



Apprendre à Ordonner 'Learn to Rank'

par

Dr. Samira LAGRINI

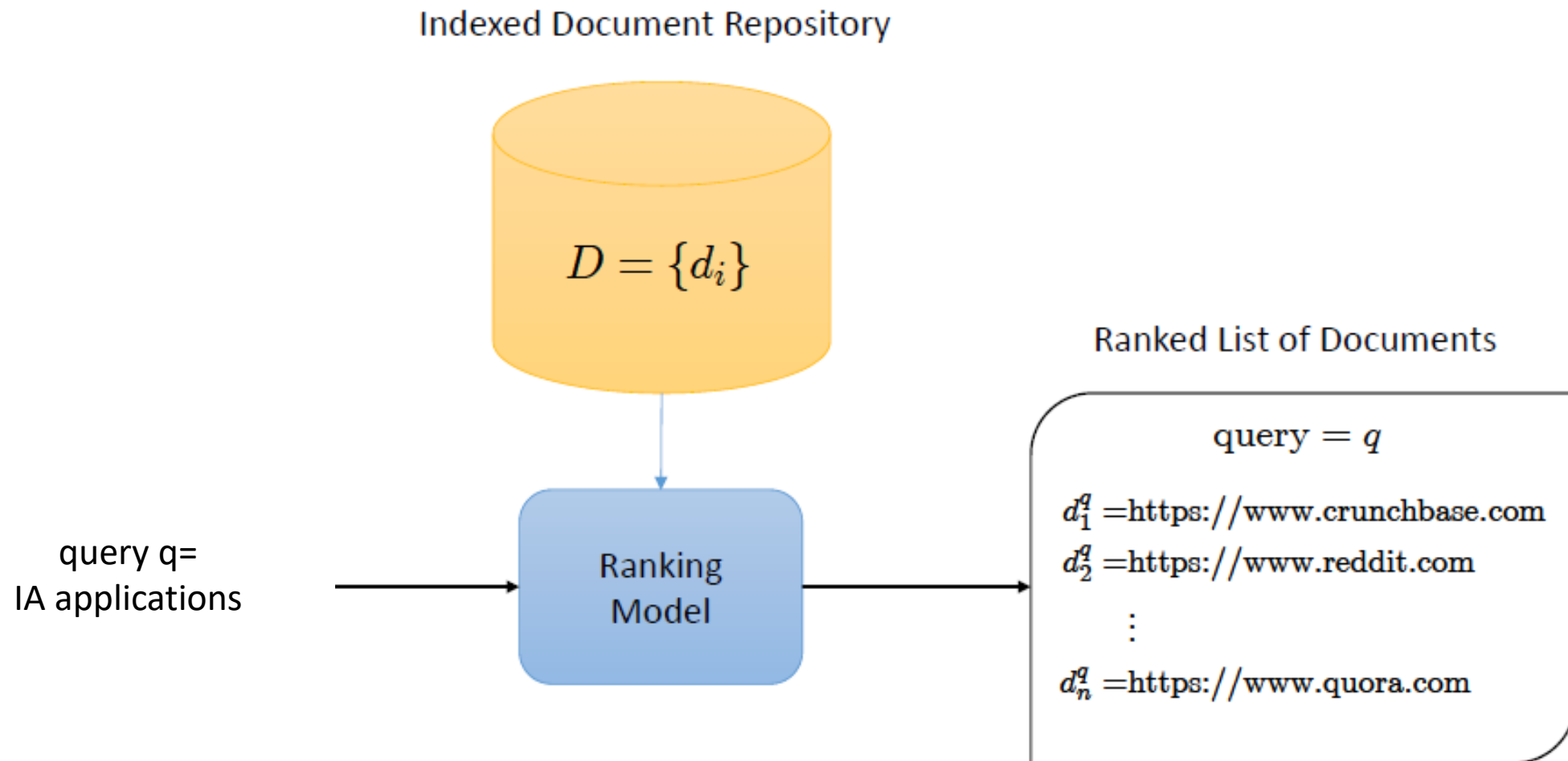


Année universitaire:2024/2025

Introduction

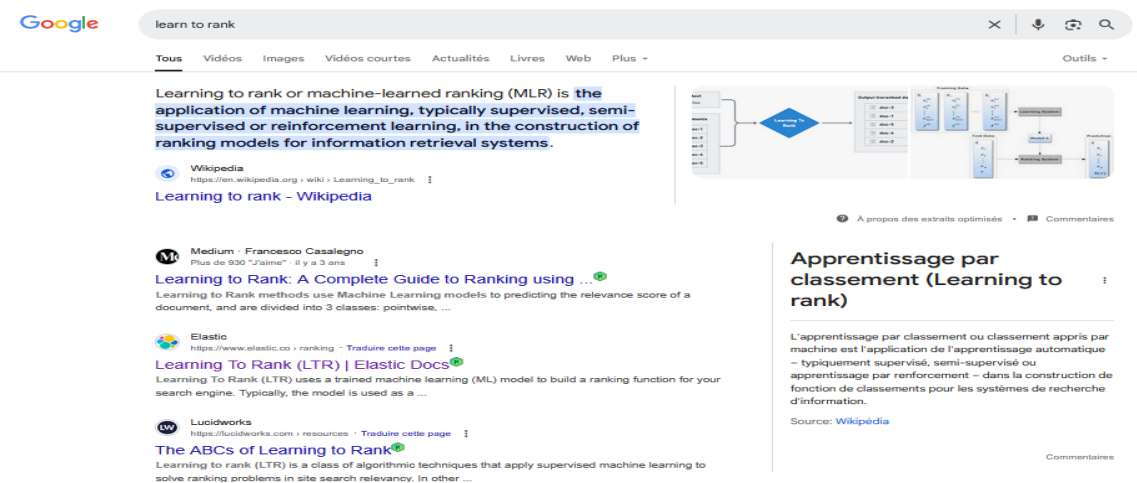
- ❑ Apprendre à ordonner (Learning to Rank, LTR) est un problème de l'apprentissage supervisé qui a pour objectif d'apprendre à un modèle à attribuer un classement (ou un score de pertinence) à des éléments (documents, produits..) dans une liste.
- ❑ Le but est de trier des éléments en fonction de leur pertinence pour une requête donnée.
- ❑ Plutôt que de prédire une simple étiquette binaire (pertinent ou non pertinent), le modèle LTR doit apprendre à ordonner les éléments selon un ordre de pertinence.

Introduction



Applications Pratiques de LTR

Optimisation des résultats des moteurs de recherche

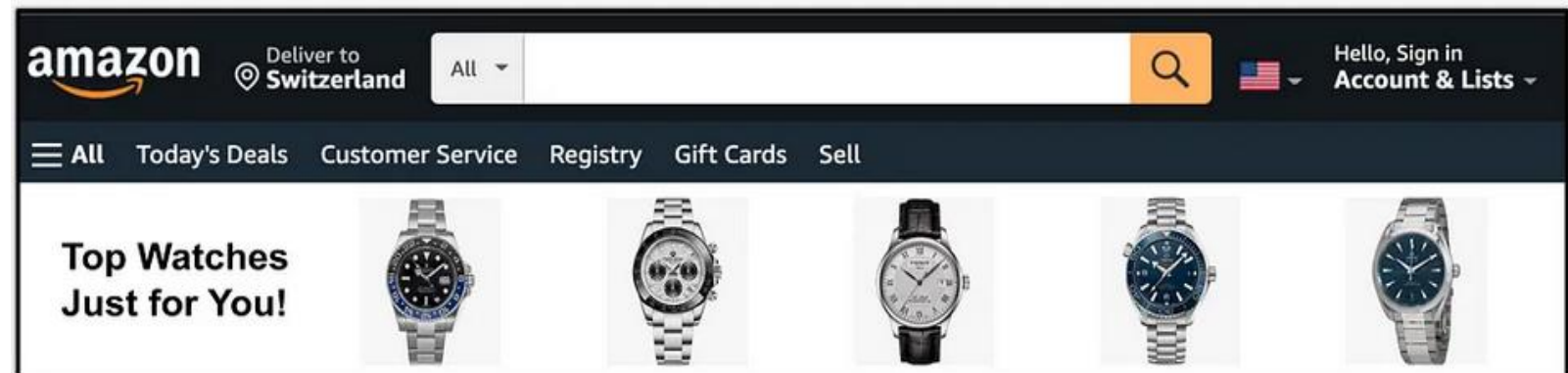


The screenshot shows a Google search for "learn to rank". The search bar at the top contains the text "learn to rank". Below the search bar, there are tabs for "Tous", "Vidéos", "Images", "Vidéos courtes", "Actualités", "Livres", "Web", and "Plus". The search results are displayed below the tabs. The first result is from Wikipedia, titled "Learning to rank or machine-learned ranking (MLR) is the application of machine learning, typically supervised, semi-supervised or reinforcement learning, in the construction of ranking models for information retrieval systems." The second result is from Medium, titled "Learning to Rank: A Complete Guide to Ranking using ...". The third result is from Elastic, titled "Learning To Rank (LTR) | Elastic Docs". The fourth result is from Lucidworks, titled "The ABCs of Learning to Rank". To the right of the search results, there is a diagram illustrating the Learning to Rank process. The diagram shows a flow from "Input" to "Learning to Rank" (represented by a blue diamond) to "Output". The "Output" is a list of items, each with a score. The diagram also shows a "Feedback" loop from the "Output" back to the "Learning to Rank" process.

E-commerce et Publicité en ligne

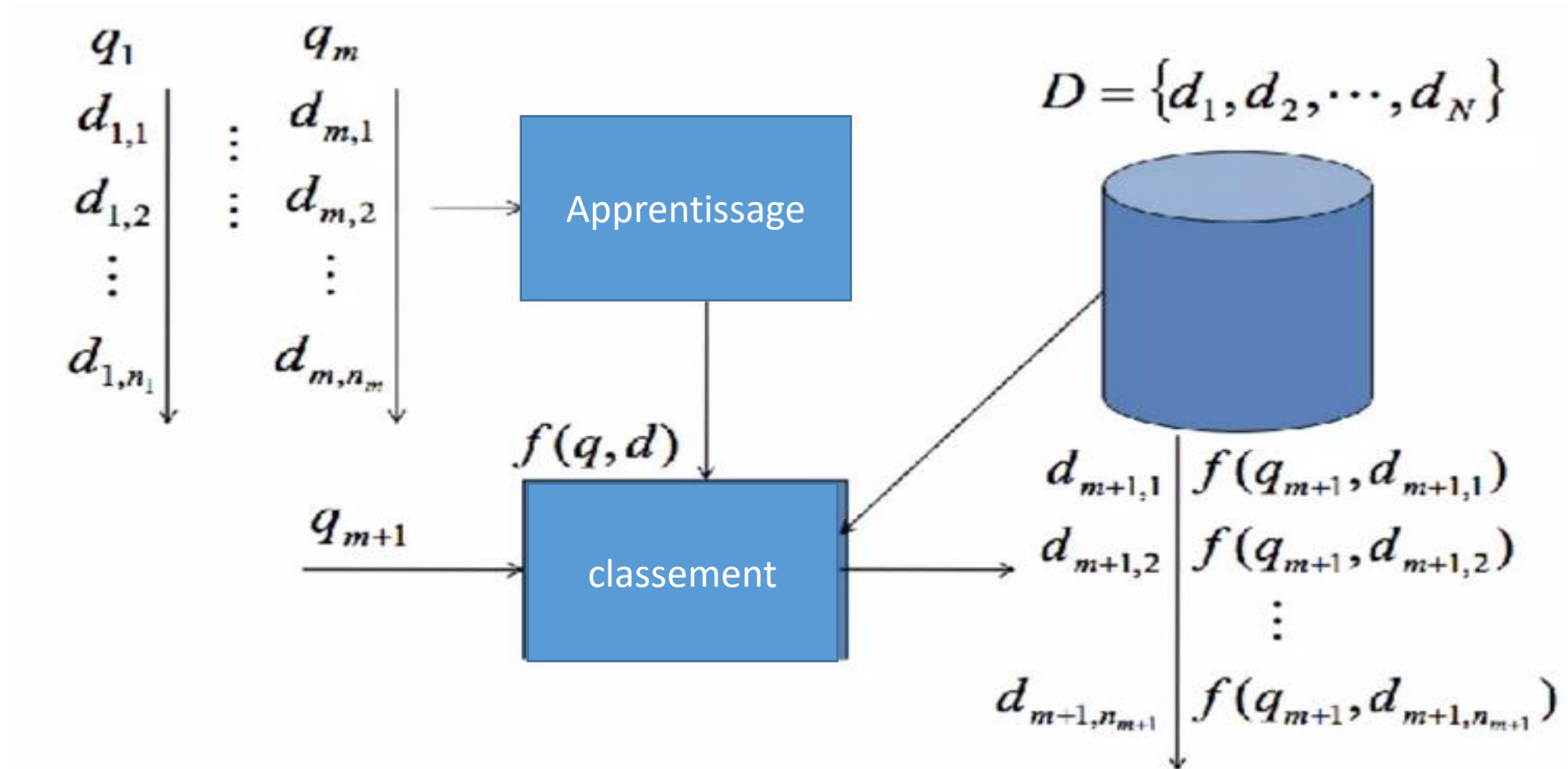


Systèmes de
Recommandation



The screenshot shows the Amazon website. The top navigation bar includes the Amazon logo, "Deliver to Switzerland", a search bar, and a "Hello, Sign in Account & Lists" link. Below the navigation bar, there is a horizontal menu with links for "All", "Today's Deals", "Customer Service", "Registry", "Gift Cards", and "Sell". The main content area features a recommendation for "Top Watches Just for You!". The recommendation includes five watch images: a blue and silver watch, a silver chronograph watch, a black leather watch, a blue and silver watch, and a silver watch.

Construire un Modèle LTR



Collecte des données d'entraînement

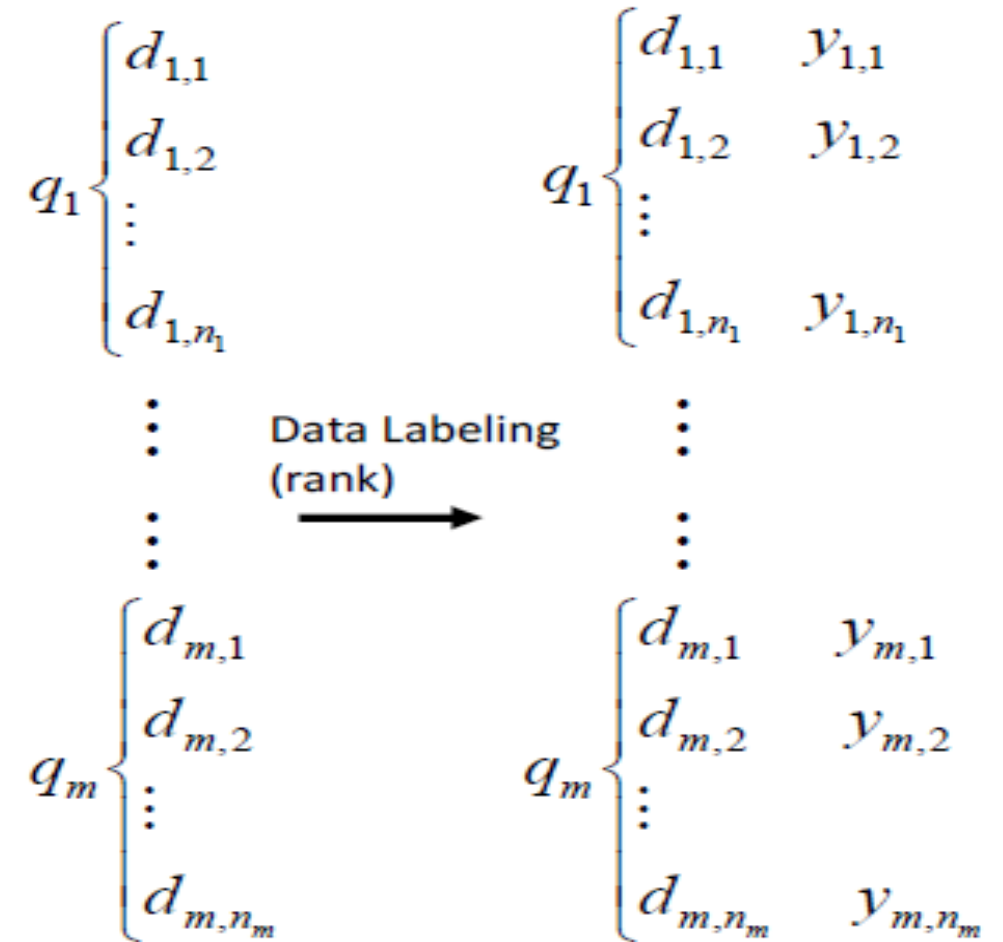
- Les données d'entraînement sont des exemples d'éléments classés par pertinence :

➤ Requêtes et résultats :

Liste de requêtes et de documents associés.

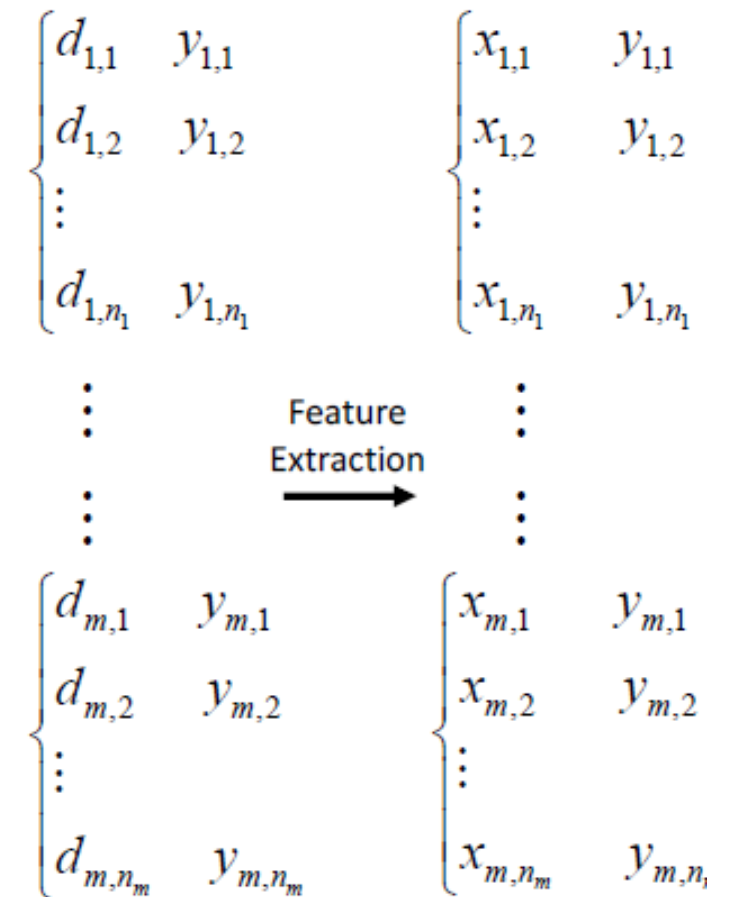
➤ Pertinence

Scores ou labels de pertinence



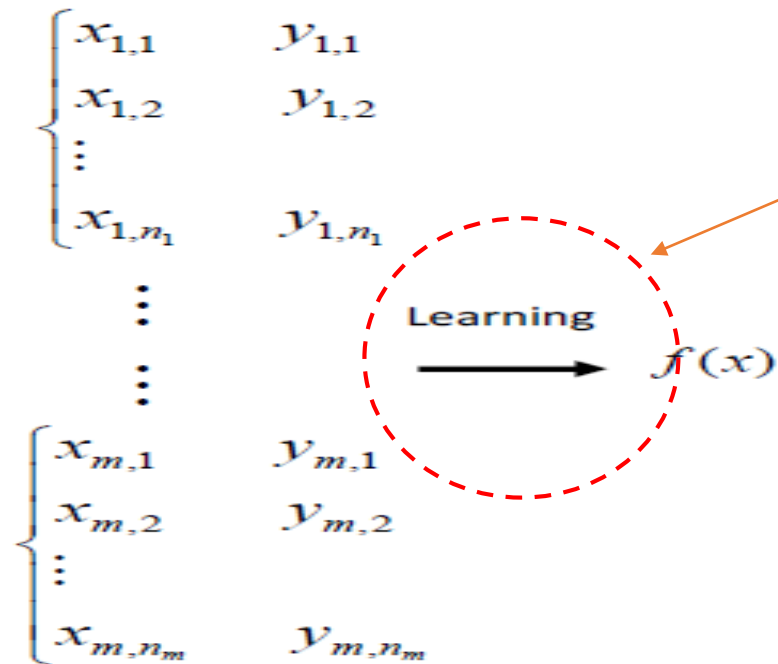
Extraction des caractéristiques

- Pour chaque élément à classer, extraire des caractéristiques (features) qui peuvent influencer son classement:
 - ✓ Fréquences de mots de requete, longueur du texte, etc.
 - ✓ Date de publication, auteur, etc.
 - ✓ **Interactions utilisateur** : Clics,.. etc.
 - ✓
- Chaque élément est transformé en un vecteur de caractéristiques qui sera utilisé pour l'entraînement du modèle.



Entraîner le modèle

- Sélectionner un algorithme d'apprentissage supervisé pour entraîner le modèle → Cela dépend de l'approche LTR



Quelle approche ????



Approches d'apprentissage LTR

Given a query q and a set of documents $D = (d_1, \dots, d_n)$:

Pointwise

Input: Single candidate

$$x = (q, d_i)$$



Compute **score** between candidate and query



Solution: Transform task into Regression.

Pairwise

Input: Pair of candidates

$$x_i = (q, d_i) \quad \text{and} \quad x_j = (q, d_j)$$



Given a **pair of candidates** decide which one **rank higher**



Solution: Transform task into Binary Classification.

Listwise

Input: Whole list of candidates

$$x_1 = (q, d_1) \quad \dots \quad x_n = (q, d_n)$$



optimise its **order**



Solution: Incorporate evaluation metrics (e.g. DCG) into loss.

Approche point par point (Pointwise)

- Apprendre à classer chaque élément de la liste indépendamment des autres. → input: $x = (q, d_i)$
 - On attribue un score de pertinence entre chaque élément et la requête.
- Modéliser la tâche comme un problème de régression

Algorithms courants

OC-SVM (One-Class Support Vector Machine), Régression linéaire, régression logistique.

Avantages : Simple à implémenter.

Limites : Ignore les relations entre les documents.

Approche Pointwise

Exemple

Supposons que vous avez les résultats suivants pour une requête "chatons" :

- Page A : Note 5
- Page B : Note 3
- Page C : Note 1
- L'objectif de l'approche pointwise est de prédire cette note pour chaque page en fonction de ses caractéristiques (par exemple, la présence du mot "chaton", le nombre de vues, etc.).

Approche par paire (Pairwise)

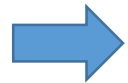
- Au lieu de classer directement chaque élément, on compare des paires d'éléments pour déterminer lequel des deux est plus pertinent. → input : $x_i=(q,d_i)$ $x_k=(q,d_j)$
- Le modèle apprend à prédire quel élément parmi une paire est le plus pertinent et ensuite à en déduire un classement global. → Modéliser la tâche comme un problème de classification binaire.

Modèles courants

RankNet (réseaux de neurones), Ranking SVM, RankBoost, LambdaRank, and LambdaMART

Approche Listwise

- L'approche Listwise optimise directement l'ordre complet de tous les éléments dans la liste



Prendre en compte la relation entre les éléments d'une liste plutôt que de les traiter individuellement.

Modèles courants

ListNet, ListMLE, AdaRank, SVM MAP, Soft Rank, and AppRank.

Évaluation des performance

Évaluer la qualité du modèle à l'aide de métriques tel que:

Métrique	Description
NDCG (Normalized Discounted Cumulative Gain)	Mesure l'efficacité du classement en tenant compte de la pertinence et de la position des éléments dans la liste.
MAP (Mean Average Precision)	Moyenne de la précision pour plusieurs requêtes, prenant en compte la position des éléments pertinents.
P@k (Precision at k)	Mesure la proportion d'éléments pertinents parmi les k premiers résultats du classement.
R@k (Recall at k)	Mesure la proportion des éléments pertinents trouvés dans les k premiers résultats par rapport au total des éléments pertinents.

