**Seminar ML : Youtube Video Recommondations**

**Problem Statement**

**-**Xây dựng đề xuất video cho người dùng YouTube. Chúng tôi muốn tối đa hóa mức độ tương tác của người dùng cũng như đề xuất các loại nội dung mới cho người dùng.

**Metrics design and Requirements**

**\*Metrics:**

**-**For offline metrics: sử dụng độ chính xác (tỷ lệ các trường hợp liên quan trong số các trường hợp đã được truy xuất), độ bao phủ (tỷ lệ các trường hợp liên quan thực tế được truy xuất so với tổng số trường hợp liên quan), loss xếp hạng và log loss.

-For online metrics: sử dụng kiểm tra A/B và so sánh tỷ lệ nhấp chuột (CTR), thời gian xem và tỷ lệ chuyển đổi.

**\*Requirements**

**-Training:**

+ Hành vi của người dùng thường không thể đoán trước, và video có thể trở nên lan truyền chỉ sau một đêm. Lý tưởng nhất là chúng ta muốn huấn luyện mô hình nhiều lần trong ngày để nắm bắt được những thay đổi theo thời gian.

**-Inference(Suy luận):**

+ Đối với mỗi người dùng truy cập trang chủ, chúng ta cần đề xuất 100 video cho họ. Thời gian trễ phải dưới 200ms, hoặc lý tưởng là dưới 100ms.

+ Đối với các đề xuất trực tuyến, điều quan trọng là phải tìm được sự cân bằng giữa khám phá và khai thác(exploitation). Nếu các mô hình khai thác quá mức dữ liệu lịch sử, các video mới có thể không được tiếp cận người dùng. Chúng ta muốn cân bằng giữa mức độ liên quan và nội dung mới.

**-Summary:**

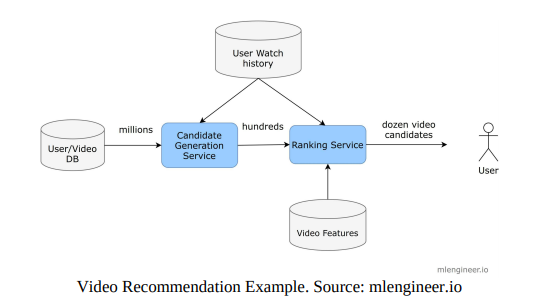
+ Các chỉ số: Độ chính xác hợp lý(reasonable precision), độ bao phủ(recall) cao.

+ Huấn luyện với thông lượng cao(high và khả năng tái huấn luyện nhiều lần trong ngày.

+ Suy luận cần có độ trễ(latency) thấp, từ 100ms đến 200ms.

+ Chúng ta cần có cách để điều chỉnh cân bằng giữa khám phá và khai thác trong quá trình suy luận.

**Multistage Models**

****

-Có hai giai đoạn(stage): tạo ứng viên(candidate generation) và xếp hạng(ranking). Lý do thêm hai giai đoạn là để mở rộng hệ thống:

+ Mô hình tạo ứng viên sẽ tìm các video liên quan dựa trên lịch sử xem của người dùng và loại video mà họ đã xem.

+ Mô hình xếp hạng sẽ tối ưu hóa khả năng xem (tức là những video có khả năng xem cao sẽ được xếp hạng cao). Điều này phù hợp tự nhiên với logistic regression althorithm.

**\*\*Model training**

**\*Candidate Generation Model**

**-Training Data:**

+ Để tạo dữ liệu huấn luyện, chúng ta có thể tạo một không gian user-video watch.

+Có thể bắt đầu bằng cách chọn một khoảng thời gian dữ liệu: tháng trước, sáu tháng trước, v.v. Điều này sẽ giúp tìm ra sự cân bằng giữa thời gian huấn luyện và độ chính xác của mô hình.

**-Feature Enginneering:**

+ Mỗi người dùng có một danh sách các video đã xem (video, số phút đã xem).

**-Model:**

+ Quá trình tạo ứng viên có thể được thực hiện bằng phương pháp phân tích ma trận (matrix factorization).Mục đích của việc tạo ứng viên là tạo ra somewhat relevant content với người dùng dựa trên lịch sử xem của họ. Danh sách ứng viên cần đủ lớn để nắm bắt các kết quả tiềm năng, giúp mô hình hoạt động tốt với thời gian trễ mong muốn.

+Một giải pháp là sử dụng collaborative algorithms vì thời gian suy luận nhanh và có thể nắm bắt được sự tương đồng between user taste in the user-video space.

**\*Ranking model**

- Trong quá trình suy luận, the ranking model nhận một danh sách các video ứng viên do candidate generation model cung cấp.

- Đối với mỗi ứng viên, mô hình xếp hạng ước lượng xác suất video đó sẽ được xem.

- Sau đó, nó sắp xếp các video ứng viên dựa trên xác suất đó và trả lại danh sách cho quá trình phía trên(upstream process).

**-Traning data:**

+ Chúng ta có thể sử dụng dữ liệu lịch sử xem của người dùng. Thông thường, tỷ lệ giữa video đã xem và video không được xem là 2/98. Điều này có nghĩa là phần lớn thời gian, người dùng không xem video.

**-Features Engineering:**

+ Watched video Ids: áp dụng nhúng video(video embedding).

+Truy vấn tìm kiếm lịch sử(historical search query): áp dụng nhúng văn bản(text embedding). For example, word2vec or use pretrained word embedding, such as Global Vectors for Word Representation (GloVe) or Bidirectional Encoder Representations from Transformer (BERT).

+Training: optimize for high throughput with the ability to retrain many times per day

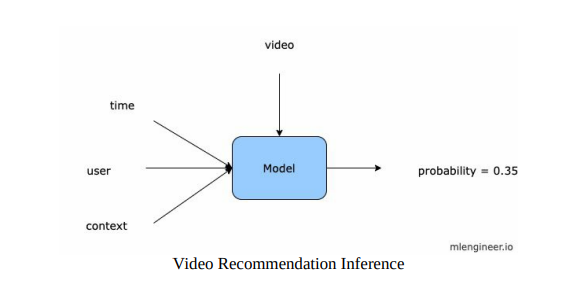
+Location: sử dụng nhúng địa chỉ(geolocation embedding.)

+Các đặc trưng liên quan đến người dùng(user associated features): age, gender with & Normalization or Standardization

+Tổng hợp ấn tượng(aggregated impression) with Normalization or Standardization

+Time-related features: Month, week\_of\_year, holiday, day\_of\_week,hour\_of\_day. **-Model:**

+ Chúng ta có thể sử dụng DCNv2 với hàm kích hoạt relurelu ở các lớp ẩn và hàm kích hoạt sigmoid(probability 0->1) ở lớp cuối cùng. Hàm mất mát có thể là cross-entropy loss.



**Calculation and Estimation**

**Assumptions:**

- Để đơn giản, chúng ta có thể đưa ra các giả định sau:

+ Số lượt xem video mỗi tháng là 150 tỷ.

+ 10% số video được xem đến từ các gợi ý, tổng cộng là 15 tỷ video.

+ Trên trang chính, người dùng thấy 100 video gợi ý.

+ Trung bình, một người dùng xem 2 video trong số 100 video gợi ý.

+ Nếu người dùng không nhấp vào hoặc không xem video nào trong một khoảng thời gian nhất định (ví dụ, 10 phút), thì đó được coi là một gợi ý bị bỏ lỡ.

+ Tổng số người dùng là 1,3 tỷ.

+ Trong một tháng, chúng ta đã thu thập được 15 tỷ nhãn tích cực và 750 tỷ nhãn tiêu cực.

+ Nói chung, chúng ta có thể giả định rằng mỗi điểm dữ liệu mà chúng ta thu thập sẽ có hàng trăm đặc điểm. Để đơn giản, mỗi dòng dữ liệu chiếm 500 byte để lưu trữ. Trong một tháng, chúng ta cần 800 tỷ dòng dữ liệu.

+Tổng kích thước dữ liệu:

500×800×10^9=4×10^15= 4 Petabytes.

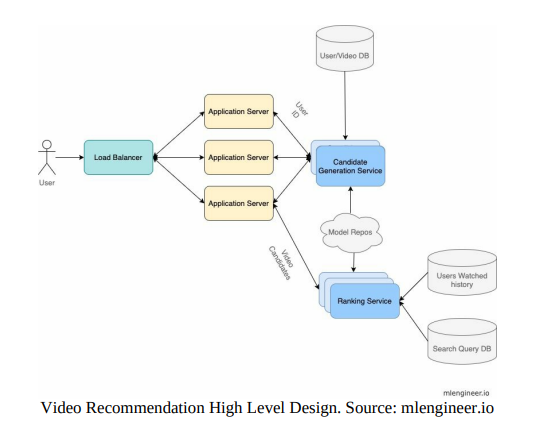
Để tiết kiệm chi phí, chúng ta có thể giữ lại dữ liệu trong 6 tháng hoặc một năm trong kho dữ liệu và lưu trữ dữ liệu cũ trong cold storage.

**Bandwidth(Băng thông) and Scale**

- Giả sử mỗi giây chúng ta phải tạo một yêu cầu gợi ý cho 10 triệu người dùng. Mỗi yêu cầu sẽ tạo ra thứ hạng cho từ 1.000 đến 10.000 video => Hỗ trợ cho 1,3 tỷ người dùng.

**System Design**

**Training**

****

**-** Database:User Watched History lưu trữ các video mà người dùng cụ thể đã xem theo thời gian. Search Query DB lưu trữ các truy vấn lịch sử mà người dùng đã tìm kiếm trong quá khứ.User/Video DB lưu trữ danh sách người dùng và hồ sơ của họ cùng với siêu dữ liệu video.User Historical Recommendations lưu trữ các đề xuất trước đó cho một người dùng cụ thể.

-Lấy mẫu lại dữ liệu(resampling data): mở rộng quy trình huấn luyện bằng cách giảm mẫu các trường hợp tiêu cực (downsampling negative samples).

-Features pipeline: một chương trình dòng để tạo ra tất cả các đặc trưng cần thiết cho mô hình huấn luyện. Điều quan trọng là dòng dữ liệu đặc trưng phải cung cấp thông lượng rất cao vì chúng ta cần tái huấn luyện mô hình nhiều lần trong một vài ngày. Chúng ta có thể sử dụng Spark (một công cụ tính toán trong bộ nhớ), Elastic MapReduce hoặc Google Dataproc (Spark và Hadoop trên đám mây).

- Kho lưu trữ mô hình (Model Repos): nơi lưu trữ tất cả các mô hình, sử dụng AWS S3 (AWS Simple Cloud Storage) là một lựa chọn phổ biến..

**Challenges**

- Dữ liệu khổng lồ. Giải pháp: Chọn dữ liệu trong một tháng hoặc sáu tháng gần nhất. Huấn luyện và tinh chỉnh mô hình học sâu (DL) hiện có (phần [retraining]).

- Dữ liệu không cân bằng. Giải pháp: Thực hiện giảm mẫu ngẫu nhiên các trường hợp tiêu cực (random negative downsampling).

- Tính khả dụng cao(high availability). Giải pháp: Cung cấp mô hình dưới dạng dịch vụ (Model-as-a-service), mỗi mô hình sẽ chạy trong các container Docker. Giải pháp bổ sung: Sử dụng Kubernetes để tự động mở rộng số lượng pods.

**Inference**

Khi người dùng yêu cầu đề xuất video:

+ Máy chủ ứng dụng(Application Server) sẽ yêu cầu các video ứng viên từ mô hình tạo ứng viên.

+ Khi nhận được các ứng viên, Application Server sẽ chuyển danh sách ứng viên đến ranking model để nhận thứ tự sắp xếp.

+ Ranking model ước tính xác suất xem và trả về danh sách đã được sắp xếp cho Application Server.

+ Application Server sau đó có thể trả về các video hàng đầu mà người dùng nên xem.

**Scale the Design**

- Mở rộng nhiều máy chủ ứng dụng (Application Server) và sử dụng bộ cân bằng tải (Load Balancer) để phân phối tải.

- Mở rộng nhiều dịch vụ tạo ứng viên (candidate generation) và dịch vụ xếp hạng (ranking services).

- We can also use ‘Kube-proxy’ so that the candidate generation service can call the ranking service directly and reduce latency even further. Read section Deployment [deployment].

**Summary**

- Trước tiên, chúng ta học cách tách biệt đề xuất thành hai dịch vụ: candidate generation service and ranking service.

- Chúng ta cũng đã học cách sử dụng mạng nơ-ron nhiều lớp (MLP) làm mô hình cơ bản và xử lý kỹ thuật đặc trưng (feature engineering).

- Để mở rộng hệ thống và giảm độ trễ, chúng ta học cách sử dụng kube-flow để dịch vụ tạo ứng viên có thể giao tiếp trực tiếp với dịch vụ xếp hạng.