**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP.HCM**



MÔN HỌC

**KHAI THÁC THÔNG TIN - GCOS107**

TIỂU LUẬN

**NGHIÊN CỨU VỀ CÁC THUẬT TOÁN HỌC ĐỂ XẾP HẠNG (LEARN TO RANK – LTR) TRONG NGỮ CẢNH ÁP DỤNG CÁC MÔ HÌNH HỌC MÁY CHO BÀI TOÁN XẾP HẠNG CÁC TÀI LIỆU TRẢ VỀ CỦA MỘT HỆ THỐNG TRUY HỒI THÔNG TIN**

|  |  |  |
| --- | --- | --- |
| **Giảng viên hướng dẫn** | : | Phạm Thế Anh Phú |
| **Học viên thực hiện** | : | Lương Xuân Nhất – 2541861021  Lục Gia Yến – 2541861041  Đặng Thanh Hoà– 2541861008  Lê Quý Thiên – 2541861032  Hồ Nhân Phước - 2541861023 |
| **Mã lớp** | : | 25SCT11 |

Thành phố Hồ Chí Minh, tháng 08 năm 2025

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP.HCM**



MÔN HỌC

**KHAI THÁC THÔNG TIN**

TIỂU LUẬN

**NGHIÊN CỨU VỀ CÁC THUẬT TOÁN HỌC ĐỂ XẾP HẠNG (LEARN TO RANK – LTR) TRONG NGỮ CẢNH ÁP DỤNG CÁC MÔ HÌNH HỌC MÁY CHO BÀI TOÁN XẾP HẠNG CÁC TÀI LIỆU TRẢ VỀ CỦA MỘT HỆ THỐNG TRUY HỒI THÔNG TIN**

|  |  |  |
| --- | --- | --- |
| **Giảng viên hướng dẫn** | : | Phạm Thế Anh Phú |
| **Học viên thực hiện** | : | Lương Xuân Nhất – 2541861021  Lục Gia Yến – 2541861041  Đặng Thanh Hoà– 2541861008  Lê Quý Thiên – 2541861032  Hồ Nhân Phước - 2541861023 |
| **Mã lớp** | : | 25SCT11 |

Thành phố Hồ Chí Minh, tháng 08 năm 2025

**NHẬN XÉT CỦA GIẢNG VIÊN**

**Điểm:**

**CHỮ KÝ GIẢNG VIÊN**

# DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ viết tắt** | **Giải thích** |
| AI | Artificial Intelligence |
| BCE | Binary Cross-Entropy |
| BM25 | Best Matching 25 |
| IR | Information Retrieval |
| LTR | Learn to Rank |
| NDCG | Normalized Discounted Cumulative Gain |
| TF-IDF | Term Frequency-Inverse Document Frequency |

# DANH MỤC HÌNH ẢNH

[Hình 1: Hình ảnh huấn luyện model RankNet 15](#_Toc205702912)

# DANH MỤC BẢNG

[Bảng 3.1. So sánh các thuật toán RankNet, LambdaRank và LambdaMART 13](#_Toc205626981)

**MỤC LỤC**

[DANH MỤC TỪ VIẾT TẮT ii](#_Toc205704028)

[DANH MỤC HÌNH ẢNH iii](#_Toc205704029)

[DANH MỤC BẢNG iv](#_Toc205704030)

[CHƯƠNG 1. TỔNG QUAN VÀ GIỚI THIỆU 1](#_Toc205704031)

[1.1. Lý do chọn đề tài 1](#_Toc205704032)

[1.2. Mục tiêu nghiên cứu 1](#_Toc205704033)

[1.2.1. Tìm hiểu và nắm vững kiến thức về Learn to rank 1](#_Toc205704034)

[1.2.2. Triển khai mô hình thực nghiệm và đánh giá, cải thiện 1](#_Toc205704035)

[1.3. Phương pháp nghiên cứu 1](#_Toc205704036)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 3](#_Toc205704037)

[2.1. Tổng quan về Learn to rank 3](#_Toc205704038)

[2.1.1. Khái niệm cơ bản 3](#_Toc205704039)

[2.1.2. Ứng dụng 3](#_Toc205704040)

[2.2. Các hướng tiếp cận của Learn to rank 3](#_Toc205704041)

[2.2.1. Phương pháp Pointwise 3](#_Toc205704042)

[2.2.2. Phương pháp Pairwise 4](#_Toc205704043)

[2.2.3. Phương pháp Listwise 4](#_Toc205704044)

[CHƯƠNG 3. CÁC THUẬT TOÁN TRONG LEARN TO RANK 5](#_Toc205704045)

[3.1. RankNet: Nền tảng của Học để Xếp hạng theo cặp 5](#_Toc205704046)

[3.1.1. Lý thuyết và cơ chế hoạt động 5](#_Toc205704047)

[3.1.2. Các bước ứng dụng RankNet vào bài toán Learn to rank thực tế 5](#_Toc205704048)

[3.1.3. Ưu và nhược điểm 6](#_Toc205704049)

[*3.1.3.1. Ưu điểm* 6](#_Toc205704050)

[*3.1.3.2. Nhược điểm* 6](#_Toc205704051)

[3.2. LamdaRank: Tối ưu hóa trực tiếp các chỉ số xếp hạng 7](#_Toc205704052)

[3.2.1. Lý thuyết và cơ chế hoạt động 7](#_Toc205704053)

[3.2.2. Các bước ứng dụng LambdaRank vào bài toán LTR thực tế: 8](#_Toc205704054)

[3.2.3. Ưu và nhược điểm 8](#_Toc205704055)

[3.3. LamdaMART: Sức mạnh của Cây Quyết định Tăng cường 9](#_Toc205704056)

[3.3.1. Lý thuyết và Cơ chế hoạt động 9](#_Toc205704057)

[3.3.2. Các bước ứng dụng LambdaMART vào bài toán LTR thực tế: 10](#_Toc205704058)

[3.3.3. Ưu và nhược điểm 10](#_Toc205704059)

[*3.3.3.1. Ưu điểm* 10](#_Toc205704060)

[*3.3.3.2. Nhược điểm* 11](#_Toc205704061)

[3.4. So sánh tổng quan các thuật toán 11](#_Toc205704062)

[CHƯƠNG 4. CÀI ĐẶT & THỰC NGHIỆM 14](#_Toc205704063)

[4.1. Dữ liệu đầu vào 14](#_Toc205704067)

[4.1.1. Thu thập dữ liệu 14](#_Toc205704068)

[4.1.2. Tiền xử lý dữ liệu 14](#_Toc205704069)

[4.2. Huấn luyện 14](#_Toc205704070)

[4.2.1. Chuẩn bị 14](#_Toc205704071)

[4.2.2. Huấn luyện 15](#_Toc205704072)

[CHƯƠNG 5. KẾT LUẬN & HƯỚNG PHÁT TRIỂN 16](#_Toc205704073)

[5.1. Kết luận 16](#_Toc205704074)

[5.2. Hướng phát triển 16](#_Toc205704075)

[TÀI LIỆU THAM KHẢO 17](#_Toc205704076)

# TỔNG QUAN VÀ GIỚI THIỆU

## Lý do chọn đề tài

Trong môn học Khai thác thông tin (Information Retrieval), có 8 đề tài trong danh sách. Thì đề tài 05: “Nghiên cứu về các thuật toán học để xếp hạng (Learn to rank – LTR) trong ngữ cảnh áp dụng các mô hình học máy cho bài toán xếp hạng các tài liệu trả về của một hệ thống truy hồi thông tin” là một đề tài hay và có tính thách thức cao đối với chúng em. Là một trong 2 đề tài “nghiên cứu” trong 8 đề tài. Nhóm chúng em cũng có sự hứng thú đặc biệt tới các bài toán Learn to rank. Các ứng dụng của Learn to rank trong thực tế cũng thật ấn tượng, ví dụ như: Công cụ tìm kiếm (Google, Bing,...), Gợi ý sản phẩm (Recommnedation) của các sàn thương mại điện tử... Vì thế sự lựa chọn đề tài này là một quyết định có chút mạo hiểm nhưng chúng em sẵn sàng đón nhận thử thách này.

## Mục tiêu nghiên cứu

Trong môn học Khai thác thông tin, khi nghiên cứu đề tài “Về các thuật toán học để xếp hạng (LTR)” thì nhóm em sẽ tập trung vào việc tìm hiểu, nắm bắt cốt lõi, triển khai mô hình, đánh giá và cải thiện nếu có thể.

### Tìm hiểu và nắm vững kiến thức về Learn to rank

* Nắm vững lý thuyết về khái niệm learn to rank, lịch sử ra đời của mô hình học máy này. Cũng như các nội dung về phương pháp học, các ví dụ về thuật toán điển hình, tìm hiểu về nhu cầu và các ứng dụng nổi tiếng sử dụng LTR.
* Đi sâu vào so sánh, phân tích đặc điểm nối bật và sự khác biệt của từng phương pháp. Hiểu về cách thức vận hành cốt lõi để cho ra kết quả chính xác cao.

### Triển khai mô hình thực nghiệm và đánh giá, cải thiện

* Từ những kinh nghiệm đã có từ việc tìm hiểu về learn to rank, nhóm sẽ thực hiện chọn một trong những mô hình thuật toán nổi bật, đó là: “ RankNet” để thực hiện triển khai demo, RankNet được coi là một trong những anh cả đời đầu của LTR. Tạo tiền đề cho nhiều thuật toán tốt hơn về sau này.

## Phương pháp nghiên cứu

Với việc xác định mục tiêu nghiên cứu rõ ràng và cụ thể thì nhóm đi đã đi đến việc phân tích trực quan về đề tài để có cái nhìn tương đối khái quát, từ đó đưa ra phương pháp nghiên cứu chủ yếu là phương pháp thực nghiệm định lượng.

Các bước cốt lõi của phương pháp thực nghiệm định lượng như sau:

1. **Xác định Vấn đề & Mục tiêu:** Tìm hiểu về thuật toán LTR nào sẽ được nghiên cứu, bộ dữ liệu nào sẽ được sử dụng, chỉ số nào sẽ được tối ưu hóa để đánh giá.
2. **Thu thập & Tiền xử lý Dữ liệu:** Các bộ dữ liệu gốc, dữ liệu đầu vào, cặp dữ liệu huấn luyện được sử dụng, xử lý ra sao, như thế nào.
3. **Lựa chọn & Triển khai Thuật toán:** Triển khai hoặc sử dụng các thư viện cho RankNet ví dụ như thư viện nổi tiếng: PyTorch…
4. **Huấn luyện & Tinh chỉnh Mô hình:** Triển khai phương pháp huấn luyện theo mô hình RankNet đã xây dựng
5. **Đánh giá Hiệu suất:** Tính toán các chỉ số đánh giá đã chọn trên tập kiểm tra.

# CƠ SỞ LÝ THUYẾT

## Tổng quan về Learn to rank

### Khái niệm cơ bản

Learn to Rank (LTR) là một mô hình của Machine Learning được sử dụng để giải quyết các vấn đề xếp hạng trong các hệ thống truy xuất thông tin. Phương pháp này bao gồm việc xây dựng các mô hình có khả năng dự đoán thứ tự sắp xếp tối ưu của một danh sách đối tượng dựa trên các đặc trưng đầu vào. Những đối tượng này có thể là kết quả tìm kiếm, gợi ý sản phẩm, hay nội dung trên mạng xã hội,...

Mục tiêu của LTR là học được một hàm xếp hạng có thể sắp xếp các đối tượng theo cách tối đa hóa mức độ phù hợp với truy vấn hoặc sở thích của người dùng. Đây là yếu tố then chốt trong các công cụ tìm kiếm như Google hay Bing, nơi thứ tự hiển thị kết quả có thể ảnh hưởng trực tiếp đến mức độ hài lòng và tương tác của người dùng.

Learn to Rank cung cấp một giải pháp có hệ thống và hiệu quả để giải quyết bài toán này.

### Ứng dụng

Learn to Rank (LTR) là một công nghệ AI quan trọng với phạm vi ứng dụng rộng rãi. LTR cải thiện hiệu quả của các công cụ tìm kiếm bằng cách sắp xếp các trang web theo mức độ phù hợp với truy vấn của người dùng, đồng thời tối ưu hóa danh sách sản phẩm trong thương mại điện tử dựa trên sở thích hoặc khả năng mua hàng của khách hàng.

Công nghệ này còn đóng vai trò thiết yếu trong việc cá nhân hóa trải nghiệm người dùng, thông qua việc đề xuất nội dung phù hợp trên mạng xã hội và các nền tảng phát trực tuyến. Ngoài ra, LTR còn được sử dụng để xếp hạng quảng cáo trực tuyến nhằm tối đa hóa tính liên quan và mức độ tương tác của người dùng. Với vai trò là yếu tố cốt lõi trong cách thông tin, sản phẩm và nội dung được khám phá trên môi trường số, LTR tiếp tục là trọng tâm nghiên cứu và phát triển trong lĩnh vực trí tuệ nhân tạo.

## Các hướng tiếp cận của Learn to rank

### Phương pháp Pointwise

Phương pháp Pointwise xem xét từng đối tượng riêng lẻ trong hàm mất mát (loss function). Cụ thể, mỗi đối tượng được xử lý độc lập, và mô hình sẽ huấn luyện một bộ phân loại hoặc hồi quy để dự đoán mức độ liên quan của đối tượng đó đối với truy vấn hiện tại. Việc xếp hạng cuối cùng được thực hiện bằng cách sắp xếp danh sách kết quả theo điểm số mà mô hình gán cho từng đối tượng. Trong phương pháp Pointwise, điểm số của mỗi đối tượng được tính toán một cách độc lập (không phụ thuộc vào các đối tượng khác trong danh sách kết quả tương ứng với truy vấn), sau đó các đối tượng được sắp xếp dựa trên điểm số đó.

Tất cả các thuật toán hồi quy và phân loại tiêu chuẩn đều có thể được sử dụng trực tiếp trong phương pháp học để xếp hạng kiểu Pointwise.

### Phương pháp Pairwise

Phương pháp Pairwise - Phương pháp này tập trung vào việc xác định đúng thứ tự giữa các cặp đối tượng trong hàm mất mát(loss function) . Bài toán xếp hạng được chuyển thành bài toán phân loại nhị phân, trong đó thuật toán học cách xác định thứ tự tối ưu giữa cặp đối tượng và so sánh với thứ tự đúng trong dữ liệu gốc(ground truth). Mục tiêu của mô hình xếp hạng là giảm thiểu số lượng hoán vị sai - tức là các trường hợp mà thứ tự của cặp tài liệu bị sắp xếp ngược so với thứ tự thực tế.

So với phương pháp Pointwise, phương pháp Pairwise thường cho kết quả tốt hơn trong thực tế, vì việc dự đoán thứ tự tương đối giữa các đối tượng phù hợp hơn với bản chất của bài toán xếp hạng so với việc dự đoán nhãn lớp hoặc điểm liên quan tuyệt đối.

Một số thuật toán LTR phổ biến nhất như RankNet, LambdaRank, và LambdaMART đều dựa trên phương pháp Pairwise.

### Phương pháp Listwise

Phương pháp Listwise — Phương pháp này xem xét toàn bộ danh sách đối tượng và tìm cách tối ưu hóa thứ tự của toàn bộ danh sách. Có hai kỹ thuật chính được sử dụng trong LTR theo phương pháp Listwise:

+ Tối ưu hóa trực tiếp các biện pháp truy xuất thông tin (IR - Information Retrieval) như NDCG (Normalized Discounted Cumulative Gain). Ví dụ như các thuật toán của Softrank, Adarank.

+ Tối thiểu hoá hàm mất mát được xác định dựa trên việc hiểu sâu về các đặc điểm riêng của loại xếp hạng mà mô hình hướng tới. Ví dụ như ListNet, ListMLE

Đây là phương pháp hướng trực tiếp đến kết quả xếp hạng cuối cùng và thường cho hiệu quả cao hơn vì nó khai thác ngữ cảnh tổng thể của danh sách.

# CÁC THUẬT TOÁN TRONG LEARN TO RANK

## RankNet: Nền tảng của Học để Xếp hạng theo cặp

### Lý thuyết và cơ chế hoạt động

RankNet, được giới thiệu bởi Christopher J.C. Burges và cộng sự tại Microsoft Research vào năm 2005 [1], là một trong những thuật toán tiên phong trong lĩnh vực LTR theo phương pháp Pairwise. Ý tưởng cốt lõi của RankNet là học một hàm xếp hạng bằng cách tối ưu hóa xác suất tương đối giữa các cặp tài liệu. Thay vì cố gắng dự đoán một điểm liên quan tuyệt đối cho từng tài liệu, RankNet tập trung vào việc học để phân biệt tài liệu nào nên được xếp hạng cao hơn tài liệu nào trong cùng một truy vấn.

Cụ thể, RankNet sử dụng một mạng nơ-ron truyền thẳng (feed-forward neural network) để ánh xạ các đặc trưng của tài liệu thành một điểm số liên quan vô hướng. Đối với mỗi cặp tài liệu (A, B) trong một truy vấn, mạng nơ-ron sẽ tính toán điểm số sA và sB. Sau đó, RankNet sử dụng hàm sigmoid để ước lượng xác suất P (A ≻ B), tức là xác suất tài liệu A được xếp hạng cao hơn tài liệu B.

Công thức tính xác suất này là: Trong đó, sA và sB là điểm số đầu ra của mạng nơ-ron cho tài liệu A và B. Mục tiêu của quá trình huấn luyện là làm cho xác suất này càng gần với xác suất thực tế (được suy ra từ nhãn liên quan của dữ liệu huấn luyện) càng tốt. Nếu tài liệu A thực sự liên quan hơn B, xác suất thực tế là 1; nếu B liên quan hơn A, xác suất thực tế là 0; nếu chúng có độ liên quan như nhau, xác suất thực tế là 0.5. Để đạt được mục tiêu này, RankNet sử dụng hàm mất mát Binary Cross-Entropy (BCE). Hàm mất mát cho một cặp tài liệu (A, B) được định nghĩa là:

*C*(*sA*, *sB*) = −*P*ˉ(*A* ≻ *B*) log *P* (*A* ≻ *B*) − (1 − *P*ˉ(*A* ≻ *B*)) log(1 − *P* (*A* ≻ *B*))

Trong đó, Pˉ(A ≻ B) là xác suất thực tế (ground truth) mà A được xếp hạng cao hơn B. Quá trình huấn luyện diễn ra bằng cách sử dụng thuật toán lan truyền ngược (backpropagation) và tối ưu hóa gradient descent để cập nhật các trọng số của mạng nơ-ron, nhằm giảm thiểu tổng hàm mất mát trên toàn bộ các cặp tài liệu trong tập huấn luyện. Điều này đảm bảo rằng mạng nơ-ron học được cách gán điểm số sao cho thứ tự tương đối giữa các tài liệu được bảo toàn.

### Các bước ứng dụng RankNet vào bài toán Learn to rank thực tế

Bước 1: Tiền xử lý dữ liệu: Đây là bước quan trọng đầu tiên, bao gồm việc thu thập dữ liệu truy vấn-tài liệu và gán nhãn liên quan (relevance labels). Các tài liệu cần được biểu diễn dưới dạng vector đặc trưng. Các phương pháp phổ biến bao gồm TF-IDF (Term Frequency-Inverse Document Frequency), BM25, hoặc các biểu diễn nhúng (embeddings) từ các mô hình ngôn ngữ lớn (ví dụ: Word2Vec, BERT, Sentence-BERT) để nắm bắt ngữ nghĩa của tài liệu và truy vấn.

Bước 2: Sinh cặp dữ liệu (Pairwise Data Generation): Từ tập dữ liệu đã được gán nhãn, chúng ta tạo ra các cặp tài liệu (A, B) cho mỗi truy vấn. Nếu tài liệu A có nhãn liên quan cao hơn tài liệu B, cặp này được gán nhãn Pˉ(A ≻ B) = 1. Ngược lại, nếu B liên quan hơn A, nhãn là Pˉ(A ≻ B) = 0. Nếu cả hai có cùng mức độ liên quan, nhãn là Pˉ(A ≻ B) = 0.5. Việc tạo ra các cặp này cần đảm bảo tính cân bằng để tránh thiên vị trong quá trình huấn luyện.

Bước 3: Xây dựng và Huấn luyện Mạng Nơ-ron: Một mạng nơ-ron truyền thẳng được thiết kế để nhận đầu vào là vector đặc trưng của một tài liệu và trả về một điểm số vô hướng. Mạng này thường có một hoặc nhiều lớp ẩn với các hàm kích hoạt phi tuyến tính (ví dụ: ReLU). Quá trình huấn luyện bao gồm việc đưa các cặp tài liệu đã sinh vào mạng, tính toán điểm số sA, sB, sau đó tính xác suất P (A ≻ B) và hàm mất mát BCE. Thuật toán tối ưu (như Adam, SGD) được sử dụng để cập nhật trọng số của mạng.

Bước 4: Đánh giá Mô hình: Sau khi huấn luyện, hiệu suất của mô hình RankNet được đánh giá trên một tập dữ liệu kiểm tra độc lập. Các chỉ số đánh giá phổ biến trong LTR bao gồm Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), và Precision@k. Các chỉ số này đo lường mức độ hiệu quả của mô hình trong việc xếp hạng các tài liệu liên quan lên các vị trí cao hơn.

### Ưu và nhược điểm

### *3.1.3.1. Ưu điểm*

Đơn giản và dễ áp dụng: So với các phương pháp LTR phức tạp hơn, RankNet có cấu trúc tương đối đơn giản, dễ hiểu và triển khai. Nó chỉ yêu cầu nhãn liên quan tương đối giữa các cặp tài liệu, không cần nhãn liên quan tuyệt đối chính xác, giúp giảm gánh nặng gán nhãn dữ liệu.

Khả năng tổng quát hóa tốt: Mạng nơ-ron có khả năng học các mối quan hệ phi tuyến tính phức tạp giữa các đặc trưng, cho phép RankNet tổng quát hóa tốt trên các tập dữ liệu đa dạng.

Nền tảng cho các thuật toán sau này: RankNet đã đặt nền móng cho nhiều thuật toán LTR tiên tiến hơn như LambdaRank và LambdaMART, chứng minh hiệu quả của phương pháp Pairwise và việc sử dụng mạng nơ-ron trong bài toán xếp hạng.

### *3.1.3.2. Nhược điểm*

Không tối ưu trực tiếp các chỉ số xếp hạng: RankNet tối ưu hàm mất mát BCE, vốn là một proxy cho các chỉ số xếp hạng thực tế như NDCG. Điều này có nghĩa là việc giảm thiểu hàm mất mát không nhất thiết dẫn đến sự cải thiện tối ưu trên các chỉ số xếp hạng mà người dùng quan tâm. Ví dụ, một lỗi xếp hạng ở vị trí cao có thể có tác động lớn hơn nhiều so với một lỗi ở vị trí thấp, nhưng BCE không phân biệt được điều này một cách rõ ràng.

Bỏ qua thông tin toàn bộ danh sách: RankNet chỉ xem xét các cặp tài liệu một cách độc lập. Nó không tận dụng được thông tin về cấu trúc toàn bộ danh sách các tài liệu được trả về cho một truy vấn. Điều này có thể dẫn đến việc bỏ lỡ các mối quan hệ phức tạp hơn giữa nhiều tài liệu trong cùng một ngữ cảnh xếp hạng.

Chi phí tính toán cao: Việc tạo ra và huấn luyện trên tất cả các cặp tài liệu có thể rất tốn kém về mặt tính toán, đặc biệt khi số lượng tài liệu cho mỗi truy vấn lớn. Số lượng cặp tăng theo cấp số nhân với số lượng tài liệu, làm tăng đáng kể thời gian huấn luyện.

## LamdaRank: Tối ưu hóa trực tiếp các chỉ số xếp hạng

### Lý thuyết và cơ chế hoạt động

LambdaRank, được giới thiệu bởi Christopher J.C. Burges và cộng sự vào năm 2006 [2], là một bước tiến quan trọng từ RankNet. Nhược điểm chính của RankNet là nó tối ưu một hàm mất mát proxy (Binary Cross-Entropy) thay vì trực tiếp các chỉ số xếp hạng phi khả vi như NDCG (Normalized Discounted Cumulative Gain) hay MAP (Mean Average Precision). Các chỉ số này thường không liên tục và không khả vi, gây khó khăn cho việc áp dụng các phương pháp tối ưu hóa dựa trên gradient truyền thống.

LambdaRank giải quyết vấn đề này bằng một cách tiếp cận khéo léo: thay vì định nghĩa một hàm mất mát mới, nó điều chỉnh trực tiếp gradient được sử dụng trong quá trình lan truyền ngược. Ý tưởng chính là 'giả lập' gradient của một hàm mất mát lý tưởng mà nếu được tối ưu, sẽ trực tiếp cải thiện các chỉ số xếp hạng. Các gradient này được gọi là 'lambda gradients' hoặc 'lambda'.

Cụ thể, đối với mỗi cặp tài liệu (A, B) trong một truy vấn, RankNet tính toán gradient của hàm mất mát BCE theo điểm số sA và sB. LambdaRank sửa đổi gradient này bằng cách nhân nó với một hệ số ΔNDCGAB, là sự thay đổi của chỉ số NDCG nếu thứ tự của tài liệu A và B bị hoán đổi. Công thức cho lambda gradient đối với tài liệu i và j là:

. ∣Δ*NDCGij* ∣

Trong đó, là gradient của hàm mất mát RankNet thông thường, và ∣Δ*NDCGij* ∣ là giá trị tuyệt đối của sự thay đổi NDCG khi hoán đổi vị trí của tài liệu i và j. Dấu của ΔNDCGij được sử dụng để điều chỉnh hướng của gradient. Bằng cách này, LambdaRank ưu tiên các cặp tài liệu mà việc xếp hạng sai có tác động lớn nhất đến chỉ số NDCG. Điều này có nghĩa là mô hình sẽ học cách sửa chữa những lỗi xếp hạng quan trọng hơn trước. Quá trình huấn luyện của LambdaRank tương tự như RankNet, sử dụng mạng nơ-ron và lan truyền ngược. Tuy nhiên, thay vì sử dụng gradient thông thường, nó sử dụng các lambda gradient đã được điều chỉnh. Tổng hợp các lambda gradient cho mỗi tài liệu sẽ cho ta gradient cuối cùng để cập nhật trọng số của mạng nơ-ron.

### Các bước ứng dụng LambdaRank vào bài toán LTR thực tế:

Bước 1: Chuẩn bị dữ liệu: Tương tự như RankNet, dữ liệu cần được tiền xử lý, vector hóa và gán nhãn liên quan. Các cặp tài liệu cũng được sinh ra.

Bước 2: Tính điểm liên quan và Gradient RankNet: Mạng nơ-ron được sử dụng để tính điểm số sA và sB cho các cặp tài liệu, và gradient thông thường của hàm mất mát RankNet được tính toán.

Bước 3: Tính toán ΔNDCG: Đây là bước then chốt của LambdaRank. Đối với mỗi cặp tài liệu (A, B), chúng ta cần tính toán sự thay đổi của chỉ số NDCG nếu thứ tự của A và B bị hoán đổi. Việc này đòi hỏi phải biết điểm liên quan thực tế của các tài liệu và vị trí của chúng trong danh sách xếp hạng hiện tại. Việc tính toán ΔNDCG có thể phức tạp và tốn kém về mặt tính toán, đặc biệt với danh sách dài.

Bước 4: Tính toán Lambda Gradient: Gradient thông thường được nhân với ∣ΔNDCGAB∣ để tạo ra lambda gradient. Các lambda gradient này sau đó được tổng hợp cho mỗi tài liệu.

Bước 5: Cập nhật trọng số mạng nơ-ron: Các lambda gradient tổng hợp được sử dụng để cập nhật trọng số của mạng nơ-ron thông qua thuật toán tối ưu hóa (ví dụ: Gradient Descent).

Bước 6: Lặp lại huấn luyện: Quá trình này được lặp lại qua nhiều vòng (epochs) cho đến khi mô hình hội tụ và đạt được hiệu suất mong muốn trên tập validation.

### Ưu và nhược điểm

*3.2.3.1. Ưu điểm*

Tối ưu trực tiếp các chỉ số xếp hạng: Đây là ưu điểm lớn nhất của LambdaRank. Bằng cách điều chỉnh gradient dựa trên tác động đến NDCG (hoặc các chỉ số khác), LambdaRank trực tiếp tối ưu hóa những gì người dùng thực sự quan tâm, dẫn đến hiệu suất xếp hạng tốt hơn so với RankNet.

Không cần định nghĩa hàm mất mát mới: LambdaRank đạt được mục tiêu tối ưu trực tiếp mà không cần phải thiết kế một hàm mất mát phi khả vi phức tạp. Nó chỉ sửa đổi gradient của hàm mất mát hiện có, giúp việc triển khai dễ dàng hơn.

Hiệu quả hơn trong việc sửa lỗi quan trọng: Do ưu tiên các cặp tài liệu có tác động lớn đến NDCG, LambdaRank có xu hướng sửa chữa các lỗi xếp hạng ở các vị trí cao (quan trọng hơn) trước, dẫn đến cải thiện đáng kể chất lượng của top-k kết quả.

*3.2.3.1. Nhược điểm*

Phức tạp hơn trong tính toán ΔNDCG: Việc tính toán ΔNDCG cho mỗi cặp tài liệu có thể rất phức tạp và tốn kém về mặt tính toán, đặc biệt khi danh sách tài liệu dài. Điều này đòi hỏi phải có kiến thức sâu về các chỉ số xếp hạng và cách chúng thay đổi khi thứ tự tài liệu thay đổi.

Yêu cầu thư viện hỗ trợ: Do sự phức tạp trong việc tính toán lambda gradient, việc triển khai LambdaRank từ đầu có thể khó khăn. Thông thường, người ta sẽ sử dụng các thư viện học máy chuyên biệt đã tích hợp sẵn LambdaRank hoặc các biến thể của nó (ví dụ: LightGBM, XGBoost).

Vẫn là phương pháp Pairwise: Mặc dù cải thiện đáng kể so với RankNet, LambdaRank vẫn là một phương pháp Pairwise. Nó vẫn không tận dụng được đầy đủ thông tin về cấu trúc toàn bộ danh sách các tài liệu, điều mà các phương pháp Listwise có thể làm được.

## LamdaMART: Sức mạnh của Cây Quyết định Tăng cường

### Lý thuyết và Cơ chế hoạt động

LambdaMART (Lambda Multiple Additive Regression Trees) là một trong những thuật toán Học để Xếp hạng (LTR) mạnh mẽ và được sử dụng rộng rãi nhất hiện nay, đặc biệt trong các hệ thống tìm kiếm thương mại. Nó là sự kết hợp tinh tế giữa ý tưởng của LambdaRank và Gradient Boosted Decision Trees (GBDT), cụ thể là Multiple Additive Regression Trees (MART) [3]. Trong khi RankNet và LambdaRank sử dụng mạng nơ-ron, LambdaMART tận dụng sức mạnh của các cây quyết định tăng cường để học hàm xếp hạng.

Ý tưởng cốt lõi của LambdaMART là xây dựng một ensemble các cây quyết định. Mỗi cây mới được thêm vào sẽ cố gắng dự đoán 'lambda gradients' (được tính toán tương tự như trong LambdaRank) cho từng tài liệu. Các lambda gradients này biểu thị mức độ quan trọng của việc thay đổi điểm số của một tài liệu để cải thiện chỉ số xếp hạng tổng thể (ví dụ: NDCG).

Thay vì tối ưu hóa một hàm mất mát trực tiếp, LambdaMART tối ưu hóa các lambda gradients này, từ đó gián tiếp tối ưu hóa các chỉ số xếp hạng phi khả vi.

Quá trình huấn luyện của LambdaMART diễn ra theo từng bước (boosting). Ở mỗi bước, một cây quyết định mới được huấn luyện để dự đoán các lambda gradients còn lại (residual lambda gradients) từ các cây trước đó. Các dự đoán của cây mới sau đó được cộng vào tổng điểm số của tài liệu. Quá trình này lặp lại cho đến khi đạt được một số lượng cây nhất định hoặc khi hiệu suất không còn cải thiện đáng kể.

### Các bước ứng dụng LambdaMART vào bài toán LTR thực tế:

Bước 1: Tiền xử lý dữ liệu: Tương tự như các thuật toán LTR khác, dữ liệu cần được chuẩn hóa và biểu diễn dưới dạng các vector đặc trưng. Việc sinh đặc trưng cho tài liệu và truy vấn là cực kỳ quan trọng đối với hiệu suất của LambdaMART.

Bước 2: Khởi tạo mô hình: Ban đầu, mô hình có thể được khởi tạo với một giá trị điểm số cố định cho tất cả các tài liệu, hoặc bằng một cây quyết định đơn giản.

Bước 3: Tính toán Lambda Gradients: Ở mỗi vòng lặp boosting, dựa trên điểm số hiện tại của các tài liệu, chúng ta tính toán các lambda gradients cho từng tài liệu. Việc này bao gồm việc tính toán sự thay đổi của chỉ số xếp hạng (ví dụ: NDCG) nếu điểm số của tài liệu đó thay đổi, tương tự như cách LambdaRank tính toán ΔNDCG.

Bước 4: Huấn luyện cây quyết định mới: Một cây quyết định hồi quy (regression tree) mới được huấn luyện. Mục tiêu của cây này là dự đoán các lambda gradients đã tính toán ở bước trước. Các lá của cây sẽ chứa các giá trị dự đoán cho lambda gradients.

Bước 5: Cập nhật điểm số tài liệu: Điểm số của mỗi tài liệu được cập nhật bằng cách cộng thêm giá trị dự đoán từ cây quyết định mới (thường có một hệ số học tập - learn rate).

Bước 6: Lặp lại: Các bước 3, 4, 5 được lặp lại cho đến khi đạt được số lượng cây mong muốn hoặc khi mô hình hội tụ.

Bước 7: Đánh giá mô hình: Sau khi huấn luyện, mô hình được đánh giá bằng các chỉ số xếp hạng như NDCG, MAP, Precision@k trên tập dữ liệu kiểm tra.

### Ưu và nhược điểm

### *3.3.3.1. Ưu điểm*

Hiệu suất cao: LambdaMART thường đạt được hiệu suất xếp hạng vượt trội so với RankNet và LambdaRank, đặc biệt trên các tập dữ liệu lớn và phức tạp. Sự kết hợp giữa tối ưu hóa gradient hiệu quả của LambdaRank và khả năng xử lý các mối quan hệ phi tuyến tính phức tạp của GBDT làm cho nó trở thành một lựa chọn mạnh mẽ.

Xử lý tốt dữ liệu rời rạc và hỗn hợp: Cây quyết định có khả năng xử lý tốt cả đặc trưng liên tục và rời rạc mà không cần tiền xử lý phức tạp. Điều này làm cho LambdaMART rất linh hoạt với các loại dữ liệu khác nhau.

Khả năng giải thích (ở mức độ nhất định): Mặc dù là một mô hình ensemble, các cây quyết định riêng lẻ có thể được phân tích để hiểu được tầm quan trọng của các đặc trưng trong quá trình xếp hạng.

Tối ưu trực tiếp chỉ số xếp hạng: Giống như LambdaRank, LambdaMART trực tiếp tối ưu hóa các chỉ số xếp

### *3.3.3.2. Nhược điểm*

Cần điều chỉnh nhiều tham số (Hyperparameter Tuning): LambdaMART có nhiều siêu tham số cần được điều chỉnh (ví dụ: số lượng cây, độ sâu tối đa của cây, tốc độ học, tỷ lệ lấy mẫu). Việc tìm kiếm bộ tham số tối ưu có thể tốn thời gian và tài nguyên tính toán.

Xử lý chậm trên tập dữ liệu rất lớn: Mặc dù hiệu quả, quá trình huấn luyện của GBDT có thể chậm hơn so với mạng nơ-ron trên các tập dữ liệu cực lớn, đặc biệt nếu không được tối ưu hóa tốt hoặc không sử dụng các thư viện hiệu quả (như LightGBM, XGBoost).

Dễ bị quá khớp (Overﬁtting): Nếu không được điều chỉnh tham số cẩn thận, LambdaMART có thể dễ dàng bị quá khớp với dữ liệu huấn luyện, dẫn đến hiệu suất kém trên dữ liệu mới.

## So sánh tổng quan các thuật toán

Để có cái nhìn toàn diện về ba thuật toán RankNet, LambdaRank và LambdaMART, bảng dưới đây tóm tắt các đặc điểm chính, ưu nhược điểm và phạm vi ứng dụng của từng phương pháp. Việc lựa chọn thuật toán phù hợp phụ thuộc vào nhiều yếu tố như yêu cầu về độ chính xác, khả năng tính toán, loại dữ liệu và mục tiêu cụ thể của bài toán xếp

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí | RankNet | LambdaRank | LambdaMART |
| Loại học | Pairwise (học theo cặp) | Pairwise (học theo cặp) | Listwise (học theo danh sách) |
| Mô hình học | Mạng nơ-ron truyền thẳng | Mạng nơ-ron truyền thẳng | Cây quyết định tăng cường (GBDT) |
| Mục tiêu tối ưu | Hàm mất mát Binary Cross- Entropy (proxy) | Trực tiếp tối ưu chỉ số xếp hạng (ví dụ: NDCG) thông qua lambda gradients | Trực tiếp tối ưu chỉ số xếp hạng (ví dụ: NDCG) thông qua lambda gradients |
| Tối ưu NDCG trực tiếp | Không | Có | Có |
| Đầu vào | Cặp tài liệu (đặc trưng và nhãn liên quan tương đối) | Cặp tài liệu (đặc trưng và nhãn liên quan tương đối) | Tập hợp tài liệu cho một truy vấn (đặc trưng và nhãn liên quan |
| Độ phức tạp triển khai | Trung bình | Cao (do tính toán  ΔNDCG) | Cao (do điều chỉnh siêu tham số và cấu trúc GBDT) |
| Hiệu suất xếp hạng | Tốt (nền tảng) | Rất tốt (cải thiện đáng kể so với RankNet) | Xuất sắc (thường là tốt nhất trong thực tế) |
| Khả năng xử lý dữ liệu phi tuyến tính | Cao | Cao | Rất cao |
| Khả năng xử lý dữ liệu rời rạc/hỗn hợp | Trung bình | Trung bình | Rất tốt |
| Rủi ro quá khớp | Trung bình | Trung bình | Cao (nếu không điều chỉnh tham số cẩn thận) |
| Tốc độ huấn luyện | Nhanh (với mạng nơ-ron nhỏ) | Trung bình | Chậm (với tập dữ liệu rất lớn và nhiều cây) |

Bảng 3.1. So sánh các thuật toán RankNet, LambdaRank và LambdaMART

Từ bảng so sánh trên, có thể thấy rằng mỗi thuật toán đều có những điểm mạnh và hạn chế riêng. RankNet là một điểm khởi đầu vững chắc, dễ hiểu và triển khai, nhưng không trực tiếp tối ưu các chỉ số xếp hạng quan trọng. LambdaRank khắc phục nhược điểm này bằng cách điều chỉnh gradient để tập trung vào việc cải thiện NDCG, mang lại hiệu suất tốt hơn. Cuối cùng, LambdaMART kết hợp ý tưởng tối ưu hóa của LambdaRank với sức mạnh của GBDT, tạo ra một mô hình cực kỳ hiệu quả và linh hoạt, thường là lựa chọn hàng đầu cho các ứng dụng thực tế quy mô lớn. Tuy nhiên, sự phức tạp trong việc điều chỉnh tham số và nguy cơ quá khớp đòi hỏi người triển khai phải có kinh nghiệm và sự cẩn trọng.

# CÀI ĐẶT & THỰC NGHIỆM



## Dữ liệu đầu vào

### Thu thập dữ liệu

Chủ đề traning của nhóm em là về “Ca dao tục ngữ Việt Nam”. Dữ liệu gốc được lưu dạng json với cấu trúc như dưới:  
[

  {

    "id": 1,

    "value": "Ăn quả nhớ kẻ trồng cây"

  }

]  
Trong đó:

* “id” là khóa để xác định câu ca dao tục ngữ.
* “value” là trường chứa nội dung ca dao tục ngữ.

Nhóm đã thực hiện tìm kiếm trên internet và cào về được khoảng gần 1000 câu ca dao tục ngữ Việt Nam để phục vụ cho các bước tiền xử lý đầu vào.

### Tiền xử lý dữ liệu

Khi có bộ dữ liệu. Ta tiến hành bước tiền xử lý dữ liệu để cho ra được input của mô hình RankNet bằng các bước như sau:  
- Thực hiện tách từ, xóa các dấu câu dữ thừa  
- Xóa bỏ hư từ (stopword)  
- Thực hiện vector hóa với thư viện sentence\_transformers

Sau khi thực hiện xong các bước trên, ta tiếp tục thực hiện tạo cặp khóa huấn luyện với mỗi cặp tài liệu, ký hiệu là di và dj . Và gán nhãn cho các cặp này.  
Việc gán nhãn cặp tài liệu có thể thực hiện ở bước chuẩn bị dữ liệu hoặc ngay trong bước chạy huấn luyện.

## Huấn luyện

### Chuẩn bị

Để tiến hành chạy demo với mô hình RankNet ta cần import các thư viện cần thiết như

* PyTorch: PyTorch là một framework học máy mã nguồn mở được sử dụng rộng rãi để xây dựng và huấn luyện các mô hình học sâu (deep learning). Được phát triển bởi Facebook (nay là Meta), PyTorch nổi bật với khả năng xử lý tính toán tensor mạnh mẽ và khả năng hỗ trợ GPU, giúp tăng tốc đáng kể quá trình huấn luyện mô hình.
* Scikit-learn: là một thư viện mã nguồn mở phổ biến của **Python** dành cho **học máy (machine learning)**. Nó cung cấp một bộ công cụ toàn diện và hiệu quả để thực hiện các tác vụ phân tích dữ liệu và xây dựng mô hình học máy.
* **SentenceTransformers:** là một thư viện Python mã nguồn mở dựa trên PyTorch và thư viện Transformers của Hugging Face. Mục tiêu chính của nó là **cung cấp các mô hình học sâu hiệu quả** để tạo ra các vector nhúng (embeddings) dày đặc cho câu, đoạn văn và thậm chí cả hình ảnh. Các vector nhúng này được thiết kế sao cho các văn bản có ý nghĩa tương tự sẽ có các vector gần nhau trong không gian đa chiều.

### Huấn luyện

Chạy huấn luyện với google colab . Tận dụng được các CPU, GPU mạnh thì chạy huấn luyện với lượng dữ liệu nhỏ sẽ nhanh rất nhiều. Hình bên dưới là log cho thấy biên độ giá trị của hàm mất mát đang được tối ưu xuống.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Nội dung do AI tạo ra có thể không chính xác.

Hình 1: Hình ảnh huấn luyện model RankNet

# KẾT LUẬN & HƯỚNG PHÁT TRIỂN

## Kết luận

Learning to Rank là một hướng tiếp cận mạnh mẽ và hiệu quả trong các bài toán **xếp hạng** dữ liệu, đặc biệt trong **truy hồi thông tin**, **hệ thống tìm kiếm** và **hệ thống gợi ý**.  
Bằng cách tận dụng sức mạnh của **học máy** (từ mô hình truyền thống đến học sâu), LTR cho phép mô hình:

1. **Tự động học** cách sắp xếp kết quả dựa trên dữ liệu huấn luyện thay vì chỉ dựa vào luật thủ công.
2. **Tối ưu trực tiếp** các chỉ số đánh giá (như NDCG, MAP, MRR) phù hợp với yêu cầu thực tế.
3. **Linh hoạt** trong việc kết hợp nhiều loại đặc trưng (nội dung, ngữ cảnh, hành vi người dùng).

Các phương pháp LTR hiện nay có thể phân thành ba nhóm chính:

* **Pointwise**: coi bài toán như dự đoán giá trị cho từng tài liệu.
* **Pairwise**: tối ưu thứ hạng dựa trên so sánh từng cặp tài liệu (ví dụ: RankNet).
* **Listwise**: tối ưu toàn bộ danh sách kết quả theo các thước đo xếp hạng.

RankNet một trong những thuật toán nổi tiếng tạo nền tảng cho các thuật toán tốt hơn, mạnh mẽ hơn sau này.

## Hướng phát triển

Triển khai mô hình học máy với RankNet là một quá trình học tập và phát triển tốt cho chúng ta có nền tảng vững chắc hơn, chuyên sâu hơn trong việc học nâng cao trong lĩnh vực machine learning.

Đề tài và mô hình huấn luyện vẫn còn một số thiếu xót như:

* Thời gian chạy huấn luyện còn tốn kém
* Tập dữ liệu đầu vào còn chưa đa dạng
* Phần xử lý còn hơi rườm rà, lộn xộn

Do đó, việc tối ưu mô hình cũng như việc cải thiện nội dung đề tài là việc cần thiết và nhóm sẽ cập nhật, bổ sung sau để có đề tài tốt hơn.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | C. J. C. S. T. R. E. L. A. D. M. H. N. &. S. R. Burges, "Learning to Rank Using Gradient Descent," in *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005. |
| [2] | C. J. C. Burges, "From RankNet to LambdaRank to LambdaMART: An overview," 2006. |
| [3] | C. J. C. Burges, "From RankNet to LambdaRank to LambdaMART: An Overview.," in *Microsoft Research Technical Report MSR-TR-2006-82.*, 2006. |