## **1️⃣ Feature Score - Các chỉ số đánh giá Job cho User**

Mỗi user và job sẽ được đánh giá dựa trên các yếu tố sau:

### ****1. User Position Score (Dùng BERT)****

* Đánh giá mức độ phù hợp giữa vị trí công việc và kinh nghiệm của user.
* Công thức:

**user\_position=α×max(experience.position)+(1−α)×max(position.interest)**

### ****2. Description Match Score (Dùng BERT)****

* Đánh giá mức độ khớp nội dung giữa mô tả công việc và kinh nghiệm của user.
* Công thức:

**description\_match\_score=max(cos(jobexperience.descriptioni​, jobposting.text))**

### ****3. View History Score****

* Đánh giá mức độ quan tâm của user đối với job dựa trên lịch sử xem.
* Công thức: view\_history\_score =

### ****4. Location Score (Dùng BERT)****

* Đánh giá mức độ khớp giữa địa điểm làm việc và nơi ở của user.
* Công thức:

**location\_score=max(cos(experience.city,jobposting.city))**

### ****5. Job Popularity Score****

* Đánh giá độ phổ biến của job dựa trên số lượt xem và số ứng tuyển.

### Công thức:​

### ****6. Final Score****

* Tổng hợp các chỉ số trên để đánh giá mức độ phù hợp cuối cùng.
* Công thức: Final\_Score ×Status
  + **Status = 1 nếu job đang mở tuyển, = 0 nếu không.**

## **2** Các bước xây dựng hệ thống gợi ý việc làm

### ****Bước 1: Xử lý Dữ liệu (Data Preprocessing)****

✔ **Xử lý dữ liệu thiếu (Missing Data)**

* Điền "Unknown" cho experience.position, jobposting.position, jobexperience.description, jobposting.text nếu bị thiếu.
* Điền "Unknown City" cho experience.city, jobposting.city nếu bị thiếu.

✔ **Chuẩn hóa dữ liệu (Standardization)**

* Chuyển tất cả dữ liệu về chữ thường (lowercase).
* Loại bỏ ký tự đặc biệt, khoảng trắng dư thừa.

### ****Bước 2: Tính các Chỉ số Đánh giá Job cho User****

* Áp dụng các **Feature Score** đã tổng hợp ở trên.
* Lưu kết quả vào một bảng hoặc ma trận lưu điểm số giữa User và Job.

### ****Bước 3: Xây dựng Ma trận User-Job****

📌 **Mô hình hóa dữ liệu**

* **Tạo ma trận User-Job**, trong đó:
  + **Hàng = User**
  + **Cột = Job**
  + **Giá trị = Final Score**

### ****Bước 4: Xây dựng Ma trận Job-Job****

📌 **Mô hình hóa dữ liệu**

* **Tạo ma trận Job-Job**, trong đó:
  + **Hàng = Job**
  + **Cột = Job**
  + **Giá trị = Độ tương đồng Cosine giữa 2 job**

## Bước 5: ARM Filtering

* **Nếu {jobA ⬄ jobB} và {jobB⬄jobC}** có độ tương đồng cosin >= ngưỡng nhất định thì gom lại thành {job A, job B , jobC}

📌 **Gợi ý dựa trên Itemset**

* Nếu user có điểm phù hợp được tính ban đầu là một phần của **itemset**, hệ thống sẽ gợi ý toàn bộ job còn lại trong itemset.
  + Ví dụ: { Job A, Job B, Job C,job D } → nếu user có điểm cosin hợp lý với A và B, thì có thể gợi ý C, D.
* **Bước 1: Tính các Chỉ số Đánh giá Job cho User**

FUNCTION ComputeEvaluationScore(user, job, viewHistory):

userPositionScore = ComputeUserPositionScore(user, job)

descriptionMatchScore = ComputeDescriptionMatchScore(user, job)

viewHistoryScore = ComputeViewHistoryScore(user, job, viewHistory)

locationScore = ComputeLocationScore(user, job)

jobPopularityScore = ComputeJobPopularityScore(job)

finalScore = AggregateScore(userPositionScore, descriptionMatchScore, viewHistoryScore, locationScore, jobPopularityScore)

IF job.status == "open" THEN:

RETURN finalScore

ELSE:

RETURN 0

FUNCTION ComputeUserPositionScore(user, job):

experiencePositionMax = MAX(CosSimilarity(user.experience.position, job.jobposting.position))

positionInterestMax = MAX(CosSimilarity(user.interest, job.jobposting.position))

RETURN alpha \* experiencePositionMax + (1 - alpha) \* positionInterestMax

FUNCTION ComputeDescriptionMatchScore(user, job):

RETURN MAX(CosSimilarity(user.experience.description, job.jobposting.text))

FUNCTION ComputeViewHistoryScore(user, job, viewHistory):

# Compute based on user's view history

# ...

RETURN viewHistoryScore

FUNCTION ComputeLocationScore(user, job):

RETURN CosSimilarity(user.experience.city, job.jobposting.city)

FUNCTION ComputeJobPopularityScore(job):

# Compute based on job views and applications

# ...

RETURN jobPopularityScore

FUNCTION AggregateScore(userPositionScore, descriptionMatchScore, viewHistoryScore, locationScore, jobPopularityScore):

# Compute final score based on different factors

# ...

RETURN finalScore

FUNCTION CosSimilarity(text1, text2):

# Use BERT to compute cosine similarity

# ...

RETURN cosineSimilarity

**Bước 2: Xây dựng Ma trận User-Job**

FUNCTION BuildUserJobMatrix(userList, jobList, viewHistory):

matrix = InitializeMatrix(length(userList), length(jobList))

FOR i FROM 0 TO length(userList) - 1:

FOR j FROM 0 TO length(jobList) - 1:

matrix[i][j] = ComputeEvaluationScore(userList[i], jobList[j], viewHistory)

RETURN matrix

**Bước 3: Xây dựng Ma trận Job-Job**

FUNCTION LocARM(jobJobMatrix, similarityThreshold)

itemsets = FindItemsets(jobJobMatrix, similarityThreshold)

RETURN itemsets

FUNCTION FindItemsets(jobJobMatrix, similarityThreshold)

# Find itemsets based on cosine similarity and threshold

# ...

RETURN itemsets

**Bước 4: ARM Filtering**

FUNCTION LocARM(jobJobMatrix, similarityThreshold)

itemsets = FindItemsets(jobJobMatrix, similarityThreshold)

RETURN itemsets

FUNCTION FindItemsets(jobJobMatrix, similarityThreshold)

# Find itemsets based on cosine similarity and threshold

# ...

RETURN itemsets

**Bước 5: Gợi ý dựa trên Itemset**

Function RecommendJobs(user, userJobMatrix, itemsets):

userScores = userJobMatrix[user]

recommendedJobs = []

FOR each itemset IN itemsets:

IF AnyJobInItemsetMatchesUser(userScores, itemset):

recommendedJobs.extend(RemainingJobsInItemset(itemset))

RETURN recommendedJobs

Function AnyJobInItemsetMatchesUser(userScores, itemset):

# Check if any job in the itemset matches the user

# ...

RETURN true OR false

Function RemainingJobsInItemset(itemset):

# Return the remaining jobs in the itemset

# ...

RETURN jobList