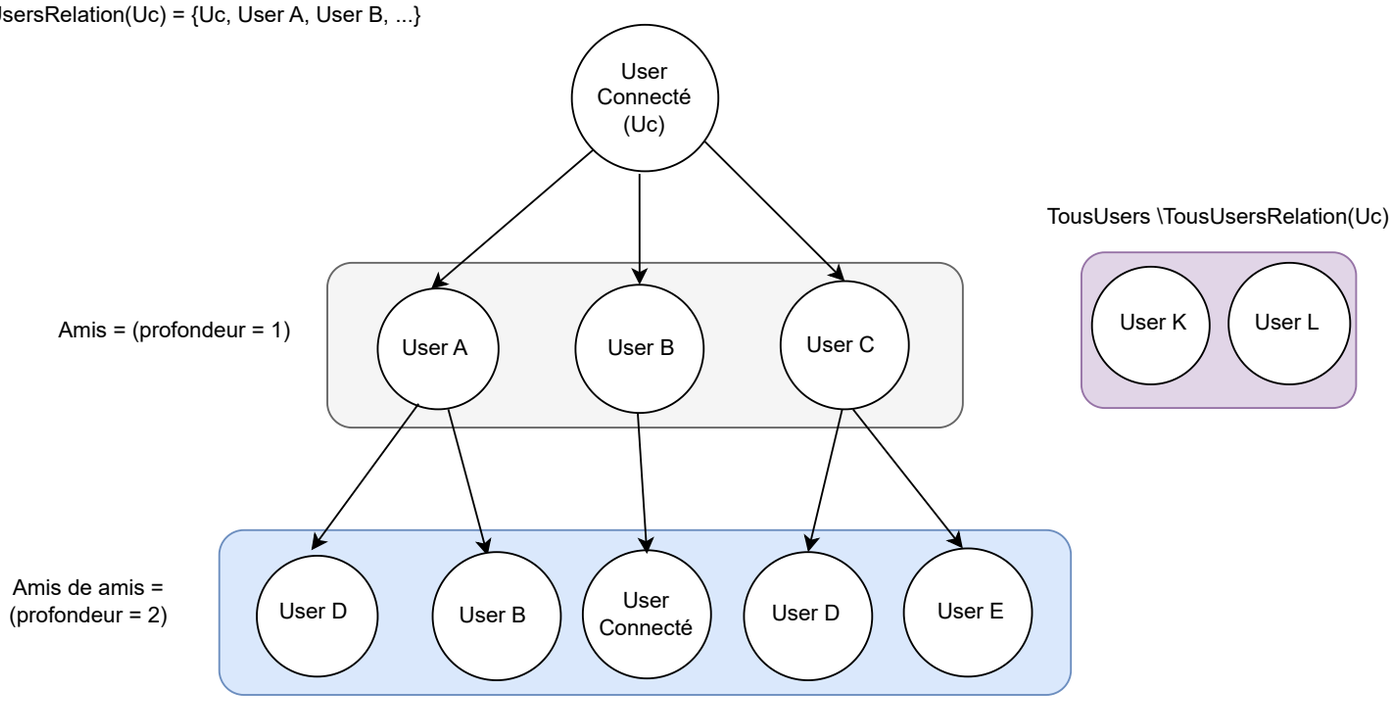
Recommendation des utilisateurs

TousUsers = {Uc, User A, User B, User K, User L ...}

TousUsersRelation(Uc) = {Uc, User A, User B, ...}



Recommendation en fonction des relations

M = Max(enfant pour un user)
h = hauteur du graph (=2)

Complexité : $O(M^h) = O(M^2)$

{ D : 2, E : 1}
--> Trier la map par valeur décroissante

Recommendation en fonction des tags

$$\{x \in TousUser \setminus Amis, UserDéjaRecommandé \mid J(Uc, x)\}$$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Exemple :

Tags(Uc) = {Technologie, Anime, Cinéma}
Tags(User K) = {Technologie, Arts}

$$\longrightarrow \quad J(Uc, User\ K) = \frac{1}{5-1} = \frac{1}{4}$$

Complexité : $O(|AllUsers|)$

Recommendation des posts

4 critères d'évaluation :

1. Profondeur de l'auteur du post dans le graph centré en Uc (40 %)
2. Indice de Jaccard de l'auteur du post (relatif au tag) (20 %)
3. Score en fonction de l'ancienneté de publication du post (20 %)
4. Score de similitude entre l'auteur du post et Uc (20 %)

Prérequis

Profondeur de l'auteur du post dans le graph centré en Uc (1)

Prérequis : On crée une Map qui associe pour chaque utilisateur la profondeur minimale dans le graph centré en Uc. Pour les utilisateurs ne figurant pas dans le graph, on fixe une profondeur arbitraire (-1 par ex) et pour Uc on fixe 1.

Ceci est effectué grâce à un BFS (récursif) en gardant en variable les nœuds visités pour avoir la profondeur minimale uniquement.

{ "Uc" : 1, "user A " : 1, "user B" : 1, "user C" : 1, "user D" : 2, "user E" : 1, "user , K" : -1, "user L": -1 }

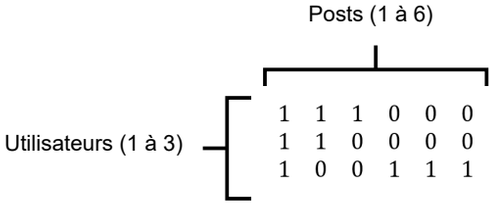
Complexité : $O(M^h)$

Score de similitude entre l'auteur du post et Uc (4)

Prérequis : On crée une matrice A de taille du nombre de tous les utilisateurs le nombre de posts totaux.

$$A_{i,j} = 1 \; si \; l'utilisateur \; i \; a \; liké \; le \; post \; j, = 0 \; sinon$$

Par exemple :



Pour remplir cette matrice, on l'initialise à la matrice nulle puis on parcourt la liste des utilisateurs et on effectue une requête SQL pour récupérer les posts likés pour cet utilisateur et on remplit cette matrice

On récupère la ligne correspondante à l'utilisateur connecté dans la matrice de similitude A. (vecteur u)

Complexité : $O(|AllUsers| \cdot |AllPosts|)$

Calcul des scores pour les 4 critères

On parcourt la liste de tous les posts dans le but d'associer à chaque post un score total puis :

Complexité : $O(|AllPosts|)$

Pour le critère (1) :

--> On détermine l'auteur du post et on récupère sa profondeur sur la Map construite dans le prérequis .

Le score pour ce critère est alors de $\frac{1}{profondeur}$.

NB : Si l'utilisateur a une profondeur de -1 (valeur qu'on a fixé arbitrairement) alors on renvoie 0.

Complexité : $O(1)$

Pour le critère (2) :

--> On détermine l'auteur du post et on appelle la même fonction qui calcule l'indice de Jaccard (entre l'auteur du post et Uc) : fonction déja expliqué dans la recommendation des utilisateurs

Complexité : $O(1)$

Pour le critère (3) :

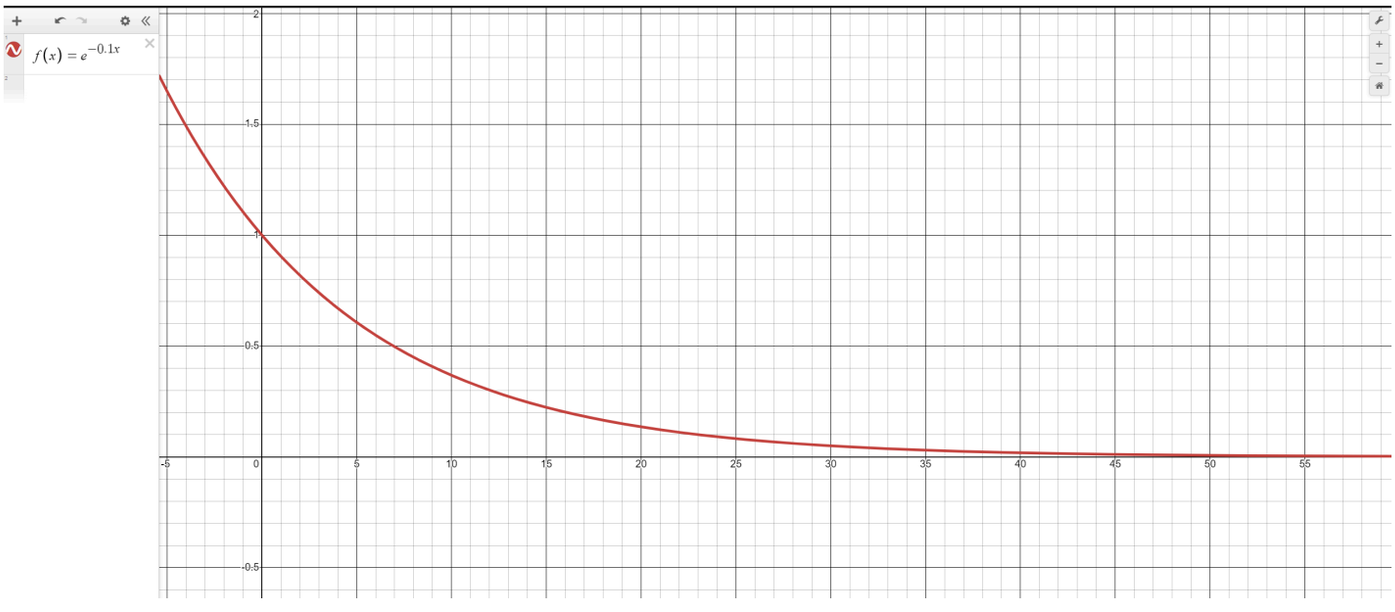
--> On détermine la date de publication du post et on calcule le nombre de jours d'ancienneté du post, qu'on appellera t.

On souhaite avoir un score compris entre 0 et 1. On veut que le score soit à 1 si le post vient d'être publié et qui tendent vers 0 plus le nombre de jour d'ancienneté est grand.

On décide de modéliser avec la fonction suivante qui respecte ce critère (il existe une infinité de telle fonction ...) :

$f(t) = e^{-0.1}t$

Complexité : $O(1)$



Pour le critère (4) :

Puis lors de la boucle sur l'ensemble des posts, on récupère l'auteur du post et la ligne correspondante à cet utilisateur dans la matrice de simulitude A. (vecteur v)

On calcule alors la similarité cosinus :

$$\cos(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|} = \frac{\sum_{i=1}^{n} u_i \cdot v_i}{\sqrt{\sum_{i=1}^{n} {u_i}^2} \cdot \sqrt{\sum_{i=1}^{n} {v_i}^2}}$$

Si le résultat est proche de 1, cela signifie que les 2 vecteurs sont "similaires", c'est à dire que l'utilisateur associé au vecteur v à une tendance similaire à liker les mêmes postes que Uc.

Si le résultat est proche de 0, cela signifie que les 2 vecteurs ne sont pas similaires, l'utilisateur associé au vecteur v et Uc ne likent pas les mêmes choses.

NB : Cette valeur ne peut pas être négative car les vecteurs ne contiennent pas de valeur négative.

-> La valeur ainsi obtenue est associée aux posts de l'utilisateur associés au vecteur v.

Complexité : $O(|AllPosts|)$

Score total

Score total = 0.4 * Score profondeur + 0.2 * Score indice de Jaccard + 0.2 * Score ancienneté + 0.2 * Score similitude

Amélioration possible : Après qu'on ait beaucoup d'utilisateurs sur l'application, utiliser un modèle de machine learning pour déterminer les pondérations des différents critères (coefficients). En effet, on peut utiliser un modèle de machine learning qui maximise le taux de conversion aux likes.

Complexité totale: $O(|AllPosts|)^2$