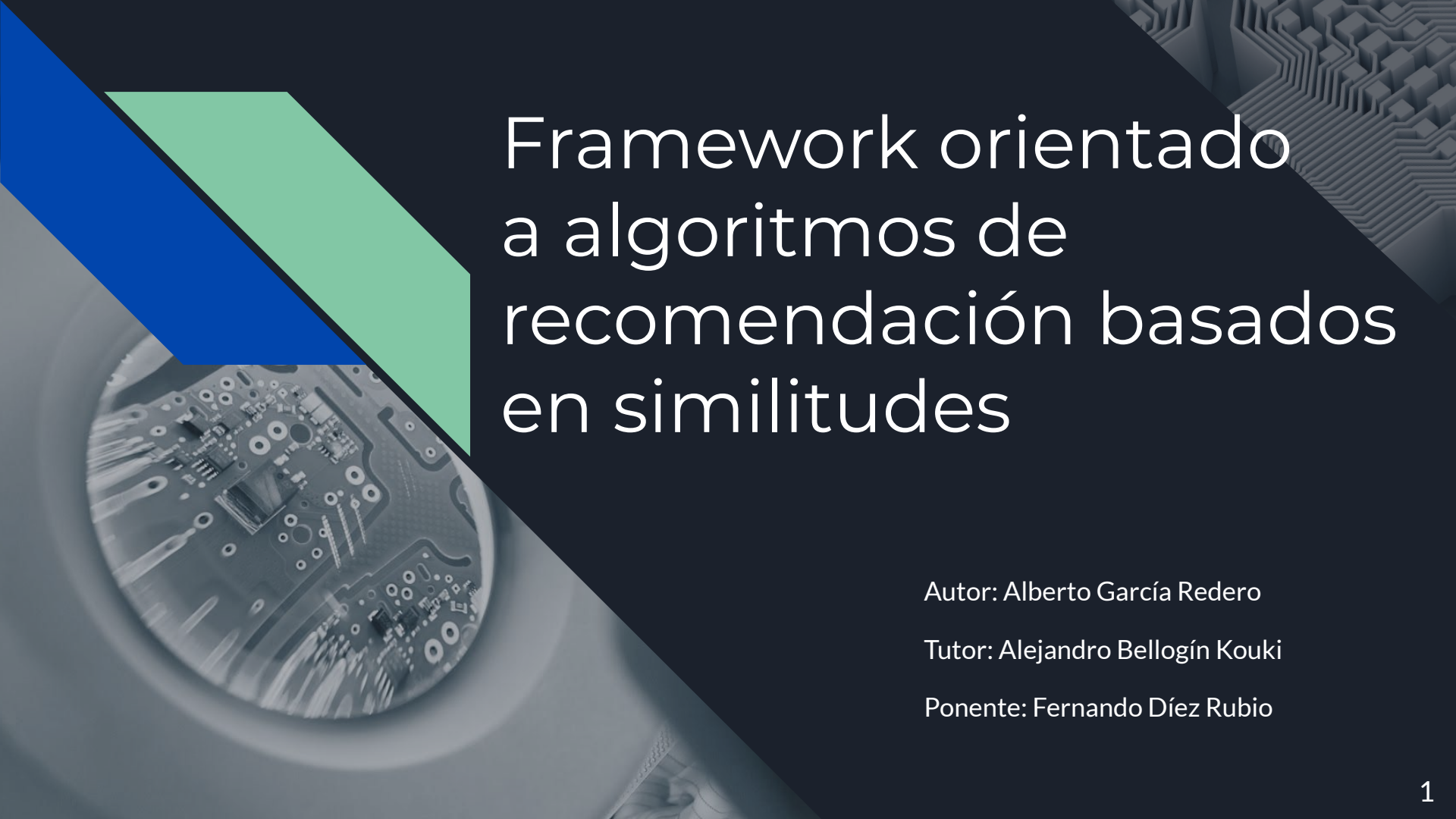


Framework orientado a algoritmos de recomendación basados en similitudes

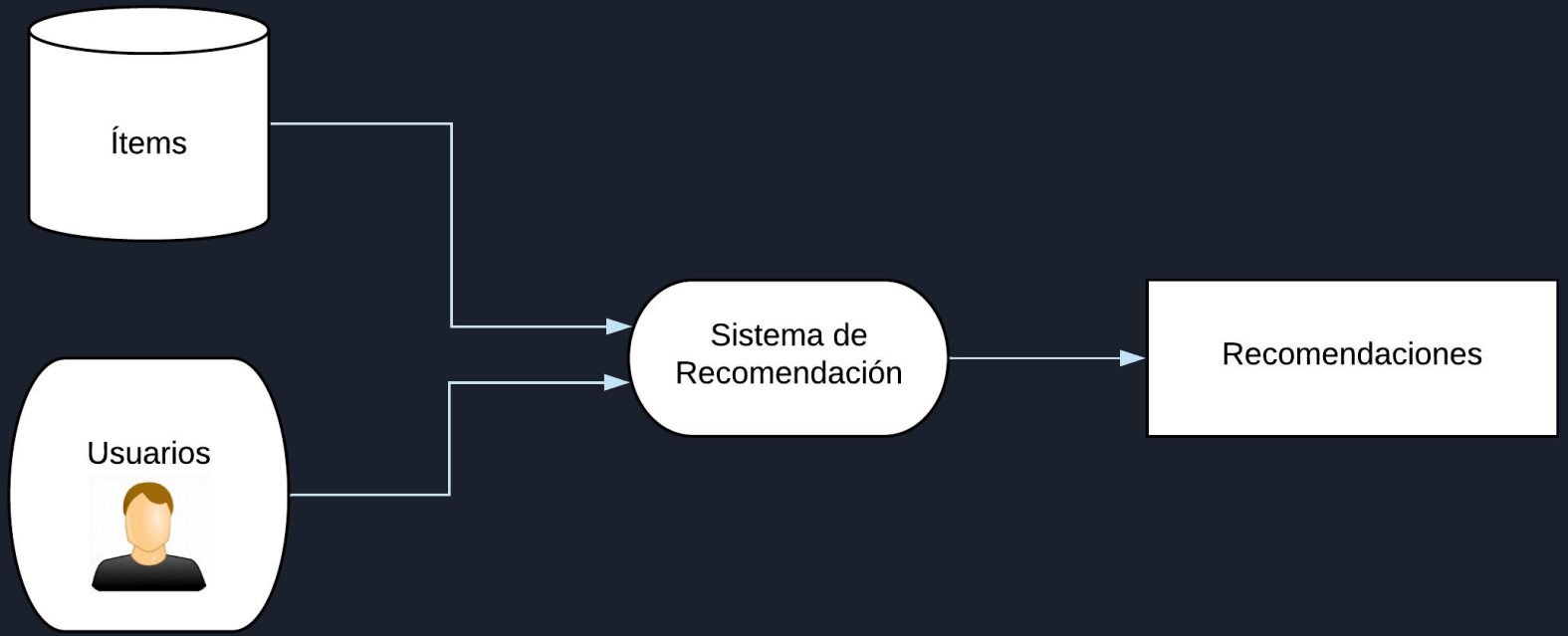
Autor: Alberto García Redero

Tutor: Alejandro Bellogín Kouki

Ponente: Fernando Díez Rubio

Índice

Introducción - Los Sistemas de Recomendación



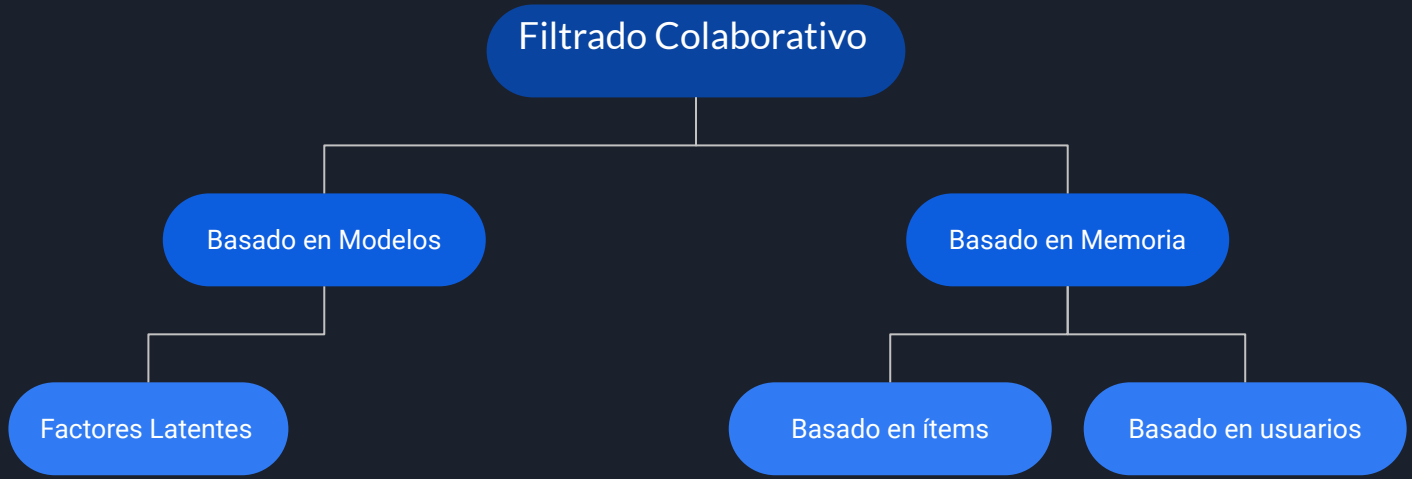
Objetivos

- Construcción de un recomendador **basado en similitudes**.

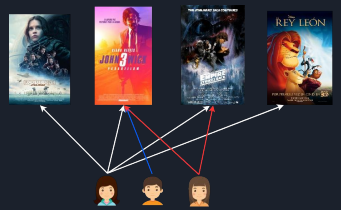
- **Estudiar distintas variantes** del recomendador para comprobar su funcionamiento.

- Comprobar su **rendimiento** frente a algunos de los Sistemas de Recomendación más comunes actualmente.

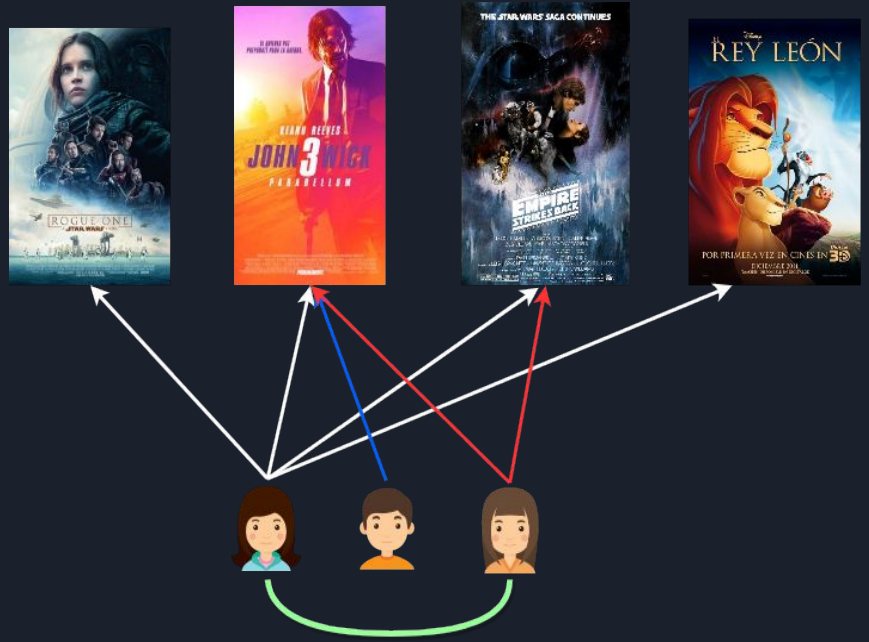
Estado del arte - Filtrado Colaborativo



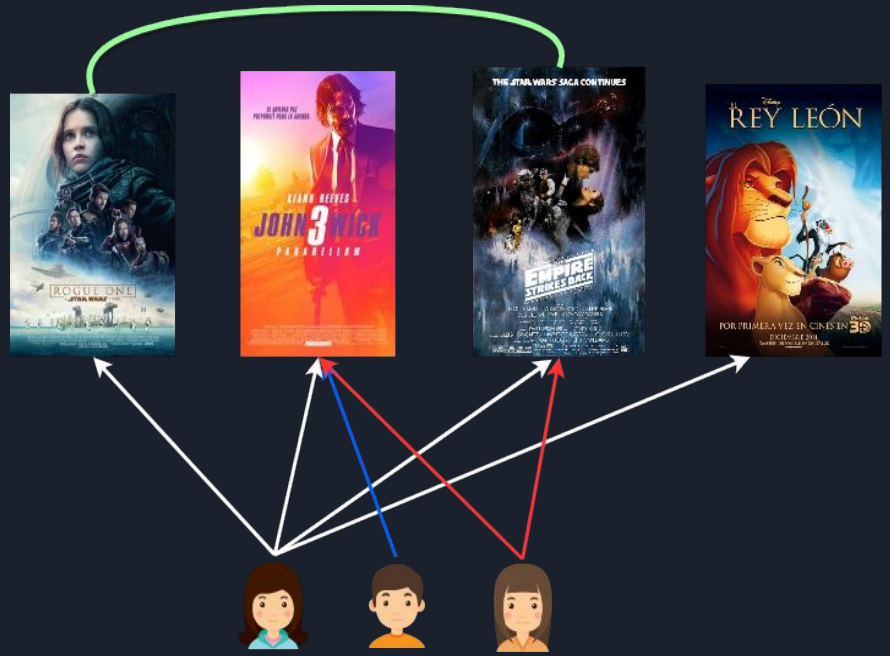
0.62	0.43	0.95	0.72	0.82	0.26	0.34



Estado del arte - Vecinos próximos



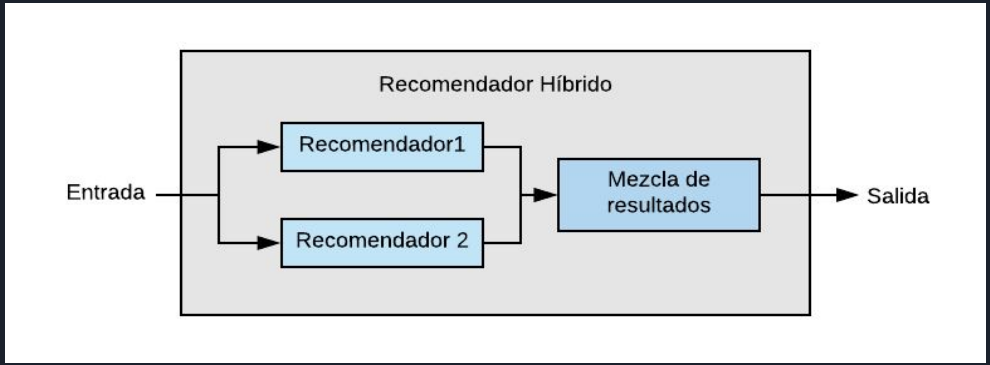
Orientado a usuarios

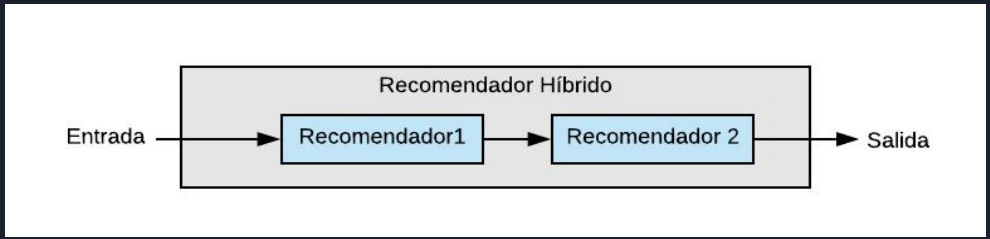


Orientado a ítems

Estado del arte - Modelos híbridos

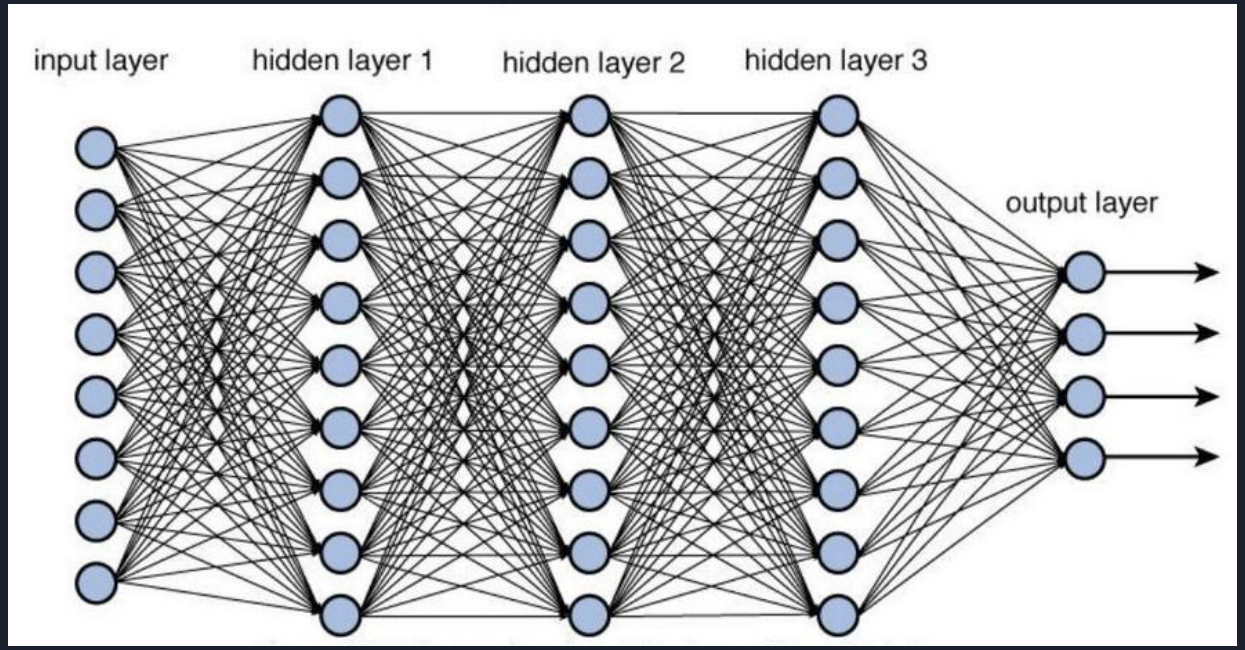
Ejemplo de estructura de Recomendadores Híbridos





Solución implementada en el proyecto.

Estado del arte - Redes Neuronales

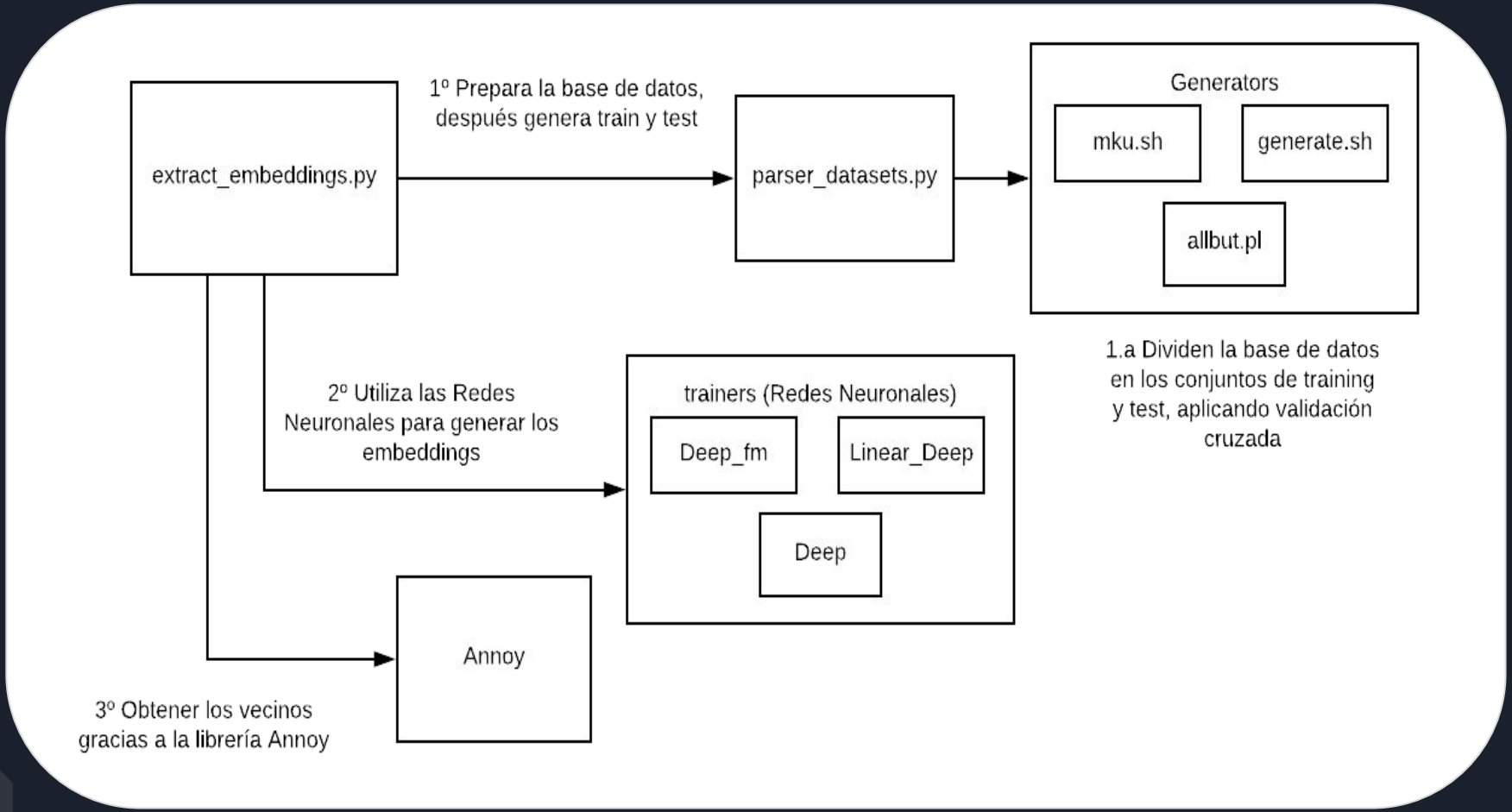


Red Profunda

Solución Propuesta

Modelo Híbrido en Cascada, en el que cada subrecomendador realiza las siguientes acciones:

1. La Red Neuronal Profunda extrae los embeddings (factores latentes) de los elementos.
2. Estos son utilizados por Vecinos Próximos para realizar las recomendaciones.


extract_embeddings.py
1º Prepara la base de datos, después genera train y test
parser_datasets.py
Generators
mku.sh
generate.sh
allbut.pl
1.a Dividen la base de datos en los conjuntos de training y test, aplicando validación cruzada
2º Utiliza las Redes Neuronales para generar los embeddings
trainers (Redes Neuronales)
Deep_fm
Linear_Deep
Deep
Annoy
3º Obtener los vecinos gracias a la librería Annoy

Implementación - Subsistema de estandarización de los Datos

Problema inicial:

- Los datasets pueden tener una estructura y características distintas.

Por lo que necesitamos una manera de estandarizar los datos para que puedan ser utilizados por las Redes Neuronales.

Implementación - Subsistema de generación de Embeddings

- Punto de partida del proyecto: GITHUB[1]
- Objetivo: entrenar las Redes generando a partir de ellas los embeddings.
- Redes Neuronales que lo componen:
  - Deep_fm
  - Deep
  - Linear_Deep

[1] https://github.com/yxtay/recommender-tensorflow

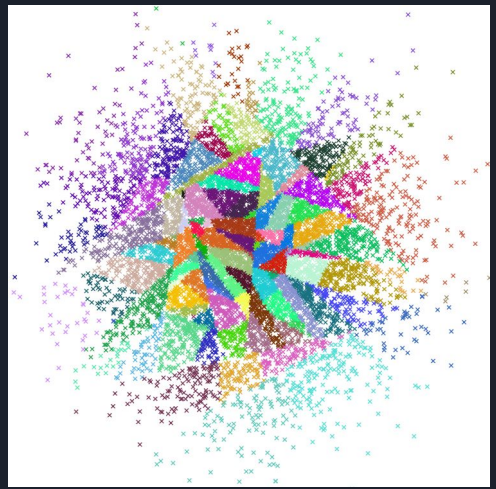
Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... Shah, H. (2016). Wide & Deep Learning for Recommender Systems. arXiv:1606.07792 [cs.LG].

Implementación - Subsistema de extracción de vecinos

Utilizamos Annoy.

Ventajas:

- Rápida.
- Utilizada por Spotify.
- Estructura de datos eficiente.
- Fácil de integrar en Python.



Funcionalidad - Películas similares

```
Movie: Star Wars (1977)
49                          Star Wars (1977)
194                  Terminator, The (1984)
198    Bridge on the River Kwai, The (1957)
277                     Bed of Roses (1996)
690                        Dark City (1998)
```

deep_fm

```
Movie: Star Wars (1977)
49                  Star Wars (1977)
172     Princess Bride, The (1987)
267              Chasing Amy (1997)
418             Mary Poppins (1964)
1644       Butcher Boy, The (1998)
```

deep

```
Movie: Star Wars (1977)
15                     French Twist (Gazon maudit) (1995)
49                                          Star Wars (1977)
229                  Star Trek IV: The Voyage Home (1986)
489                                 To Catch a Thief (1955)
1532    I Don't Want to Talk About It (De eso no se ha...
```

linear_deep

Funcionalidad - Recomendar al usuario

```
User: 259
45                                  Exotica (1994)
63                  Shawshank Redemption, The (1994)
87                       Sleepless in Seattle (1993)
137                      D3: The Mighty Ducks (1996)
182                                   Alien (1979)
189                                 Henry V (1989)
221                  Star Trek: First Contact (1996)
```

deep_fm

```
User: 259
43                          Dolores Claiborne (1994)
48                                    I.Q. (1994)
52                        Natural Born Killers (1994)
165       Manon of the Spring (Manon des sources) (1986)
212                     Room with a View, A (1986)
217                             Cape Fear (1991)
228          Star Trek III: The Search for Spock (1984)
```

deep

```
User: 259
74      Brother Minister: The Assassination of Malcolm...
79                        Hot Shots! Part Deux (1993)
80                        Hudsucker Proxy, The (1994)
114       Haunted World of Edward D. Wood Jr., The (1995)
160                                 Top Gun (1986)
192                          Right Stuff, The (1983)
230                            Batman Returns (1992)
```

linear_deep

Funcionalidad

Películas más populares

	movieId	title	genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
6	7	Twelve Monkeys (1995)	Drama|Sci-Fi
49	50	Star Wars (1977)	Action|Adventure|Romance|Sci-Fi|War
55	56	Pulp Fiction (1994)	Crime|Drama
97	98	Silence of the Lambs, The (1991)	Drama|Thriller
99	100	Fargo (1996)	Crime|Drama|Thriller
120	121	Independence Day (ID4) (1996)	Action|Sci-Fi|War
126	127	Godfather, The (1972)	Action|Crime|Drama
173	174	Raiders of the Lost Ark (1981)	Action|Adventure
180	181	Return of the Jedi (1983)	Action|Adventure|Romance|Sci-Fi|War
257	258	Contact (1997)	Drama|Sci-Fi
285	286	English Patient, The (1996)	Drama|Romance|War
287	288	Scream (1996)	Horror|Thriller
293	294	Liar Liar (1997)	Comedy
299	300	Air Force One (1997)	Action|Thriller

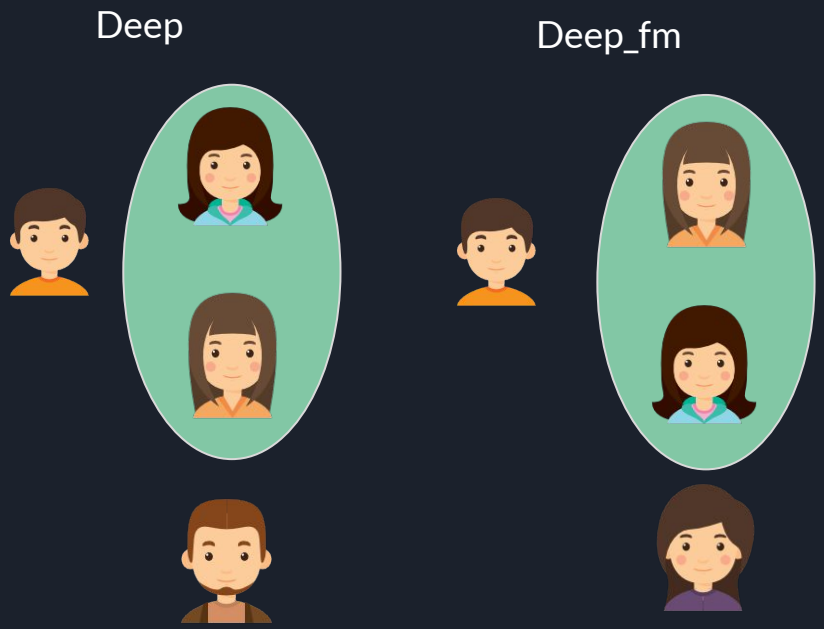
Pruebas y resultados

- Diferencias entre las Redes Neuronales implementadas.
- Comparación de vecinos del modelo híbrido frente al estándar.
- Comparación con otros sistemas de recomendación.

Datasets utilizados:

- Movielens 100K. 943 usuarios y 1682 ítems.
- Movielens 20M. 138493 usuarios y 27278 ítems.

Método comparación de los vecinos de las distintas Redes utilizadas



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

$$J(Deep, Deep\_fm) = \frac{2}{3+3-2} = \frac{1}{2}$$

Pruebas y resultados - Diferencias entre las Redes Neuronales implementadas

	@5	@10	@15	@20	@50
Deep vs Deep_fm	1.661E-5	2.975E-5	4.641E-5	6.261E-5	1.545E-4
Deep_fm vs Linear_deep	1.704E-5	3.062E-5	5.334E-5	6.17E-5	1.527E-4
Deep vs Linear_deep	1.718E-5	3.293E-5	5.368E-5	7.134E-5	1.775E-4

Jaccard entre usuarios de las distintas redes.

	@5	@10	@15	@20	@50
Deep vs Deep_fm	1.188E-4	2.290E-4	3.645E-4	4.257E-4	1.024E-3
Deep_fm vs Linear_deep	1.015E-4	1.588E-4	2.615E-4	3.785E-4	9.191E-4
Deep vs Linear_deep	1.102E-4	1.739E-4	2.636E-4	3.592E-4	9.150E-4

Jaccard entre ítems de las distintas redes.

Pruebas y resultados - Comparación de usuarios e ítems

	Precisión	Recall	nDCG	EPC	AD	Gini
Modelo Híbrido, Deep	0.025	0.041	0.035	0.829	20693.19	0.188
Modelo Híbrido, Deep_fm	0.025	0.042	0.042	0.829	20259.59	0.194
Modelo Híbrido, Linear_deep	0.025	0.042	0.034	0.829	20259.59	0.191
Modelo estándar	0.019	0.032	0.026	0.889	8694.60	0.078

Comparación vecinos orientados a usuarios del modelo estándar vs el modelo híbrido.

	Precisión	Recall	nDCG	EPC	AD	Gini
Modelo Híbrido, Deep	5.832E-4	9.198E-4	7.466E-4	0.988	8460.60	0.117
Modelo Híbrido, Deep_fm	5.083E-4	8.431E-4	6.299E-4	0.977	8366.80	0.103
Modelo Híbrido, Linear_deep	5.468E-4	7.967E-4	6.073E-4	0.978	8507.20	0.122
Modelo estándar	0.023	0.039	0.032	0.849	20259.59	0.226

Comparación vecinos orientados a ítems del modelo estándar vs el modelo híbrido.

Pruebas y resultados - Comparación con otros sistemas de recomendación

	Precisión	Recall	nDCG	EPC	AD	Gini
Random	2.563E-4	4.329E-4	3.474E-4	0.993	19752.01	0.844
Popularity	0.027	0.045	0.037	0.818	8596.69	0.048
Factorización de Matrices	0.025	0.042	0.036	0.840	1913.99	0.020
Mejor modelo estándar	0.023	0.039	0.032	0.849	20259.59	0.226
Mejor modelo Híbrido	0.025	0.042	0.042	0.829	20259.59	0.194

Resultados obtenidos contra otros recomendadores populares.

Discusión Resultados

- Mejora a Vecinos Próximos en la versión orientada a usuarios.
- Es posible obtener buenos resultados sin necesidad de utilizar Redes Neuronales muy complejas.
- Redes Neuronales similares devuelven resultados dispares pero de rendimiento parecido.

Conclusiones

- He aprendido nuevas técnicas de cálculo de vecinos. (Annoy)
- He practicado el uso de Redes Neuronales y mejorado mis conocimientos.
  - He descubierto qué son los embeddings y sus aplicaciones.
  - TensorFlow.
- He ganado experiencia utilizando python3, Java8 y en menor medida C++.

El framework resultante del proyecto se encuentra disponible en: https://bitbucket.org/albergr/tfg/

Trabajo Futuro

- Probar el sistema con otro tipo de Red Neuronal. Por ejemplo, Convolucional o Recurrente.
- Crear la red con Keras en vez de con estimadores.
- Estudiar en profundidad las métricas de distancia que Annoy nos aporta.
- Probar distintos tamaños del vector de embeddings.
- Comparar la caracterización de los elementos realizada por las redes con la factorización de matrices

Preguntas

Estructura TensorFlow

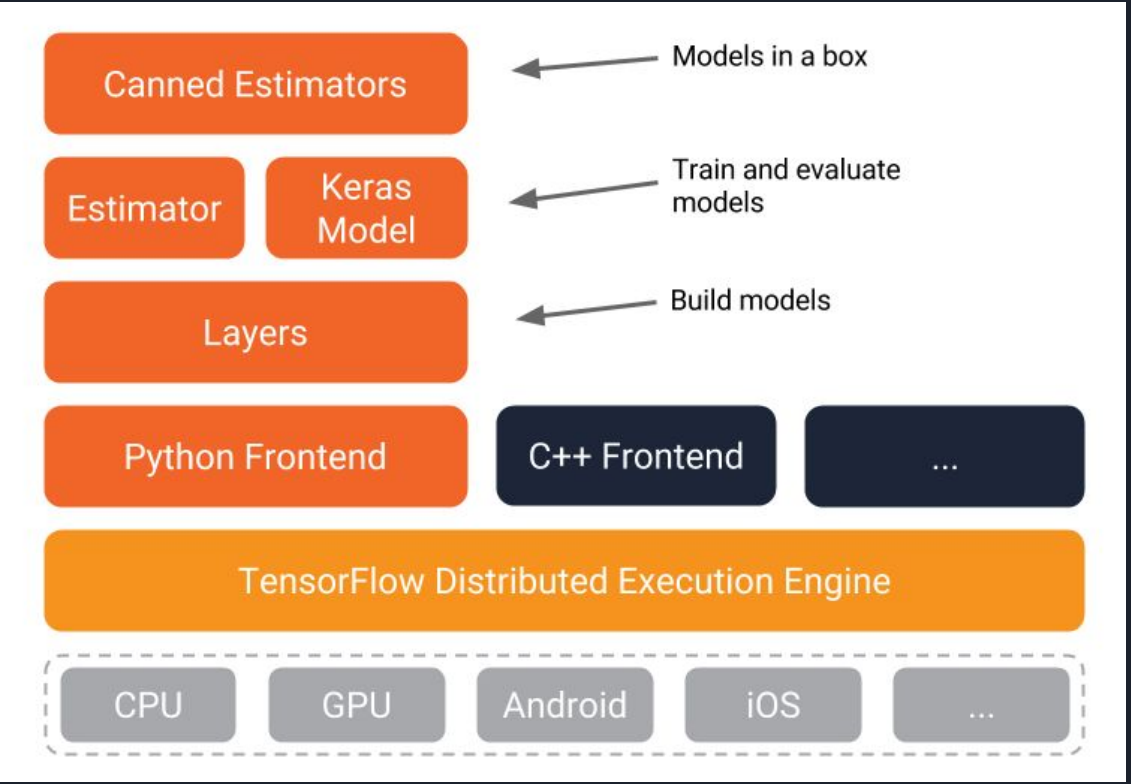
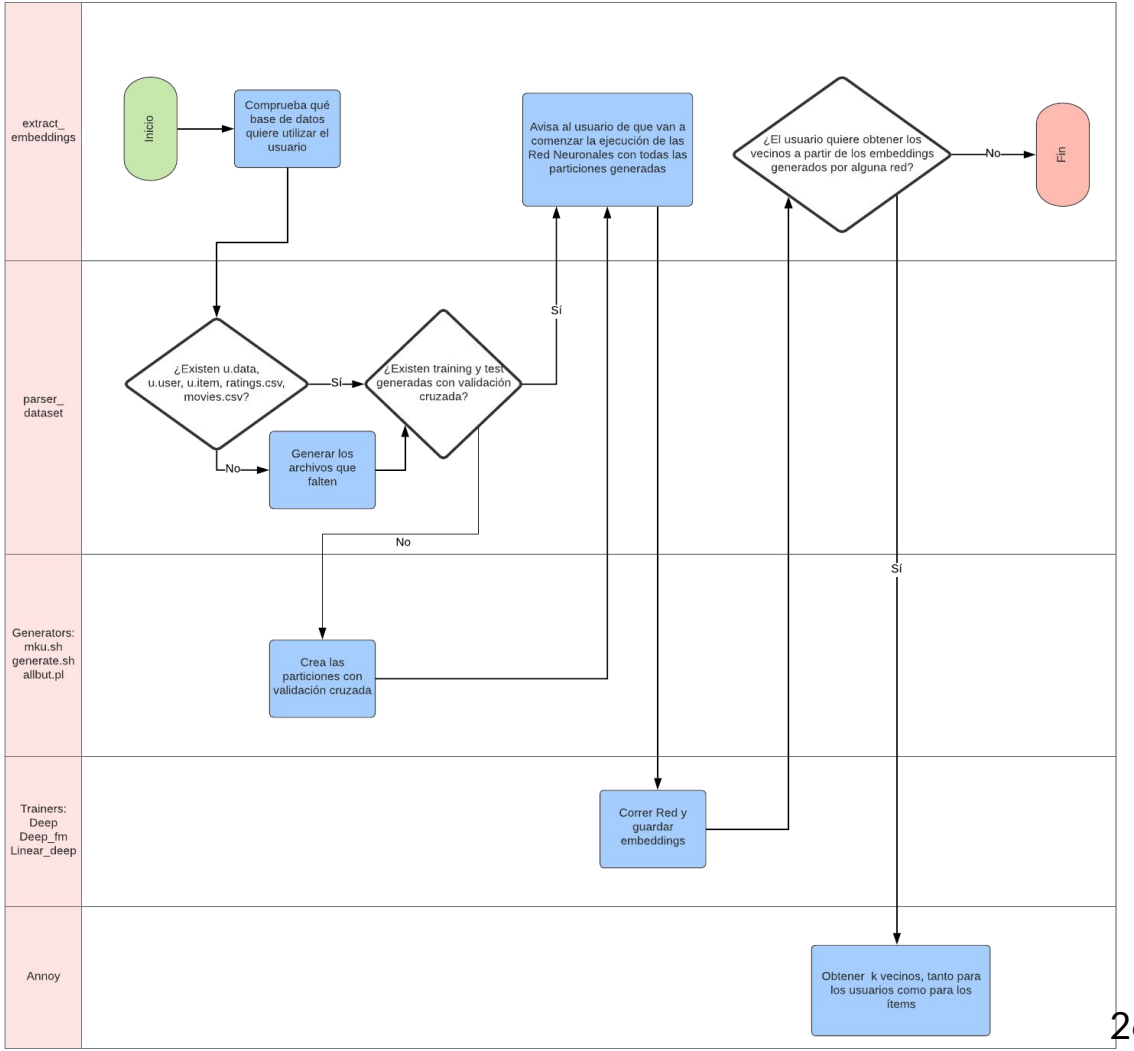
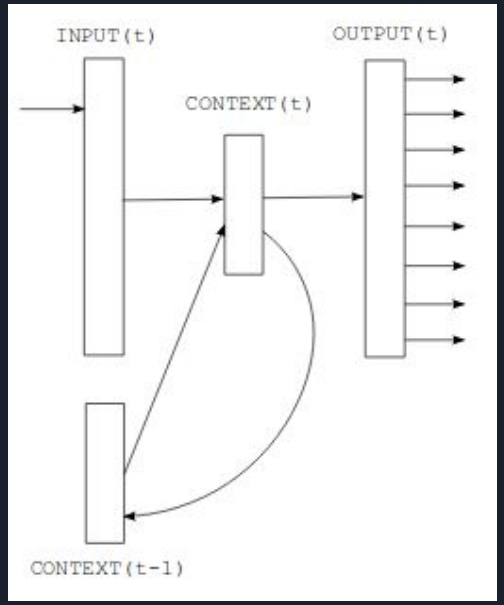


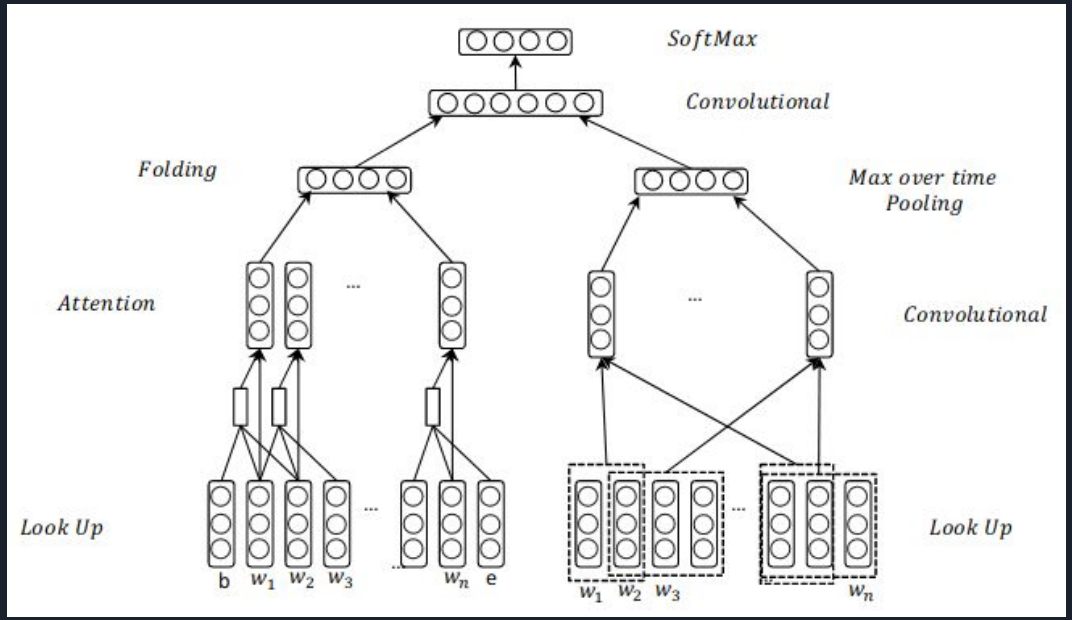
Diagrama Secuencia: Proceso automático



Estado del arte - Redes Neuronales

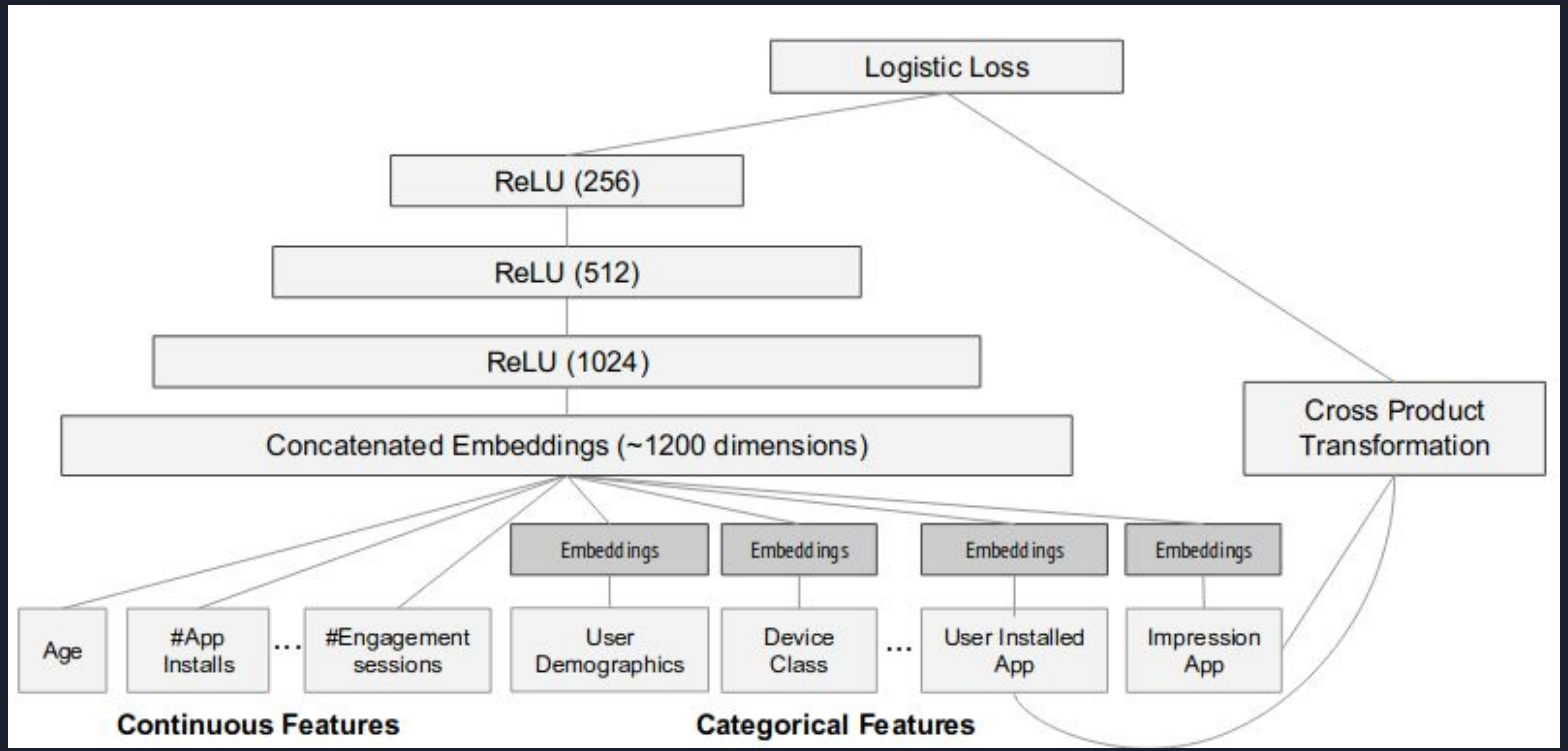


Red Recurrente



Red Convolucional

Estructura de Deep_fm



Estructura de Deep y Linear_deep

Deep

Características:

- Densa, todas las neuronas conectadas a las de la capa siguiente.
- Prealimentada, las conexiones entre neuronas no forman un bucle.

tf.estimator.DNNClassifier

Linear_deep

Características:

- Red Lineal.
- Densa.

tf.estimator.DNNLinearCombinedClassifier

Diferencias datasets Movielens (1)

	Ratings	Usuarios	Ítems
Movielens 100K	100000	943	1682
Movielens 20M	20000263	138493	27278

	Movielens 100K	Movielens 20M	¿Necesario?	Equivalente a
u.data	YES	NO	YES	ratings.csv
u.genre	YES	NO	NO	
u.info	YES	NO	NO	
u.item	YES	NO	YES	movies.csv, pero faltan campos
u.occupation	YES	NO	NO	
u.user	YES	NO	YES	Solo los IDs desde ratings.csv
links.csv	NO	YES	NO	
movies.csv	NO	YES	YES	u.item
ratings.csv	NO	YES	YES	u.data
tags.csv	NO	YES	NO	