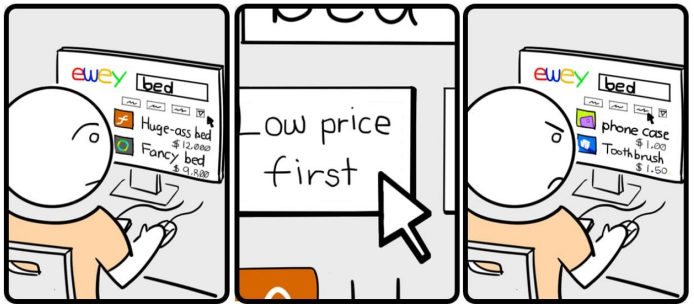
Learning to Rank

Cristian Cardellino - Luis Biedma

Introducción

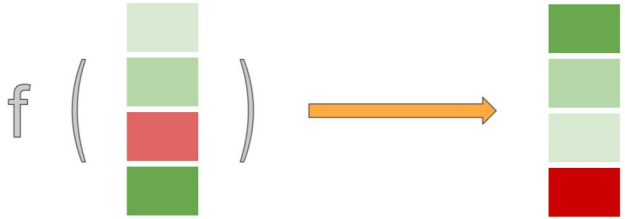


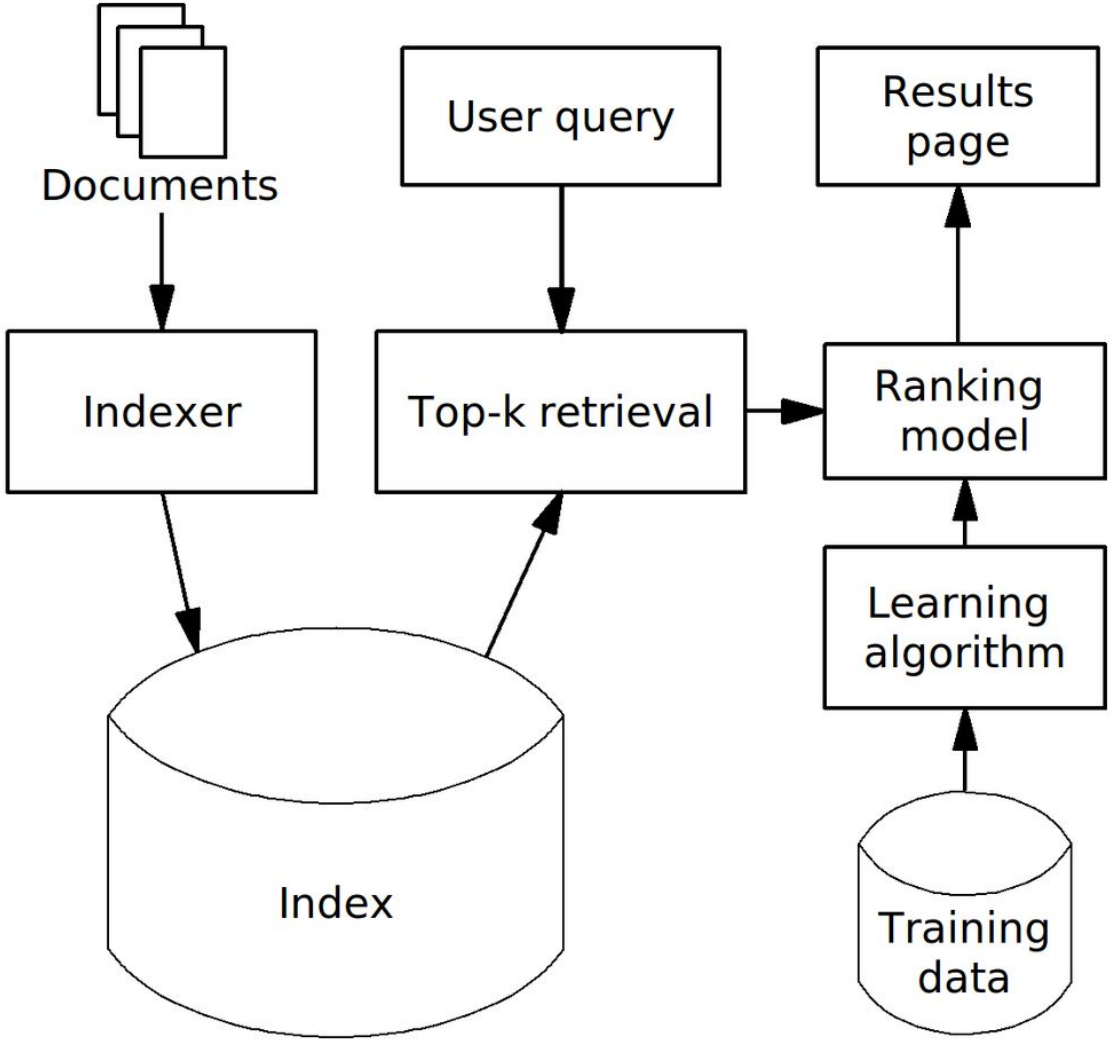
Usos Conocidos

- Nace en el area de Information Retrieval
- Traducción automática: Rankear candidatos a traducciones
- Biología computacional: Rankear estructuras 3D para problemas de predicción de estructuras de proteínas
- Ingeniería del software: Localización de fallas
- **Sistemas de recomendación: Identificar una lista rankeada de recomendaciones de noticias relacionadas a una noticia leída.**

Learning to Rank Model (Ranker)

- Objetivo: Ordenar una colección de **documentos o resultados de búsqueda** de acuerdo a su nivel de **relevancia** asociado a una **query**.
- Ejemplo: Rankear items para el newsfeed de un usuario
  - No nos importa solamente la probabilidad de click
  - Nos conviene darle algo que termine de leer
  - Quizás dependemos de más de una métrica
  - Esto es entrenable!




Documents
User query
Results page
Indexer
Top-k retrieval
Ranking model
Learning algorithm
Index
Training data

Modelo del Problema

Problema: Aprender una función f* para ordenar una lista de objetos.

- **INPUT:** Lista de ejemplos (con contexto)
  - Bag of Features + relevancia (ordenamiento)
- **OUTPUT:** Función que produce el ordenamiento óptimo
  - Parametrizada mediante funciones como regresiones, redes, etc.
  - No necesariamente da un orden completo

$$\psi = (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^n \times \mathbb{R}^n$$

$$\mathcal{L}(f) = \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \Psi} \ell(\boldsymbol{y}, f(\boldsymbol{x}))$$

Algoritmos para LtR

Los modelos y sus algoritmos asociados están categorizados de acuerdo a las funciones de pérdida que se usen:

Enfoque Pointwise

- Pérdida definida en base sólo a cada documento
- Termina siendo un problema de regresión/clasificación
- Ejemplo (delta es la función de pérdida L):

$$\Delta(\pi(f, D_q), y_q) = \frac{1}{n(q)} \sum_{i=1}^{n(q)} (f(d_i^q) - y_i^q)^2$$

- No son lo mejorcito...

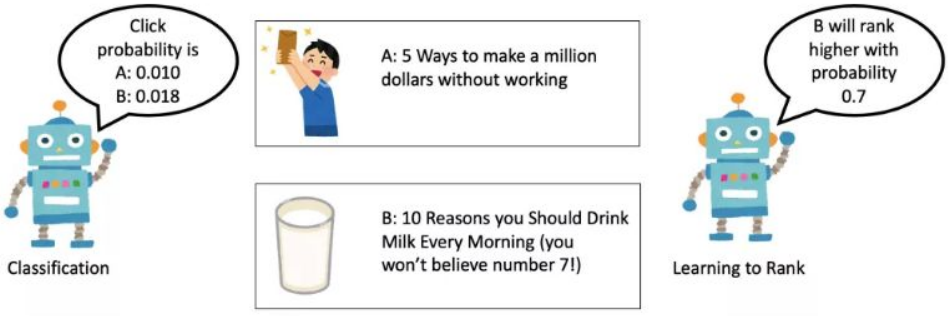
Algoritmos para LtR

Enfoque Pairwise

- Pérdida se define en base a pares de documentos con diferentes juicios de relevancia
- Reduce el problema a clasificación
- Los más famosos salen de aplicar el principio de máxima verosimilitud (como casi cualquier algoritmo conocido de Machine Learning)

Algoritmos para LtR

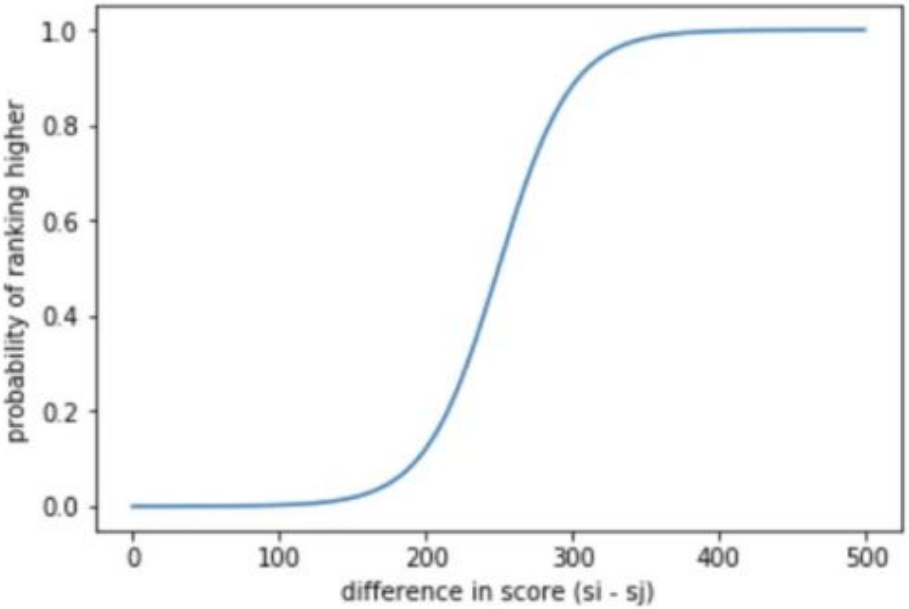
Enfoque Pairwise

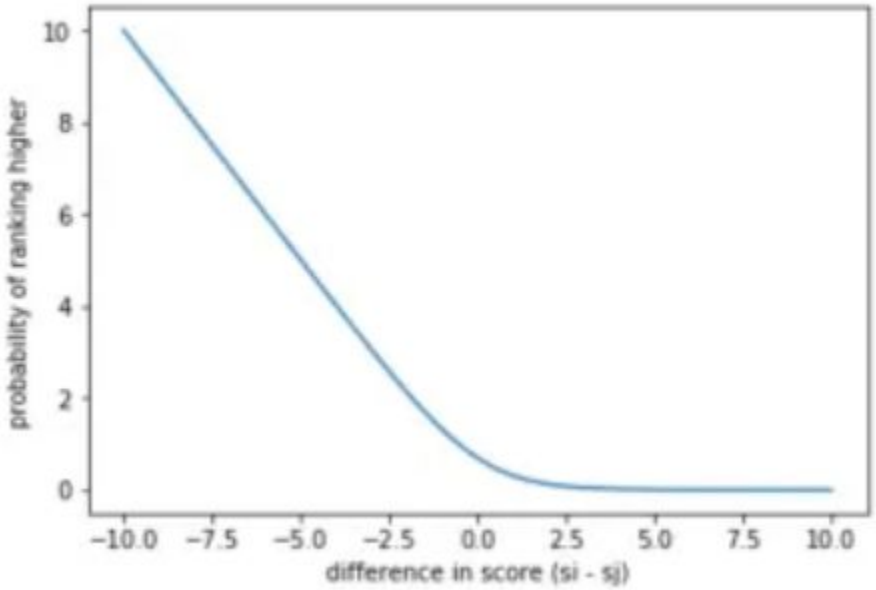


- Probabilidad de que el item *i* rankee mejor que el item *j*.
- Mayor diferencia en scores -> mayor diferencia en probs.
- Scores iguales -> p = 0.5
- Simetría

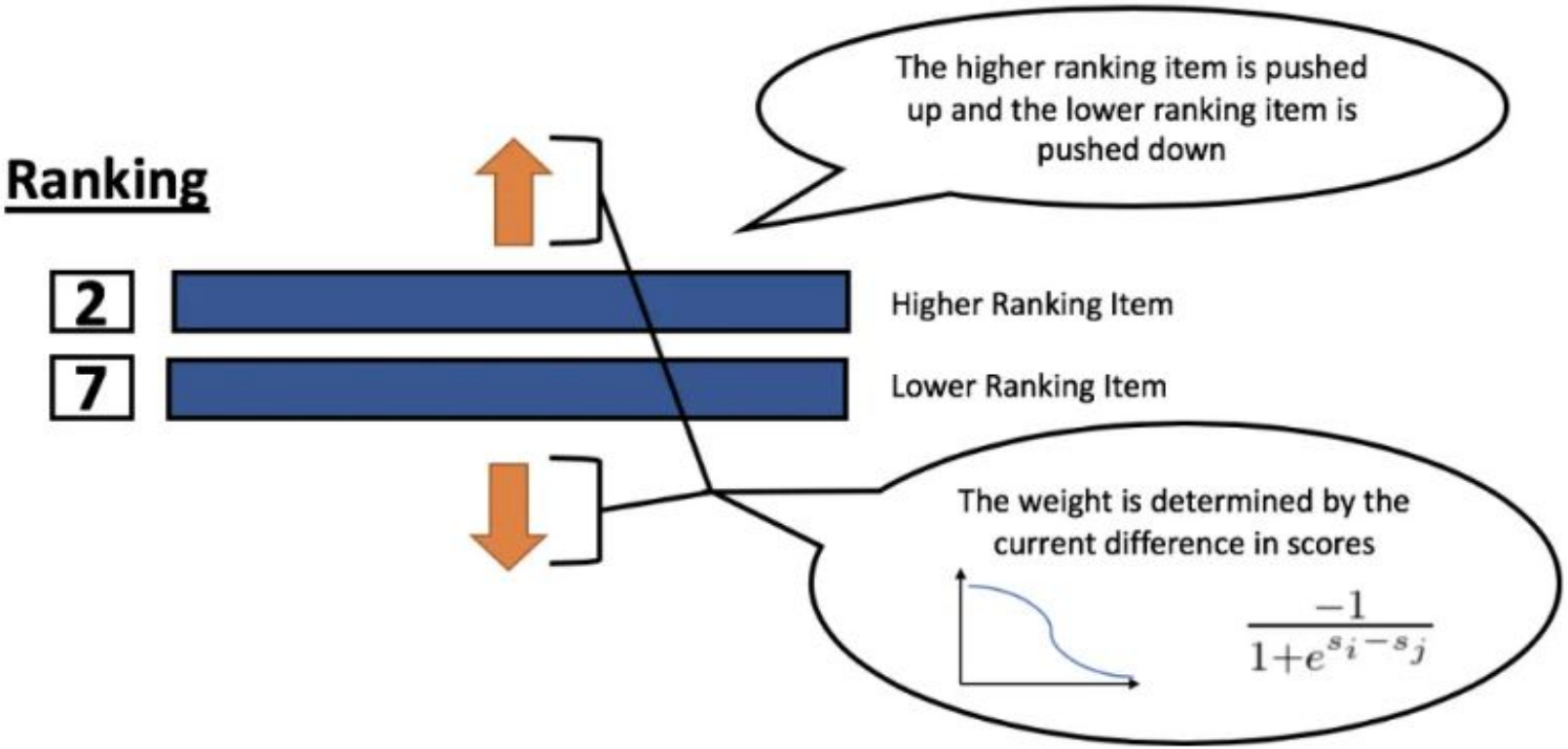
Algoritmos para LtR

$$P(\text{rank}(i) > \text{rank}(j)) = \frac{1}{1+e^{-\alpha(s_i - s_j)}} \longrightarrow J_{ij} = -\log(\frac{1}{1+e^{-(s_i - s_j)}}) = \log(1 + e^{s_j - s_i})$$

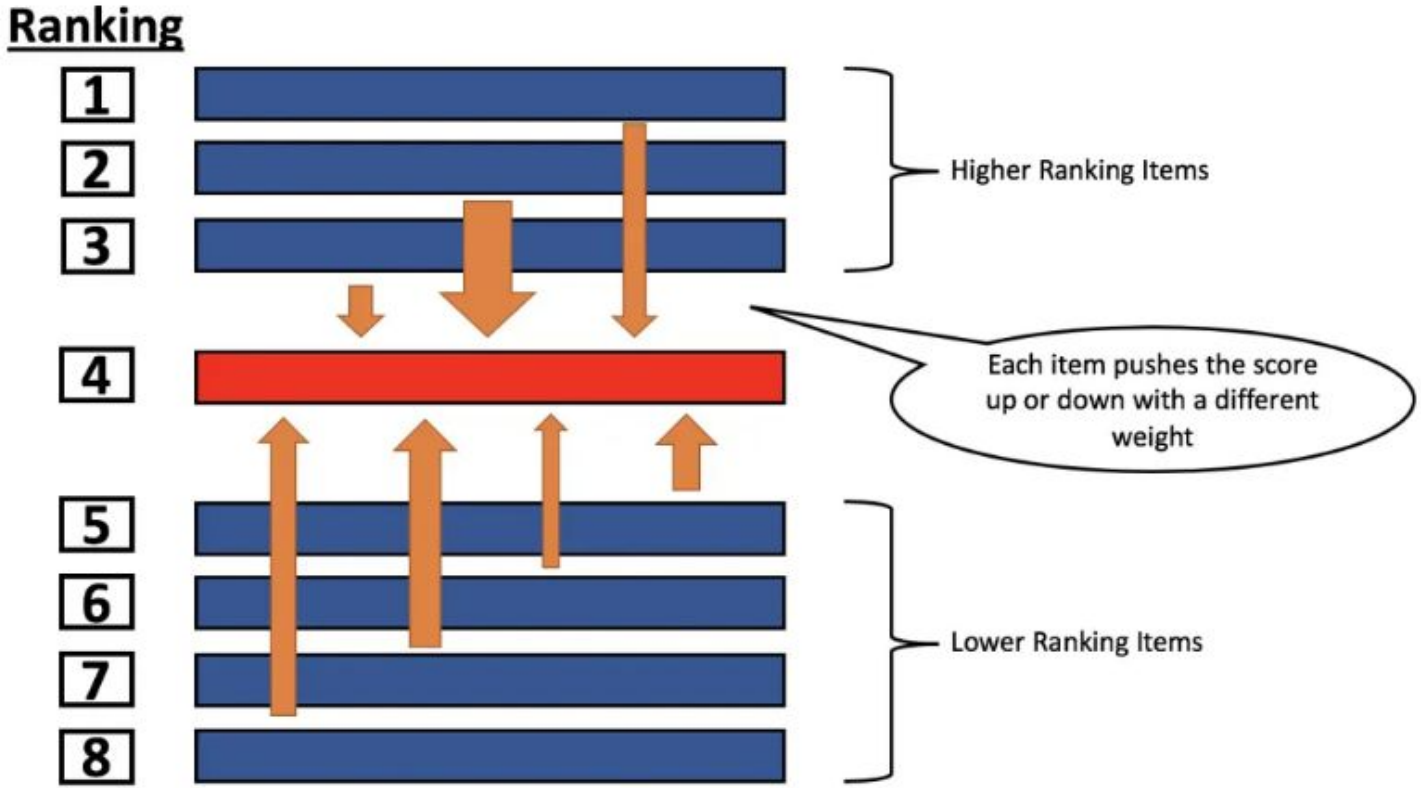




RankNet


The higher ranking item is pushed up and the lower ranking item is pushed down
Ranking
2
7
Higher Ranking Item
Lower Ranking Item
The weight is determined by the current difference in scores
$\frac{-1}{1+e^{s_i - s_j}}$

RankNet, SVM, RankBoost…



Algoritmos para LtR

Enfoque Listwise

- Pérdida definida en base a la lista completa
- Estoy dando una estructura más fuerte
- Primera solución? Agregar una “penalidad”

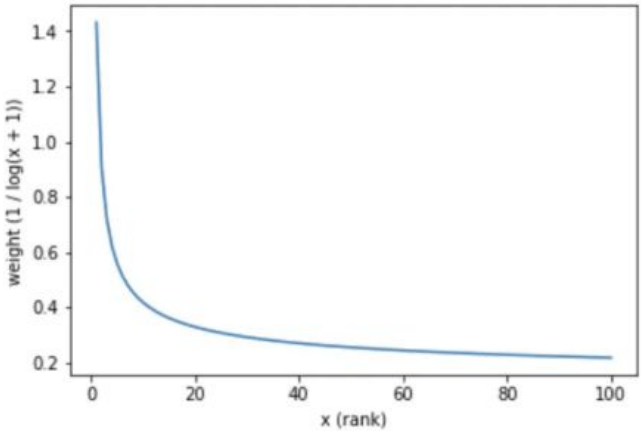
$$\lambda_{ij} = \frac{-1}{1+e^{s_i - s_j}} \quad \Rightarrow \quad \lambda_{ij} = \frac{-|\Delta(i,j)|}{1+e^{s_i - s_j}}$$

- Penalidad? (Normalized) Discounted Cumulative Gain

Discounted Cumulative Gain

- Medida de calidad de rankeo más usada
- Si rel_i es la relevancia del item en el lugar i:

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$$



Normalized Discounted Cumulative Gain

- IDCG (o *GDCG*): El máximo de todos los DCG (existe porque el espacio siempre sería finito)
- NDCG = DCG / IDCG
- Se entrena igual que antes, usando SGD
- Este enfoque se llama **LambdaRANK**

En general,

pointwise < pairwise < listwise

RankMART y LambdaMART son iguales a sus compatriotas, excepto por el entrenamiento: **Boosting.**

Otras Métricas (también para Evaluación!)

- Precision at K (P@K):

  P@K(u) = #contenido relevante en top K / K

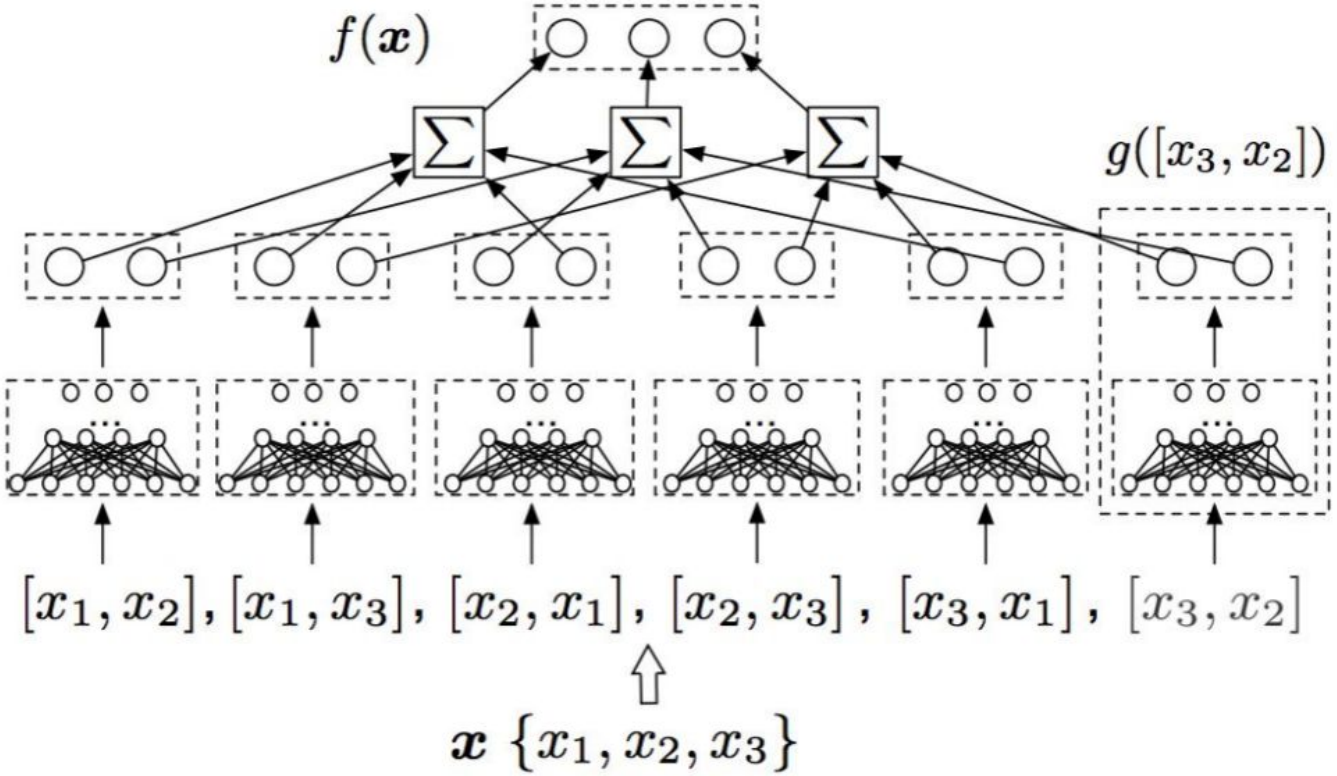
- Average Precision (AP):

$$AP(u) = \frac{\sum_{k=1}^{m} P@k(u)}{m}$$

- Mean Average Precision (MAP):

$$MAP = \frac{\sum_{u \in U} AP(u)}{|U|}$$

Más Modelos? Bueno, dale...



Un poco de Código

https://github.com/Microsoft/LightGBM