Máquinas de Factorización
{ Factorization Machines (FM) }

Denis Parra

Sistemas Recomendadores

IIC 3633

2do semestre de 2017

Agenda Semestral

	Semana	Martes	Jueves	Enunciados	Deadlines/controles	Lecturer
3-ago	0		Intro (Blog) + UB CF			
8 y 10 ago	1	CF item-based (Vicente)	Slope One (Vicente)	Tarea 1		**VICENTE**
15 y 17 ago	2	feriado virgencita	Factorizacion Matricial			
22 y 24 ago	3	Evaluacion de RecSys	Implicit Feedback			
29 y 31 ago	4	Practico Tarea (Ivania)	Content-based (Ivania)		Deadline tarea	**IVANIA**
5 y 7 sept	5	Resumen RecSys + Híbridos	Intro proyect y practico content-based			
12 y 14 sept	6	Context-aware RecSys	Maquinas de Factorizacion	Enunciado Proyecto final		
19 y 21 sept	7	feriado fiestas patrias	Practico Maquinas de Factorizacion			
26 y 28 de sept	8	***Presentacion:Proy final***	***Presentacion:Proy final***			
3 y 5 de oct	9	User-centric RecSys/Interfaces	student presentation		FEEDBACK 1	**VICENTE**
10 y 12 de oct	10	Active Learning/Ranking	student presentation			
17 y 19 oct	11	Graph-based	student presentation		Informe de avance	
24 y 26 oct	12	Deep-Learning	student presentation		FEEDBACK 2	
31 oct y 2 nov	13	Learning to Rank	student presentation			
7 y 9 nov	14	Aplicaciones / Trust / Ethics	student presentation			
14 y 16 nov	15	LIBRE PARA PROYECTO FINAL			informe final	
Vi 24 de Nov	16	***Exámenes (Presentaciones finales)***	***Exámenes (Presentaciones finales)***			

En esta clase

- Factorization Machines

- 2 Ejemplos de Proyecto final

Máquinas de Factorización (2010)

- Inspiradas en SVM, permiten agregar un número arbitrario de features (user, item, contexto) pero funcionan bien con “sparse data” al incorporar variables latentes factorizadas (inspiradas en Factorización Matricial). No se necesitan vectores de soporte para optimizar el modelo.
- Generalizan diversos métodos de factorización matricial.
- Disminuyen la complejidad de aprendizaje del modelo de predicción respecto de métodos anteriores.

Motivación de FM

- Cada tarea de recomendación (implicit feedback, agregar tiempo, incorporar contexto) requiere rediseño del modelo de optimización y re-implementación del algoritmo de inferencia
- Lo ideal sería usar alguna herramienta como libSVM, Weka, … agregar los vectores de features
- Pero para manejar datos tan dispersos, se podrían mantener las factorizaciones!

Ejemplo

- Supongamos los siguientes usuarios, items y transacciones

$$U = \{\text{Alice (A)}, \text{Bob (B)}, \text{Charlie (C)}, \ldots\}$$
$$I = \{\text{Titanic (TI)}, \text{Notting Hill (NH)}, \text{Star Wars (SW)}, \text{Star Trek (ST)}, \ldots\}$$

Let the observed data S be:

$$S = \{(\text{A}, \text{TI}, \text{2010-1}, 5), (\text{A}, \text{NH}, \text{2010-2}, 3), (\text{A}, \text{SW}, \text{2010-4}, 1) (\text{B}, \text{SW}, \text{2009-5}, 4), (\text{B}, \text{ST}, \text{2009-8}, 5), (\text{C}, \text{TI}, \text{2009-9}, 1), (\text{C}, \text{SW}, \text{2009-12}, 5)\}$$

Representación Tradicional

Example for data:

User \ Movie	TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...

Matrix Factorization:

$$\hat{Y} := W H^t, \quad W \in \mathbb{R}^{|U| \times k}, H \in \mathbb{R}^{|I| \times k}$$

$$\hat{y}(u,i) = \hat{y}_{u,i} = \sum_{f=1}^{k} w_{u,f} h_{i,f} = \langle \mathbf{w}_u, \mathbf{h}_i \rangle$$

k is the rank of the reconstruction.

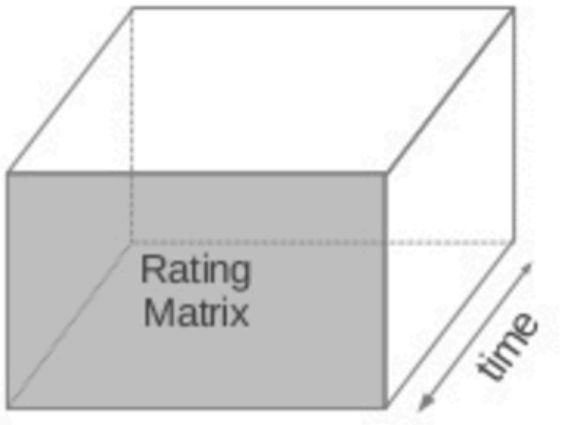
$$\min_{q*,p*} \sum_{(u,i) \in K} (r_{ui} - q_i^T \cdot p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

http://dparra.sitios.ing.uc.cl/classes/recsys-2016-2/clase8_factorizacion_matricial.pdf

Otros Modelos

Ejemplos de Datos:

User \ Movie	TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...



Ejemplos de Modelos:

$$\hat{y}^{\text{MF}}(u,i) := \sum_{f=1}^{k} v_{u,f} v_{i,f} = \langle \mathbf{v}_u, \mathbf{v}_i \rangle$$

$$\hat{y}^{\text{SVD++}}(u,i) := \left\langle \mathbf{v}_u + \sum_{j \in N(u)} \mathbf{v}_j, \mathbf{v}_i \right\rangle$$

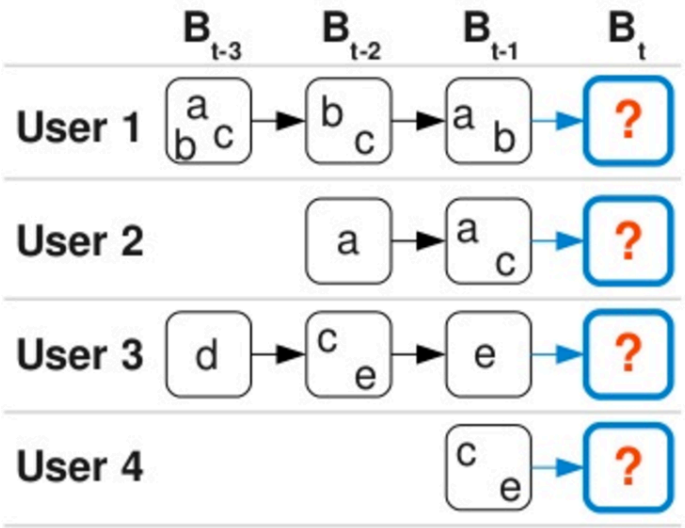
$$\hat{y}^{\text{Fact-KNN}}(u,i) := \frac{1}{|R(u)|} \sum_{j \in R(u)} r_{u,j} \langle \mathbf{v}_i, \mathbf{v}_j \rangle$$

$$\hat{y}^{\text{timeSVD}}(u,i,t) := \langle \mathbf{v}_u + \mathbf{v}_{u,t}, \mathbf{v}_i \rangle$$

$$\hat{y}^{\text{timeTF}}(u,i,t) := \sum_{f=1}^{k} v_{u,f}\, v_{i,f}\, v_{t,f}$$

$$\ldots$$

Modelos de Factorización Secuencial



$$\hat{y}^{\text{FMC}}(u, i, t) := \sum_{l \in B_{t-1}} \langle \mathbf{v}_i, \mathbf{v}_l \rangle$$

$$\hat{y}^{\text{FPMC}}(u, i, t) := \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \sum_{l \in B_{t-1}} \langle \mathbf{v}_i, \mathbf{v}_l \rangle$$

$$\dots$$

Modelos de Factorización

- Ventaja:
  - Permiten estimar interacciones entre dos (o más) variables incluso si la interacción no es observada explícitamente.
- Desventajas:
  - Modelos específicos para cada problema
  - Algoritmos de aprendizaje e implementaciones están diseñados para modelos individuales

Datos y Representación de Variables

- Muchos modelos de ML usan vectores de valores reales como input, lo que permite representar, por ejemplo:
  - Cualquier número de variables
  - Variables categóricas -> dummy coding
- Con este modelo estándar podemos usar regresión, SVMs, etc.

Modelo de Regresión Lineal

- Equivale a un polinomio de grado 1
- Queremos aprender w_0 y los p parámetros w_j
- No logra capturar interacciones latentes como la factorización matricial

$$\check{y}(x) = w_0 + \sum_{j=1}^{p} w_j x_j$$

- *O(p)* parámetros en el modelo.

Modelo con interacciones (d=2)

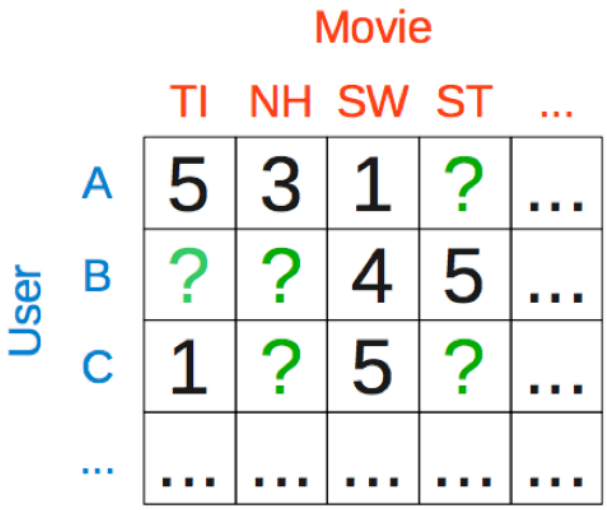
- Regresión Polinomial

$$\check{y}(x) = w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{\acute{j}=j+1}^{p} x_j x_{\acute{j}} w_{j,\acute{j}}$$

- $O(p^2)$ parámetros en el modelo

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{W} \in \mathbb{R}^{p \times p}$$

Representación Matricial como Vector de Features

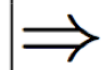


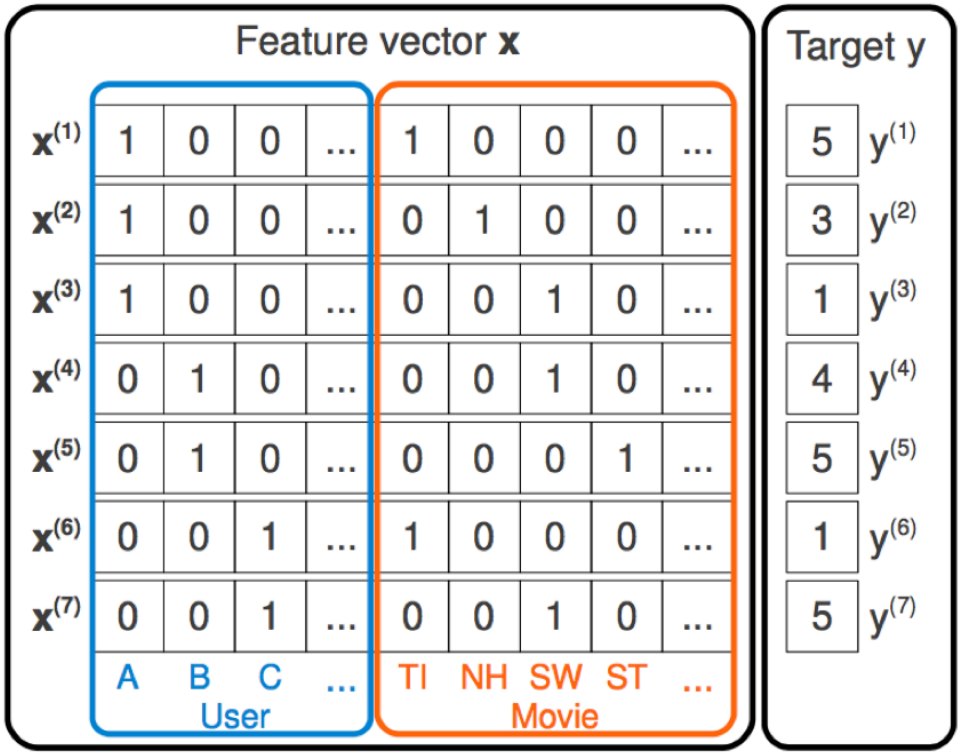
⇔

#	User	Movie	Rating
1	**A**lice	**T**itanic	5
2	**A**lice	**N**otting **H**ill	3
3	**A**lice	**S**tar **W**ars	1
4	**B**ob	**S**tar **W**ars	4
5	**B**ob	**S**tar **T**rek	5
6	**C**harlie	**T**itanic	1
7	**C**harlie	**S**tar **W**ars	5
...	...	...	...

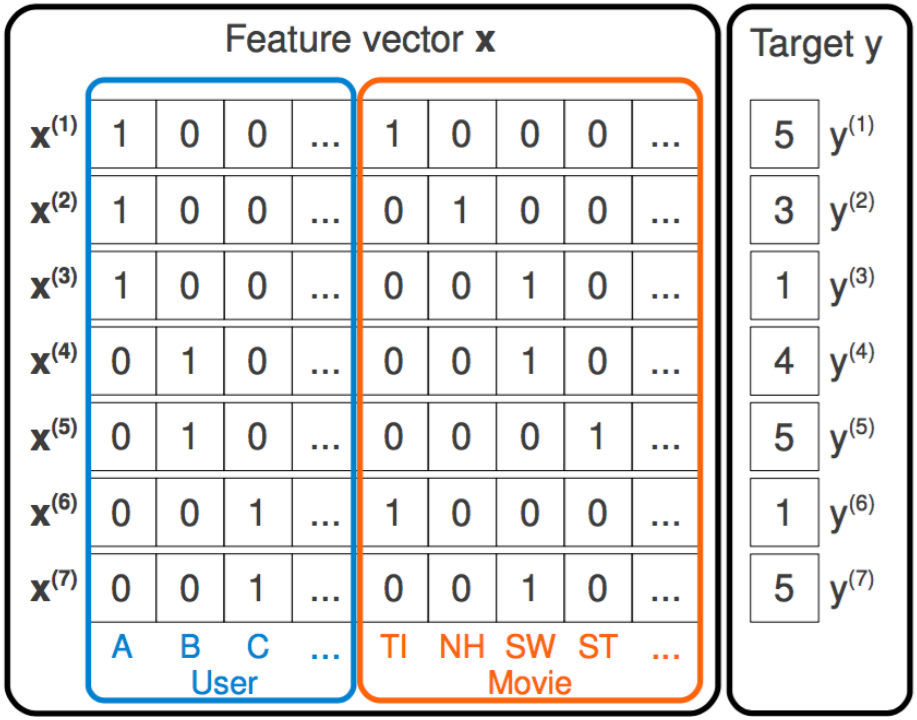
Representación Matriz como Vector de Features

#	User	Movie	Rating
1	**A**lice	**T**itanic	5
2	**A**lice	**N**otting **H**ill	3
3	**A**lice	**S**tar **W**ars	1
4	**B**ob	**S**tar **W**ars	4
5	**B**ob	**S**tar **T**rek	5
6	**C**harlie	**T**itanic	1
7	**C**harlie	**S**tar **W**ars	5
...	...	...	...





Aplicación de Regresión



- Regresión Lineal: $\hat{y}(\mathbf{x}) = w_0 + w_u + w_i$
- Regresión Polinomial: $\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + w_{u,i}$
- Factorización Matricial: $\hat{y}(u, i) = \langle \mathbf{w}_u, \mathbf{h}_i \rangle$

Problemas con Regresión Tradicional

- Regresión lineal no considera interaciones usuario-item : poder de expresión muy bajo

- Regresión Polinomial incluye interacciones de pares pero no se puede estimar porque
  - $n << p^2$: nro. de casos mucho menor que el número de parámetros.
  - Regresión polinomial no puede generalizar para cualquier efecto de pares de variables.

Modelo con interacción d=2 y factores latentes vs. Regresión polinomial

- Máquina de Factorización

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i\, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}$$

- Regresión Polinomial

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j \geq i}^{p} w_{i,j}\, x_i\, x_j$$

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{W} \in \mathbb{R}^{p \times p}$$

F.M. dado un modelo con d=2

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{j,f}\, v_{j',f},$$

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad V \in \mathbb{R}^{p \times k}.$$

Sesgo (bias) global

Coeficientes de regresión de la j-ésima variable

Interacción de features

Factorización (variables latentes)

F.M. dado un modelo con d=3

- Modelo

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i\, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

$$+ \sum_{i=1}^{p} \sum_{j>i}^{p} \sum_{l>j}^{p} \sum_{f=1}^{k} v_{i,f}^{(3)}\, v_{j,f}^{(3)}\, v_{l,f}^{(3)}\, x_i\, x_j\, x_l$$

- Parámetros

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}, \quad \mathbf{V}^{(3)} \in \mathbb{R}^{p \times k}$$

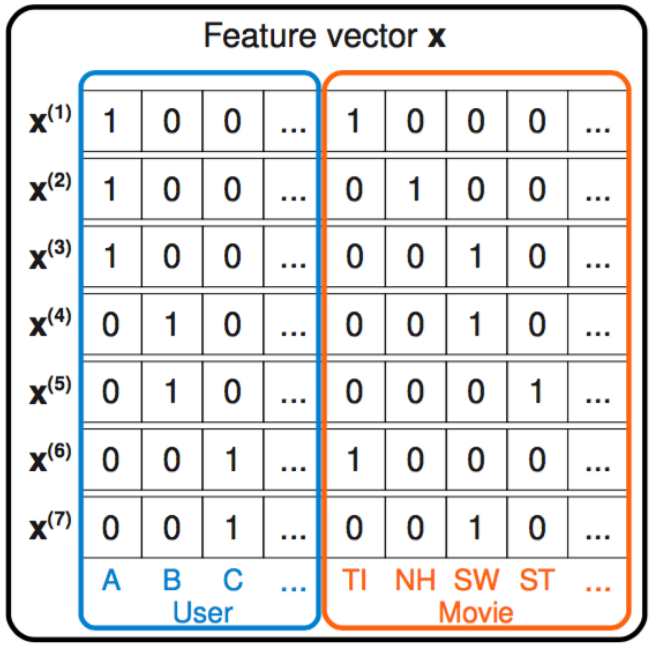
En suma

- FMs usan como entrada datos numéricos reales
- FMs incluyen interacciones entre variables como la regresión polinomial
- Los parámetros del modelo para las interacciones son factorizados
- Número de parámetros es $O(kp)$ vs. $O(p^2)$ en regresión polinomial.

Ejemplos

A. Dos variables categóricas
B. Tres variables categóricas
C. Dos variables categóricas y tiempo como predictor continuo
D. Dos variables categóricas y tiempo discretizado en bins
E. SVD++
F. Factorized Personalized Markov Chains (FPMC)

A. Dos variables categóricas



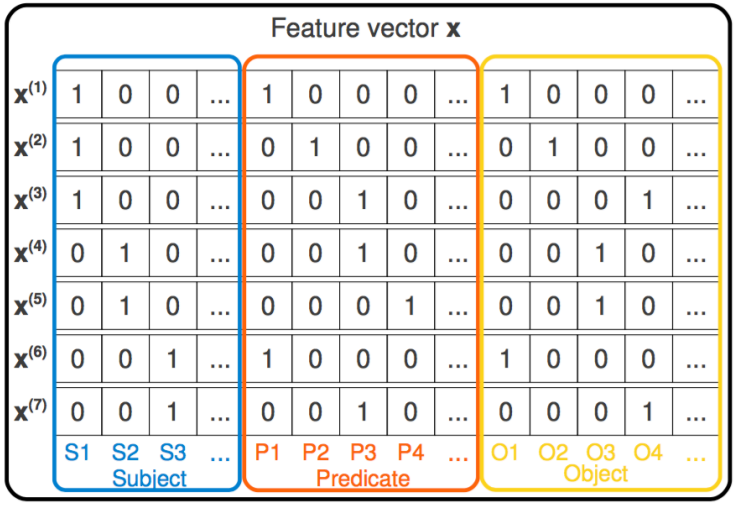
- Así, modelo corresponde a MF con biases

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + \underbrace{\langle \mathbf{v}_u, \mathbf{v}_i \rangle}_{\text{MF}}$$

libFM, $k = 128$, MCMC inference, Netflix RMSE=0.8937

B. Tres variables categóricas

- Predicción de tripletas RDF con FM

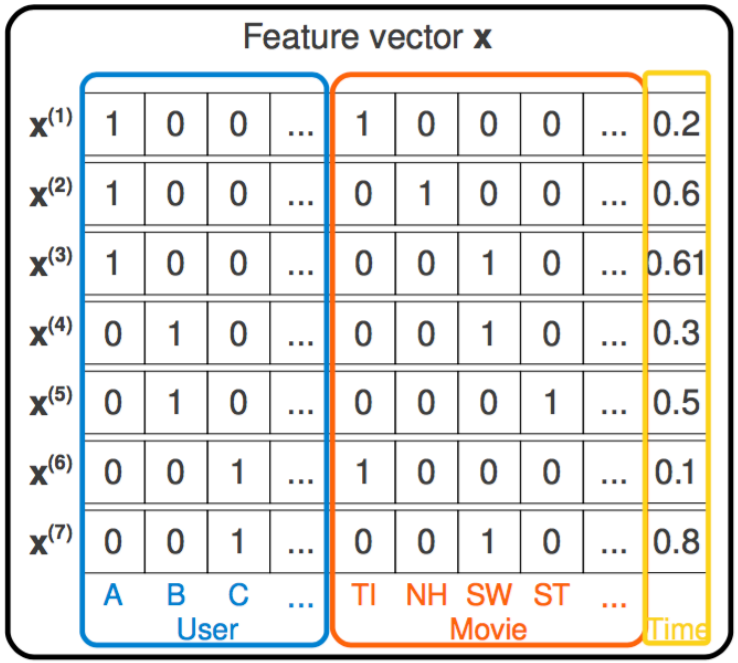


- Equivalente a PITF (recomendación de tags)

$$\hat{y}(\mathbf{x}) := w_0 + w_s + w_p + w_o + \langle \mathbf{v}_s, \mathbf{v}_p \rangle + \langle \mathbf{v}_s, \mathbf{v}_o \rangle + \langle \mathbf{v}_p, \mathbf{v}_o \rangle$$

[PITF: Rendle et al. 2010, WSDM Best Student Paper, ECML 2009 Best DC Award]

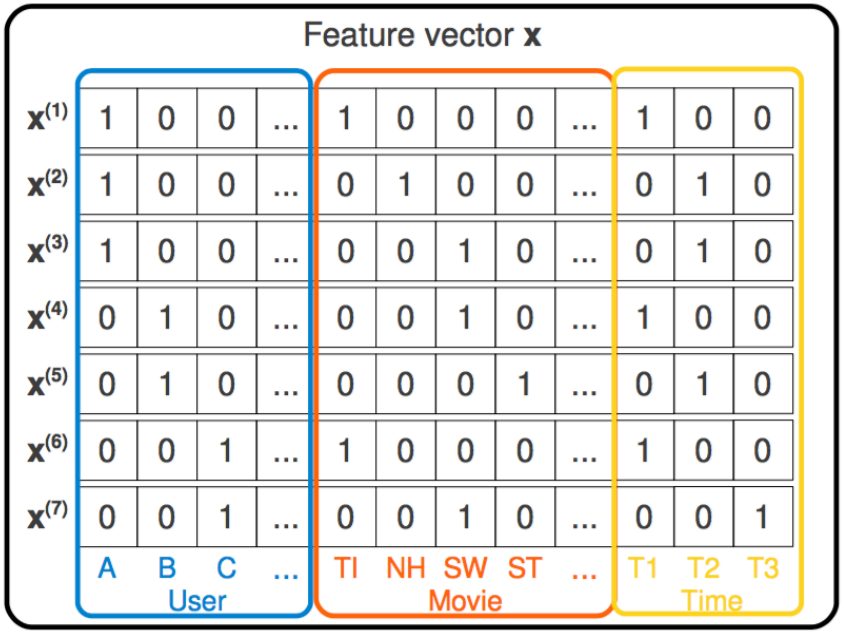
C. Dos variables categóricas y tiempo como predictor continuo



- Modelo corresponde a:

$$\hat{y}(\mathbf{x}) := w_0 + w_i + w_u + t\, w_{\text{time}} + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + t \langle \mathbf{v}_u, \mathbf{v}_{\text{time}} \rangle + t \langle \mathbf{v}_i, \mathbf{v}_{\text{time}} \rangle$$

D. Dos variables categóricas y tiempo discretizado en bins

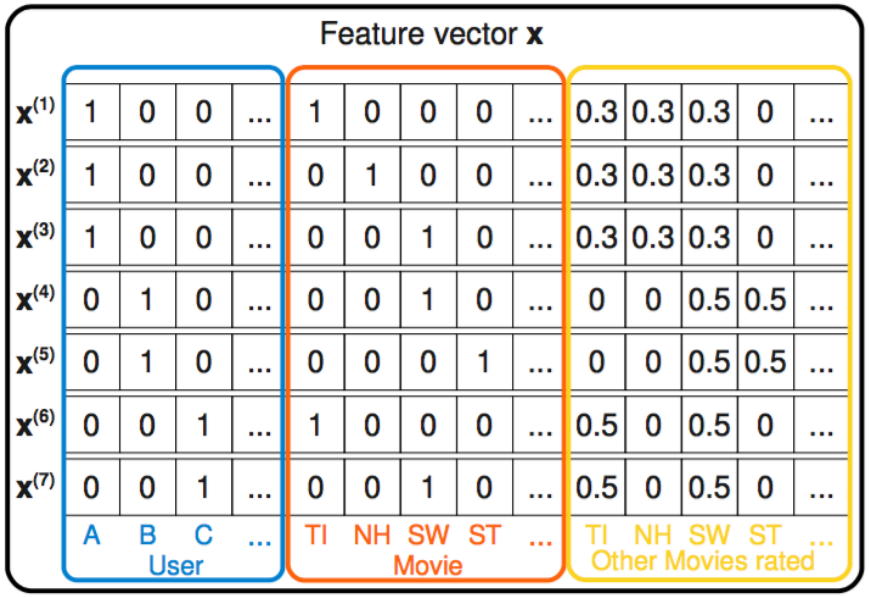


- Modelo corresponde a:

$$\hat{y}(\mathbf{x}) := w_0 + w_i + w_u + w_{b(t)} + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \langle \mathbf{v}_u, \mathbf{v}_{b(t)} \rangle + \langle \mathbf{v}_i, \mathbf{v}_{b(t)} \rangle$$

libFM, $k = 128$, MCMC inference, Netflix RMSE=0.8873

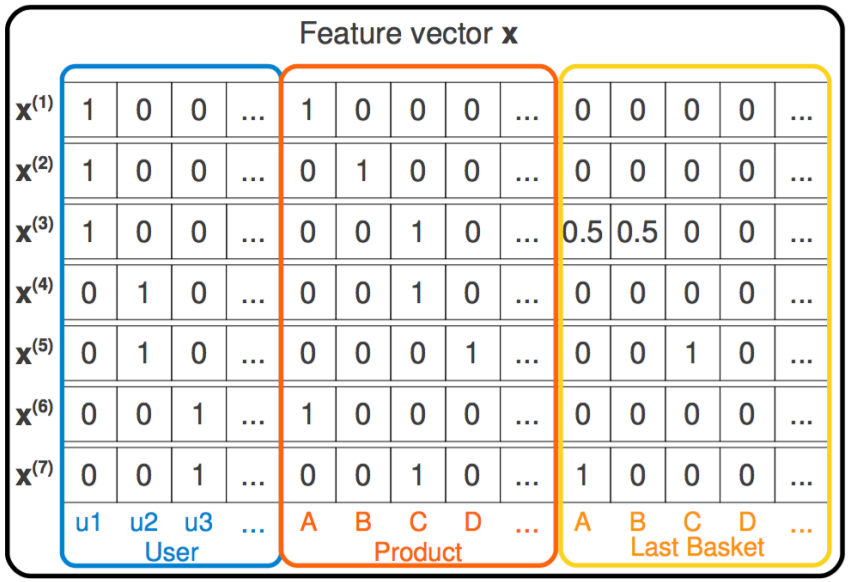
E. SVD++

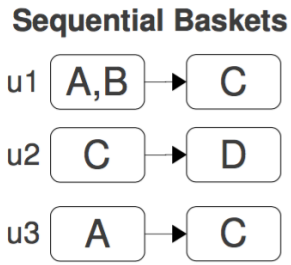


- Modelo idéntico a:

$$\hat{y}(\mathbf{x}) = \overbrace{w_0 + w_u + w_i + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \frac{1}{\sqrt{|N_u|}} \sum_{l \in N_u} \langle \mathbf{v}_i, \mathbf{v}_l \rangle}^{\text{SVD++}}$$
$$+ \frac{1}{\sqrt{|N_u|}} \sum_{l \in N_u} \left(w_l + \langle \mathbf{v}_u, \mathbf{v}_l \rangle + \frac{1}{\sqrt{|N_u|}} \sum_{l' \in N_u, l' > l} \langle \mathbf{v}_l, \mathbf{v}'_l \rangle \right)$$

F. FPMC





- Equivalente a:

$$\hat{y}(\mathbf{x}) := w_0 + w_u + w_i + \frac{1}{|B_{t-1}|}\sum_{j \in B_{t-1}} w_j + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \frac{1}{|B_{t-1}|}\sum_{j \in B_{t-1}} \langle \mathbf{v}_i, \mathbf{v}_j \rangle + \dots$$

[Rendle et al. 2010, WWW Best Paper]

Comparación con otros modelos

- En el paper Rendle, S. (2010, December). **Factorization machines**, se muestra como desde FM se puede derivar:
  - Matrix Factorization
  - SVD++
  - Pair-wise Interaction Tag-Factorization (PITF)
  - Factorized Personalized Markov Chains (FPMC)

Propiedades

- Expresividad* (cualquier matrix semi-definida positiva)
- Multilinearidad**
- **Complexity**

O(kn^2) -> O(kn)
Y debido a dispersión de los datos, $O(km_D)$

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j\rangle x_i\, x_j$$

$$=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j\rangle x_i\, x_j - \frac{1}{2}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{v}_i\rangle x_i\, x_i$$

$$=\frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{k} v_{i,f}\, v_{j,f}\, x_i\, x_j - \sum_{i=1}^{n}\sum_{f=1}^{k} v_{i,f}\, v_{i,f}\, x_i\, x_i\right)$$

$$=\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i\right)\left(\sum_{j=1}^{n} v_{j,f}\, x_j\right) - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2\right)$$

$$=\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i\right)^2 - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2\right)$$

*, ** ver detalles en Rendle, S. (2010, December). **Factorization machines**.

Complejidad

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{j,f}\, v_{j',f},$$

Número de parámetros :

$$1 + p + k*p$$

lineal respecto al tamaño del input y el tamaño de los factores latentes

Reducción del modelo

1) Model Equation: The model equation for a factorization machine of degree $d = 2$ is defined as:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \qquad (1)$$

where the model parameters that have to be estimated are:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k} \qquad (2)$$

And $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \qquad (3)$$

Aprendizaje

- Regularización L2 para regresión y clasificacion
  - SGD
  - ALS
  - MCMC
- Ranking regularizado L2

Todos los algoritmos tienen tiempo de ejecución $O(k\ N_z(x)\ i)$ donde i: iteraciones, $N_z(X)$: elementos no-cero, y k: nro. de factores latentes.

Software: LibFM

- LibFM implementa FMs
  - Modelos: FMs de 2do orden
  - Aprendizaje: SGD, ALS, MCMC
  - Clasificación y regresión
  - Formato de datos: sparse (LIBSVM, LIBLINEAR, SVMlight, etc.)
  - Soporta agrupación de variables
  - Open Source: GPLv3

www.libfm.org

Open Screenshot

libFM: Factorization Machine Library

Author: Steffen Rendle

Factorization machines (FM) are a generic approach that allows to mimic most factorization models by feature engineering. This way, factorization machines combine the generality of feature engineering with the superiority of factorization models in estimating interactions between categorical variables of large domain. libFM is a software implementation for factorization machines that features stochastic gradient descent (SGD) and alternating least squares (ALS) optimization as well as Bayesian inference using Markov Chain Monte Carlo (MCMC).

Source code

- **github** repository: https://github.com/srendle/libfm.
- Please acknowledge the software (i.e. cite the paper *Factorization Machines with libFM*) if you publish results produced with this software.

Latest release

- **Source code (C++) libfm-1.42.src.tar.gz (2014-09-14)**, GPL v3 license
- **Windows Executable libfm-1.40.windows.zip (2013-07-12)**
- The license is included in the archive -- please see the file `license.txt` for details.
- Please acknowledge the software (i.e. cite the paper *Factorization Machines with libFM*) if you publish results produced with this software.

Usage

Please see the libFM 1.4.2 manual for details about how to use `libFM`. This manual is also included in the tar.gz archive of the source code.

References

If you use libFM please cite the following paper:

> Steffen Rendle (2012): *Factorization Machines with libFM*, in ACM Trans. Intell. Syst. Technol., 3(3), May. [PDF]

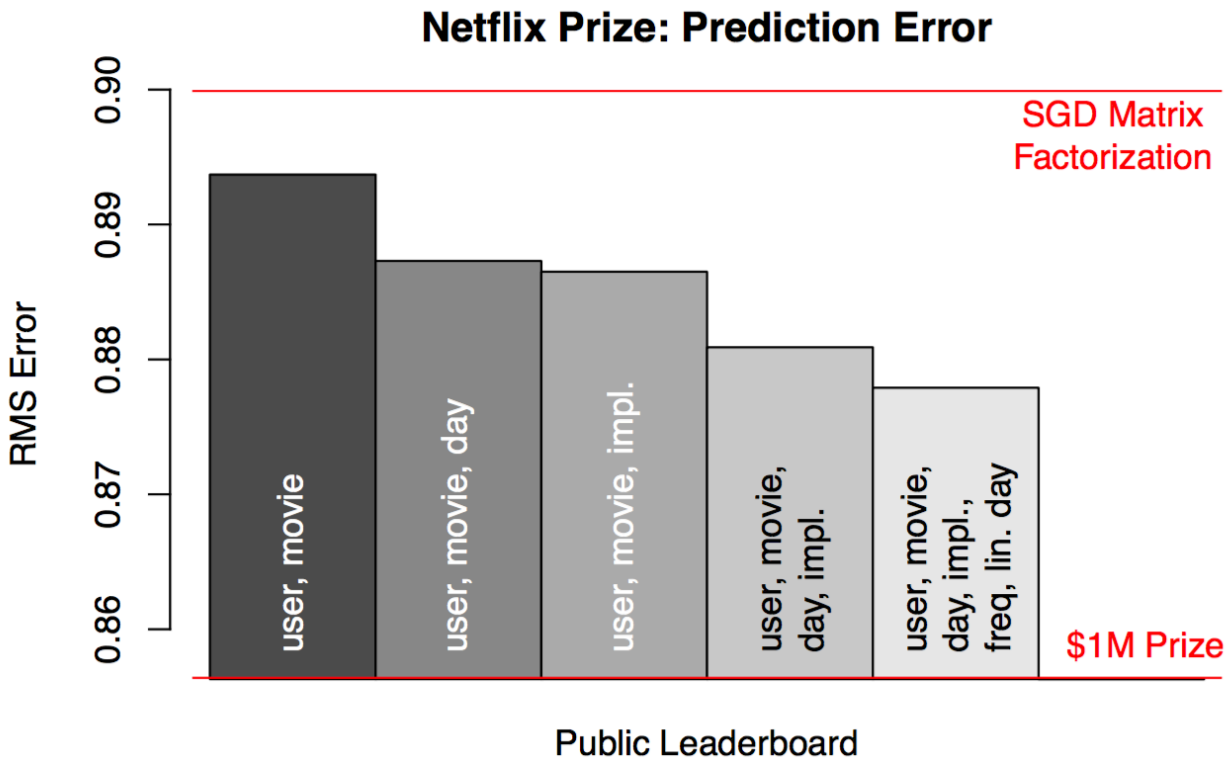
BibTeX:

```
@article{rendle:tist2012,
        author = {Rendle, Steffen},
        title = {Factorization Machines with {libFM}},
        journal = {ACM Trans. Intell. Syst. Technol.},
        issue_date = {May 2012},
        volume = {3},
        number = {3},
        month = May,
        year = {2012},
        issn = {2157-6904},
        pages = {57:1--57:22},
        articleno = {57},
        numpages = {22},
        publisher = {ACM},
        address = {New York, NY, USA},
```

Predicción de ratings (Context-aware)

- Main variables:
  - User ID (categorical)
  - Item ID (categorical)
- Additional variables:
  - time
  - mood
  - user profile
  - item meta data
  - ...
- Examples: Netflix prize, Movielens, KDDCup 2011

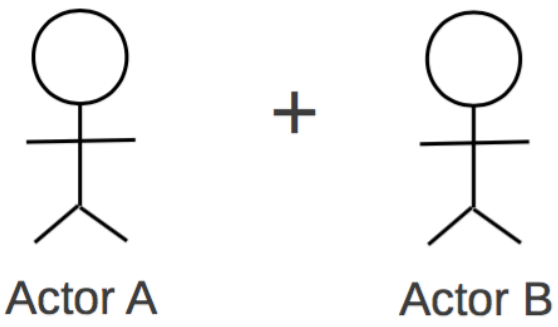
Netflix Prize



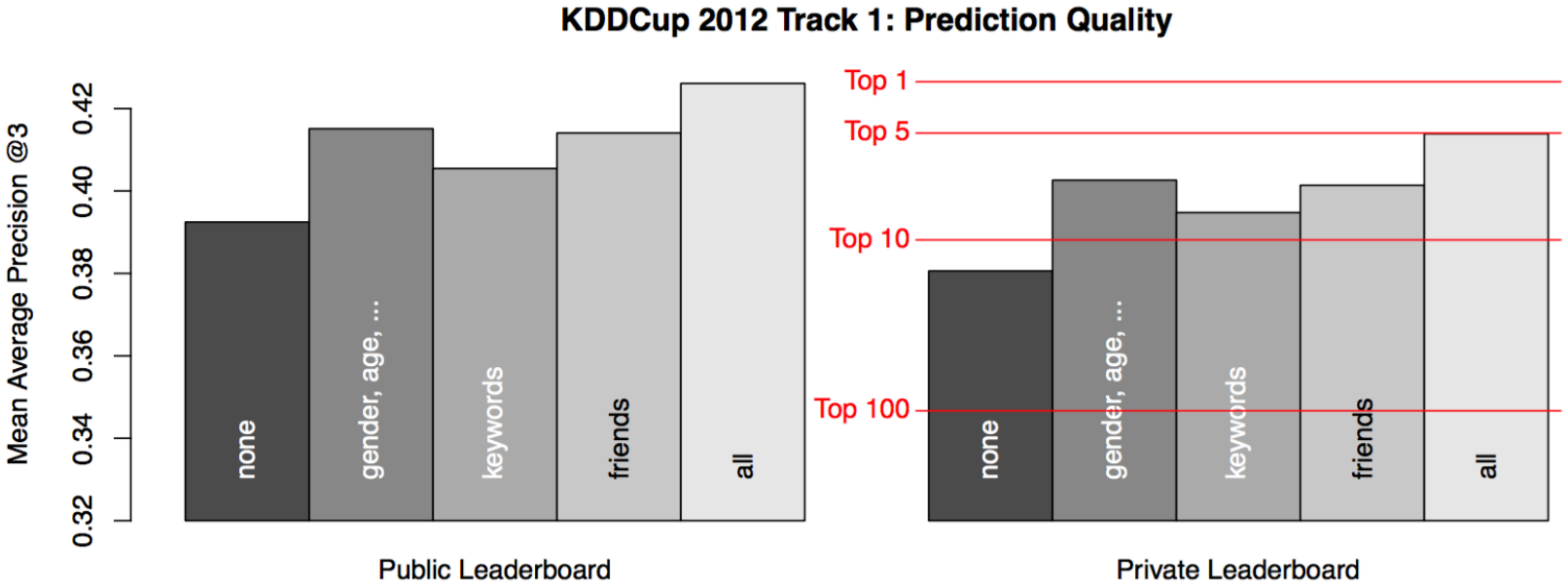
- $k = 128$ factors, 512 MCMC samples (no burnin phase, initialization from random)
- MCMC inference (no hyperparameters (learning rate, regularization) to specify)

Predicción de relaciones en Redes

- Main variables:
  - Actor A ID
  - Actor B ID
- Additional variables:
  - profiles
  - actions
  - ...



KDDCup 2012: track 1

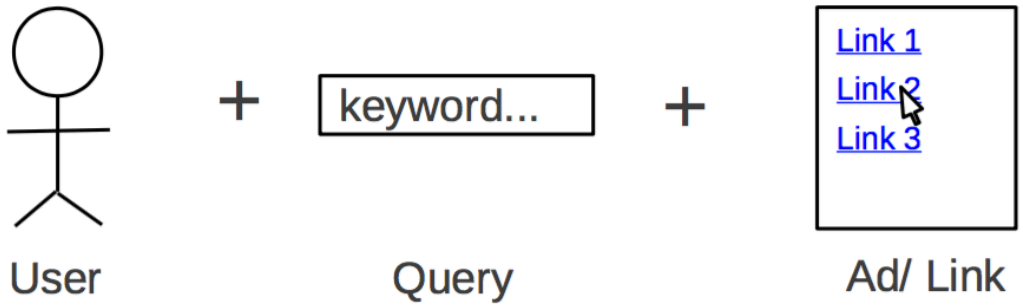


- $k = 22$ factors, 512 MCMC samples (no burnin phase, initialization from random)
- MCMC inference (no hyperparameters (learning rate, regularization) to specify)

[Awarded 2nd place (out of 658 teams)]

Predicción de Clicks

- Main variables:
  - User ID
  - Query ID
  - Ad/ Link ID
- Additional variables:
  - query tokens
  - user profile
  - . . .



KDDCup 2012: Track 2

Model	Inference	wAUC (public)	wAUC (private)
ID-based model ($k = 0$)	SGD	0.78050	0.78086
Attribute-based model ($k = 8$)	MCMC	0.77409	0.77555
Mixed model ($k = 8$)	SGD	0.79011	0.79321
Final ensemble	n/a	0.79857	**0.80178**

Ensemble

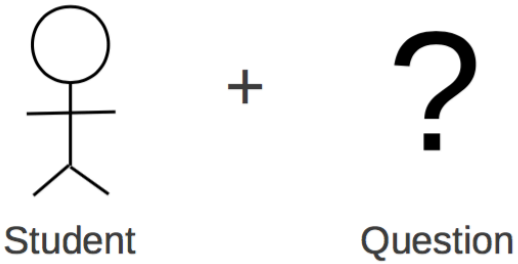
- Rank positions (not predicted clickthrough rates) are used.
- The MCMC attribute-based model and different variations of the SGD models are included.

.

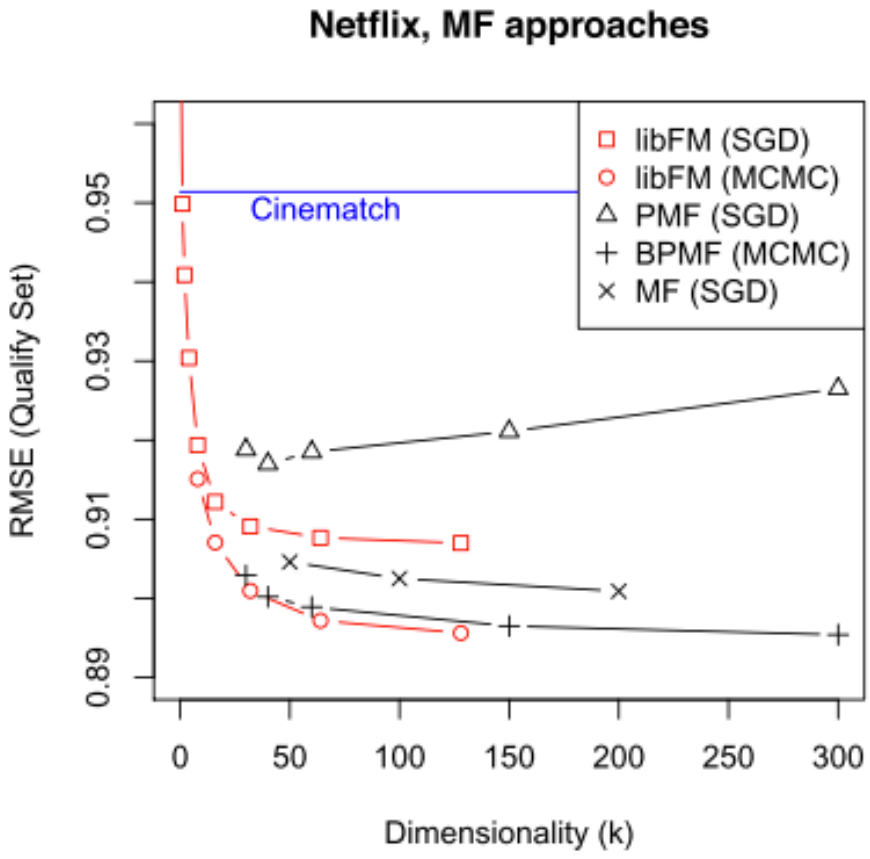
[Awarded 3rd place (out of 171 teams)]

Predecir Resultados de Estudiantes

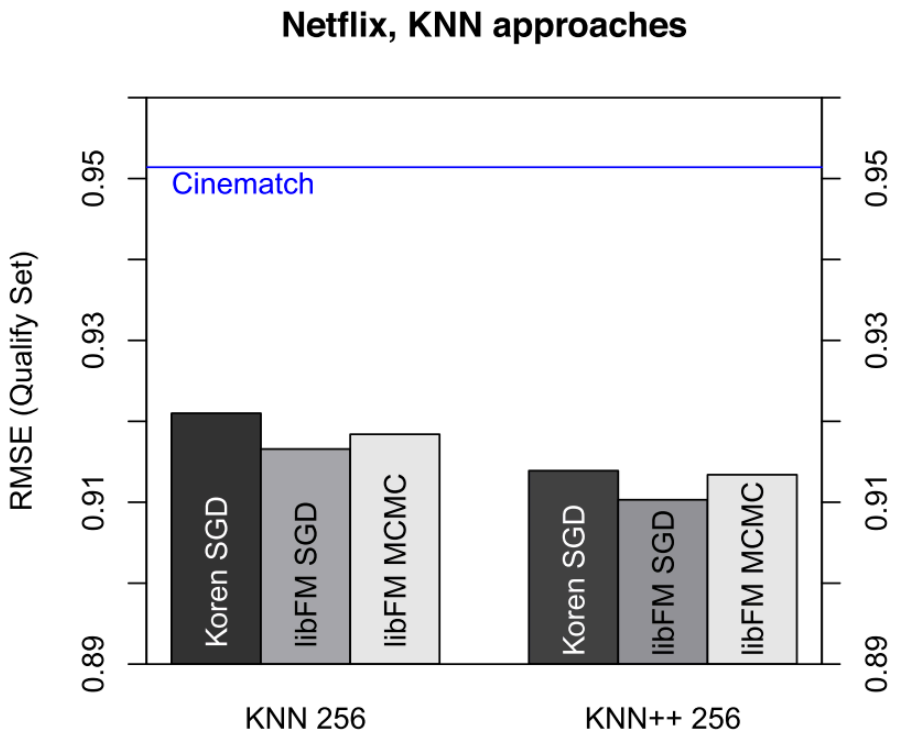
- Main variables:
  - Student ID
  - Question ID
- Additional variables:
  - question hierarchy
  - sequence of questions
  - skills required
  - ...
- Examples: KDDCup 2010, Grockit Challenge[4] (FM placed 1st/241)



Algunos Resultados

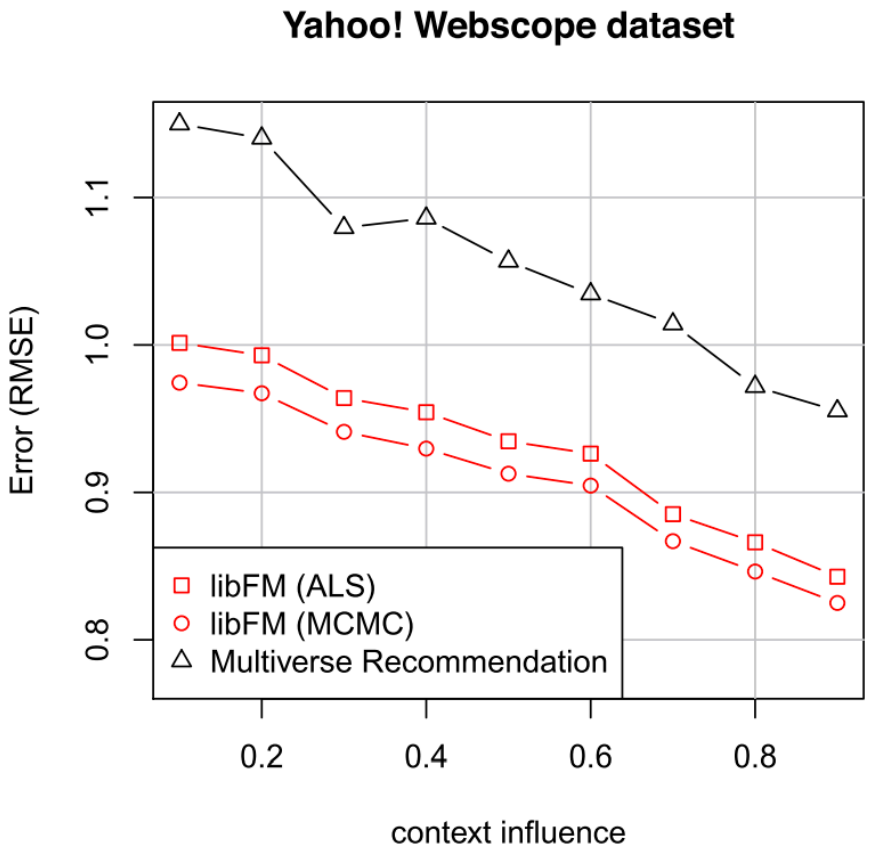


(a) Matrix factorization (MF).

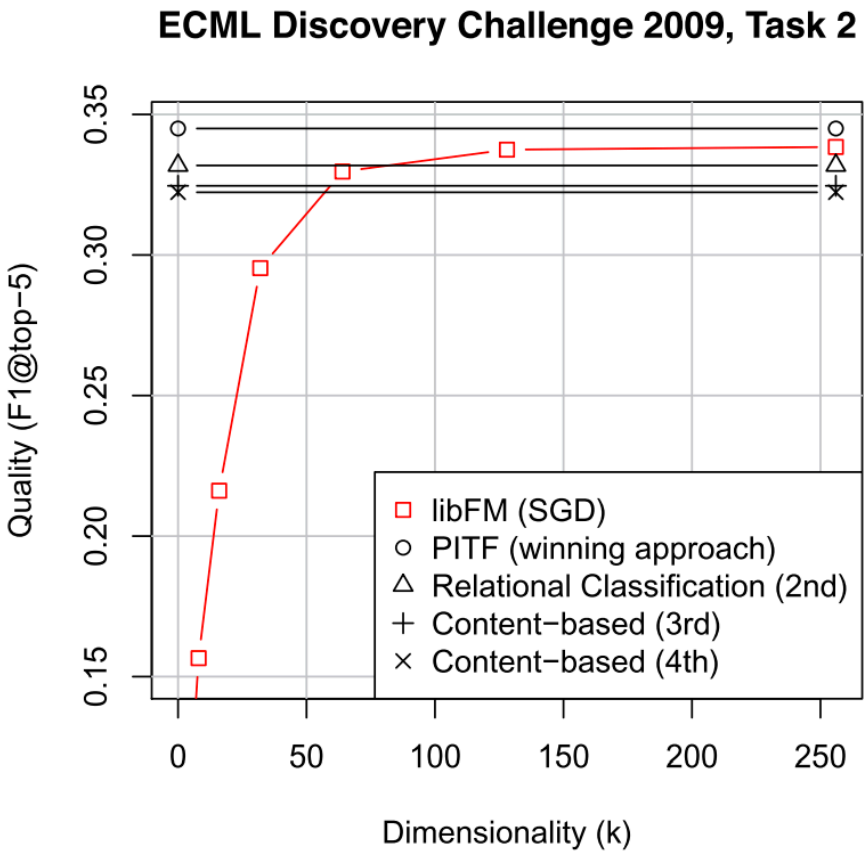


(b) Nearest neighborhood (KNN).

Algunos Resultados II



(a) Context-aware recommendation.



(b) Tag recommendation.

Using Libfm

- Llamada 1:

```
./libFM -task r -train ml1m-train -test ml1m-test -dim '1,1,8'
```

- Llamada 2:

```
./libFM -task r -train ml1m-train.libfm -test ml1m-test.libfm -dim '1,1,8' -iter 1000
  -method sgd -learn_rate 0.01 -regular '0,0,0.01' -init_stdev 0.1
```

$$X = \begin{pmatrix} 1.5 & 0.0 & 0.0 & -7.9 & 0.0 & 0.0 & 0.0 \\ 0.0 & 10^{-5} & 0.0 & 2.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 4 \\ 2 \\ -1 \end{pmatrix}$$

Example

```
4 0:1.5 3:-7.9
2 1:1e-5 3:2
-1 6:1
...
```

Ejemplo con libFMexe

- Wrapper de LibFM para R

```
library(libFMexe)

data(movie_lens)

set.seed(1)
train_rows = sample.int(nrow(movie_lens), nrow(movie_lens) * 2 / 3)
train = movie_lens[train_rows, ]
test  = movie_lens[-train_rows, ]

predFM = libFM(train, test, Rating ~ User + Movie, task = "r", dim = 10, iter = 300,
exe_loc = "/Users/denisparra/libfm-1.42.src/bin/")

head(predFM)
# How good is RMSE ?
mean((predFM - test$Rating)^2)
```

Conclusiones

- FMs combinand regresión lineal/polinomial con modelos de factorización.
- Interacción entre variables se aprenden vía representación low-rank.
- Es posible la estimación de observaciones no observadas.
- Se pueden calcular eficientemente y tienen una buena calidad de predicción.

Referencias

- **Rendle, S. (2010) "Factorization Machines" (https://www.ismll.uni-hildesheim.de/pub/pdfs/Rendle2010FM.pdf)**
- http://www.slideshare.net/hongliangjie1/libfm
- http://www.slideshare.net/SessionsEvents/steffen-rendle-research-scientist-google-at-mlconf-sf
- http://www.slideshare.net/0x001/intro-to-factorization-machines?next_slideshow=1

Proyecto Final curso RecSys 2014

- Trade-offs Between Implicit Feedback and Context-Aware Recommendation
  - Santiago Larraín, PUC Chile
  - Nicolás Risso, PUC Chile
- Moviecity Dataset

Proyecto Final curso RecSys 2014

- Moviecity

Columna	Descripción
user_id	Identificador de usuario único
version_id	Identificador único de contenido
user_watchinglist_time_minutes_spent	Consumo acumulado en minutos
DURATION_MINUTES	Largo del contenido
account_country_code	Código de país del usuario
country_description	IdentificadorNombre del país
country_region	Regíon geográfica
Kids	Marca de si el contenido es para kids o no
Genre	Genero del contenido
Subgenre	Genero primario del contenido

Dataset Moviecity

Mes	Cantidad de usuarios	Cantidad de items	Rmin	Rmax	Ravg
Junio	95013	1679	0	33.94	0.26
Julio	83924	1612	0	38.20	0.34
Agosto	95013	1679	0	33.94	0.26
Total	191657	1918	0	38.20	0.32

Month	Row count	User count	Item count
June	407.078	95.013	1.679
July	482.772	83.924	1.612
August	548.419	95.013	1.668
Total	1.438.269	191.657	1.918

Table 2: Dataset statistics by month

Dataset MovieCity II

Géneros	Subgéneros	Países	Zonas geográficas
Kids	Animation	Mexico	N
Pelicula	Family	Argentina	S
Serie	Thriller	Peru	SA
Movies And Features	Comedy	Colombia	C
Anime	Adventure	Chile	
Documental	Drama	Uruguay	
	Documentary	Venezuela	
	Horror	Rep. Dominicana	
	Action	Honduras	
	Classics	Panama	
	Musical	El Salvador	
	Science Fiction	Guatemala	
	Western	Bolivia	
		Costa Rica	
		Nicaragua	
		Paraguay	

Métodos I

- Hu and Koren ~ Implicit Feedback

$$p_{u,i} = \begin{cases} 1 & \text{if } r_{u,i} > 0 \\ 0 & \text{other case} \end{cases} \qquad c_{u,i} = 1 + \alpha r_{u,i}$$

$$\min_{x*,y*} \sum_{u,i} c_{u,i}(p_{u,i} - \vec{x}_u^T \vec{y}_i)^2 + \lambda\Big(\sum_u ||\vec{x}_u||^2 + \sum_i ||\vec{y}_i||^2\Big)$$

$$\vec{x}_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u) \qquad (4)$$

$$\vec{y}_u = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i) \qquad (5)$$

Where $C^u \in \mathbb{R}^{U \times k} : C^u_{i,i} = c_{u,i}$ and $C^i \in \mathbb{R}^{I \times k} : C^i_{u,u} = c_{u,i}$

Métodos II

- Factorización Tensorial (usando HOSVD)

If we define $\mathbf{P} \in \mathbb{R}^{U \times I \times C_1 \ldots \times C_N}$, we can decompose the origin al data as follows:

$$\mathbf{P} = S \times_U U^{(U)} \times_I U^{(I)} \times_{C_1} \ldots \times_N U(C_N) \qquad (6)$$

Where $U^{(n)} \in \mathbb{R}^{n \times n}$ is a n dimension tensor and $\times_n$ is a tensor product in dimension n.

Métodos III

- Factorization Machines, Rendle (2010)

$$y(x) := w_0 + \sum_{i=1} w_i x_i + \sum_{i=1} \sum_{j=i+1} (\vec{v_i} \cdot \vec{v_j}) x_i x_j$$

$$y(x) := w_0 + \sum_{i=1} w_i x_i + \sum_{l=1} \sum_{i_i=1} \cdots \sum_{i_l=i_{l-1}+1} \left(\prod_{j=1} x_{i_j} \right) \left(\sum_{f=1} \prod_{j=1} v_{i_j} f \right)$$

Métricas de Evaluación

- RMSE: Diferencia de tiempo entre programa visto y lo predicho

1. $RMSE = \sqrt{\sum_{(u,i) \in Train} (r_{u,i} - \hat{r}_{u,i})^2}$
2. $MAE = \sum_{(u,i) \in Train} |r_{u,i} - \hat{r}_{u,i}|$
3. $\overline{rank} = \frac{\sum_{(u,i) \in Train} r_{u,i} rank_{u,i}}{\sum_{(u,i) \in Test} r_{u,i}}$

Optimización de los modelos

Parameters			RMSE	MAE	rank
k	α	λ			
10	20	75	0.6869	0.4686	0.0500
10	20	150	0.7420	0.5172	0.0639
10	20	250	0.7687	0.5529	0.0882
40	20	150	0.7281	0.4936	0.0496
40	20	250	0.7844	0.5615	0.0872
40	40	75	0.5934	0.3567	0.0151
40	40	150	0.6331	0.4053	0.0252
40	40	250	0.6935	0.4681	0.0544
40	60	75	**0.5931**	**0.3392**	**0.0138**
40	60	150	0.5994	0.3674	0.0197
40	60	250	0.6443	0.4211	0.0363

Implicit Feedback

k	RMSE	MAE	rank
10	1.024	0.7787	0.5092
25	1.028	0.7807	0.5059
40	**1.014**	**0.7683**	**0.5003**

Tensor Factorization

k	RMSE	MAE	rank
10	**0.6019**	**0.3958**	**0.4181**
25	0.6862	0.4333	0.4398
40	0.6310	0.4396	0.4272

Factorization Machines

Comparación de los Modelos

Model	RMSE	MAE	rank
Matrix factorization	0.7404	0.4820	**0.1334**
Tensor factorization	1.0024	0.7553	0.5117
Factorization machine	**0.6105**	**0.4148**	0.4092

- Matrix Factorization: $k = 40$; $\lambda = 75$; $\alpha = 60$.
- Tensor Factorization: $k = 40$.
- Factorization Machine: $k = 10$.

Conclusiones

- Error de MAE entre 40% y 70%: diferencia promedio entre el tiempo predicho y el tiempo que el usuario realmente vio. Mejor método es Factorization Machines, indicando que para esta tarea el contexto ayuda.
- Ranking: el mejor método es Implicit Feedback recommender. Extrañamente, esto indica que para rankear, el mejor método no requiere contexto.

Propuesta de Proyecto Final curso RecSys 2016

- Clustering de documentos con social tags
  - Lucas Zorich, PUC Chile
  - Mauricio Troncoso, PUC Chile

Clustering de documentos con *social tags*

Mauricio Troncoso, Lukas Zorich

Pontificia Universidad Católica de Chile

{*mjtroncoso,lezorich*}*@uc.cl*

27 de septiembre de 2016

Contexto

Contexto

Hacer *clustering* de documentos permitiría:

- Agrupar los resultados de búsquedas en categorías
- Entregar diversidad en los resultados de las búsquedas
- Mejorar la presentación de la información al usuario

Problema y solución propuesta

El principal problema estriba en cómo detectar el **tópico** del que trata cada documento.

Problema y solución propuesta

Usar *tags* como información adicional:

- Los *tags* entregan información semántica importante.
- Mucho contenido en internet tiene *tags*.

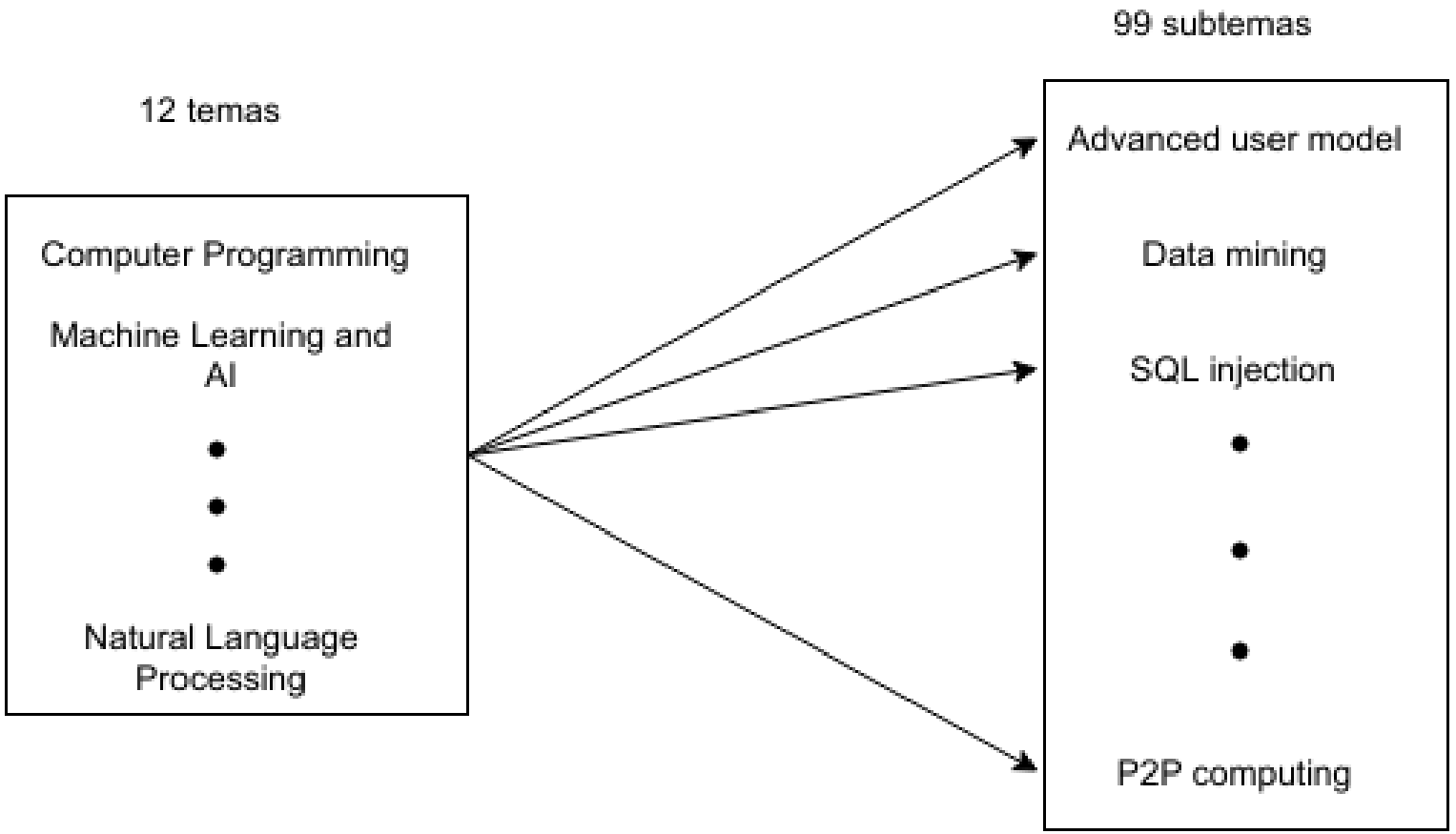
Problema y solución propuesta

Compararemos diferentes algoritmos de *clustering* en documentos utilizando *tags* como información adicional.

Objetivo

¿Los *tags* de usuarios son una fuente significativa de información al hacer *clustering* de documentos?

Dataset


12 temas
Computer Programming
Machine Learning and AI
Natural Language Processing
99 subtemas
Advanced user model
Data mining
SQL injection
P2P computing

Dataset

- Cada subtema es considerado como una *query* ejecutada en CiteULike.
- Estudiantes de postgrado expertos clasificaron el resultado de la *query* como:
  - No relevante
  - Poco relevante
  - Muy relevante
- Los documentos relevantes de cada *query* será el *ground truth*.
- Los *tags* se conseguirán haciendo *scraping* de los documentos en CiteULike.

Algoritmos a comparar

- K-Means
- Non-negative Matrix Factorization
- LDA
- MM-LDA (opcional)

Experimentos

Por cada algoritmo, se probarán diferentes *features*:

- Solo palabras
- Solo *tags*
- *Tags* como palabras por n

Evaluación

- Siguiendo a (Ramage *et al*, 2008) ocuparemos el *Cluster-F1 score* como métrica de evaluación.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Propuesta de proyecto: **Una comparación de diferentes técnicas de *clustering* de documentos**

IIC3633 Sistemas Recomendadores

Mauricio Troncoso mjtroncoso@uc.cl
Lukas Zorich lezorich@uc.cl

Índice

1. Contexto del Problema

Actualmente, los motores de búsqueda modernos están encargados de devolver los resultados más relevantes en el momento en el que un usuario hace una *query*, la que generalmente es bastante ambigua. Estos resultados más relevantes se tienen que encontrar de entre millones de documentos, lo que no es una tarea fácil.
La gran mayoría de las veces, para encontrar los resultados más relevantes el motor de búsqueda utiliza técnicas de *ranking* basadas en las conexiones entre *links*, textos de los *links*, *clicks* del usuario en los distintos datos, o buscar en el texto de la página. Sin embargo, el mayor desafío es manejar la ambigüedad que existe en la *query* del usuario. Esta *query* generalmente tiene pocas palabras y puede tener varios significados. Una manera de manejar esta ambigüedad es mediante *clustering* del contenido. Si una *query* ambigua puede tener varios significados, y asumiendo que un significado es un *cluster* de documentos de un mismo tópico, se le puede devolver al usuario documentos de varios de estos tópicos lo que aumentaría la diversidad de la respuesta. Por otro lado, si se sabe el significado de la *query* del usuario, se podrían obtener fácilmente documentos relevantes, ya que se sabría que el usuario quiere documentos de un tópico en particular.
Por lo tanto, poder agrupar distintos documentos que pertenezcan a un mismo tópico es fundamental para que el usuario pueda obtener información relevante fácilmente.

2. Problema y propuesta de solución

La principal problemática que se busca atacar con este proyecto, es la de identificar tópicos de interés en documentos y poder usarlos para hacer *clusters* de documentos, lo que permitiría a los usuarios encontrar más fácilmente documentos que le interesan. El problema entonces, se representa por la pregunta: cuál es la mejor forma de hacer *clustering* de documentos?.
La solución propuesta entonces, es comparar diferentes técnicas que existen hoy en día para agrupar distintos documentos.

3. Objetivos

El objetivo de este proyecto es poder compara diferentes algoritmos de *clustering* de documentos. Además, un objetivo ideal, aunque se dejará opcional, sería el de responder la pregunta: son los tags entregados por usuarios en sitios como CiteULike una fuente significativa de información al intentar hacer un *clustering* de documentos?.

4. Definición de experimentos

4.1. Conjunto de datos

Los experimentos se correrán en el *dataset* CiteEval [2]. Este *dataset* fue creado definiendo 12 áreas de interés, o categorías, dentro de la ciencia de la computación y ciencia de la información. Dentro de estas categorías se encuentran las áreas de Aprendizaje de Máquinas e IA, Bases de Datos, Arquitectura de Computadores, entre otras. Para cada una de estas categorías se definieron un subconjunto de tópicos representados como *querys*, las cuales son 99 en total. Cada una de estas 99 *querys* fueron ejecutadas en el sitio CiteULike. El resultado de cada *query* fue guardado en una colección de documentos que incluyen el id del documento, el título y el *abstract*. Para

saber la relevancia de cada documento en la *query*, se les preguntó a estudiantes de postgrado expertos en la categoría o subtópico de esa *query*. Se ocupará esta relevancia como el *ground truth* al evaluar los diferentes algoritmos.

4.2. Algoritmos a comparar

Los algoritmos que se compararán son:

- K-*means*.
- *Hierarchical clustering*. En particular, se utilizará AgglomerativeClustering.
- *Latent dirichlet allocation* (LDA).

Para el algoritmo K-means y *Hierarchical clustering*, se ocupará tfidf para representar cada palabra.
Además, dejaremos como opcional probar el algoritmo MM-LDA [1]. Para poder probar este algoritmo, será necesario extraer tags disponibles desde la página de CiteULike para los documentos que están en nuestro conjunto de datos.

Referencias

[1] Ramage, D., Heymann, P., Manning, C. D., & Garcia-Molina, H. (2009, February). Clustering the tagged web. In Proceedings of the Second ACM International Conference on Web Search and Data Mining (pp. 54-63). ACM.

[2] Yue, Z., Harpale, A., He, D., Grady, J., Lin, Y., Walker, J., ... & Yang, Y. (2009, July). CiteEval for evaluating personalized social web search. In SIGIR 2009 Workshop on the Future of IR Evaluation (p. 23).

Recomendación de artículos académicos utilizando etiquetas sociales

Lukas Zorich
Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile
lezorich@uc.cl

Mauricio Troncoso
Departamento de Ingeniería Matemática
Pontificia Universidad Católica de Chile
mjtroncoso@uc.cl

ABSTRACT

El almacenamiento digital de material científico ha dificultado considerablemente la tarea, de los investigadores, de encontrar trabajos pertinentes a su labor académica, luego, es pertinente idear métodos de recomendación que se ajusten a las nuevas necesidades del mundo de la academia. Para esto se emplea un método ya utilizado para esta labor que consiste en recomendar material nuevo y antiguo en base a, respectivamente, modelación de tópicos y filtrado colaborativo. En este trabajo se extiende la labor más reciente sobre la base de datos de CiteULike, considerando la información entregada por las etiquetas que los mismos usuarios asocian a cada artículo de su interés. Se propone el modelo MM-CTR que permite agregar etiquetas de los documentos al hacer recomendaciones. De acuerdo a los experimentos, se comprueba que el uso de etiquetas permite hacer mejores recomendaciones.

1. INTRODUCCIÓN

Usualmente, los investigadores realizan las búsquedas de artículos que son de su interés siguiendo las citaciones de aquellos que ya han estudiado, esto es completamente sensato pero sus resultados están inevitablemente sesgados hacia artículos altamente citados, además, de esta forma es muy poco probable que haya movimiento entre disciplinas, esto es, que desde un trabajo, por ejemplo, de biología se llegue a otro relacionado, en el campo de la estadística.

Otra alternativa es la búsqueda a través de palabras clave, que a pesar de ser una herramienta poderosa presenta problemas en los casos en los que el investigador no sabe realmente qué está esperando encontrar, además, las búsquedas son realizadas principalmente en base a contenido y no consideran que un artículo valioso es uno que también es valorado por otros usuarios. Al respecto, recientemente sitios como CiteULike[1] han permitido a los investigadores crear sus propias librerías con los artículos que son de su interés agregándoles *tags* descriptivos para ser compartidos por la comunidad. El objetivo del presente trabajo es recomendar a cada usuario un conjunto de artículos que no figuran en su librería personal y dar cuenta de qué forma las etiquetas mejoran la recomendación, ya que al ser ingresadas por los mismos usuarios representan una personalización de la evaluación hecha a cada documento.

Al momento de realizar las recomendaciones se considerarán dos importantes criterios: los usuarios quieren artículos antiguos cuando están buscando introducirse en una nueva área de estudios o cuando quieren aprender los fundamentos de ésta, al respecto, los trabajos importantes estarán en las librerías de muchos usuarios. Por otro lado, los investigadores también necesitan ser informados de las publicaciones más recientes de su disciplina que, al no tener muchas lecturas, deben ser recomendadas en base al contenido de las mismas. Así, se hará uso de dos tipos de datos: el contenido de los artículos y las librerías de otros usuarios que, inherentemente, representan una evaluación de los documentos. En ese sentido, el modelo propuesto que se explicará más adelante es un modelo híbrido, ya que ocupa filtrado colaborativo y el contenido de los documentos para hacer la recomendación. La recomendación es hecha en base a filtrado colaborativo cuando el archivo ha sido evaluado por uno o más usuarios previamente, por otro lado, para el caso de archivos sin evaluación se utilizará modelación de tópicos (*topic modeling*) para permitir recomendaciones acordes a los intereses que ya ha mostrado el usuario.

2. ESTADO DEL ARTE

El presente trabajo aborda dos casos en los que se realizan recomendaciones, primero cuando se tienen documentos que han sido evaluados anteriormente, y segundo cuando se tienen nuevos items sin evaluación previa, este caso es conocido como el *cold start problem*.

Para la recomendación de artículos académicos con y sin evaluaciones previas, en [1] se introduce el método seguido en este trabajo, pero sin la consideración de las etiquetas que permite introducir CiteULike. En [8] agregan las etiquetas y la variable temporal para realizar recomendaciones sobre, entre otras, una base de datos de CiteULike y obtienen un mejor comportamiento en la mayoría de las métricas. Por otra parte, en [7], introducen etiquetas para recomendar páginas web y proponen distintos espacios para la consideración de dichas etiquetas, en este ámbito, en [9] realizan recomendaciones personalizadas de páginas web utilizando la información de las etiquetas para crear los grupos que guían la recomendación.

[1] http://www.citeulike.org

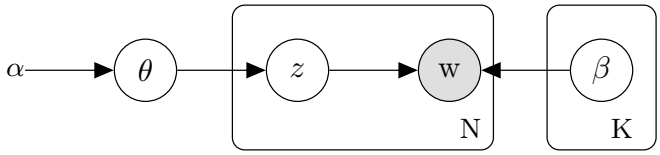


Figure 1: Modelo gráfico de LDA.

3. BACKGROUND

El método propuesto utiliza la factorización matricial usual explicada en [2] para el caso en el que el artículo al que se le desea predecir la evaluación ha sido evaluado con anterioridad, a este caso le llamaremos, siguiendo a [1], *in-matrix prediction*. En el otro caso, en el que no hay evaluaciones previas, *out-of-matrix prediction*, la recomendación es realizada a través de un modelamiento de tópicos como se explicará en la siguiente sección.

3.1 Modelos probabilísticos de tópicos

Modelar tópicos, en este caso, tiene como objetivo dar cuenta de cómo se generan documentos, a través del descubrimiento de tópicos sobre una base de datos con muchos de ellos. El término probabilístico aparece porque cada tópico es representado como una distribución de probabilidad sobre un vocabulario que está centrada en las palabras más representativas del tópico en cuestión.
El modelo más simple de modelación probabilística de tópicos [1] es el llamado LDA por sus siglas en inglés (*Latent Dirichlet Allocation*), cuyo funcionamiento consta principalmente de dos etapas, una asignación de K tópicos para toda la base de datos, y una asignación de palabras para cada tópico, estos pasos se describen a continuación y se grafican en la Figura 1.
Para cada uno de los J documentos:

- Obtener una distribución de tópicos $\theta_j \sim$Dir(α), donde Dir$(\cdot)$ es una distribución Dirichlet uniforme con parámetro α.
- Para cada palabra w_{jn} en el documento:
  - Obtener un tópico específico $z_{jn} \sim$ multi(θ_j), donde multi$(\cdot)$ es una distribución multinomial de parámetro W (la cantidad total de palabras). ón de la evaluación hecha a cada documento.
  - Obtener una palabra $w_{jn} \sim \beta_{z_{jn}}$.

El proceso anterior clarifica de qué forma se hace que cada palabra en un documento provenga de una mezcla de tópicos.

3.2 Multi-Multinomial LDA

El modelo MM-LDA (Multi-Multinomial LDA) [7], busca incluir la información que entregan las etiquetas ingresadas por los usuarios en CiteULike para realizar la recomendación, básicamente agregando una variable a LDA, explicado en la sección anterior, para la generación de etiquetas desde los K tópicos iniciales. El proceso se muestra a continuación y se grafica en la Figura 2.
Para cada uno de los J documentos:

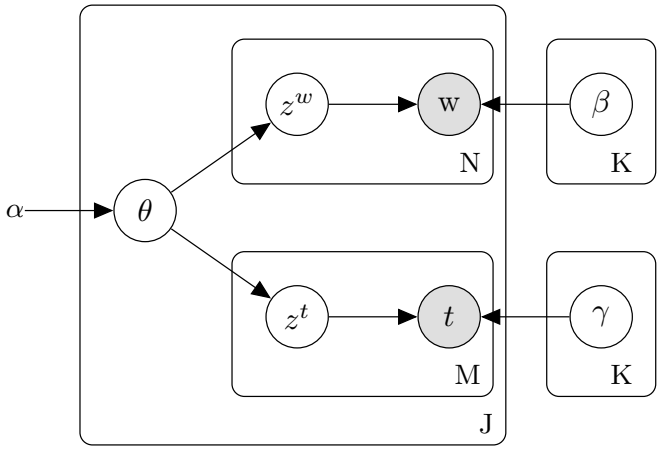


Figure 2: Modelo gráfico de MM-LDA.

- Obtener una distribución de tópicos $\theta_j \sim$ Dir(α).
- Para cada tópico $k = 1, \ldots, K$ obtener una distribución multinomial β_k de tamaño W. β_k es la probabilidad de observar cada palabra, dado el tópico k.
- Para cada tópico $k = 1, \ldots, K$ obtener una distribución multinomial γ_k de tamaño T (cantidad total de etiquetas). γ_k es la probabilidad de observar cada etiqueta, dado el tópico k.
- Para cada palabra w_{jn} del documento j,
  - Obtener un tópico $z^w_{jn} \sim$ Multi(θ_j).
  - Obtener una palabra $w_{jn} \sim \beta_{z^w_{jn}}$.
- Para cada etiqueta (tag) t_{jm} del documento j.
  - Obtener un tópico $z^t_{jm} \sim$ Multi(θ_j).
  - Obtener una etiqueta $t_{jm} \sim$ Multi$(\gamma_{z^t_{jm}})$.

3.3 Collaborative Topic Regression

El modelo *Collaborative Topic Regression* (CTR) combina filtrado colaborativo con modelamiento de tópicos. Es importante destacar que los "ítems" del filtrado colaborativo y los "documentos" de LDA se refieren a lo mismo en este trabajo, y por lo tanto se usarán ambos términos de manera de manera intercambiable.

Al igual que en LDA, en CTR a cada documento se le asigna una proporción de tópicos θ_j, que es usado para generar las palabras. Luego, ocupa esta distribución de tópicos para generar el vector latente de los ítems v_j. El proceso generativo de CTR es el siguiente:

1. Para cada usuario i, se obtiene el vector latente del usuario $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$.
2. Para cada ítem j,
   a) Se obtiene la distribución de tópicos $\theta_j \sim Dirichlet(\alpha)$.
   b) Se obtiene el *offset* latente del ítem $\epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I_K)$ y se asigna el vector latente del ítem el valor $v_j = \epsilon_j + \theta_j$.

c) Para cada palabra w_{jn},

1) Se obtiene el tópico $z_{jn} \sim Cat(\theta_j)$.
2) Se obtiene la palabra $w_{jn} \sim Cat(\beta_{z_{jn}})$.

3. Para cada par usuario-ítem (i, j), se obtiene el *rating* $r_{ij} \sim \mathcal{N}(u_i^T v_j, c_{ij}^{-1})$, donde c_{ij} es el parámetro de precisión para r_{ij}.

4. MULTI-MULTINOMIAL COLLABORATIVE TOPIC REGRESSION

El modelo multi-multinomial *collaborative topic regression* es una extensión al modelo CTR que permite agregar la información de las etiquetas de los documentos para representar de mejor manera los tópicos latentes.

Un primer *approach* para agregar información de los *tags* al modelo, es utilizar esta información como parte del mismo documento. Sin embargo, en [7] se estudio esto y se comprobó empíricamente que hay mejores formas de agregar esta información extra. Insipirado en el modelo propuesto propuesto en [7], se decidió tomar las palabras y los *tags* de los documentos como un conjunto de observaciones independientes. El proceso generativo de MM-CTR es el siguiente:

1. Para cada tópico k,

a) Se obtiene la distribución de palabras $\beta_k \sim Dirichlet(\eta_w)$.
b) Se obtiene la distribución de *tags* $\gamma_k \sim Dirichlet(\eta_t)$.

2. Para cada usuario i, se obtiene el vector latente del usuario $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$.

3. Para cada ítem j,

a) Se obtiene la distribución de tópicos $\theta_j \sim Dirichlet(\alpha)$.
b) Se obtiene el *offset* latente del ítem $\epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I_K)$ y se asigna el vector latente del ítem el valor $v_j = \epsilon_j + \theta_j$.
c) Para cada palabra w_{jn},

1) Se obtiene el tópico $z_{jn}^w \sim Cat(\theta_j)$.
2) Se obtiene la palabra $w_{jn} \sim Cat(\beta_{z_{jn}^w})$.

d) Para cada *tag* t_{jm},

1) Se obtiene el tópico $z_{jm}^t \sim Cat(\theta_j)$.
2) Se obtiene el *tag* $t_{jm} \sim Cat(\gamma_{z_{jm}^t})$.

4. Para cada par usuario-ítem (i, j), se obtiene el *rating* $r_{ij} \sim \mathcal{N}(u_i^T v_j, c_{ij}^{-1})$, donde c_{ij} es el parámetro de precisión para r_{ij}.

El modelo gráfico de la propuesta se puede ver en la Fig. 3.

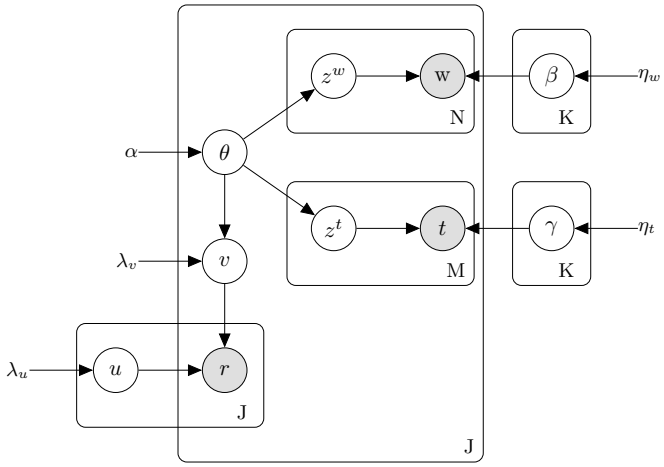


Figure 3: Modelo gráfico de MM-CTR.

4.1 Inferencia

Calcular la posterior completa de u_i, v_i y θ_j es intratable. Por lo tanto, se maximizará la *log likelihood* de U, V, $\Theta = \theta_{1:J}$ y R, dado λ_u, λ_v y $\boldsymbol{\beta}$,

$$\mathcal{L} = \sum_j \sum_m log \left[\sum_k \gamma_{k t_{jm}} \theta_{jk} \right] + \sum_j \sum_n log \left[\sum_k \beta_{k w_{jn}} \theta_{jk} \right] - \frac{\lambda_u}{2} \sum_i u_i^T u_i - \frac{\lambda_v}{2} \sum_j (v_j - \theta_j)^T (v_j - \theta_j) - \sum_i \sum_j \frac{c_{ij}}{2} (r_{ij} - u_i^T v_j)^2 \quad (1)$$

Se omitió una constante y se dejó $\alpha = 1$. Una de las formas que se puede optimizar esta función es utilizando ascenso coordinado (*coordinate ascent*), optimizando iterativamente primero u_i y v_i, y luego la proporción de tópicos θ_j. Otra forma, es obtener primero un θ_j estimado a partir de los documentos y los *tags*, y luego dejar fijo θ_j al optimizar $\mathcal{L}$. Con los datos que se están ocupando, esto no tiene mayores pérdidas de *performance* y ahorra tiempo de computación [1]. En este trabajo se optimizará $\mathcal{L}$ utilizando la segunda opción debido a la necesidad urgente de tener resultados rápido.

Para estimar θ_j, se pueden utilizar varias técnicas. Inferencia Variacional [5] y Gibbs *sampling* [6] son las técnicas generales que más se utilizan para aprender los parámetros en este tipo de modelo. Se decidió ocupar Gibbs *sampling* debido a que es más fácil de implementar. En Gibbs *sampling*, secuencialmente se *samplean* todas las variables latentes condicionadas en todas las demás. En otras palabras, la probabilidad de que un tópico k sea asignado a una palabra w_n depende de la probabilidad de que se haya asignado ese mismo tópico a todas las otras palabras en todos los otros documentos. Formalmente, estas probabilidades se pueden escribir de la siguiente manera

$$p(z_i^w = k \mid \cdot) \propto \frac{C_{w_n,k}^{WT} + \eta_w}{\sum_{w=1}^W C_{wk}^{WT} + W\eta_w} \frac{C_{d_i,k}^{DT} + \alpha}{\sum_{k=1}^K C_{d_i,k}^{DT} + K\alpha} \quad (2)$$

$$p(z_i^t = k \mid \cdot) \propto \frac{C^{TT}_{t_n,k} + \eta_t}{\sum_{t=1}^{M} C^{TT}_{tk} + M\eta_t} \frac{C^{DT}_{d_i,k} + \alpha}{\sum_{k=1}^{K} C^{DT}_{d_i,k} + K\alpha} \quad (3)$$

donde C^{WT}_{wk} es el número de palabras w asignadas al tópico k, C^{TT}_{tk} es el número de *tags* t asignados al tópico k y C^{DT}_{dk} es el número de documentos d asignados al tópico k. Finalmente, la distribución de tópicos θ se puede estimar utilizando la siguiente expresión

$$\theta_d = \frac{C^{DT}_{dk} + \alpha}{\sum_{k=1}^{K} C^{DT}_{dk} + K\alpha} \quad (4)$$

donde C^{DT}_{dk} es el número de documentos d asignados al tópico k. Notar que la expresión para estimar θ_d es la misma que en [6].

Dado θ_j, se puede estimar u_i y v_j de manera similar a como se hace al hacer factorización matricial [4]. Calculando el gradiente de $\mathcal{L}$ con respecto a u_i y v_j, y luego igualando a cero, se obtiene

$$u_i \leftarrow (VC_iV^T + \lambda_u I_k)^{-1} VC_iR_i \quad (5)$$

$$v_j \leftarrow (UC_jU^T + \lambda_v I_k)^{-1} (UC_jR_j + \lambda_v \theta_j), \quad (6)$$

donde $U = (u_i)_{i=1}^{I}$, $V = (v_j)_{j=1}^{J}$, C_i es una matriz diagonal con $c_{i,j}$ $j = 1, \ldots, J$ como sus elementos diagonales y $R_i = (r_{ij})_{j=1}^{J}$ para el usuario i. En la ecuación 6 se puede observar como afecta la proporción de tópicos del documento, θ_j, al vector latente v_j, donde λ_v actúa como el regularizador.

4.2 Predicción

Después de estimar los parámetros U, V y Θ, la predicción se hace igual que en el modelo CTR, es decir, para *in-matrix prediction*, se puede estimar r_{ij} con

$$r_{ij} \approx (u_i)^T(\theta_j + \epsilon_j) = u_i^T v_j \quad (7)$$

donde $v_j = \theta_j + \epsilon_j$. Para una predicción *out-of-matrix*, el artículo es nuevo y por lo tanto no hay *ratings* disponibles. Luego, r_{ij} se calcula como

$$r_{ij} \approx u_i^T \theta_j \quad (8)$$

5. EXPERIMENTOS

5.1 Dataset

La base de datos utilizada es la misma que en [1]. Este *dataset* tiene usuarios y documentos que esos usuarios tienen en sus librerías de CiteULike[2]. A este *dataset*, además se le agregó la información de las etiquetas entregadas por los usuarios, que fueron obtenidas entrecruzando la primera base de datos con el *snapshot*[3] de CiteULike.

Debido a que se contaba con pocos recursos computacionales, el *dataset* original fue reducido. El resumen del *dataset* se puede observar en la Tabla 1.

Table 1: Resumen de la base de datos.

Ítem	# de instancias
Documentos	5.000
Usuarios	1.749
Pares Usuario-Documento	8.745.000
tags	56.077

Con respecto a los artículos, se concatenó el título y el *abstract* de cada uno. Luego, se preprocesó cada artículo. En este preprocesamiento, se tokenizó el texto, eliminando los *tokens* que correspondían a caracteres de puntuación, palabras con menos de 2 caracteres, palabras que incluían un número y las *stop words* del inglés. Cada palabra se pasó a minúscula, y luego se aplicó lematización. Una vez que se aprendió el vocabulario, se aplicó el método tf-idf y se seleccionaron las 6.000 palabras más importantes. El vocabulario final consta por lo tanto, de 6.000 palabras diferentes para los documentos, lo que conforma un *corpus* de 340 mil palabras. En promedio, cada artículo tiene 69 palabras.
Con respecto a los *tags*, basado en [7], no se realizó un mayor preprocesamiento. Lo único que se hizo fue convertir los *tags* a minúscula, y eliminar los *tags* que aparecen menos de dos veces en todos los documentos. El vocabulario de los *tags* cuenta con 6.127 palabras diferentes. En promedio, cada artículo tiene 11 *tags*. En el Cuadro 2 se puede observar un cuadro resumen donde se muestran los *tags* con mayor frecuencia, donde se incluye el número de documentos distintos donde aparece cada uno.

Table 2: Resumen de los 10 *tags* de mayor frecuencia.

#	Tag	# de ocurrencias	# de documentos
1	networks	1903	433
2	network	1329	386
3	import	1280	1185
4	bibtex	1151	1069
5	social	1012	229
6	review	958	344
7	information	951	300
8	learning	893	256
9	theory	872	341
10	evolution	808	262

5.2 Evaluación

La evaluación se hará en artículos que se sacaron de las librerías de los usuarios (*held out data*). Se recomendarán M documentos a los usuarios, y se verá si esos documentos estaban o no en su librería personal.

Para poder compararse mejor con [1], se utilizó como métrica de evaluación el *recall*. La razón de esto, es que se conocen los documentos que están en la librería del usuario, y que

[2]http://www.citeulike.org/

[3]http://www.citeulike.org/faq/data.adp

por lo tanto tienen $r_{ij} = 1$, pero eso no quiere decir que los documentos con *rating* $r_{ij} = 0$ no le hayan gustado a ese usuario. Quizás que el usuario simplemente no los ha leído. Por lo tanto, utilizando el *recall*, se evaluarán los verdaderos positivos. El *recall* se calcula como

$$recall@M = \frac{\text{n de docs que le gustan al usuario en top M}}{\text{total de docs que le gustan al usuario}}$$

Como se dijo anteriormente, se consideraron dos formas de recomendación, *in-matrix prediction* y *out-of-matrix prediction*.

5.2.1 *In-matrix prediction*

In-matrix prediction toma el caso en que un usuario tiene un conjunto de artículos que no ha visto, pero que otros usuarios sí han visto.

Esta forma de predicción es muy similar al filtrado colaborativo. Para evaluar este caso, se utilizó una validación cruzada de 5-*folds*. Se pusieron en los conjunto de pruebas solo los artículos que se encuentran en las librerías de más de 5 usuarios. De este modo es seguro que todos los artículos que estan en el conjunto de prueba hayan aparecido en el conjunto de entrenamiento.

Se corrió el modelo utilizando el conjunto de entrenamiento y se hicieron las M mejores predicciones para el conjunto de prueba.

5.2.2 *Out-of-matrix prediction*

La predicción *out-of-matrix* es el caso en que hay un artículo nuevo que nadie ha visto. En este caso no es tan real el uso de *tags*, ya que si el artículo es nuevo no tendría ninguna etiqueta, pero se estudiará de todas formas.

Esta forma de predicción es puramente basada en contenido. Al igual que antes, se hará validación cruzada de 5-*folds*. En cada *fold*, se dejan 1/5 de los artículos como conjunto de prueba, y el resto de los artículos queda como el conjunto de prueba. En este caso, no se entrena con ningún artículo que está en el conjunto de pruebas, es decir, al entrenar los usuarios no tienen esos artículos en su librería.

Nuevamente se corrió el modelo utilizando el conjunto de entrenamiento y se hicieron las mejores M predicciones con el conjunto de prueba.

5.3 Configuración de experimentos

La implementación de Gibbs *sampling* se hizo modificando una versión ya existente de este algoritmo para LDA, la cual estaba en Cython[4]. La optimización de u_i y v_j se hizo en Python puro. Todos los experimentos se corrieron en un computador con un procesador Intel Xeon 2.6GHz y 16GB de memoria RAM.

Para comparar el modelo propuesto, se utilizará el modelo CTR. Además, para comparar el *performance* de manera más justa, se corrió el modelo CTR con la información de las etiquetas como parte de los documentos. Para esto, se definieron dos vocabularios separados, el vocabulario del contenido del documento, y el vocabulario de las etiquetas. Luego, estos dos vocabularios se concatenaron. Se llamará a este modelo CTR-tags.

[4]http://cython.org/

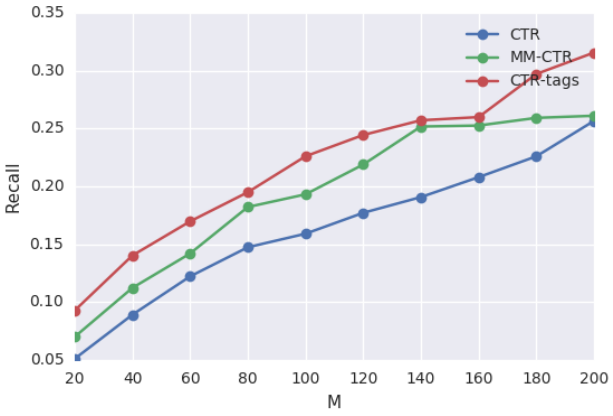


Figure 4: Comparación del *recall* en *in-matrix prediction* de los modelo CTR, MM-CTR y CTR utilizando los tags (CTR-tags) al variar el número de recomendaciones M, con $\lambda_v = 100$.

Todos los experimentos se corrieron con 80 tópicos. Además, se fijó el valor de $\alpha = 1$, $\eta_w = 0,1$ y $\eta_t = 0,1$. Por otro lado, $\lambda_u = 0,01$.

5.4 Resultados

En la Fig. 4 se puede observar el resultado del modelo MM-CTR, CTR y CTR-tags cuando se varía el número de recomendaciones $M = 20, 40, \ldots, 200$. Para todos se fijó $\lambda_v = 100$. Como se puede observar en la figura, al añadir la información de los *tags* efectivamente se tiene una mejora en la *performance* del modelo propuesto y del modelo CTR. Sin embargo, y contrario al resultado obtenido en [7], se obtuvo mejor *recall* al añadir la información de las etiquetas como palabras nuevas del documento que al añadirlas como nuevas observaciones. Además, al igual que los resultados que se obtuvo en [1], a medida que aumenta M se obtiene mejor un mejor *recall*.

En la Fig. 5 se estudia el efecto del número de recomendaciones M al hacer predicciones *out-of-matrix*. Como se ve en la figura, a medida que aumenta el número de recomendaciones M, el *recall* aumenta, al igual que en [1]. Además, se observa que en este caso el modelo MM-CTR es superior al modelo CTR utilizando etiquetas. En este caso, el uso de etiquetas también es superior a solo ocupar el contenido de los documentos.

En la Fig. 7 se estudió el efecto de λ_v en las recomendaciones. Como se explicó anteriormente, λ_v es el parámetro que dice cuánto pesa el contenido en comparación al los *ratings*. Cuando λ_v es pequeño, el vector latente v_j puede diferir bastante de la proporción de tópicos θ_j. Por otro lado, cuando λ_v es grande, se tiene el efecto contrario, el vector latente v_j es más parecido a la proporción de tópicos, y por lo tanto,

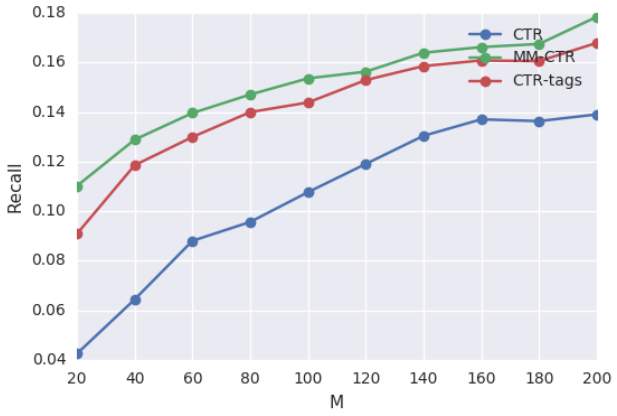


Figure 5: Comparación del *recall* en *out-of-matrix prediction* de los modelos CTR, MM-CTR y CTR utilizando los tags (CTR-tags) al variar el número de recomendaciones M, con $\lambda_v = 100$.

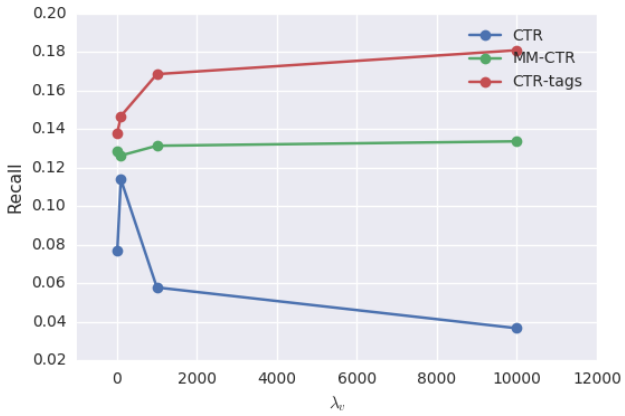


Figure 6: Comparación del *recall* en *in-matrix prediction* entre el modelo CTR, MM-CTR y CTR utilizando los tags (CTR-tags) al variar λ_v, con $M = 50$.

el contenido de los documentos tiene más peso.

Se corrieron experimentos con $\lambda_v = 10, 100, 1000, 10000$, con $M = 50$. Como se observa en la figura, a medida que aumenta λ_v, el modelo CTR tiene menor *performance*, lo que quiere decir que el contenido de los documentos por sí solo no basta para hacer una buena recomendación. Por otro lado, en CTR-tags y MM-CTR, el *recall* aumenta a medida que aumenta λ_v. Esto confirma que las etiquetas que la comunidad agrega a los documentos sí son una fuente de información importante, y pueden suplir la información de los *ratings* de los demás usuarios. Sin embargo, nuevamente añadir las etiquetas como parte del documento tiene mejor *performance* que el modelo propuesto.

Con la Tabla 3 y 4 se puede hacer un análisis más cualitativo de los resultados. En la Tabla 3 se muestran los tópicos más preferidos por el usuario U1, junto con las palabras y los *tags* que mejor describen ese tópico, y en la Tabla 4 se

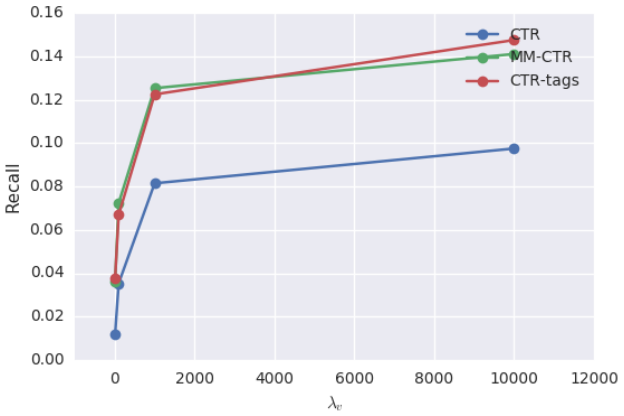


Figure 7: Comparación del *recall* en *out-of-matrix prediction* entre el modelo CTR, MM-CTR y CTR utilizando los tags (CTR-tags) al variar λ_v, con $M = 50$.

muestra el resultado de una recomendación de 10 documentos para ese mismo usuario. En la tabla de los tópicos se puede observar que las palabras y los *tags* de ese tópico sí están relacionados. Por ejemplo, el tópico 1 tiene palabras como cerebro, señales, y actividad, y ese mismo tópico tiene etiquetas como neuroimagen (*neuroimaging* en inglés) y descanso. Esta última etiqueta puede tener que ver con la actividad del cerebro en períodos de descanso.

Con respecto a la recomendación que se le hizo al mismo usuario U1, no se tuvieron buenos resultados, ya que se recomendó solo 3 documentos que eran parte de la librería del usuario. Sin embargo, se puede ver que algunos documentos recomendados sí tenían que ver con los tópicos preferidos por el usuario de acuerdo al modelo MM-CTR. Por ejemplo, el documento 1 se refiere al aprendizaje en la sala de clases, y el tópico 2 incluye palabras como "estudiante", y "libro". Además, varios documentos recomendados están relacionados con la biología, y uno de los topicos que el usuario prefiere tiene que ver con la actividad del cerebro, temas que están relacionados de alguna manera.

6. CONCLUSIONES

La primera conclusión del trabajo es que efectivamente las etiquetas que los usuarios agregan a los documentos son una fuente de información adicional. Al usar etiquetas se obtuvo una mejor *performance* en todos los experimentos que se corrieron.

En este trabajo se propuso un modelo que tenía la intención de ser efectivo a la hora de agregar etiquetas al modelo CTR. Sin embargo, como se vio en los resultados el modelo propuesto no fue mejor que considerar las etiquetas como parte de los documentos. En ese sentido, no se confirmaron los resultados de [7]. Es necesario hacer más experimentos y verificar que la implementación de MM-CTR fue correcta, lo que queda como trabajo a futuro. Además, es necesario ver cómo se comportan los modelos al cambiar el número de tópicos o el valor de los *priors*.

Table 3: Top 3 tópicos del usuario U1

Top 3 tópicos	
1	**palabras**: brain, functional, region, signal, task, subject, result, activity, suggest, human **tags**: fmri, brain, imaging, state, resting, methods, bold, meditation, mvpa, neuroimaging
2	**palabras**: book, include, topic, author, edition, chapter, new, student, introduction, cover **tags**: book, books, textbook, first, compression, computer, temporary, econometrics, textbook, skr
3	**palabras**: large, high, number, level, order, small, low, cost, however, performance **tags**: fpga, reconfigurable, rc, reconfigurable_computing, trends, reconfigurable_platforms, threads, classic, hierarchical, level

Table 4: Top 10 documentos recomendados para usuario U1. Los documentos marcados con una Y quiere decir que ese documento sí era parte de la librería del usuario.

Top 10 documentos	Título documento	
1	Assessment and classroom learning	X
2	Friends and neighbors on the web	X
3	Kernel methods for relation extraction	X
4	Persuasive technology: using computers to change what we think and do	Y
5	GroupLens: An Open Architecture for Collaborative Filtering of Netnews	X
6	Reverse Engineering of Biological Complexity	X
7	Sequencing and comparison of yeast species to identify genes and regulatory elements	Y
8	A database of macromolecular motions	X
9	A phylogenomic approach to bacterial phylogeny: evidence of a core of genes sharing a common history	X
10	The Stanford Microarray Database	Y

Por último, el modelo propuesto produce resultados fácilmente interpretables. Si bien no se obtuvieron los mismos resultados que en [1], al ver los tópicos que los usuarios prefieren y sus recomendaciones se ve que el modelo no funciona tan mal.

7. REFERENCES

[1] Wang, C., & Blei, D. M. (2011, August). Collaborative topic modeling for recommending scientific articles. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 448-456). ACM.

[2] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

[3] Murphy, Kevin P. (2012). Machine learning : a probabilistic perspective. Cambridge, MA: MIT Press. pp. 151–152.

[4] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[5] D.M. Blei and M.I. Jordan. Modeling annotated data. In SIGIR '03.

[6] T.L. Griffiths. Finding scientific topics. Proceedings of the National Academy of Sciences, 101:5228–5235, '04

[7] Ramage, D., Heymann, P., Manning, C. D., & Garcia-Molina, H. (2009, February). Clustering the tagged web. In Proceedings of the Second ACM International Conference on Web Search and Data Mining (pp. 54-63). ACM.

[8] Lacic, E., Kowald, D., Seitlinger, P., Trattner, C., & Parra, D. (2014). Recommending items in social tagging systems using tag and time information. arXiv preprint arXiv:1406.7727.

[9] Shepitsen, A., Gemmell, J., Mobasher, B., & Burke, R. (2008, October). Personalized recommendation in social tagging systems using hierarchical clustering. In Proceedings of the 2008 ACM conference on Recommender systems (pp. 259-266). ACM.