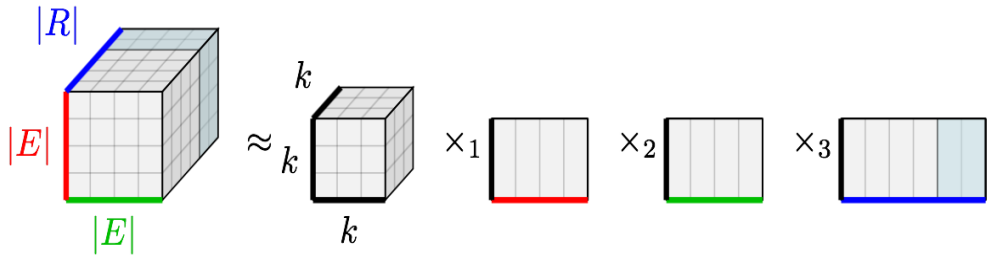
Tucker Decomposition

(Tucker, 1966)

- Given: $\mathcal{T} \in \mathbb{R}^{p \times q \times r}$
- $\mathcal{T} \approx \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$
  - $\mathcal{G}$ is called **core tensor**
  - Models linear interactions between all three variables
  - $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are called **loading matrices**
  - $\mathcal{G} \in \mathbb{R}^{N_1 \times N_2 \times N_3}, \mathbf{A} \in \mathbb{R}^{N_1 \times p}, \mathbf{B} \in \mathbb{R}^{N_2 \times q}, \mathbf{C} \in \mathbb{R}^{N_3 \times r}$

Example

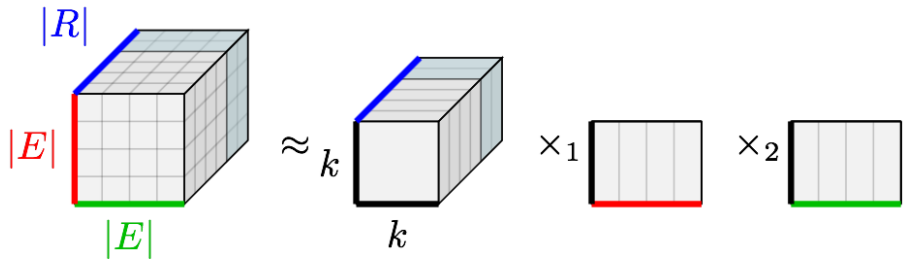
- $\mathcal{T} \approx \mathcal{G} \times_1 \mathbf{R} \times_2 \mathbf{E_1} \times_3 \mathbf{E}_2$
  - $\mathcal{T} \in \mathbb{R}^{|R| \times |E| \times |E|}, \mathcal{G} \in \mathbb{R}^{k \times k \times k}$
  - $\mathbf{R} \in \mathbb{R}^{|R| \times k}, \mathbf{E}_1 \in \mathbb{R}^{|E| \times k}, \mathbf{E}_2 \in \mathbb{E}^{|R| \times k}$



- Problem: in practice hard to estimate parameters
  - SGD expensive for core tensor $\mathcal{G}$
  - Tucker decomposition is not unique
- Often one mode is kept fixed (**Tucker2**)

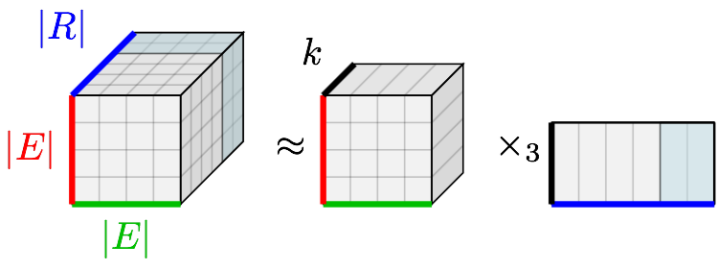
Tucker2 Decomposition

- Full Tucker decomposition is called **Tucker3**
  - Factorizes along all three modes
- **Tucker2** factorizes only along two modes
  - One of the loading matrices is the identity matrix
  - For instance: $\mathcal{T} \approx \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 I$
- Example:



Matrix Factorization as Tucker1 Decomposition

- **Tucker1** factorizes only along one mode
- Is an instance of matrix factorization
  - Learns vectors for pairs of variables of two modes
  - Two of the loading matrices are the identity
  - For instance: $\mathcal{T} \approx \mathcal{G} \times_1 I \times_2 I \times_3 \mathbf{C}$
- Example:



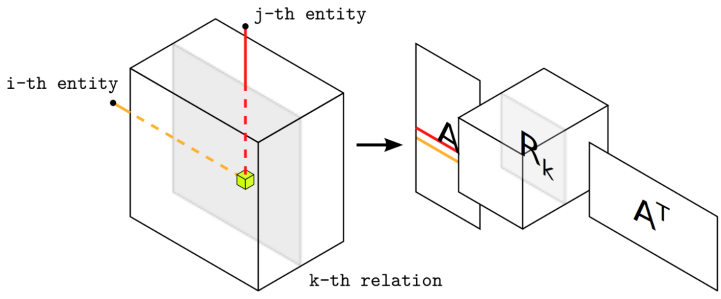
CANDECOMP/PARAFAC

(Carroll and Chang, 1970; Harshman, 1970)

- Approximate $\mathcal{T}$ with a sum of n rank-1 tensors
  $\mathcal{T} \approx \sum_{i=1}^{n} \mathbf{a}_i \otimes \mathbf{b}_i \otimes \mathbf{c}_i$
- Loss
  $\min_{\mathbf{a}_i \mathbf{b}_i \mathbf{c}_i} \|\mathcal{T} - \sum_{i=1}^{n} \mathbf{a}_i \otimes \mathbf{b}_i \otimes \mathbf{c}_i\|_F^2$

RESCAL

(Nickel et al, 2011; 2012)



RESCAL

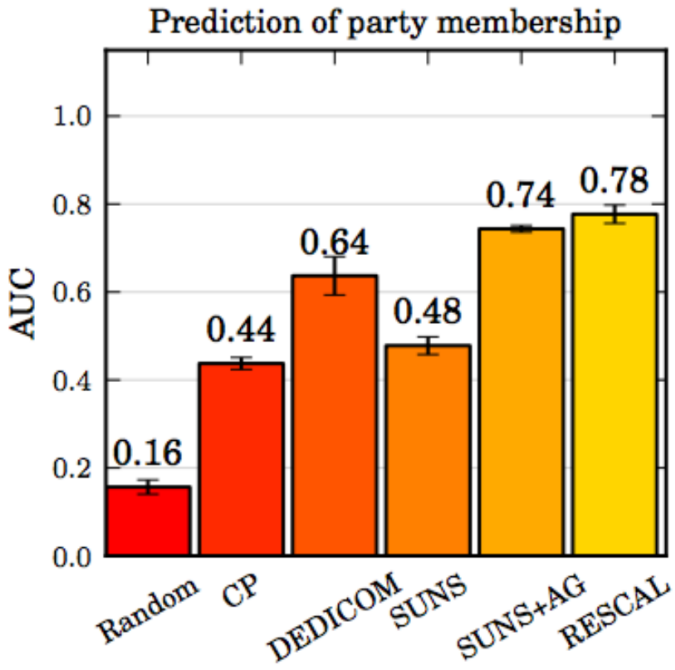
(Nickel et al, 2011; 2012)

- Is an instance of Tucker2: $\mathcal{T} \approx \mathcal{R} \times_1 \mathbf{A} \times_2 \mathbf{A}$
  - $\mathcal{R}$ holds one slice $\mathbf{R}_s \in \mathbb{R}^{k \times k}$ for each relation s
  - $\mathbf{A} \in \mathbb{R}^{|E| \times k}$ is the dictonary-matrix for all entity embeddings
  - Symmetry assumption: an entity has the same embedding whether used as first or second argument
- Per relation view (s slices of $\mathcal{T}$ and $\mathcal{R}$): $\mathbf{T}_s \approx \mathbf{A}\mathbf{R}_s\mathbf{A}^T$
- Loss function

$$\min_{\mathbf{R}_s, \mathbf{A}} \sum_s \|\mathbf{T}_s - \mathbf{A}\mathbf{R}_s\mathbf{A}^T\|_F^2 + \lambda_R \|\mathbf{R}_s\|_F^2 + \lambda_A \|\mathbf{A}\|_F^2$$

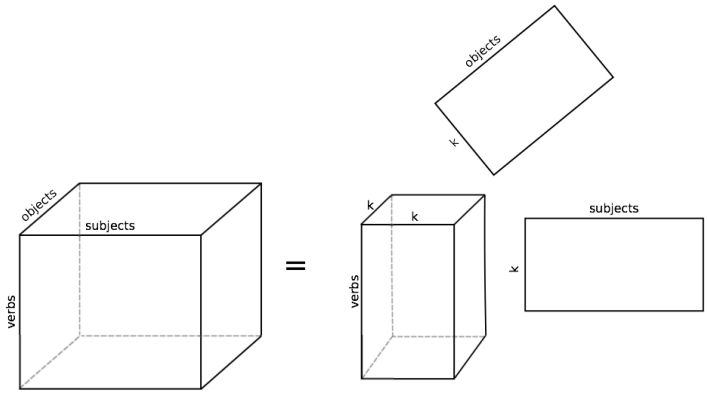
- Optimized using Alternating Least Squares

Results



Factorization Model of Semantic Compositionality

(Van de Cruys et al, 2013)



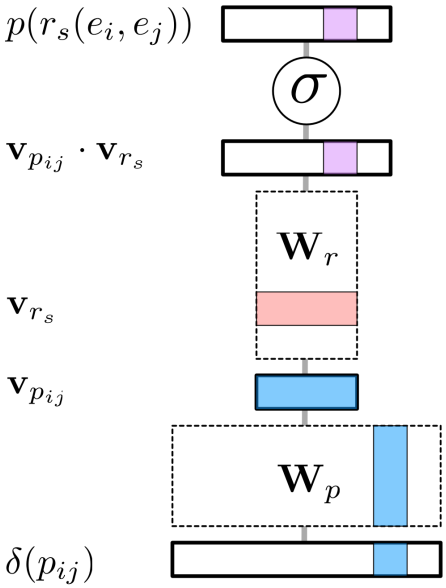
Factorization Model of Semantic Compositionality

(Van de Cruys et al, 2013)

- Captures three-way interaction of subject, verb, object triplets
- Instance of Tucker2 decomposition
  - $\mathcal{T} \approx \mathcal{V} \times_1 \mathbf{N} \times_2 \mathbf{N}$
  - $\mathbf{N} \in \mathbb{R}^{|N| \times k}$ is the dictionary matrix of noun embeddings
  - $\mathcal{V}$ holds slice $\mathbf{V}_s \in \mathbb{R}^{k \times k}$ for each verb s

Relationship to Neural Networks

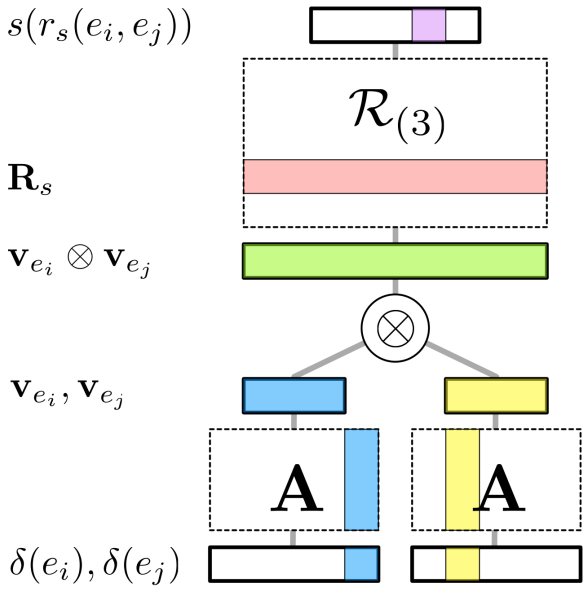
Matrix Factorization: $\mathbf{T} \approx \sigma(\mathbf{W}_p^t \mathbf{W}_r)$



Relationship to Neural Networks

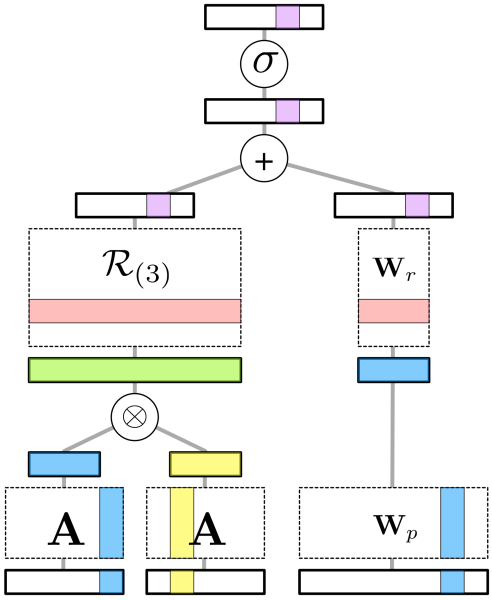
(Nickel, 2015)

RESCAL: $\mathcal{T} \approx \mathcal{R} \times_1 \mathbf{A} \times_2 \mathbf{A}$



Combining Matrix and Tensor Factorization

(e.g. Chang et al. 2014, Singh et al. 2015)

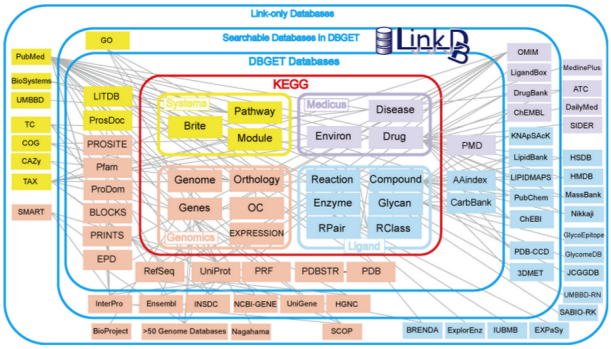


Collective Matrix Factorization

Arbitrary Database Factorization

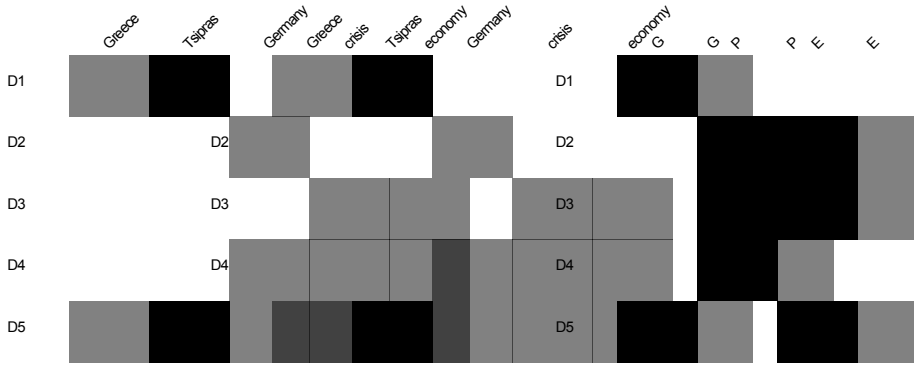
Objective: A probabilistic model for any database

Example: Health data

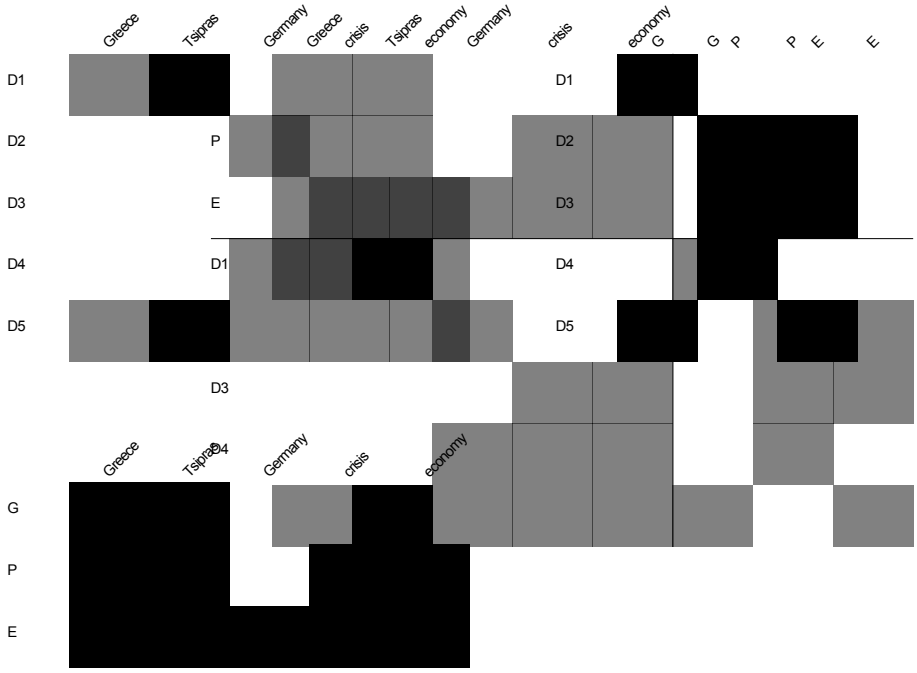


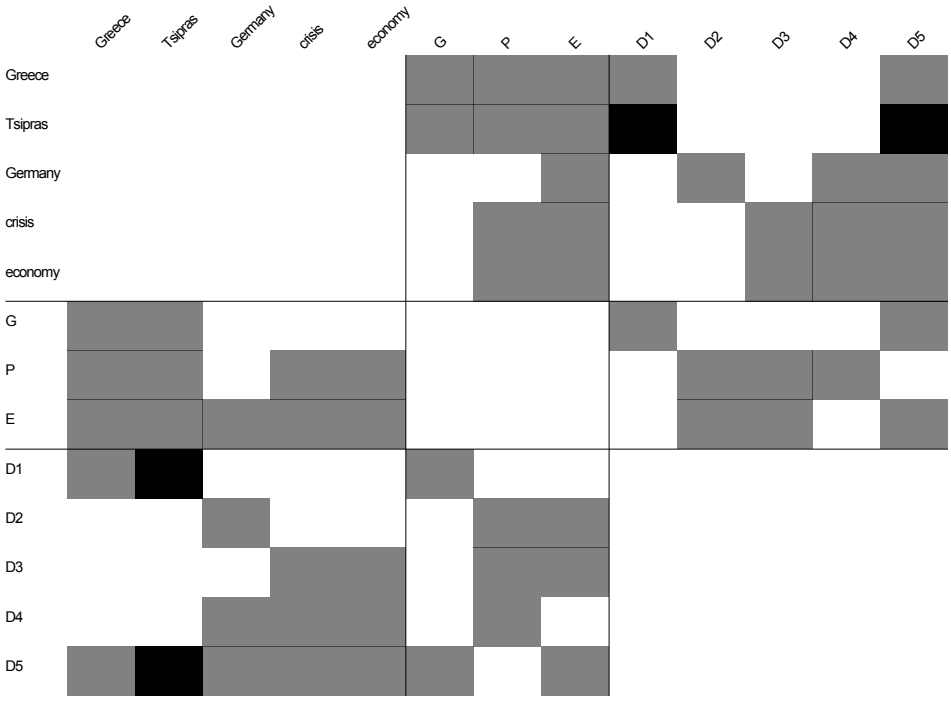
?

Modeling Two Relations



Third Relation: Feature Labelling





Modeling Two Relations

Assume we have 2 relations:

1. Document - Word matrix $Y^{(1)} \in (\mathfrak{R} \cup \{?\})^{n \times F}$
2. Document - Label matrix $Y^{(2)} \in \{0, 1, ?\}^{n \times K}$

Representing is a a 2-slices tensor is overly complicated and not needed. A simple and effective way to represent it is by concatenating the two matrices into a big $n \times (F + K)$ matrix:

$$Y := [Y^{(1)} \; Y^{(2)}]$$

We can estimate a probabilitic model on Y by factorizing it.

- Handles missing data in the input
- Can work with partially labelled documents
- Takes advantage of unlabelled data

Concatenation

- Document - Word matrix $Y^{(1)} \in (\mathfrak{R} \cup \{?\})^{N \times F}$
- Document - Label matrix $Y^{(2)} \in \{0, 1, ?\}^{N \times K}$
- Word - Label matrix $Y^{(3)} \in \{0, 1, ?\}^{K \times F}$

Naive concatenation:

$$Y := \begin{bmatrix} Y^{(3)} & [?]_{K \times K} \\ Y^{(1)} & Y^{(2)} \end{bmatrix}$$

We can estimate a probabilitic model on Y by factorizing it. This additional relation enables feature labelling and increase prediction capabilities when similar rare words share similar labels

Symmetric Block Matrix

- Document - Word matrix $Y^{(1)} \in (\mathfrak{R} \cup \{?\})^{N \times F}$
- Document - Label matrix $Y^{(2)} \in \{0, 1, ?\}^{N \times K}$
- Word - Label matrix $Y^{(3)} \in \{0, 1, ?\}^{F \times K}$

A $(n + F + K) \times (n + F + K)$ symmetric block-matrix:

$$Y := \begin{bmatrix} [?]_{F \times F} & Y^{(3)T} & Y^{(1)T} \\ Y^{(3)} & [?]_{K \times K} & Y^{(2)T} \\ Y^{(1)} & Y^{(2)} & [?]_{N \times N} \end{bmatrix}$$

Let factorize it!

Collective Matrix Factorization

We have T types of entity with embeddings $\mathbf{U} = (\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(T)})$ and we observe relations between all possible entity types $\mathbf{Y} = (\mathbf{Y}^{(1,1)}, \mathbf{Y}^{(1,2)}, \cdots, \mathbf{Y}^{(T,T)})$

General Loss functions $\ell_{tt'}$:

$$\mathcal{L}(\mathbf{U}) = \sum_{t=1}^{T} \sum_{t<t'<T} \ell_{tt'}(\mathbf{U}^{(t)}\mathbf{U}^{(t')T}; \mathbf{Y}^{(t,t')}\|)$$

Let's try SGD!

```
val entityNames = Seq(
Seq("Greece","Tsipras","Germany","crisis","economy"),
Seq("G","P","E"),
Seq("D1","D2","D3","D4","D5"))
val n1 = entityNames(0).length
val n2 = entityNames(1).length
val n3 = entityNames(2).length
val tuples31 = toTuples(parseMatrix(
"""1 2 0 0 0
0 0 1 0 0
0 0 0 1 1
0 0 1 1 1
1 2 1 1 1""".stripMargin), n1 + n2, 0)
val tuples32 = toTuples(parseMatrix(
"""1 0 0
0 1 1
0 1 1
0 1 0
1 0 1""".stripMargin), n1 + n2, n1)
val tuples21 = toTuples(parseMatrix(
"""1 1 0 0 0
1 1 0 1 1
1 1 1 1 1""".stripMargin), n1, 0)
val T1 = new Table(l2_type, tuples31)
val T2 = new Table(l2_type, tuples32)
val T3 = new Table(l2_type, tuples21)
```

?

```
def optimizeCMF(database: Seq[Table], tables_weights: Seq[D
Int, iters: Int, step_size:Double = 0.01, initScale:Double = 1.0):
Seq[Vect] = {
val U = initial_embeddings(num_of_entities(database), rk,
initScale) // N*rank embedding matrix
for (it <- 0 until iters) {  // Loop over SGD iterations
// Sampling an observation
val relation = sample_discrete(tables_weights)
val table = database(relation)
val observation = sample_seq(table.observations)
//Gradient Step
val (i, j, gold) = observation
val v1 = U(i).copy
val v2 = U(j).copy
val dloss = loss_gradient(table.loss_type, v1 dot v2, gold)
U(i) -= v2 * step_size * dloss
U(j) -= v1 * step_size * dloss
}
U //output the embedding matrix}
}
```

?

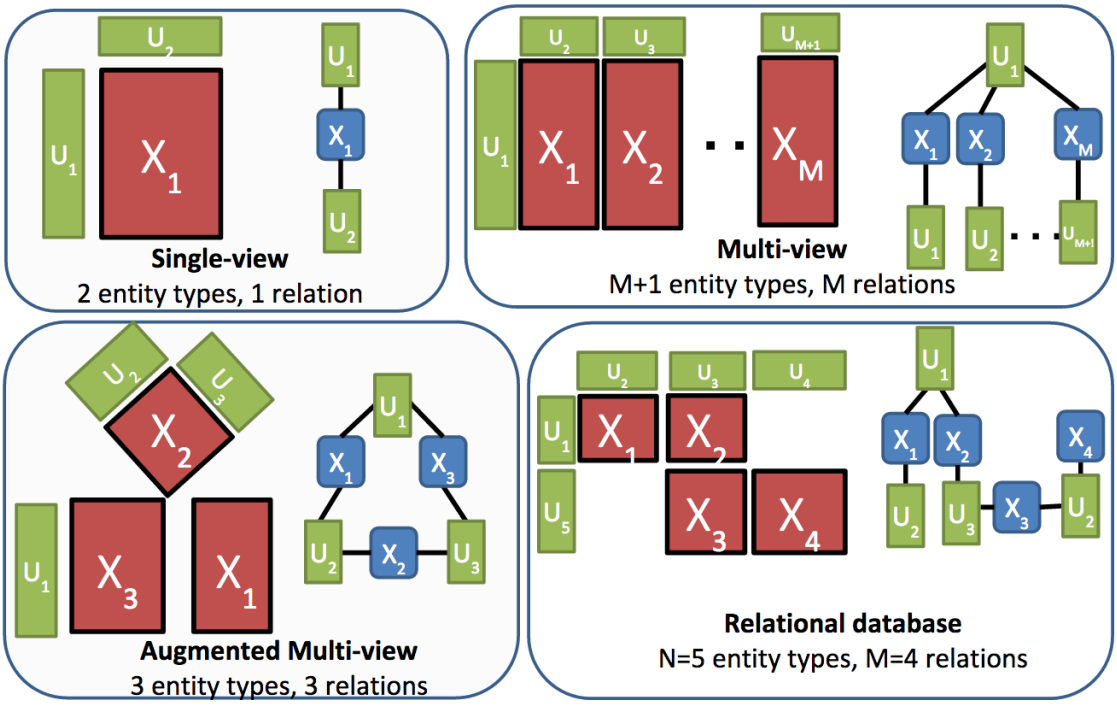
#Result

```
val U = optimizeCMF0(Seq(T1, T2, T3), Seq(.3, .3, .4), 3, 20000,.0
showCMF(U)
```

?

#generalization

Generalization to Arbitrary Databases

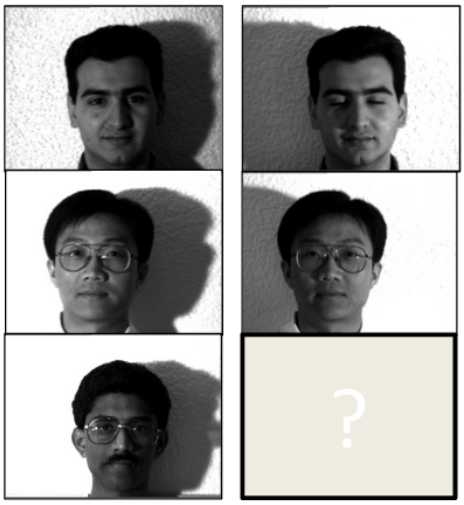


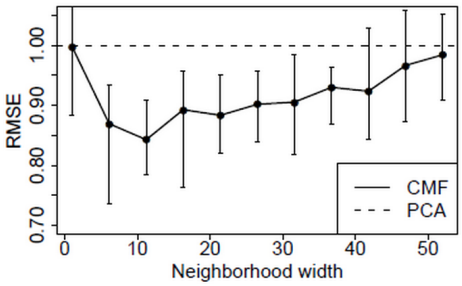
Example: Augmented Multi-View

Objective: Predict one view given the other

Pixel similarity should help. We create a binary matrix linking pixels.

- Relation 1: pixel intensity value in the first image (real)
- Relation 2: pixel intensity value in the second image (real)
- Relation 3: Relative similarity between pixels (binary)





Example: Social Media Data (Flickr)

Bayesian Modelling

CMF: many regularization parameters.

Tuning is painful

Bayesian learning: principled automatic tuning

1. Choice of a prior $P(\mathbf{U}, \mathbf{V})$
2. Find (approximate!) the posterior $P(\mathbf{Y}|\mathbf{U}, \mathbf{V})$

Algorithms: - Sampling - Gibbs sampling: Salakhutdinov et al., 2008 - HMC: Mohamed et al., 2009 - Variational Inference - Variational Bayes: Raiko et al., 2008
- Expectation Propagation: Stern et al., 2009

Bayesian Matrix Factorization

Independent Gaussian Priors: $P(\mathbf{U}, \mathbf{V}) \propto e^{-\lambda(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2)}$

Homoscedastic Gaussian likelihood:

$$P(\mathbf{Y}|\mathbf{U}, \mathbf{V}) \propto e^{-\frac{1}{\sigma^2}(\|\mathbf{U}\mathbf{V}^T - \mathbf{Y}\|_F^2)}$$

Algorithms for Bayesian Matrix Factorization

Block Gibbs sampling

- Sample $\mathbf{u_i}|\mathbf{V}, \mathbf{Y}_{i:}$ for $i = 1, \cdots, N$
- Sample $\mathbf{v_j}|\mathbf{U}, \mathbf{Y}_{:j}$ for $j = 1, \cdots, M$

Variational Bayes : Fully factorized approximation:

$$Q(\mathbf{U}, \mathbf{V}) = \prod_{i=1}^{N} Q(\mathbf{u_i}) \prod_{j=1}^{M} Q(\mathbf{v_j})$$

- Compute $Q(\mathbf{u}_i)$ based on $\mathbf{V}$ and $\mathbf{Y}_{i:}$
- Compute $Q(\mathbf{v}_j)$ based on $\mathbf{U}$ and $\mathbf{Y}_{:j}$

Bayesian Collective Matrix Factorization

We have T types of entity with embeddings $\mathbf{U} = (\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(T)})$ and we observe relations between all possible entity types $\mathbf{Y} = (\mathbf{Y}^{(1,1)}, \mathbf{Y}^{(1,2)}, \cdots, \mathbf{Y}^{(T,T)})$

Independent Gaussian Priors: $P(\mathbf{U}) \propto e^{-\sum_{t=1}^{T} \lambda_t (\|\mathbf{U}^{(t)}\|_F^2}$

Per-relation Homoscedastic Gaussian likelihood:

$$P(\mathbf{Y}|\mathbf{U}) \propto \prod_{t=1}^{T} \prod_{t<t'<T} e^{-\frac{1}{\sigma_{tt'}^2}(\|\mathbf{U}^{(t)}\mathbf{U}^{(t')T} - \mathbf{Y}^{(t,t')}\|_F^2)}$$

Algorithms for Bayesian CMF

Block Gibbs sampling

- For each type $t = 1, \cdots, T$
  - Sample $\mathbf{u}_\mathbf{i}^{(\mathbf{t})} | \mathbf{U}^{(-\mathbf{t})}, \mathbf{Y}$ for $i = 1, \cdots, N_t$

Variational Bayes : Fully factorized approximation:

$$Q(\mathbf{U}) = \prod_{t=1}^{T} \prod_{i=1}^{N_t} Q(\mathbf{u}_i^{(t)})$$

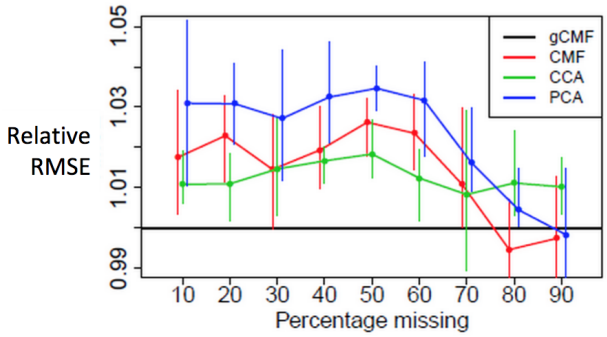
- Compute $Q(\mathbf{u}_i^{(t)})$ based on $\mathbf{U}^{(-t)}$ and $\mathbf{Y}$ -

Now we can maximize the likelihood with respect to $\sigma_{tt'}$ and λ_t with t, t'

Gene Expression Data Experiment

40 patients with breast cancer

- *Views*: 2 measurements of 4287 genes.
- *Task*: predicting random missing entries



Conclusion to Collective Matrix Factorization

Collective Matrix Factorization and Tensor Factorization:

- Learn §more by fusing multiple databases
- flexible and generic model for relational data
- Bayesian learning : automatic tuning of parameters and complexity

Collective Matrix Factorization vs. Tensor Factorization:

- CMF is easier because it deals only with matrices
- CMF = typed data. Tensor = multi-relational
- Augmented multi-view: common setup

Discriminative Factorial Models

Discriminative Factorial Models

Factorization is an efficient way of reducing the effective number of parameters in discriminative/predictive models

We introduce:

- Factorization Machines
- Multi-Task/Multi-Label Learning with Matrix Factorization
- Structured Prediction with Factorized Parameters

Factorization Machines

Binary Matrix Factorization as a Logistic Regression Matrix $Y \in \Re^{n,d}$. For any K:

$P(\mathbf{Y}_{ij} = 1|i,j) = \sigma(\langle \mathbf{U}_{i:}, \mathbf{V}_{j:} \rangle)$ where $\mathbf{U} \in \Re^{N \times K}$ and $\mathbf{V} \in \Re^{M \times K}$

Is equivalent to:

$$P(y_t = 1|x_t = (i,j)) = \sigma(\langle \mathbf{\Theta}, \Phi(x_t) \rangle)$$

u.c. rank($\mathbf{\Theta}$) = K

- $\Phi(x_t) = \mathbf{e}_i \otimes \mathbf{e}_j$: standard MF
- $\Phi(x_t) = [1; \mathbf{e}_i] \otimes [1; \mathbf{e}_j]$: MF with row and columns bias

→ called **factorization machine** [Rendle 2010]

Reduced Rank Regression

Multiple Linear Regression:

- d Inputs: $\mathbf{X} \in \Re^{n \times d}$
- K Outputs: Matrix $\mathbf{Y} \in \Re^{n \times K}$

Linear regression:

$$\min \sum_{k=1}^{K} \sum_{i=1}^{n} (y_{ik} - x_{i:}^T \theta_k)^2 = ||\mathbf{Y} - \mathbf{X}\mathbf{\Theta}||_F^2$$

- Parameters: Matrix $\mathbf{Y} \in \Re^{n \times K}$

Reduced Rank Regression assumes $\mathbf{\Theta} = \mathbf{U}\mathbf{V}^T$

Multi-Task Learning

- Inputs: d feature $\phi(x_i) \in \Re^d$
- Outputs: K labels $\mathbf{y}_i \in \{0, 1\}^K$

$$P(\mathbf{y}_i|x_i) = \prod_{k=1}^{K} \sigma(\phi(x_i)\mathbf{U}\mathbf{v}_{k:}^T)$$

Can be interpreted as label embedding: $\mathbf{v}_{k:}$ is the R-dimensional embedding of the k-th label.

Structured Prediction

Goal: Predict $y^* = \arg\max_{y \in T(\mathbf{x})} S(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$

- $\mathbf{y}$ is a structured output
- $S(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}) =< \mathbf{\Phi}(\mathbf{x}, \mathbf{y}), \mathbf{\Theta} >$ is a score function
- $T(\mathbf{x})$ is all possible structures

In NLP $\mathbf{\Theta}$ is often a vector, but it can be a matrix or tensor.

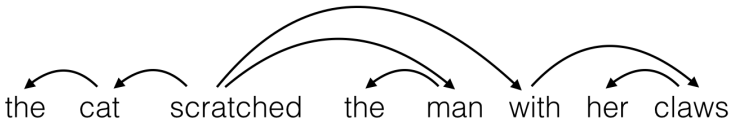
Examples of NLP Structured Prediction

Lei et al.: Low-Rank Tensors for Scoring Dependency (ACL 2014)

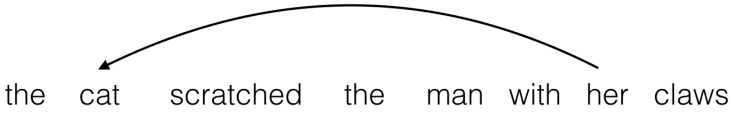
Sequence Tagging: y* is a sequence of tags

DT	NN	VB	DT	NN	PP	PR	NN
the	cat	scratched	the	man	with	her	claws

Dependency Parsing: y* is a directed acyclic graph



Coreference Resolution: y* is a set of coreference chains



Standard Vector Features

- The score function S is the product of two vectors:

$$S(x,y;\Theta) = \langle \Theta, \phi(x,y) \rangle$$

  - A feature function ϕ, a sparse vector of feature counts, derived from a set of manually-crafted feature templates:

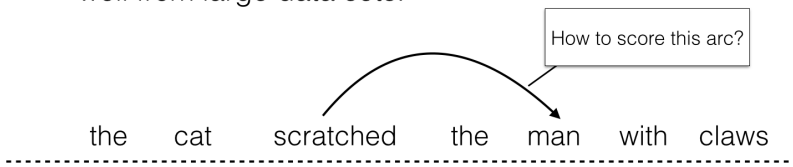
$\phi(x,y) =$ | 0 | 0 | 1 | … | 1 |

  - And a dense parameter vector:

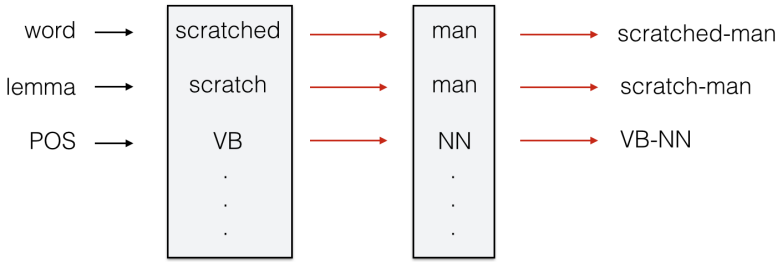
$\Theta =$ | 0.6 | 1.4 | 0.3 | … | 0.1 |

High Order Features

- Simple features can be informative and estimated well from large data sets:



- Concatenations of simple, single-token features:



Difficulties of Manual Feature Selection

Few Templates:

- Poor Performance

Many Templates:

- High Performance $\Longrightarrow$ Many Parameters
- Useful feature interactions hard to know a priori

Alternatives? Automatic Feature Selection:

- Effective, but "messy" (Zhao, 2009)
- Computationally expensive

Solution: Learn templates from data using matrix factorization

Step 1: Identify Simple Features

Feature vectors for each type of information:

- ϕ_{head}, vector $\in \mathfrak{R}^n$ for head token
- ϕ_{child}, vector $\in \mathfrak{R}^n$ for child token
- ϕ_{arc}, vector $\in \mathfrak{R}^n$ for arc information

Step 2: Define Composite Features

- $\phi_{head} \otimes \phi_{child}$, matrix $\in \mathfrak{R}^{n \times n}$
- $\phi_{arc} \otimes \phi_{child}$, matrix $\in \mathfrak{R}^{n \times n}$
- $\phi_{head} \otimes \phi_{arc}$, matrix $\in \mathfrak{R}^{n \times n}$
- $\phi_{head} \otimes \phi_{child} \otimes \phi_{arc}$, matrix $\in \mathfrak{R}^{n \times n \times n}$

Step 3: Formulate Model Parameters Tensor

A tensor $\mathfrak{R}^{n \times n \times d}$, describes the concatenation of all three feature vectors, so replace Θ with tensor A:

$$S(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}) = < \phi_{head} \otimes \phi_{child} \otimes \phi_{arc}, \mathbf{\Theta} >$$

Can be huge! A better option? Low rank approximation. Calculate A as:

$$\mathbf{\Theta} = \mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W}$$

where U, V, and W are dense low-dimensional representations $\in \mathfrak{R}^{n \times r}$

Learning

Training Objective:

$$C \sum_i \eta_i + \|U\|_F^2 + \|V\|_F^2 + \|W\|_F^2 + \|A\|_F^2$$

Non-convex optimization with regularization

Can optimize with variant of Passive Aggressive (Crammer, 2006)

Learning

Online method tailored for tensors [Lei et al., 2014]

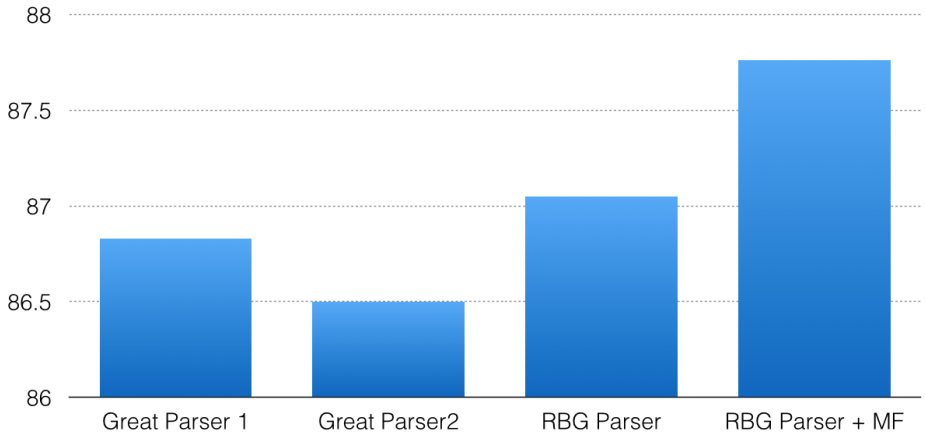
Iterate over data and for each instance, update unary feature weights θ and choose one of the feature tensors as follows (assuming U was chosen):

- $\theta^{(t+1)} = \theta^{(t)} + \Delta\theta$
- $\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} + \Delta\mathbf{U}$

With sub-problem (solvable with closed-form solution):

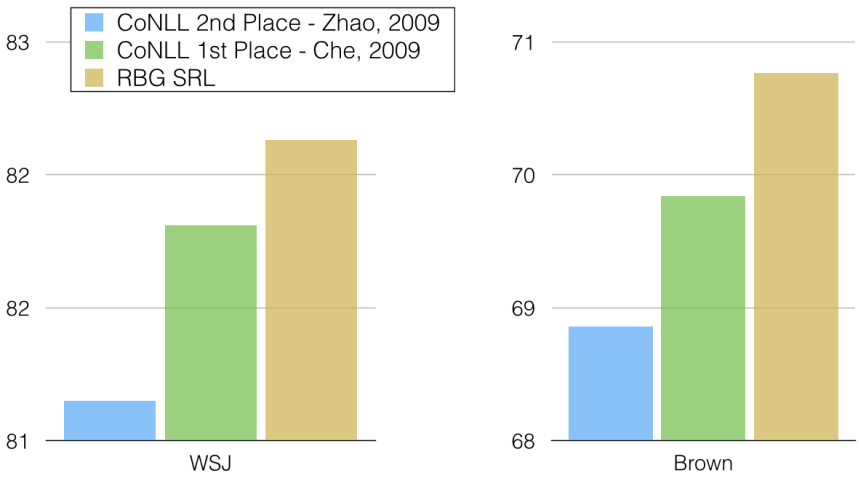
$$\min_{\Delta\theta, \Delta\mathbf{U}} \frac{1}{2}\|\Delta\theta\|_F^2 + \frac{1}{2}\|\Delta\mathbf{U}\|_F^2 + C\eta_i$$

Performance - Parsing



- RBG + MF model trained with rank $r = 50$ tensors
- Results averaged over 14 languages (CoNLL data)

Performance - SRL



- Baselines have many features and automatic feature selection
- Outperforms previous best system across many languages

Conclusion

- **Discriminative** models can be parameterized with matrices or tensors
- **Low-rank** asumption enables **parameter sharing**
  - **Multi-task** learning: corresponds to label-embedding
  - **Vanilla matrix factorization** can be expressed as a $N \times M$-dimensional **logistic regression with rank constraint**
  - **structured prediction** with low rank parameters can improve performance over heavily engineered state-of-the-art systems

Convexification

Convexification

Matrix $Y \in \mathfrak{R}^{n,d}$. For any $K \in \mathbb{N}$,

(1) $\min_{\mathbf{U}\in\mathfrak{R}^{N\times K},\mathbf{V}\in\mathfrak{R}^{M\times K}} \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F$

Is equivalent to:

(2) $\min_{\mathbf{X}\in\mathfrak{R}^{N\times M},\mathtt{rank}(\mathbf{X})\leq K} \|\mathbf{Y} - \mathbf{X}\|_F$

Which is relaxed into:

(3) $\min_{\mathbf{X}\in\mathfrak{R}^{N\times M},\|\mathbf{X}\|_*\leq K} \|\mathbf{Y} - \mathbf{X}\|_F$

(3) is convex, (2) is not

What Do We Gain By Making the Problem Convex?

We have access to all the convex optimization tools!

- Theoretical guarantees
- More importantly, new algorithms

Theory

- Convergence to the global optimum, no need for smart initialization
- Speed of convergences
- Polynomial time guarantees for a fixed accuracy

New Algorithms

- Proximal methods (ISTA and FISTA)
- Frank-Wolfe (Conjugate Gradient) algorithms
- Augmented Lagrangian approaches (ADMM, splitting methods)
- Stochastic variants (e.g. SAG)

Combining Regularizations

Convex penalties can be added to get better models

- Sparse plus low-rank

  $\min_{\mathbf{S},\mathbf{L}} \ell(\mathbf{S}+\mathbf{L};\mathbf{Y}) + \lambda\|\mathbf{S}\|_1 + \mu\|\mathbf{L}\|_*$

  Sum of trace norms

  $\min_{\mathbf{L}_1,\mathbf{L}_2} \ell(\mathbf{X};\mathbf{Y}) + \lambda_1\|\mathrm{blk}_1(\mathbf{X})\|_* + \lambda_2\|\mathrm{blk}_2(\mathbf{X})\|_*$

e.g.: Convex Cannonical Correlation Analysis

Very popular in image processing ==> should also be useful in NLP!

Some Pointers to Convex Factorization

- **2001**: Fazel, Hindi & Boyd. "A rank minimization heuristic with application to minimum order system approximation.
- **2009**: Candès & Recht. "Exact matrix completion via convex optimization.".
- **2009**: Wright, Ganesh, Rao, Peng & Ma. "Robust principal component analysis".
- **2009**: Hsu, Kakade and Zhang. A Spectral Algorithm for Learning Hidden Markov Models
- **2011**: Tamioka & Suzu. Statistical performance of convex tensor decomposition.
- **2013**: Bouchard, Guo & Yin. Convex Collective Matrix Factorization.
- **2014**: Bailly, Carreras & Quattoni: Unsupervised Spectral Learning of Finite-state Transducers

Application: Spectral Learning in NLP

A promising application of convex factorization: **polynomial-time** learning of

- HMMs
- WFSAs
- Grammars
- Even NMF!

Idea: factorization of redundant moment matrices.

Example: Empirical Hankel Matrix of a String

Matrix of counts of all possible prefixes and suffixes

$$S = \left\{ \begin{array}{c} aa,\ b,\ bab,\ a, \\ b,\ a,\ ab,\ aa, \\ ba,\ b,\ aa,\ a, \\ aa,\ bab,\ b,\ aa \end{array} \right\} \longrightarrow \hat{\mathbf{H}} = \begin{array}{c} \\ \epsilon \\ a \\ b \\ ba \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \left[\begin{array}{cc} .19 & .25 \\ .31 & .06 \\ .06 & .00 \\ .00 & .13 \end{array} \right] \end{array}$$

Summary of Convex Factorization

- **Trace norm** is great
  - from intractable to tractable problems
- **Spectral learning** combines well with trace norm
  - Learning latent variable with computational guarantees

Summary

Before the break

1. Matrices/Tensor = natural concepts for relations
2. Low-Rank = natural regularization
3. Non-negativity => Clustering
4. Binary loss: much better than squared loss

After the break

1. Prior knowledge included using *logic*-based loss models
2. Multi-relational learning
   - Tensors: *many* different possible factorizations
   - Collective Matrix Factorization for *typed* data
3. Discriminative factoriation
   - Reduced rank regression/Multi-task learning
   - Factorization machines
   - Structured prediction
4. Convexification
   - The *trace norm* is a real breakthrough
   - Theoretical guarantees, new algorithms, structured regularization

Take-home messages

NLP is about learning relations. Low-rank factorization simply *works*

- in *theory*: well understood model with nearly **linear-time** complexity
- in *practice*: many **large scale** examples

The real reason why it works:

Replaces **combinatorics** by **linear algebra**

Toolkits

- R Package for Collective Matrix Factorization (Klami) https://cran.r-project.org/web/packages/CMF
- MyMediaLite (Ganter & Rendle): http://mymedialite.net/news/index.html
- scikit-tensor (Nickel): http://github.com/mnick/scikit-tensor
- NMF-MATLAB (Yi & Ngom): https://sites.google.com/site/nmftool/
- Theano (Bengio et al.): http://deeplearning.net/software/theano/
- Wolfe (Riedel et al.): http://www.wolfe.ml
  - used in this presentation via Moro by Sameer Singh

I CAN READ IT

Annotated Bibliography

Kolda, Tamara G., and Brett W. Bader. "Tensor decompositions and applications." SIAM review 51.3 (2009): 455-500.
Extensive overview of the topic, we adopted their notation in the tutorial

Goldberg, Andrew B., Zhu, Xiaojin, Recht, Benjamin, Xu, Jun-Ming, Nowak, Robert. "Transduction with Matrix Completion: Three Birds with One Stone." In Advances in Neural Information Processing Systems (NIPS) 24. 2010.
First work to formulate semi-supervised multi-task learning with missing data explicitly as matrix completion

Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 8 (2009): 30-37.
Description of SGD and ALS algoirthms for matrix factorization

Wang, Quan, et al. "Regularized latent semantic indexing." Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 2011.
Introduces regularized latent semantic indexing

Lee, Daniel D., and H. Sebastian Seung. "Algorithms for non-negative matrix factorization." Advances in Neural Information Processing Systems (NIPS) 15. 2001.
Multiplicative updates algorithm for NMF

Kim, Hyunsoo, and Haesun Park. "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method." SIAM Journal on Matrix Analysis and Applications 30.2 (2008): 713-730.
Alternating least squares for NMF

Gaussier, Eric, and Cyril Goutte. "Relation between PLSA and NMF and implications." Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2005.
First paper to point out the connection between PLSA and NMF

Arora, Sanjeev, Rong Ge, and Ankur Moitra. "Learning topic models--going beyond SVD." Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on. IEEE, 2012.
Theoretical justification of topic modeling as NMF

Tucker, Ledyard R. "Some mathematical notes on three-mode factor analysis." Psychometrika 31.3 (1966): 279-311.

Carroll, J. Douglas, and Jih-Jie Chang. "Analysis of individual differences in multidimensional scaling via an N-way generalization of “Eckart-Young” decomposition." Psychometrika 35.3 (1970): 283-319.

Harshman, Richard A. "Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis." (1970): 1-84. APA

Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel. "A three-way model for collective learning on multi-relational data." Proceedings of the 28th international conference on machine learning (ICML-11). 2011.

Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel. "Factorizing YAGO: scalable machine learning for linked data." Proceedings of the 21st international conference on World Wide Web. ACM, 2012. APA

Goller, Christoph, and Andreas Kuchler. "Learning task-dependent distributed representations by backpropagation through structure." Neural Networks, 1996., IEEE International Conference on. Vol. 1. IEEE, 1996.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shia Shalev-Shwartz, and Yoram Singer. "Online Passive-Aggressive Algorithms".JMLR. 2006.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay and Tommi Jaakkola. " Low-Rank Tensors for Scoring Dependency Structures."In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 1381--1391, Baltimore, Maryland, June.

Hai Zhao, Wenliang Chen, Chunyu Kit, Guodong Zhou. "Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing", CoNLL, 2009.

Rendle, Steffen. "Factorization machines." Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 2010.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. 2014. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. Journal of Machine Learning Research.

Bailly, R., Carreras, X., Luque, F., and Quattoni, A. 2013. Unsupervised spectral learning of WCFG as low-rank matrix completion.

Klami, Arto, Guillaume Bouchard, and Abhishek Tripathi. "Group-sparse Embeddings in Collective Matrix Factorization." Proceedings of International Conference on Learning Representations (ICLR) 2014.

Salakhutdinov, Ruslan, and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

Mohamed, Shakir, Zoubin Ghahramani, and Katherine A. Heller. "Bayesian exponential family PCA." Advances in Neural Information Processing Systems. 2009.

Stern, David H., Ralf Herbrich, and Thore Graepel. "Matchbox: large scale online bayesian recommendations." Proceedings of the 18th international conference on World wide web. ACM, 2009.

Raiko, Tapani, Alexander Ilin, and Juha Karhunen. "Principal component analysis for sparse high-dimensional data." Neural Information Processing. Springer Berlin Heidelberg, 2008.