SOMMAIRE

Introduction :

Après avoir décrit la conception de notre système, nous allons passer dans cette dernière partie à sa réalisation. Nous allons présenter tout d'abord, l'environnement et les outils que nous avons utilisé, ensuite nous verrons les étapes par lesquelles nous sommes passées dans la réalisation de notre système de recommandation. Ensuite, la partie expérimentation et tests de nos diverses propositions serons détaillés.

1 Réalisation

L'implémentation du système a comme objectif en un premier temps de déterminer et d'expliquer la modélisation, et à donner une vue sur les outils nécessaires à l'exécution du système dans un deuxième temps, tout en schématisant les différentes étapes qui nécessitent une explication.

1.1 Environnement de développement

1.1.1 NetBeans:

NetBeans est un environnement de développement intégré (EDI) [5], placé en *open source* par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans supporte une large variété de langages de programmation et d'outils de collaboration. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris, Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme.

L'IDE NetBeans s'enrichit à l'aide de plugins.

1.1.2 Pourquoi la bibliothèque Spmf ?

SPMF est une bibliothèque de data mining open-source écrit en Java, spécialisée dans le pattern mining. Elle est distribuée sous la licence v3 de la GPL.

Elle utilise les fichiers texte et implémente 93 algorithmes de data mining, pour :

- sequential pattern mining,
- association rule mining,
- itemset mining,
- sequential rule mining,
- clustering.

Le code source de chaque algorithme peut être intégré dans un autre logiciel Java. Aussi SPMF peut être utilisé comme un programme autonome avec une interface simple pour l'utilisateur ou à partir de la ligne de commande.

La version actuelle est v0.96r18 et elle a été libérée le 26 mai 2015.

1.2 Les étapes de réalisation :

1.2.1 Les classe et les méthodes utilisées

Pour réaliser notre système on a utilisé plusieurs classes présentées dans la **figure** 73.

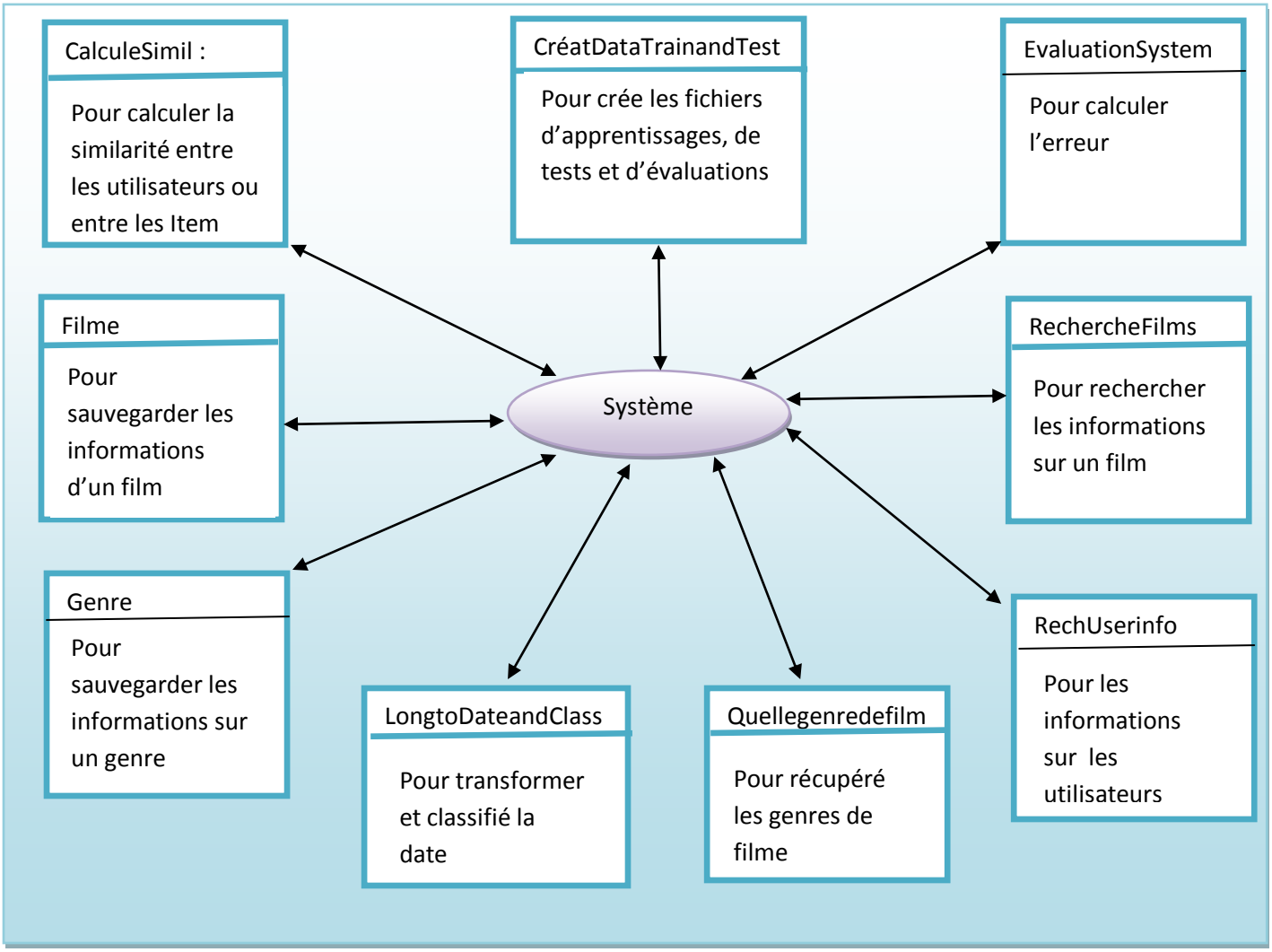

Figure 73 : Toutes Les classes utilisées dans notre système

a) La classe filme :

L'utilité de cette classe est de sauvegarder les informations des films tels que le nom, la date, l'id et la note.

Elle contient 2 méthodes :

- **Affich ()** : pour afficher les informations.
- **Affich2 ()** : pour retourner une chaine de caractère (String) contienne toutes les informations.

La **figure** 74 illustre le code source de cette classe.

```
public class Filme {
    public String Nom;
    public String Date;
    public String Id;
    public int Note;

    public void affich(){
         System.out.println("Id : "+Id+" Nom Du filme : "+Nom+"| Date : "+Date+"\n");
    }

    public String affich2(){
        String k=("Id: "+Id+" | Nom Du filme: "+Nom+" | Date: "+Date+" | Nbr De Vu: "+ Note +"\n");
        return k;
    }
}
```

Figure 74 : Code Source de la classe filme

b) La classe genre :

L'utilité de cette classe est de sauvegarder les informations sur les genres tels que le nom du genre et la note.

c) La classe long to date and class:

Cette classe converti le temps en milliseconde en une date et les classifiés on utilisant les deux méthodes :

- **ClassificationUtilisantUnInstant (**long Time**)**

La **figure** 75 présente le code source de cette méthode.

```
public void classificationUtilisantUnINstant(long time){
   // String dt= dateString[1]+"/"+dateString[2];
     this.time=time;
    instant=java.time.Instant.ofEpochSecond(time); // jdk 8 pour convertir ce temp since 1970 to instant
    date=Date.from(instant); // instant on le convertit on date pour le format bèh apré ntésté 3la les jrs
    DateFormat fullDateFormat = DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL); // type de date li fihe l
    dateString= fullDateFormat.format(date).toString().split(" "); // on  co,nverti notre date en String et on le metttre dans un ta

    int t=Integer.parseInt(dateString[1]); //
   System.out.println(TypeJr);

    if(dateString[0].equalsIgnoreCase("vendredi") || dateString[0].equalsIgnoreCase("samedi")) {TypeJr="Vacance";}
        if(15<=t && t<=30){
             if (dateString[2].equalsIgnoreCase("novembre") || dateString[2].equalsIgnoreCase("mars")) {TypeJr="Vac
                      }
    if(dateString[2].equalsIgnoreCase("décembre") || dateString[2].equalsIgnoreCase("janvier") || dateString[2].equ
    if(dateString[2].equalsIgnoreCase("mars") || dateString[2].equalsIgnoreCase("avril") || dateString[2].equalsIgn
    if(dateString[2].equalsIgnoreCase("juin") || dateString[2].equalsIgnoreCase("juillet") || dateString[2].equalsI
    if(dateString[2].equalsIgnoreCase("septembre") || dateString[2].equalsIgnoreCase("octobre") || dateString[2].eq
    Jour=dateString[0];
    TempHeur=dateString[4]+dateString[5];
    System.out.print("Jour : "+Jour+"\n Type du Jour : "+TypeJr+"\n Saison : "+Saison+"\n Temp : "+TempHeur+"\n");
```

Figure 75 : code Source de la méthode Classification utilisant un instant (long time)

- **ClassificationUtilisantUneDate (**String [] date**)**

La **figure** 76 présente le code source de cette méthode.

```
public void classificationUtilisantUneDate(String[] date){
    this.dateString=date;
    // String dt= dateString[1]+"/"+dateString[2];
    int t=Integer.parseInt(dateString[1]); //
   //System.out.println(TypeJr);

    if(dateString[0].equalsIgnoreCase("vendredi") || dateString[0].equalsIgnoreCase("samedi")) {TypeJr="Vacance";}
        if(15<=t && t<=30){
             if (dateString[2].equalsIgnoreCase("novembre") || dateString[2].equalsIgnoreCase("mars")) {TypeJr="Vac
                        }
    if(dateString[2].equalsIgnoreCase("décembre") || dateString[2].equalsIgnoreCase("janvier") || dateString[2].equ
    if(dateString[2].equalsIgnoreCase("mars") || dateString[2].equalsIgnoreCase("avril") || dateString[2].equalsIgn
    if(dateString[2].equalsIgnoreCase("juin") || dateString[2].equalsIgnoreCase("juillet") || dateString[2].equalsI
    if(dateString[2].equalsIgnoreCase("septembre") || dateString[2].equalsIgnoreCase("octobre") || dateString[2].eq
    Jour=dateString[0];
    TempHeur=dateString[4]+dateString[5];
    System.out.print("Jour : "+Jour+"\n Type du Jour : "+TypeJr+"\n Saison : "+Saison+"\n Temp : "+TempHeur+"\n");

}
```

Figure 76 : Code source utilisant une date (String [] date)

d) La classe QuelleGenreDeFilme :

Pour connaitre le genre de chaque film on utilise cette classe qui contient les méthodes suivantes :

- **ReturnUnSeulGenre(String idItem)** : a partir d'un iditem on obtient une chaine de caractère contient un seul genre de cet item.
- **Recherch(String idItem)** : elle retourne une chaine de caractère contient tout les genres d'un item données.
- **RecherchParGenre(String [] Genre)** : elle retourne une liste de type <filme> contient tout les films appartient a ce genre.
- **TrieListFilm(LinkedList<Filme> filme) :** cette méthode consiste a trié une liste des filmes par ordre décroisant.

e) La classe RecherchUserInfo :

Cette classe nous permet de connaitre les informations sur un utilisateur donné utilisant les deux méthodes suivantes :

- **RecherchUserParID (String Id) :** chercher un utilisateur donné par l'id.
- **IntervalAge (String Age) :** classifié un âge donné dans des intervalles d'âge, la **figure** 77 illustre le code source de cette méthode.

```
public String IntervalAge(String Age){
    int i=Integer.parseInt(Age);
    if(10<=i && i<18) return "Adolescent";
            else if(18<=i && i<=30) return "JeuneHomme";
            else if (30<i && i<=40) return "Homme";
            else if(40<i && i<=70) return "Adulte";
            else return"Vieux";
}
```

Figure 77 : Code source de la méthode IntervalAge

f) La classe RecherchFilms :

Cette classe a pour objectif de récupérer les informations des filmes utilisant les méthodes suivantes :

- **RechercheParId (String Id) :** chercher les informations à partir de l'id, et nous donne comme résultat une chaine de caractère.
- **RechercheParId (String Id) :** chercher les informations a partir de l'id, et nous donne comme résultat un objet de type Filme (voir section 1.1.1).
- **ListDesFilmesLesMieuxNotéDunUser (String IdUser) :** consiste à chercher les filmes mieux noté par utilisateur donné.
- **RetourNoteDeFilmeParId (String IdItem) :** nous donnes comme résultat la note d'un item donné.

g) La classe Evaluation:

Pour calculer les deux erreurs RMSE et MAE on utilise une méthode qui appartient a cette classe, le code source est présenter dans la **figure** 78.

```
    public void CalculeErreur (EvaluationSystéme Parent,String Fichier,int i,int j) throws FileNotFoundExceptio
        BufferedReader br = new BufferedReader(new FileReader(Fichier));
        String line;
        int cpt=0;
        double RSE = 0,MAE=0;
        while((line = br.readLine()) != null) {
            String[] values = line.split("\\ ");
             Parent.Affichage.add(values[i]+" | "+values[j]);
           if(cpt==0);
            else
            {
               try{
                  RSE=RSE+Math.pow (Math.abs(Double.parseDouble (values[i])-Double.parseDouble (values[j])),2);
                  MAE=MAE+(Math.abs(Double.parseDouble (values[i])-Double.parseDouble (values[j])));
               } catch (Exception e) {
                   if(values[j].equalsIgnoreCase("null")){
                    RSE=RSE+Math.pow (Double.parseDouble (values[i])-0,2);
                    MAE=MAE+(Double.parseDouble (values[i])-0);}
                }

            }
            cpt+=1;


        }
        br.close();
  Parent.RMSE.setText(String.valueOf(Math.sqrt(RSE/cpt)));
        Parent.MAE.setText(String.valueOf(MAE/cpt));
    }
}
```

Figure 78 : Code Source de la méthode calculeErreur appartient a la classe Evaluationsystem

h) La classe CreateDataTrainAndTest:

Cette classe a pour objectif de créé les fichiers d'apprentissage et de test qui nous aident à évaluer notre système, les méthodes utiliser dans cette classe sont :

- **CreatDataTrainBasic (**String FileIn,String FileOut**) :** cette méthode consiste a crée le fichier d'apprentissage ou de test pour les arbres de décision basic.
- **TrainArbreBasic(**String FileTrain,String FileTest, String FileEvaluationOut**):** Cette méthode sert a crée le fichier d'évaluation basic après la phase d'apprentissage et test de l'arbre de décision.
- **CreatDataTrainContext (**String FileIn,String FileOut**) .**
- **TrainArbreContext (**String FileTrain,String FileTest, String FileEvaluationOut**).**
- **CreatDataTrainContent (**String FileIn,String FileOut**) .**
- **TrainArbreContent (**String FileTrain,String FileTest, String FileEvaluationOut**).**
- **CreatDataTrainContentContext (**String FileIn,String FileOut**) .**
- **TrainArbreContentContext (**String FileTrain,String FileTest, String FileEvaluationOut**).**

- **CreationDataTrainForOneUser (**String File,String FileOut,String IdUser**) :** cette méthode consiste a crée le fichier d'apprentissage ou de test pour un utilisateur donné.
- **TrainArbreForOneUser (**String FileTrain,String FileTrainOut,String FileEvaluation**) :** cette méthode consiste a crée le fichiers d'évaluation pour un seul utilisateur après la phase d'apprentissage et de test qu'elle a fait sur l'arbre de décision.
- **TempHeurToTemp(**String tp**) :** classé le temps dans un intervalle, la **figure** 79 illustre le code source de cette méthode.

```
public String TempHeurToTemp(String tp){
   String []d=tp.split("h");
  if(5<Integer.parseInt(d[0]) && Integer.parseInt(d[0])<=12) return "Matin";
  else if(12<Integer.parseInt(d[0]) && Integer.parseInt(d[0])<=19) return "ApréMidi";
  else if(00<=Integer.parseInt(d[0]) && Integer.parseInt(d[0])<5) return "Nuit";
  else return "Soir";
}
```

Figure 79 : Code source de la méthode TempHeurToTemp

i) La classe CreateDataTrainAndTest:

Cette classe consiste à calculer la similarité entre les utilisateurs ou les item avec la méthode qu'on a proposé dans le (chapitre 4 section 1.2.2), Cette classe contient les méthodes suivantes :

- **DistanceDeJaccard (**DefaultTable User1,DefaultTable User2**)** : calculer la distance de Jaccard mais avec la méthode qu'on a proposé dans le (chapitre 4 section 1.2.2). la **figure** 80 représente cette méthode.

```
public double DistanceJaccard(DefaultTableModel f1,DefaultTableModel f2){
    double Res=0,cpt=0,Inter=0;
    int Intersection=0,Union=0;
    for(int i=0;i<f1.getRowCount();i++){
        for(int j=0;j<f2.getRowCount();j++){
           if(String.valueOf(f1.getValueAt(i,0)).equalsIgnoreCase(String.valueOf(f2.getValueAt(j,0).toString()))) {
                if(f1.getValueAt(i,1).toString().equalsIgnoreCase(f2.getValueAt(j,1).toString())) cpt=cpt+0.5;
                if(f1.getValueAt(i,2).toString().equalsIgnoreCase(f2.getValueAt(j,2).toString()))cpt=cpt+0.2;
                if(f1.getValueAt(i,3).toString().equalsIgnoreCase(f2.getValueAt(j,3).toString())) cpt=cpt+0.3;
                Intersection+=1;
                Inter+=cpt;
                cpt=0;
                //System.out.println(f1.getValueAt(i,1)+" | ");
            }
        }
    }
    Union=(f1.getRowCount()+ f2.getRowCount())-Intersection;
   // System.out.println("Jaccard Distance : "+ (Union-Inter)/Union);
    return (Union-Inter)/Union;
}
```

Figure 80 : Code Source de la méthode distance de jaccard

- **ListInfoUserSimil (**String IdUser**):** créé la matrice de similarité entre les utilisateurs (cf. **figure** 81).

```
public DefaultTableModel ListInfoUserSimil (String idUser) throws FileNotFoundExceptio
     BufferedReader br = new BufferedReader(new FileReader("Data/SimilUser.txt"));
    String line;
    DefaultTableModel InfoFilmUser=new DefaultTableModel();

       InfoFilmUser.setColumnIdentifiers(new String[]{"Id","Note","TypeJr","Temp"});
    while((line = br.readLine()) != null) {
        String[] values=line.split("\\ ");
        SimilInfo in=new SimilInfo();
        if(values[0].equalsIgnoreCase(idUser)) {
        InfoFilmUser.addRow(new Object[]
                {values[1],values[2],values[3],values[4]});
        }
    }
    br.close();
    return InfoFilmUser;

}
```

Figure 81: Code source de la méthodeListInfoUserSiml

- **ListInfoItemSimil (**String IdItem**):** créé la matrice de similarité entre les items.
- **UserSimil (**String IdSimil**)** : cherché l'utilisateur le plus similaire. La **figure** 82 représente le code source de cette méthode.

```
public String UserSimil(String idUser) throws FileNotFoundException, IOExce
    BufferedReader br = new BufferedReader(new FileReader("Data/SimilUser.t
    String line,IdSimil = null;
 DefaultTableModel InfoFilmUser=ListInfoUserSimil(idUser);
   double Res=1;

    for(int i=1;i<=943;i++){
   if(idUser.equalsIgnoreCase(String.valueOf(i))==false){
        DefaultTableModel InfoDUser=ListInfoUserSimil(String.valueOf(i));
     Double d=DistanceJaccard(InfoFilmUser, InfoDUser);
      System.out.println("Id User : "+idUser + "| Id Autre User : "+i+" | S
        if(d<Res){Res=d;IdSimil=String.valueOf(i);}
       }

    }
    return IdSimil;


 }
```

Figure 82 : Code Source de la méthode UserSimil

- **ItemSimil (**String Item**) :** chercher l'item le plus similaire.

1.3 Les fonctionnalités du système

Notre système englobe les fonctionnalités suivantes :

- Des interfaces graphiques conviviales et faciles à utiliser.
- Recommandation Coté Profil.
- Recommandation Coté historique dans des temps différents
- Calcule similarité entre les utilisateurs ou les items.
- Création et affichage des arbres de décision.
- Extraction de règles d'association avec le calcule de confiance maximal.
- Affichages des filmes trié par ordre décroisant
- Evaluation des différents systèmes.

La **figure** 822 schématise les différentes étapes des systèmes.

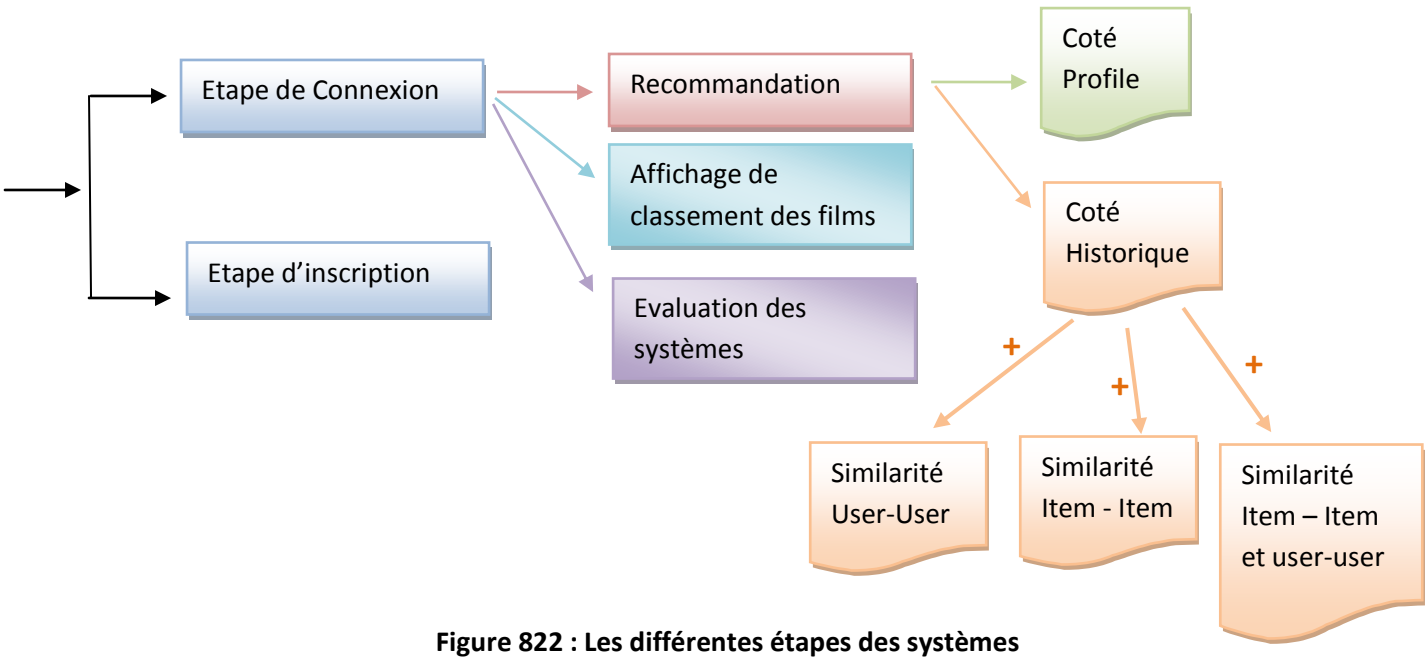

Figure 822 : Les différentes étapes des systèmes

1.4 Interface du système

1.4.1 Interface de connexion

L'interface de connexion est un objet héritant de la classe JFrame. Elle représente une fenêtre standard affichée en plein écran (cf. **figure** 83).

Figure 83 : interface de connexion au système

1 : Pour écrire l'id de l'utilisateur.

2 : Pour écrire l'âge de l'utilisateur.

3 : Pour connecter a notre système.

4 : Pour ajouter un nouvel utilisateur (cf. figure 84)

Figure 84 : Ajouter Un Nouvel utilisateur

1.4.2 Interface Principale

Une fenêtre de chargement des données et de création des arbres apparaissent (cf. **figure** 85) avant l'interface principale.

Figure 85 : Interface du chargement des données

Après la fin du chargement l'interface principale apparaisse (cf. **figure** 86).

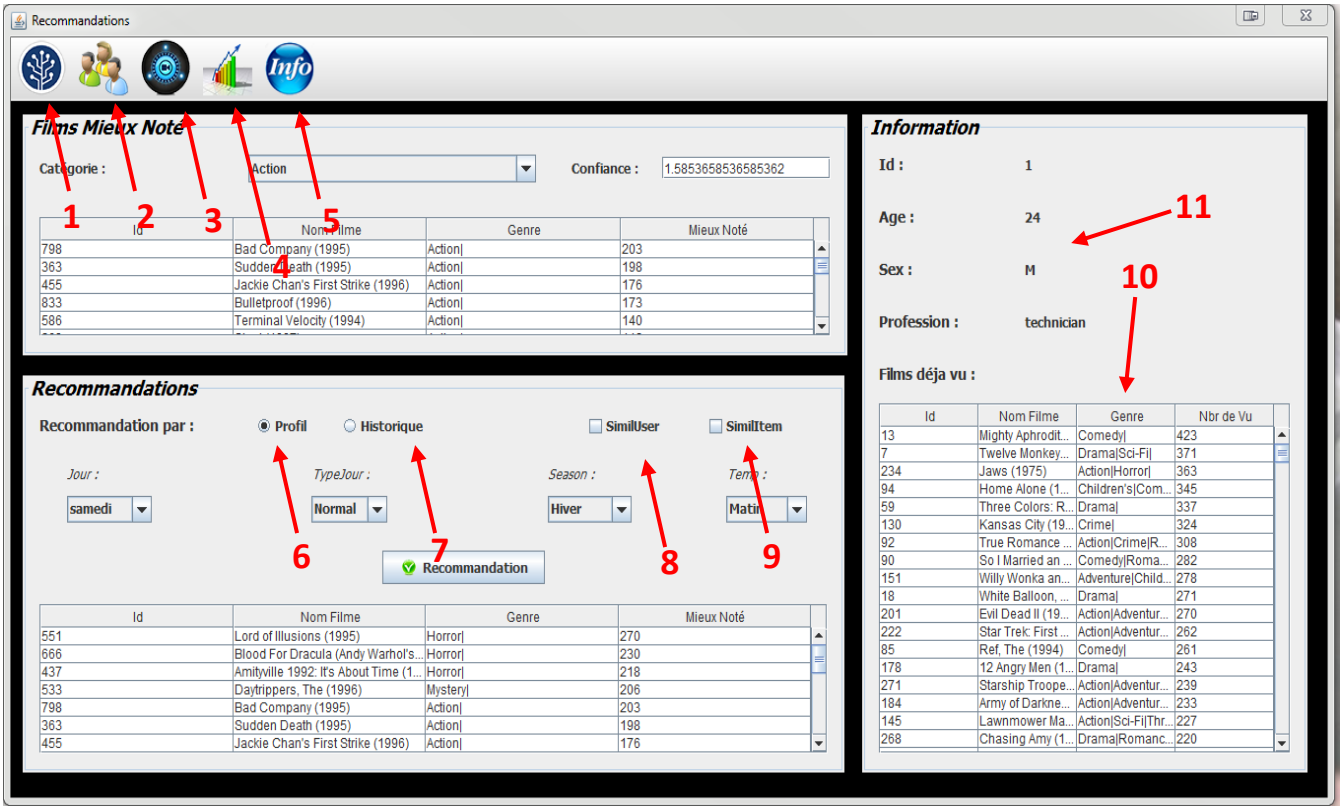

Figure 86 : Interface principale

1 : Afficher l'arbre de décision.

2 : Calculer la similarité entre les utilisateurs.

3 : Calculer la similarité entre les filmes.

4 : Evaluation des systèmes de recommandation (cf. **figure** 87).

5 : Information sur la personne qui à réaliser le système (cf. **figure** 88).

6 : pour une recommandation cotée profil.

7 : pour une recommandation cotée historique.

8 : pour une recommandation utilisant les arbres de décision et la similarité entre les utilisateurs.

9 : pour une recommandation utilisant les arbres de décision et la similarité entre les films.

10 : les informations sur l'utilisateur connecté.

11 : tous les films déjà vu par l'utilisateur.

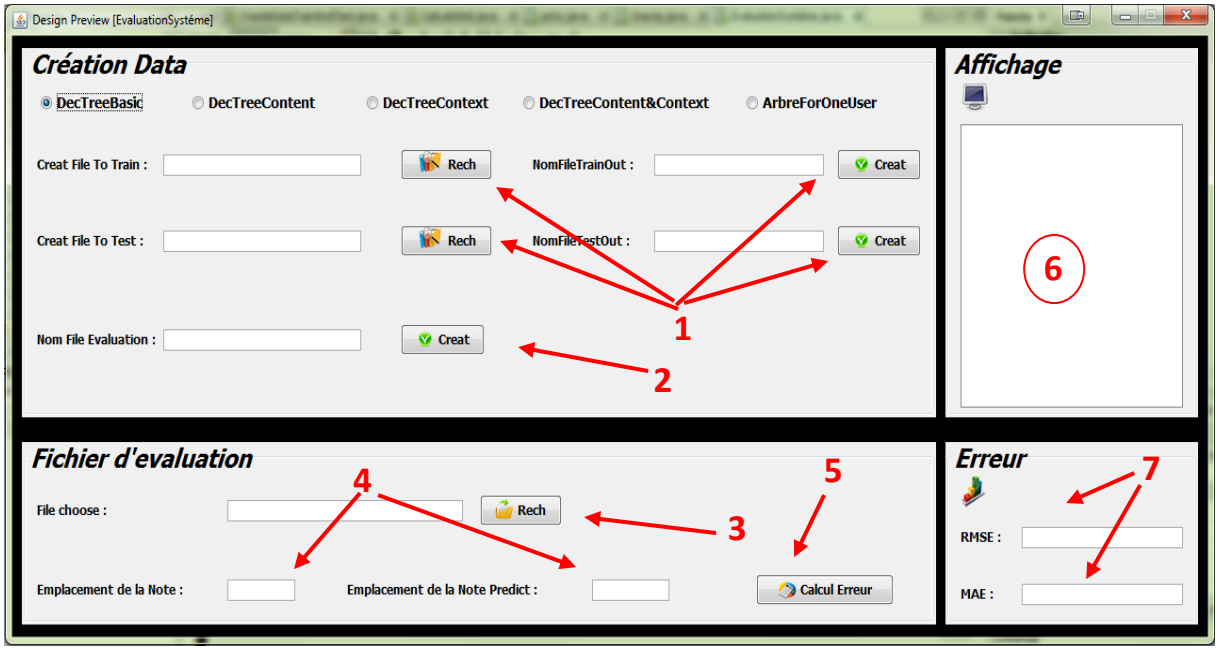

Figure 87 : Interface pour l'évaluation des systèmes

Dans la **figure** 87, on a 5 système a évalué il faut juste sélectionner un seul, après en suivre les étapes suivante :

1 : Chargement et création des fichiers d'apprentissage et test.

2 : Création de fichier d'évaluation

3 : Chargement du fichier d'évaluation

4 : Emplacement de la note de test et la note prédis par l'arbre

5 : Calcule des erreurs

6 : Affichage de la note de test et la note prédis par l'arbre.

7 : Afficher le taux d'erreur.

Figure 88 : Information sur la personne qui à réaliser le système

Conclusion :

Ce chapitre englobe la démarche suivie et synthétise l'ensemble des résultats obtenus.