Une méthode KNN sans paramètre pour prédire les notes des utilisateurs

Junior Medjeu Fopa,*, Modou Gueye,*, Samba Ndiaye,*, Hubert Naacke**

*LID, Université Cheikh Anta Diop
BP. 16432, Dakar-Fann, Sénégal
http://www.ucad.sn

**LIP6 - Sorbonne Universités
4 place Jussieu 75005, Paris, France
http://www.lip6.fr

Résumé. Parmi les algorithmes de filtrage collaboratif les plus populaires figurent les méthodes basées sur les avis des K voisins les plus proches. Dans leur fonctionnement de base, ces méthodes considèrent un nombre fixe de voisins pour élaborer des recommandations. Cependant, il est difficile de choisir un nombre approprié de voisins. Une valeur non adéquate affecterait négativement la qualité des recommandations. Ainsi, il est généralement fixé à une valeur calibrée au préalable.

Dans la littérature, certains auteurs ont abordé le problème de rechercher dynamiquement un nombre approprié de voisins. Mais ils utilisent des paramètres supplémentaires nécessitant d'être également calibrés. Ce qui limite leurs propositions.

Nous proposons une méthode KNN sans paramètre pour la prédiction de note. Elle est capable de sélectionner dynamiquement un nombre approprié de voisins à utiliser. Les expériences que nous avons menées sur trois jeux de données accessibles publiquement démontrent l'efficacité de notre proposition.

1 Introduction

L'objectif des systèmes de recommandation (SR) est de déterminer parmi une grande quantité de contenu (films, livres, musique, etc.) [1] lesquels intéresseront le plus un utilisateur donné. Ils ont une valeur commerciale capitale pour tout type de commerce électronique (Schafer et al., 1999; Fleder et Hosanagar, 2007; Jannach et Hegelich, 2009; Linden et al., 2003).

Le filtrage collaboratif est une catégorie de systèmes de recommandation largement utilisée. Il repose sur l'analyse des relations existantes entre les utilisateurs et les produits pour identifier des centres d'intérêt concernant les utilisateurs (Su et Khoshgoftaar, 2009; Koren

1. De façon générale, nous parlerons de produits.

et al., 2009; Paterek, 2007). Sur la base de ces identifications, des recommandations sont déduites. Parmi les algorithmes de filtrage collaboratif les plus populaires, on peut en citer ceux basés sur la méthode des K plus proches voisins (KNN [2]). Ces derniers automatisent le principe commun du bouche-à-oreille, où l'on s'appuie sur les avis de proches ou d'autres personnes partageant les mêmes centres d'intérêt pour évaluer la valeur d'un produit. Ces algorithmes sont reconnus comme étant simples d'utilisation avec des recommandations justifiables, en plus d'être efficaces et stables (Ricci et al., 2011; Su et Khoshgoftaar, 2009). Dans la littérature, la méthode KNN est bien étudiée et de nombreuses approches ont été proposées pour prédire l'avis qu'un utilisateur pourrait avoir sur un produit (Koren et al., 2009; Koenigstein et al., 2011; Ma et al., 2011). L'idée derrière ces méthodes est que l'avis d'un utilisateur u sur un produit i est probablement proche de celui d'un autre utilisateur v, si les deux utilisateurs u et v ont des avis similaires sur d'autres produits.

Dans leur fonctionnement de base, le nombre K de voisins à considérer est le seul paramètre utilisé. Cependant trouver une valeur optimale du paramètre K reste un défi. Une valeur de K non appropriée peut affecter négativement la qualité des recommandations. Par conséquent, le paramètre K est généralement fixé à une valeur calibrée à l'avance. Certains auteurs ont déjà abordé le problème de la recherche dynamique d'une valeur de K appropriée afin d'optimiser la qualité des recommandations (Ougiaroglou et al., 2007; Zeybek et Kaleli, 2018; Kaleli, 2014). Dans leurs approches, ils introduisent des paramètres additionnelles permettant de déterminer une valeur de K appropriée mais nécessitant d'être aussi calibrées.

Nous proposons ici une méthode KNN ne nécessitant aucun paramètre que nous avons dénommée *freeKNN*. Elle est capable de sélectionner dynamiquement un nombre approprié de voisins à prendre en compte en fonction de l'utilisateur et du produit à noter. Les expériences que nous avons menées sur trois jeux de données publics démontrent son efficacité.

La suite de ce papier est organisée comme suit. Dans la section 2, nous présentons des travaux connexes relatifs à la recherche dynamique d'une valeur de K appropriée afin d'optimiser la qualité des recommandations. Dans la section 3, nous détaillons la méthode *freeKNN*qui n'utilise aucun paramètre pour faire des prédictions. Nous exposons son fonctionnement et expliquons comment déterminer dynamiquement un nombre approprié de voisins à considérer. La section 4, présente les résultats de notre évaluation qui démontrent l'efficacité de *freeKNN*. Enfin, dans la section 5, nous concluons ce papier.

2 Travaux connexes

A la base, les systèmes de recommandation (SR) cherchent à suggérer aux utilisateurs des produits à haut intérêt, que ce soit en termes de prédiction de la note qu'un utilisateur donnerait à un produit ou d'estimation de la probabilité d'interaction entre l'utilisateur et le produit (achat, lecture, etc.).

Dans ce contexte, la tâche de recommandation peut se résumer à trouver les N produits pour lesquels un utilisateur u donnerait les notes les plus élevées et à les lui recommander. De façon formelle, la tâche peut être définie comme suit :

$$Top(u, N) = \underset{i \in I}{argmax}^{N} \hat{r}_{ui} \tag{1}$$

2. *K-Nearest neighbors*, en anglais

où I est l'ensemble des produits pouvant être recommandés (par exemple, le catalogue d'une boutique électronique), N un nombre donné de produits à recommander et $\hat{r}_{ui}$ la note prédite par le SR que l'utilisateur u donnerait au produit i.

Comme introduit précédemment, l'idée-clé des recommandations basées sur la méthode KNN est que l'avis d'un utilisateur u sur un produit i serait probablement proche de ceux de ses voisins qui ont eu à noter le produit i. Par la notion de voisinage, nous sous-entendons les autres utilisateurs qui lui sont similaires en termes de notation des produits. Plus formellement, considérons les définitions suivantes.

Definition 1 *$\mathcal{N}_{ui}$ est la liste des voisins de u ayant noté le produit i, classés par ordre décroissant de leurs similarités avec u.*

Definition 2 *$\mathcal{N}_{ui}^K$ est la sous-liste de $\mathcal{N}_{ui}$ constituée des K premiers voisins.*

Definition 3 *$\mathcal{R}$ est l'ensemble des notes*[3] *possibles que les utilisateurs peuvent utiliser pour donner leurs avis sur les produits*[4].

La prédiction de la note $\hat{r}_{ui}$ que l'utilisateur u donnerait au produit i est déduite à partir de l'agrégation des notes de ses K plus proches voisins ayant noté le produit i selon la formule suivante :

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}_{ui}^K} s_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v \in \mathcal{N}_{ui}^K} |s_{uv}|} \tag{2}$$

où $r_{vi} \in \mathcal{R}$ est la note que le voisin v a donné au produit i et $\bar{r}_u$ (respectivement, $\bar{r}_v$) la moyenne des notes de l'utilisateur u (respectivement, du voisin v).

Dans l'équation 2, s_{uv} représente la similarité entre les utilisateurs u et v. Elle est généralement calculée avec le coefficient de corrélation de Pearson (Herlocker et al., 2004). Ce dernier prend en compte dans quelle mesure les notes des deux utilisateurs u et v tendent à varier de façon similaire ou non. Le calcul est effectué comme suit :

$$s_{uv} = \frac{\sum\limits_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2 \sum\limits_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}} \tag{3}$$

avec I_{uv} qui représente l'ensemble des produits communément notés par les deux utilisateurs.

La qualité des prédictions issues de la méthode KNN dépend principalement de la métrique de similarité utilisée et du choix d'un nombre de voisins K optimal à considérer. Le calibrage du paramètre K sur des échantillons est souvent utilisé mais il demeure une approche difficile car le nombre optimal de voisins à considérer évolue en fonction de la distribution des données. Avec un nombre K trop petit, des voisins avec des avis importants peuvent être ignorés alors qu'avec un nombre K trop grand, les avis de voisins moins similaires pourraient causer du bruit dans la prédiction finale. Pour surmonter ce problème, plusieurs méthodes KNN à

3. Autrement dit, les différentes classes de note
4. Il s'agit généralement de valeurs entières. L'échelle d'une à cinq étoiles étant très populaire sur le Web.

nombre de voisins dynamiquement choisi ont été proposées dans la littérature. Nous les appelons les méthodes dynamiques.

On note entre autres, la méthode incrémentale de Ougiaroglou et al. (2007) qui à partir d'un nombre maximal de voisins, K, à prendre tout au plus, cherche à trouver un sous-nombre de voisins plus optimal. Pour ce faire, elle débute à partir d'un nombre minimal de voisins, $MinNN$, et puis progressivement y ajoute d'autres voisins selon leur ordre de similarité jusqu'à ce qu'une condition d'arrêt déterminée par une heuristique soit atteinte ou qu'elle ait atteint le nombre maximal de voisins fixé au départ. Ils ont ainsi défini trois heuristiques d'arrêt : l'heuristique de rupture simple, celle de classe indépendante, celle de classe M-fois majeure. L'heuristique de rupture simple arrête son parcours dès qu'une des classes de note $r \in \mathcal{R}$ atteint un pourcentage de vote, $PMaj$, de la part des voisins en cours de visite dans $\mathcal{N}_{ui}$. L'heuristique de classe indépendante, quant à elle, définit plutôt un facteur de supériorité, $IndFactor$, des votes de la classe de note dominante par rapport au reste des classes. Enfin l'heuristique de classe M-fois majeure ajoute à celle de rupture simple une condition supplémentaire qui est l'existence de M voisins consécutifs dans $\mathcal{N}_{ui}$ ayant voté pour la classe majeure.

A côté, on peut citer la méthode de filtrage collaboratif avec un K dynamique de Zeybek et Kaleli (2018). Les auteurs y proposent deux phases de traitement : une première phase consistant à déterminer pour chaque utilisateur la valeur optimale de K sur un ensemble de valeurs fournis au préalable avec un jeu de données d'apprentissage, et une seconde phase de prédiction au cours de laquelle la valeur de K apprise lors de l'apprentissage est utilisée. Demirelli Okkalıoğlu (2020) propose une amélioration de cette méthode en modifiant notamment la métrique de similarité et aussi en choisissant les K plus proches voisins seulement parmi les voisins qui ont interagi avec le produit. Ce qui n'est pas le cas dans Zeybek et Kaleli (2018).

Kaleli (2014) propose une méthode de sélection de voisins combinant le degré d'incertitude de la répartition des notes des voisins comparée à celle de l'utilisateur et leurs similarités. Le degré d'incertitude détermine ici dans quelle mesure un utilisateur utilise une classe de note et peut être estimé en calculant l'entropie de son vecteur de préférences. Ainsi si deux utilisateurs ont des degrés d'incertitude proches, Kaleli en déduit qu'ils notent de la même manière. De plus, si leur similarité basée sur la corrélation de Pearson est également élevée, intrinsèquement, il en conclut qu'ils ont une forte relation de voisinage. En pratique, l'auteur assimile la combinaison des deux mesures au problème du sac à dos 0-1 car il cherche à minimiser les différences de degré d'incertitude d'un utilisateur avec ses voisins tout en maximisant leurs similitudes.

Comparées à notre approche, ces méthodes présentent de nombreuses limites telles que l'introduction de nouveaux paramètres dont le choix des valeurs a un impact considérable sur la détermination de la valeur de K. De plus, elles ont besoin d'une valeur de K de référence pour s'exécuter.

Dans la section qui suit, nous proposons une méthode dynamique n'ayant recours à aucun paramètre.

3 freeKNN

Nous avons appelé notre méthode *freeKNN*. Elle permet de prédire la note qu'un utilisateur donnerait à un produit sans fixer un nombre de voisins à considérer au préalable. Pour ce faire, nous introduisons la notion de *Top-voisins* que nous définissons comme la plus petite sous-liste de voisins de $\mathcal{N}_{ui}$ dont les opinions sont représentatives de tout l'ensemble des voisins.

3.1 Préliminaires

À partir de la définition 2, la détermination d'un nombre optimal de voisins pour la méthode KNN peut se résumer à la recherche de la meilleure sous-liste de voisins à prendre dans $\mathcal{N}_{ui}$. Plus précisément, nous devons considérer toutes les sous-listes de $\mathcal{N}_{ui}$ par ordre croissant (c-à-d., $\mathcal{N}_{ui}^1$, $\mathcal{N}_{ui}^2$, $\ldots \mathcal{N}_{ui}^{N-1}$ et $\mathcal{N}_{ui}^N$ avec $|\mathcal{N}_{ui}| = N$) en gardant les voisins triés selon leur similarité avec u, comme illustré ci-dessous.

$$\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{v_1}_{\mathcal{N}_{ui}^1} \; v_2 \; v_3 \; \ldots \; v_{N-1}}_{\mathcal{N}_{ui}^{N-1}} \; v_N}_{\mathcal{N}_{ui}^N}}_{\mathcal{N}_{ui}^2}}_{\mathcal{N}_{ui}^3}}_{\mathcal{N}_{ui}^{N-2}}}$$

Dans cette illustration, v_1 représente le voisin le plus similaire de u qui a noté le produit i. v_2 est le suivant et ainsi de suite jusqu'à v_N qui est le moins similaire. Nous définissons l'opinion globale d'une sous-liste de voisins $\mathcal{N}_{ui}^K$ comme suit :

Definition 4 *L'opinion globale $\mathcal{O}_{ui}^K$ du voisinage $\mathcal{N}_{ui}^K$ est le classement final des différentes notes $r \in \mathcal{R}$ que les voisins ont attribuées au produit i selon leurs votes tels que nous le définissons plus loin.*

Plus formellement, l'opinion globale est constituée de la liste de toutes les notes possibles, $\mathcal{O}_{ui}^K = [\, r \mid r \in \mathcal{R} \,]$, classées selon la condition suivante :

$$\forall (m,n), 0 \leq m < n < |\mathcal{O}_{ui}^K| \implies score(r_m|u,i,K) \geq score(r_n|u,i,K) \tag{4}$$

Une note r_m dans $\mathcal{O}_{ui}^K$ précède une autre r_n si le cumul des votes $score(r_m|u,i)$ de la première note est supérieur à celui de la seconde ou en cas d'égalité $r_m < r_n$.

En ce qui concerne le cumul des votes, $score(r|u,i,K)$, pour une note $r \in \mathcal{R}$, nous considérons le vote d'un voisin d'autant plus important qu'il est plus similaire à l'utilisateur que les autres voisins. De ce fait, nous prenons en compte la valeur de sa similarité avec l'utilisateur et son rang dans le voisinage $\mathcal{N}_{ui}^K$. Ainsi, nous calculons l'opinion du voisinage $\mathcal{N}_{ui}^K$ pour toute note $r \in \mathcal{R}$ selon la formule :

$$score(r|u,i,K) = \sum_{v \in \mathcal{N}_{ui}^K \wedge r_{vi} = r} \frac{s_{uv}}{\sqrt{0.5 \times (n_{rv} + rank_v)}} \tag{5}$$

Il s'agit de la somme des similarités des voisins de l'utilisateur u pondérées par leurs positions dans $\mathcal{N}_{ui}^K$. Il est facile de remarquer ici que la la contribution au vote des voisins successifs est une fonction décroissante. Notre intuition étant que le vote d'un voisin v pour une note r dépend de sa similarité avec u et aussi du nombre de voisins ayant déjà voté pour la note r et qui sont plus similaires à u que ne l'est v. Ainsi, dans l'équation 5, $rank_v$ représente la position globale du voisin v dans $\mathcal{N}_{ui}^K$ alors que n_{rv} est sa position locale parmi ceux ayant donné la note r au produit i. Pour le premier voisin v_1 dans $\mathcal{N}_{ui}^K$, son vote sera égal à s_{uv_1} alors que pour le reste, l'importance de leurs votes diminuera progressivement en fonction de leurs positions.

Definition 5 *De la définition 4, nous posons l'équivalence de deux sous-listes de voisins dans $\mathcal{N}_{ui}$ si et seulement si leurs opinions globales sont égales :*

$$\mathcal{N}_{ui}^K \equiv \mathcal{N}_{ui}^L \iff \mathcal{O}_{ui}^K = \mathcal{O}_{ui}^L \tag{6}$$

Definition 6 *De là, nous introduisons la notion de Top-voisins comme étant la plus petite sous-liste de voisins équivalente à $\mathcal{N}_{ui}$.*

Il s'agit du sous-groupe de voisins représentatif de tout le reste. Son opinion globale serait la même que celle de tous les voisins dans $\mathcal{N}_{ui}$. Sa taille K constitue un nombre de voisins approprié pouvant être considéré par la méthode KNN, avec

$$K = \min\left\{k \mid \mathcal{N}_{ui}^k \equiv \mathcal{N}_{ui}\right\} \tag{7}$$

Avec notre définition 6, nous établissons la condition d'équivalence de deux sous-listes de voisins. Il est facile à ce point de démontrer que la liste des *Top-voisins* est équivalente à toutes ses sur-listes puisque l'opinion globale des *Top-voisins* couvre le reste de tous les voisins.

$$\mathcal{N}_{ui}^K \equiv \mathcal{N}_{ui} \implies \mathcal{N}_{ui}^K \equiv \mathcal{N}_{ui}^k \quad \forall k, K < k \leq |\mathcal{N}_{ui}| \tag{8}$$

Ceci revient à dire que la considération de voisins supplémentaires n'influence en rien l'opinion globale des *Top-voisins*. De là, nous pouvons optimiser la recherche des *Top-voisins* en cherchant de manière progressive et croissante la sous-liste de voisins dont l'opinion globale ne pourrait plus évoluer dans la suite qu'importe l'intégration des avis des autres voisins. De ce fait, nous associons à chaque note $r \in \mathcal{R}$, un score maximal $maxScore(r|u,i,K)$ qu'elle ne pourrait dépasser même avec le cumul des avis du reste des voisins. Nous définissons le score maximal d'une note comme étant l'addition sur son score courant du cumul majoré du reste des voisins non encore pris et ayant donné cette note au produit. A ce niveau des statistiques sur le nombre de voisins restants, n_r, ayant noté le produit sont maintenus pour chaque note $r \in \mathcal{R}$.

La cumul majoré est décrit par l'équation 9. Le premier voisin, v_{K+1}, de ceux restants à être considéré est pris comme référence pour maximiser la contribution possible de leurs votes. Ce voisin est le plus influent parmi eux. Nous attribuons aux autres pouvant voter pour la note r la même influence.

$$maxScore(r|u,i,K) = score(r|u,i,K) + \left(n_r \times \frac{s_{uv_{K+1}}}{\sqrt{0.5 \times \left(n_{rv_{K+1}} + rank_{v_{K+1}}\right)}} \right) \tag{9}$$

Nous pouvons ainsi établir une condition d'arrêt de la recherche des *Top-voisins* à la sous-liste de voisins dont l'opinion globale n'évoluera plus puisqu'elle est représentative de celle de tous les voisins. L'équation 10 formalise notre condition d'arrêt via la stabilité du classement des notes des *Top-voisins*. Elle stipule que l'ordre des notes $r \in \mathcal{O}_{ui}^K$ ne changera plus qu'importe les votes du reste des voisins.

$$\forall (m,n) : 0 \leq m < n < |\mathcal{O}_{ui}^K|, \quad score(r_m|u,i,K) > maxScore(r_n|u,i,K) \tag{10}$$

3.2 Algorithme et optimisation

Nous présentons dans cette partie le fonctionnement de notre méthode de détermination d'un nombre optimal de voisins à considérer pour prédire la note qu'un utilisateur u donnerait à un produit i.

L'algorithme 1 détaille les étapes du processus de détermination d'un nombre optimal de voisins comme nous l'avons formalisé plus haut. Il prend en paramètres d'entrée un utilisateur u, un produit i dont on voudrait prédire la note que u attribuerait, ainsi que la liste $\mathcal{N}_{ui}$ de ses voisins ayant aussi noté le produit et la liste des notes possibles telles que définies dans $\mathcal{R}$. Il retourne un nombre optimal de voisins à considérer.

Algorithme 1 : Détermination du nombre optimal de voisins à considérer.

```
/* On a en entrée l'utilisateur u et le produit i, ainsi que la liste
   des voisins u ayant noté le produit i et la liste des notes possibles
   */
Entrées : u, i, N_ui, R
/* ... en sortie, un nombre optimal de voisins à considérer          */
Sorties : K
/* Le nombre de voisins à prendre est fixé par défaut au maximum     */
1 K ← |N_ui|;
/* Le voisinage de l'utilisateur est parcouru de façon incrémentale  */
2 pour k ← 1 to |N_ui| faire
      /* pour chaque sous-liste de voisins N_ui^k, on calcule leur opinion
         globale O_ui^k                                               */
3     Calculer O_ui^k ;
      /* puis on détermine si l'opinion est représentatif du reste des
         voisins                                                      */
4     si la condition de l'équation 10 est satisfaite par O_ui^k alors
          /* si c'est le cas, l'algorithme s'arrête et le nombre de voisins
             courant est enregistré                                   */
5         K ← k;
6         break ;
7     fin
8 fin
  /* K est retourné comme le nombre optimal de voisins à prendre      */
9 retourner K ;
```

A la ligne 1, le nombre optimal de voisins à prendre est fixé à priori sur le nombre total de voisins dans $\mathcal{N}_{ui}$. Puis le voisinage de l'utilisateur est parcouru par taille croissante et on cherche la sous-liste minimale de voisins pouvant représenter leur ensemble. Une itération est effectuée entre les lignes 2 et 8. A chaque itération, le voisin le plus similaire à l'utilisateur parmi ceux restants est ajouté à la liste de ceux déjà pris. Ce qui nous amène à des sous-listes $\mathcal{N}_{ui}^k$ croissantes. Pour chaque sous-ensemble courant, on calcule l'opinion globale $\mathcal{O}_{ui}^k$ des voisins considérés à la ligne 3. A la ligne 4, on vérifie si leur opinion globale recouvre celle du reste des voisins non encore considérés. Nous utilisons la condition d'arrêt de parcours définie par l'équation 10. Enfin si la condition est vérifiée, la taille de la sous-liste courante est prise comme le nombre optimal de voisins à prendre.

4 Expérimentation

Pour la validation de notre méthode, nous l'avons comparée à celles présentées dans notre état de l'art à la section 2. En plus de ces méthodes, nous avons aussi inclus les résultats de la méthode du KNN standard avec le paramètre K optimisé sur le meilleur nombre de voisins déterminé à partir d'une recherche exhaustive.

4.1 Jeux de données

Nous avons utilisé trois jeux de données aux caractéristiques différentes pour valider notre proposition. Ils sont populaires et librement téléchargeables. Tous les trois concernent des notes sur des films faites par des utilisateurs. Les deux premiers, Movilens 100K (M100K) et Movilens 1M (M1M), ont été collectés par le GroupLens Research Project de l'Université du Minnesota [5]. Le troisième, MovieTweetings (MT), est plus récent et est collecté depuis des tweets sur Twitter [6]. Le tableau 4.1 ci-dessous décrit chacun des jeux de données :

Nom	**#Utilisateurs**	**#Produits**	$\mathcal{R}$	**#Notes**	**Densité**
M100K	943	1 682	1-5	100 000	6,3%
M1M	6 040	3 952	1-5	1 000 000	4,18%
MT	71 041	37 543	1-10	911 241	0,034%

TAB. 1 – *Les jeux de données utilisés pour la validation.*

4.2 Méthodologie expérimentale

Nous avons utilisé la validation croisée pour évaluer notre méthode. Ainsi, nous avons divisé chacun des jeux de données en cinq échantillons et puis à tour de rôle l'un des échantillons a servi de jeu de validation pendant que les autres constituaient le jeu d'apprentissage. À l'issue de la procédure, la moyenne des erreurs sur les cinq jeux de validation est calculée et reportée dans nos résultats.

5. https://grouplens.org/datasets/movielens/
6. https://github.com/sidooms/MovieTweetings/tree/master/latest

Nous avons utilisé comme mesures de précision des prédictions : l'erreur absolue moyenne (MAE) et la racine de l'erreur quadratique moyenne (RMSE)[7]. Le MAE mesure la moyenne des valeurs absolues des écarts entre les notes prédites $\hat{r}_{ui}$ et celle réelle r_{ui}, alors que le RMSE calcule la racine carrée de la moyenne des écarts quadratiques.

Nous nous comparons aux méthodes de notre état de l'art dans leurs meilleurs configurations. Nous avons déterminé leurs paramètres optimaux permettant d'atteindre leurs meilleures prédictions en appliquant une recherche exhaustive. Nous listons ci-dessous les valeurs que nous avons trouvées pour chacune de ces méthodes et sur chaque jeu de données.

1. Pour le **KNN standard**, nous avons trouvé les valeurs 45 pour le paramètre K sur le jeu d'apprentissage de M100K et 175 sur M1M et MT.
2. Pour les heuristiques de la méthode de Ougiaroglou et al. (2007), les trois tableaux ci-dessous listent les valeurs optimales que nous avons obtenues pour leurs différents paramètres sur chacun des trois jeux de données.

Heuristique	$MinNN$	$PMaj$	$IndFactor$	K	M
Heuristique de rupture simple	9	0,8		45	
Heuristique de classe indépendante	8		9	45	
Heuristique de classe M-fois majeure	2	0,8		45	6

TAB. 2 – *Valeurs optimales avec le jeu de données M100K.*

Heuristique	$MinNN$	$PMaj$	$IndFactor$	K	M
Heuristique de rupture simple	2	1		175	
Heuristique de classe indépendante	9		8	175	
Heuristique de classe M-fois majeure	10	0,8		175	7

TAB. 3 – *Valeurs optimales avec le jeu de données M1M.*

Heuristique	$MinNN$	$PMaj$	$IndFactor$	K	M
Heuristique de rupture simple	8	0.9		175	
Heuristique de classe indépendante	9		10	175	
Heuristique de classe M-fois majeure	8	0,8		175	8

TAB. 4 – *Valeurs optimales avec le jeu de données MT.*

3. Pour la méthode de filtrage collaboratif avec un K dynamique de Zeybek et Kaleli (2018), nous avons utilisé la liste de valeurs $[40, 150]$ pour paramètre K avec le jeu de données M100K et $[160, 170, 175, 180, 185, 190, 200, 210, 220, 230, 240, 250]$ avec les deux autres. Pour avoir des listes pertinentes, nous avons à chaque fois choisi des valeurs consécutives autour de la valeur optimale de K de la méthode KNN standard.
4. Enfin, pour la méthode de sélection basée sur l'entropie de Kaleli (2014), nous avons repris les mêmes valeurs du paramètre K de la méthode *Standard KNN*. Et pour son paramètre β, on a la valeur 9 sur M100K, 10 sur M1M et MT.

7. *Mean Absolute Error* et *Root Mean Square Error*, en anglais

4.3 Résultats expérimentaux

Nous présentons dans cette partie les résultats expérimentaux des différentes méthodes présentées dans ce papier sur les trois jeux de données. Pour rappel, les résultats portent sur leurs qualités en termes de MAE et RMSE. Les tableaux 5 et 6 montrent respectivement les performances en termes d'erreurs MAE et RMSE des différentes méthodes.

Méthodes	**M100K**	**M1M**	**MT**
freeKNN	0,737	0,711	**1,293**
Heuristique de rupture simple de Ougiaroglou et al. (2007)	0,737	0,711	1,30
Heuristique de classe indépendante	0,737	0,711	1,30
Heuristique de classe M-fois majeure	0,737	0,711	1,30
Sélection par apprentissage de K de Zeybek et Kaleli (2018)	0,738	0,711	1,301
Sélection basée sur l'entropie de Kaleli (2014)	0,738	0,714	1,31
KNN standard	0,737	0,711	1,30

TAB. 5 – *Mesures de la qualité avec MAE.*

Méthodes	**M100K**	**M1M**	**MT**
freeKNN	**0,940**	0,905	**1,897**
Heuristique de rupture simple de Ougiaroglou et al. (2007)	0,941	0,904	1,902
Heuristique de classe indépendante	0,941	0,904	1,902
Heuristique de classe M-fois majeure	0,941	0,904	1,902
Sélection par apprentissage de K de Zeybek et Kaleli (2018)	0,941	0,904	1,907
Sélection basée sur l'entropie de Kaleli (2014) Kaleli (2014)	0,942	0,909	1,912
KNN standard	0,941	0,904	1,902

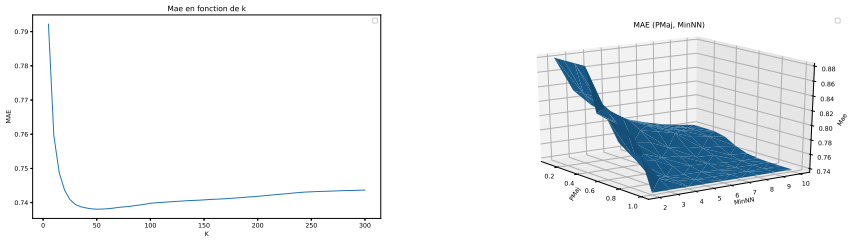
TAB. 6 – *Mesures de la qualité avec RMSE.*

On constate qu'en termes de MAE, notre solution *freeKNN*est au moins aussi performante que les autres méthodes et qu'elle est même plus performante que celles existantes en termes de RMSE. Exception faite du jeu de données M1M où *freeKNN*est est légèrement moins performante que certaines méthodes, *freeKNN*atteint leurs performances dans l'ensemble quoique les autres méthodes ont été optimisées via le calibrage de leurs paramètres à leurs meilleures valeurs.

Les figures 1(a) et 1(b) montrent l'apport du calibrage respectivement pour le KNN standard et la méthode de Ougiaroglou et al. (2007) basée sur l'heuristique de rupture simple. On peut constater que l'erreur moyenne de prédiction évolue différemment selon les valeurs des paramètres d'où la nécessité de bien les calibrer. Ce qui démontre l'efficacité de notre méthode qui n'a recours en aucun paramètre.

5 Conclusion

Dans le cadre de la prédiction de notes, nous nous sommes intéressés à la problématique d'amélioration des méthodes KNN en nous focalisant sur l'optimisation du nombre de voisins



(a) KNN standard (b) Heuristique de rupture simple

FIG. 1 – *Nécessité du calibrage des paramètres.*

à considérer. Ainsi, nous avons proposé une méthode de sélection dynamique d'un nombre approprié de voisins. Notre méthode ne nécessite aucun paramètre contrairement à celles de travaux connexes. Nos résultats expérimentaux montrent son efficacité et sa capacité à sélectionner un nombre approprié de voisins. L'absence de paramétrage rend notre méthode bien adaptée au contexte du web dynamique où les changements sur la distribution des données peuvent souvent nécessiter le calibrage de paramètres essentiels à certaines méthodes proposées jusqu'ici.

Références

Demirelli Okkalıoğlu, B. (2020). Improving prediction performance of dynamic neighbor selection in user-based collaborative filtering. *Sakarya University Journal of Computer and Information Sciences 3*, 74 – 88.

Fleder, D. M. et K. Hosanagar (2007). Recommender systems and their impact on sales diversity. In *Proceedings of the 8th ACM conference on Electronic commerce*, EC '07, New York, NY, USA, pp. 192–199. ACM.

Herlocker, J. L., J. A. Konstan, et J. Riedl (2004). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval 5*, 287–310.

Jannach, D. et K. Hegelich (2009). A case study on the effectiveness of recommendations in the mobile internet. In *RecSys*, pp. 205–208.

Kaleli, C. (2014). An entropy-based neighbor selection approach for collaborative filtering. *Knowledge-Based Systems 56*, 273–280.

Koenigstein, N., G. Dror, et Y. Koren (2011). Yahoo ! music recommendations : modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys '11, New York, NY, USA, pp. 165–172. ACM.

Koren, Y., R. Bell, et C. Volinsky (2009). Matrix factorization techniques for recommender systems. *Computer 42*, 30–37.

Linden, G., B. Smith, et J. York (2003). Amazon.com recommendations : item-to-item collaborative filtering. *IEEE Internet Computing 7*(1), 76–80.

Ma, H. et al. (2011). Recommender systems with social regularization. In *WSDM*, pp. 287–296.

Ougiaroglou, S., A. Nanopoulos, A. N. Papadopoulos, Y. Manolopoulos, et T. Welzer-Druzovec (2007). Adaptive k-nearest-neighbor classification using a dynamic number of nearest neighbors. In Y. Ioannidis, B. Novikov, et B. Rachev (Eds.), *Advances in Databases and Information Systems*, Berlin, Heidelberg, pp. 66–82. Springer Berlin Heidelberg.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 39–42.

Ricci, F. et al. (Eds.) (2011). *Recommender Systems Handbook*. Springer.

Schafer, J. B., J. Konstan, et J. Riedi (1999). Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, EC '99, New York, NY, USA, pp. 158–166. ACM.

Su, X. et T. M. Khoshgoftaar (2009). A survey of collaborative filtering techniques. *Adv. in Artif. Intell. 2009*, 4 :2–4 :2.

Zeybek, H. et C. Kaleli (2018). Dynamic k neighbor selection for collaborative filtering. *Anadolu University Journal of Science and Technology - Applied Sciences and Engineering 19*, 303 – 315.

Summary

Among the most popular collaborative filtering algorithms are methods based on nearest neighbors. In their basic operation, KNN methods consider a fixed number of neighbors to make recommendations. However, it is not easy to choose an appropriate number of neighbors. Thus, it is generally fixed by calibration to avoid inappropriate values which would negatively affect the accuracy of the recommendations.

In the literature, some authors have addressed the problem of dynamically finding an appropriate number of neighbors. But they use additional parameters which limit their proposals because these parameters also require calibration.

In this paper, we propose a parameter-free KNN method for rating prediction. It is able to dynamically select an appropriate number of neighbors to use. The experiments that we did on three publicly available datasets demonstrate the efficiency of our proposal. It rivals those of the state of the art in their best configurations.