Unidad 4: Sistemas de recomendación

Learning to rank

Blanca Vázquez y Gibran Fuentes-Pineda

23 de octubre de 2021

Motivación

- En la práctica, los sistemas de recomendación solo muestran al usuario los elementos como una lista ordenada
- Los usuarios usualmente le ponen atención a los primeros elementos de la lista
- Los modelos basados en predecir la calificación no consideran explícitamente el ordenamiento
  - Usualmente primero se realiza la predicción de la calificación de cada elemento y posteriormente se ordena a partir de esta calificación

Definición

- En *learning to rank*, se aplica aprendizaje supervisado para realizar el ordenamiento de un conjunto de elementos

- Tipos
  - A nivel elemento: cada elemento se clasifica como relevante o no relevante
  - A nivel par: predice el orden relativo de cada par de elementos
  - A nivel lista: general el orden de una lista completa

Nivel instancia

- Se basa en los atributos de cada elemento de forma individual
- Datos de entrenamiento: elementos y su calificación (e.g. relevante/no relevante o valor númerico)
- Ejemplos: Pranking o McRank

Nivel par

- Aprende una función que determina el orden relativo de dos elementos $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$, es decir, si $\mathbf{x}^{(i)}$ va antes o después que $\mathbf{x}^{(j)}$
- Datos de entrenamiento: pares de elementos y cuál es más relevante
- Ejemplo: RankNet, LambdaRank o RankingSVM

Nivel lista

- Optimiza una función de pérdida basada en el ordenamiento completo de una lista
  - Ejemplos de funciones de pérdida: *normalized cumulative discounted gain (NDCG)* y mean reciprocal rank (MRR)
- Problema: hay muchos posibles ordenamientos de una misma lista
- Ejemplos: ListNet, ListMLE o PermuRank

Learning to rank y recuperación de información

- Aplicado a motores de búsqueda donde hay un conjunto de consultas $\mathbf{Q}$, cada uno con su correspondiente conjunto de respuesta $\mathcal{S}$ y el problema es ordenar $\mathcal{S}$

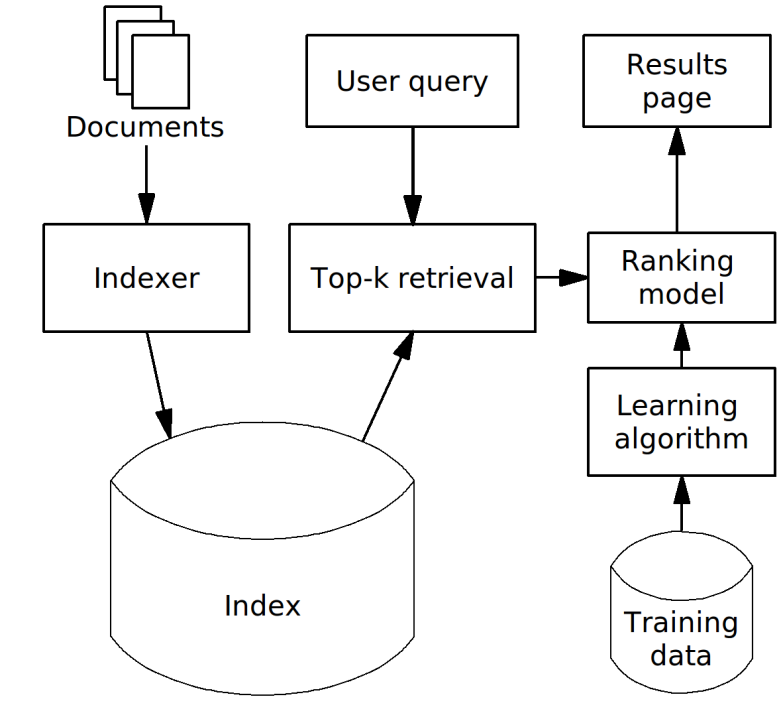

Imagen del dominio público, creada por el usuario X7q de Wikipedia.

RankNet

- Método de *learning to rank* que utiliza un modelo cuya función a optimizar es diferenciable respecto a sus parámetros
  - Usualmente una red neurona, aunque también se han usado árboles de potenciación del gradiente
  - Cada ejemplo $\mathbf{x}^{(i)}$ se mapea a un número $f(\mathbf{x}^{(i)})$
- Por ej. red neuronal tipo siamesa entrenada para aprender el orden relativo de dos elementos

Entrenamiento (1)

- Busca aprender el orden relativo de dos elementos
- Para una consulta, se elige un par de ejemplos $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ con etiquetas diferentes y se calcula

$$s^{(i)} = f(\mathbf{x}^{(i)})$$
$$s^{(j)} = f(\mathbf{x}^{(j)})$$

Entrenamiento (1)

- Estas salidas se mapean a una probabilidad de que $\mathbf{x}^{(i)}$ es más relevante que $\mathbf{x}^{(j)}$

$$P_{i,j} \equiv \frac{1}{1 + e^{-\sigma(s^{(i)} - s^{(j)})}}$$

  donde σ es un parámetro que determina la forma de la función sigmoide

- Se busca minimizar la entropía cruzada binaria

LamdaRank

- Es similar a RankNet pero en el entrenamiento se busca maximizar el *Normalized Discounted Cumulative Gain* (NDCG)

$$NDCG@T = \frac{DCG@T}{\text{máx}(DCG@T)} \in [0, 1]$$

$$DCG@T = \sum_{i=1}^{T} \frac{2^{y^{(i)}} - 1}{\log(1 + i)}$$

  donde T es el nivel de truncamiento, $y^{(i)}$ es la etiqueta asociada al i-ésimo elemento de la lista

- Se especifican los gradientes directamente, en lugar de derivarlos