Matching of MovieLens and IMDb Movie Titles

Verónika Peralta

Laboratoire PRiSM, Université de Versailles
45, avenue des Etats-Unis
78035 Versailles Cedex
FRANCE

veronika.peralta@prism.uvst.fr

Technical Report [1]

March 2007

Abstract. This report describes the procedure followed for joining MovieLens and IMDb databases. Even if both databases identify movie by title, unfortunately, they suffer from many problems such as abbreviation, translations into different languages and, generally, lack of writing standardization, which makes difficult their join. In order to match movie titles we propose a suite of approximate matches that consider different matching heuristics.

1. Introduction

This report describes the procedure followed for joining MovieLens [1] and IMDb [2] databases. Both databases deal with data about movies. The IMDb database contains rich information about films, actors, directors, the places where they are produced, their budgets, their categories and the average rank given by the users who had evaluated them. IMDb describes more than 850.000 movies at the moment we have extracted its data (October 2006). The MovieLens database contains very few information about films but provides a huge amount of evaluations given by users who have seen these films. MovieLens provides two datasets. The bigger one is composed of more than 1 million evaluations given by 6.040 users on 3.883 films; the smaller one is composed of 100.000 evaluations given by 943 users on 1.682 films. MovieLens and IMDb databases are complementary as they almost target the same movies (actually the set of films referred in MovieLens is a subset of those referred in IMDb). However, the join between the two databases is not easy to perform as there is not a universal identifier for the contained movies. The only common data is the titles of the movies but, unfortunately, they suffer from many problems such as abbreviation, translations into different languages and, generally, lack of writing standardization.

In order to match movie titles we propose a suite of approximate matches that consider different matching heuristics. This report describes the matching process and presents statistics on matching results.

The remaining sections are organized as follows: Section 2 presents an overview of the matching process and Section 3 describes matching algorithms. Finally, Section 3 concludes.

[1] This research was partially supported by the French Ministry of Research and New Technolologies under the ACI program devoted to Data Masses (ACI-MD), project #MD-33.

2. Overview of the matching process

We extracted 858.961 movies from IMDb and 3.883 movies from the larger data set of MovieLens (see [3] for details about data extraction). After extraction, we stated that all MovieLens movies were included in the IMDb data set. However, matching titles was not trivial. Furthermore, only 79% of matches had identical movie titles in both data sets. Other matching strategies were implemented, including manual look-up, in order to match the remaining titles. This section presents an overview of the matching process; next section describes the proposed algorithms.

Both, MovieLens and IMDb identify movies by ad-hoc identifiers (called movie titles) that concatenate title and running year (between brackets)[2]. But even identifiers are built in the same manner, we found discrepancies in titles. Most typical causes are inclusion of special characters, typing errors, transposition or omission of articles, use of different running years, translation of foreign titles and use of alternative titles. Table 1 illustrates differences in movie titles.

IMDb titles	MovieLens titles	Causes
¡Three Amigos! (1986)	Three Amigos! (1986)	Special characters
Shall We Dance (1937)	Shall We Dance? (1937)	
8 ½ Women (1999)	8 1/2 Women (1999)	
One Hundred and One Dalmatians (1961)	101 Dalmatians (1961)	
Two Moon Junction (1988)	Two Moon Juction (1988)	Typing errors
La Bamba (1987)	Bamba, La (1987)	Transposed articles
El Dorado (1966)	Dorado, El (1967)	
Three Ages (1923)	Three Ages, The (1923)	Omitted articles
Story of G.I. Joe (1945)	Story of G.I. Joe, The (1945)	
Tarantella (1996)	Tarantella (1995)	Different years
Supernova (2000/I)	Supernova (2000)	
Abre los ojos (1997)	Open Your Eyes (Abre los ojos) (1997)	English titles
Caro diario (1994)	Dear Diary (Caro Diario) (1994)	
Huitième jour, Le (1996)	Eighth Day, The (Le Huitième jour) (1996)	
Historia oficial, La (1985)	Official Story, The (La Historia Oficial) (1985)	
Cité des enfants perdus, La (1995)	City of Lost Children, The (1995)	
Star Wars (1977)	Star Wars: Episode IV - A New Hope (1977)	Alternative titles
Santa Claus (1985)	Santa Claus: The Movie (1985)	
Sunset Blvd. (1950)	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	
Sugar Hill (1994)	Harlem (1993)	

Table 1 – Examples of movies having different titles

In order to match MovieLens titles with IMDb titles (included in ML_Movies and IMDb_Movies tables), we followed several matching strategies. Each strategy tries to match MovieLens titles not matched by previous strategies as shown in Figure 1. To this end, each strategy considers a different heuristic for building candidate (alternative) titles for unmatched movies and compares them with IMDb titles. We implemented the following strategies:

– *Join by movie title*: This strategy computes the exact match. We joined IMDb_Movies and ML_Movies tables by the MovieTitle attribute. We obtained 3086 matches (79%).

– *Matching using the MovieLens small data set*: This strategy uses the SML_Movies table (small data set) as a mapping table between unmatched MovieLens titles and IMDb titles. Specifically, the SML_Movies table contains the URL of the IMDb web page corresponding to each movie. We formatted the URL and replaced special characters (e.g. %20 represents a blank) obtaining candidate titles. We joined this table with unmatched MovieLens movies (by the MovieTitle attribute) and with IMDb_Movies (by the just built candidate title). We had several difficulties: First, the intersection of both MovieLens collections is not very large. Second, we cannot replace all special characters. As a result, we obtained 83 new matches, totalizing 3169 matches (82%).

[2] Additionally to movies, IMDb also describes series episodes, mini-series, TV series, videos and video games (which have different identifiers). We omit them in this report because they do not match MovieLens titles, which only includes cinema movies.

- *Matching extracting foreign title*: Some MovieLens titles are translated to English but include, between brackets, the original title (see examples in Table 1). We extracted original titles (text between brackets) of unmatched MovieLens movies obtaining candidate titles. Then, we joined them with IMDb_Movies. We obtained 92 new matches, totalizing 3261 matches (84%).
- *Matching ignoring running year*: Sometimes, movie titles embeds running years, sometimes they embed diffusion years and sometimes they include additional characters (e.g. '1999/I'). This strategy consists in ignoring years for joining movie titles and manually verifying the obtained matches. We removed years from movie titles of unmatched MovieLens movies and IMDb_Movies. Then, we joined these auxiliary titles storing matches in a temporal table. A human validated matches (e.g. those differentiating in only one year) and eliminated erroneous ones. As a result, we obtained 295 new matches, totalizing 3556 matches (92%).
- *Matching of 20 first characters*: This strategy consists in truncating titles to 20 characters, joining them and manually verifying the obtained matches. We tried to solve some kinds of alternative titles (e.g. "Friday the 13th Part III (1982)" and "Friday the 13th Part 3: 3D (1982)") or special characters or truncations (e.g. "Why Do Fools Fall In Love? (1998)" and "Why Do Fools Fall In Love (1998)"). We truncated titles of unmatched MovieLens movies and IMDb_Movies. Then, we joined these auxiliary titles storing matches in a temporal table. A human validated matches by comparing whole titles and eliminated erroneous matches. As a result, we obtained 34 new matches, totalizing 3590 matches (92%).
- *Matching of 10 first characters*: This strategy repeats the previous one, but truncating titles to 10 characters. We obtained 46 new matches, totalizing 3636 matches (94%).
- *Manual look-up*: This strategy consists in manually examining unmatched MovieLens titles looking for possible matches in the IMDb_Movies table. We used intuition for finding titles in the huge IMDb collection (e.g. transposing words for "La Bamba (1987)" and "Bamba, La (1987)") and knowledge of foreign languages (e.g. "Cité des enfants perdus, La (1995)" and "City of Lost Children, The (1995)"). We obtained 116 new matches, totalizing 3752 matches (97%).
- *Web look-up*: The last strategy uses the search engine of MovieLens web site to find unmatched movies. Then, we used the link provided by MovieLens site to access the movie page at IMDb and we copied its title. We matched the remaining 131 movie titles.

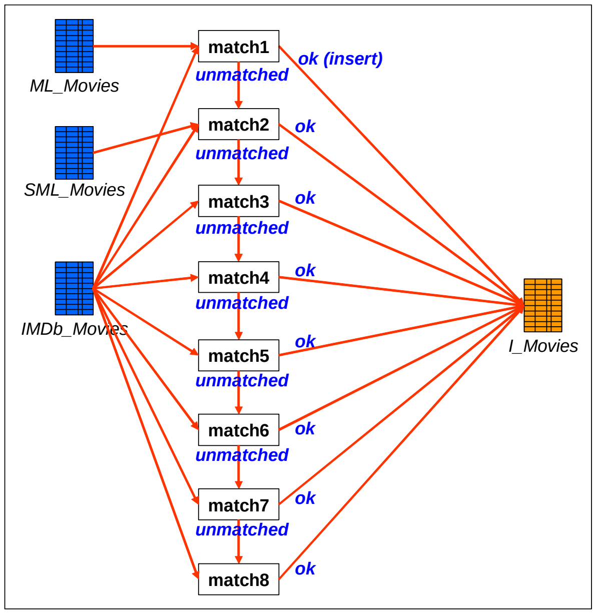

Figure 1 – Matching process

After matching titles, we found that 2 movies had the same corresponding movie at IMDb, i.e. we detected 2 duplicates. Tuples with original title were kept; those with translated title were removed.

3. Description of matching algorithms

This section details the matching procedures, describing their behaviors, inputs and outputs. Procedures were implemented as SQL sentences.

The inputs of the matching process are the following tables:

- **IMDb_Movies** (MovieTitle, Year): table containing IMDb movies, identified by MovieTitle.
- **ML_Movies** (MovieId, MovieTitle, Genres): table containing MovieLens movies, identified by MovieId. Titles differ from those of IMDb.
- **SML_Movies** (MovieId, MovieTitle, URLimdb): table containing a small set of MovieLens movies, identified by MovieId and referencing the URL of the corresponding movie in IMDb. Movie ids differ form those from ML_Movies

The following sub-sections describe each one of the matching strategies.

3.1. Join by movie title

This strategy implements the exact match. It consists of two procedures, who compute the join and compute the unsolved movies, respectively:

❑ ***Join_1_movietitle***

- Overview: Joins movies from MovieLens and IMDb par title
- Input: ML_Movies, IMDb_Movies
- Output: Movies (creation); copied to table join_1
- Query:

```
CREATE TABLE Movies AS
SELECT  ML_Movies.MovieId, ML_Movies.MovieTitle AS TitleMovieLens,
   IMDb_Movies.MovieTitle AS TitleIMDb_Movies, "title" AS Type
FROM IMDb_Movies INNER JOIN ML_Movies
   ON IMDb_Movies.MovieTitle = ML_Movies.MovieTitle;
```

- Result: 3086 tuples

❑ ***Diff_1_movietitle***

- Overview: Computes the difference between movies from MovieLens and movies already joined
- Input: ML_Movies, Movies
- Output: diff1 (creation)
- Query:

```
CREATE TABLE diff_1 AS
SELECT  ML_Movies.MovieId, ML_Movies.MovieTitle AS TitleMovieLens
FROM ML_Movies
WHERE   ML_Movies.MovieTitle NOT IN
   (SELECT TitleMovieLens FROM Movies);
```

- Result: 797 tuples

3.2. Matching using the MovieLens small data set

This strategy uses the SML_Movies table (small data set) as a mapping table between unmatched MovieLens titles and IMDb titles. It consists of nine procedures: the former joins the small data set with unmatched movies of the large data set, the following five extract a candidate movie title from the URL of the small data set substituting some special characters, the following one computes the join with IMDb titles and the two latter register solved movies and compute unsolved movies, respectively.

- ***Aux_2_1***
  - Overview: Intersects SML_Movies with movies not yet joined. Adds auxiliary attributes.
  - Input: SML_Movies, diff_1
  - Output: auxiliar_2_1 (creation)
  - Query:

```
CREATE TABLE auxiliar_2_1 AS
SELECT  diff_1.MovieId, diff_1.TitleMovieLens, SML_Movies.URLimdb,
   "" AS expr1, "" AS type
FROM SML_Movies INNER JOIN diff_1
   ON SML_Movies.MovieTitle = diff_1.TitleMovieLens;
```

  - Result: 268 tuples

- ***Aux_2_2***
  - Overview: Extracts the movie title form the URL (case 1: exact title)
  - Input: auxiliar_2_1
  - Output: auxiliar_2_1 (update)
  - Query:

```
UPDATE auxiliar_2_1
SET  auxiliar_2_1.expr1 = Mid(URLimdb,34,500),
   auxiliar_2_1.type = "url-1"
WHERE   Mid(URLimdb,25,1)="l";
```

  - Result: 239 tuples updated (268 tuples)

- ***Aux_2_3***
  - Overview: Extracts the movie title form the URL (case 2: approximate title)
  - Input: auxiliar_2_1
  - Output: auxiliar_2_1 (update)
  - Query:

```
UPDATE auxiliar_2_1
SET  auxiliar_2_1.expr1 = Mid(URLimdb,26,100),
   auxiliar_2_1.type = "url-2"
WHERE   Mid(URLimdb,25,1)<>"l";
```

  - Result: 29 tuples updated (268 tuples)

- ***Aux_2_4***
  - Overview: Substitutes special characters in the extracted movie title (case 1: character %20 → " ")
  - Input: auxiliar_2_1
  - Output: auxiliar_2_1 (update)
  - Query:

```
UPDATE auxiliar_2_1
SET  auxiliar_2_1.expr1 = Replace([expr1],"%20"," ",1,500,1);
```

  - Result: 268 tuples updated (268 tuples)

- ***Aux_2_5***
  - Overview: Substitutes special characters in the extracted movie title (case 2: character + → “ “)
  - Input: auxiliar_2_1
  - Output: auxiliar_2_1 (update)
  - Query:

```
UPDATE auxiliar_2_1
SET auxiliar_2_1.expr1 = Replace([expr1],"+"," ",1,500,1);
```

  - Result: 268 tuples updated (268 tuples)

- ***Aux_2_6***
  - Overview: Substitutes special characters in the extracted movie title (case 3: character %E9 → é)
  - Input: auxiliar_2_1
  - Output: auxiliar_2_1 (update)
  - Query:

```
UPDATE auxiliar_2_1
SET auxiliar_2_1.expr1 = Replace([expr1],"%E9","é",1,500,1);
```

  - Result: 268 tuples updated (268 tuples)

- ***Aux_2_7***
  - Overview: Joins titles obtained from URLs with IMDb movies
  - Input: auxiliar_2_1, IMDb_Movies
  - Output: auxiliar_2_7 (creation)
  - Query:

```
CREATE TABLE auxiliar_2_7 AS
SELECT  auxiliar_2_1.MovieId, auxiliar_2_1.TitleMovieLens,
IMDb_Movies.MovieTitle AS TitleIMDb,
   auxiliar_2_1.type
FROM auxiliar_2_1 INNER JOIN IMDb_Movies
   ON auxiliar_2_1.expr1 = IMDb_Movies.MovieTitle;
```

  - Result: 80 tuples

- ***Join_2_url***
  - Overview: Adds new joined tuples to movies table
  - Input: auxiliar_2_7, Movies
  - Output: movies (insert); copied to table join_2
  - Query:

```
INSERT INTO Movies ( MovieId, TitleMovieLens, TitleIMDb, Type )
SELECT auxiliar_2_7.MovieId, auxiliar_2_7.TitleMovieLens,
   auxiliar_2_7.TitleIMDb, auxiliar_2_7.type
FROM auxiliar_2_7;
```

  - Result: 83 additional tuples (3169 tuples)

- ***Diff_2_url***
  - Overview: Computes the difference between movies from MovieLens and movies already joined
  - Input: diff1, auxiliar_2_7
  - Output: diff_2 (creation)
  - Query:

```
CREATE TABLE diff_2 AS
SELECT  diff_1.MovieId, diff_1.TitleMovieLens
FROM diff_1
WHERE   diff_1.TitleMovieLens NOT IN
   (SELECT TitleMovieLens FROM auxiliar_2_7);
```

  - Result: 714 tuples

3.3. Matching stracting foreign title

This strategy uses the original title of a movie (included between brackets in some MovieLens titles, which were translated to English) for joining with IMDb titles. It consists of eight procedures: the five former extracts the original title from unmatched movies of MovieLens substituting some special characters, the following one computes the join with IMDb titles, and the two latter register solved movies and compute unsolved movies, respectively.

- ***Aux_3_1***
  - Overview: Adds auxiliary attributes to the movies not yet joined
  - Input: diff_2
  - Output: auxiliar_3_1 (creation)
  - Query:

```
CREATE TABLE auxiliar_3_1 AS
SELECT  diff_2.MovieId, diff_2.TitleMovieLens, 0 AS position1,
   0 AS position2, "" AS original
FROM diff_2;
```

  - Result: 714 tuples

- ***Aux_3_2***
  - Overview: Computes positions of brackets (which enclose original title)
  - Input: auxiliar_3_1
  - Output: auxiliar_3_1 (update)
  - Query:

```
UPDATE auxiliar_3_1
SET auxiliar_3_1.position1 = InStr(1,[TitleMovieLens],"(",1),
   auxiliar_3_1.position2 = InStr(1,[TitleMovieLens],")",1);
```

  - Result: 714 tuples updated (714 tuples)

- ***Aux_3_3***
  - Overview: Extracts the original title (between brackets)
  - Input: auxiliar_3_1
  - Output: auxiliar_3_1 (update)
  - Query:

```
UPDATE auxiliar_3_1
SET auxiliar_3_1.original =
Mid([TitleMovieLens],[Position1]+1,[Position2]-[Position1]-1)
   +Mid([TitleMovieLens],[Position2]+1,1000);
```

  - Result: 714 tuples updated (714 tuples)

- ***Aux_3_4***
  - Overview: Substitutes special characters in the extracted original title (case 1: double space)
  - Input: auxiliar_3_1
  - Output: auxiliar_3_1 (update)
  - Query:

```
UPDATE auxiliar_3_1
SET  auxiliar_3_1.original = Replace(origianl,"  (","  (",1,500,1);
```

  - Result: 714 tuples updated (714 tuples)

- ***Aux_3_5***
  - Overview: Substitutes special characters in the extracted original title (case 2: a.k.a.)
  - Input: auxiliar_3_1
  - Output: auxiliar_3_1 (update)
  - Query:

```
UPDATE auxiliar_3_1
SET  auxiliar_3_1.original = Replace(origianl,"a.k.a. ","",1,500,1);
```

  - Result: 714 tuples updated (714 tuples)

- ***Aux_3_6***
  - Overview: Joins titles obtained from URLs with IMDb movies
  - Input: auxiliar_3_1, IMDb_Movies
  - Output: auxiliar_3_6 (creation)
  - Query:

```
CREATE TABLE auxiliar_3_6 AS
SELECT  auxiliar_3_1.MovieId, auxiliar_3_1.TitleMovieLens,
   IMDb_Movies.MovieTitle AS TitleIMDB, "orig" AS Type
FROM auxiliar_3_1 INNER JOIN IMDb_Movies
   ON auxiliar_3_1.original = IMDb_Movies.MovieTitle;
```

  - Result: 92 tuples

- ***Join_3_original***
  - Overview: Adds new joined tuples to movies table
  - Input: auxiliary_2_6
  - Output: movies (insert); copied to table join_3
  - Query:

```
INSERT INTO Movies ( MovieId, TitleMovieLens, TitleIMDb, Type )
SELECT  auxiliar_3_6.MovieId, auxiliar_3_6.TitleMovieLens,
   auxiliar_3_6.TitleIMDB, auxiliar_3_6.Type
FROM auxiliar_3_6;
```

  - Result: 92 additional tuples (3261 tuples)

- ***Diff_3_original***
  - Overview: Computes the difference between movies from MovieLens and movies already joined
  - Input: diff2, auxiliar_3_6
  - Output: diff_3 (creation)
  - Query:

```
CREATE TABLE diff_3 AS
SELECT  diff_2.MovieId, diff_2.TitleMovieLens
FROM diff_2
WHERE   diff_2.TitleMovieLens NOT IN
   (SELECT TitleMovieLens FROM auxiliar_3_6);
```

  - Result: 622 tuples

3.4. Matching ignoring running year

This strategy consists in ignoring years for joining movie titles and manually verifying the obtained matches. It consists of nine procedures: the five former removes years from both, the unmatched MovieLens titles and IMDb titles, the following one computes the join among them, the following one is a manual procedure for eliminating erroneous matches, and the two latter register solved movies and compute unsolved movies, respectively.

- ***Aux_4_1***
  - Overview: Adds auxiliary attributes to the movies not yet joined
  - Input: diff_3
  - Output: auxiliar_4_1 (creation)
  - Query:

```
CREATE TABLE auxiliar_4_1 AS
SELECT diff_3.MovieId, diff_3.TitleMovieLens, 0 AS [position],
   "" AS prefix
FROM diff_3;
```

  - Result: 622 tuples

- ***Aux_4_2***
  - Overview: Computes positions of brackets (beginning of year in most cases)
  - Input: auxiliar_4_1
  - Output: auxiliar_4_1 (update)
  - Query:

```
UPDATE auxiliar_4_1
SET   auxiliar_4_1.[position] = InStr(1,[TitleMovieLens],"(",1);
```

  - Result: 622 tuples updated (622 tuples)

- ***Aux_4_3***
  - Overview: Extracts the prefix ignoring year (before brackets)
  - Input: auxiliar_4_1
  - Output: auxiliar_4_1 (update)
  - Query:

```
UPDATE auxiliar_4_1
SET   auxiliar_4_1.prefix = Mid([TitleMovieLens],1,[Position]-1);
```

  - Result: 622 tuples updated (622 tuples)

- ***Aux_4_4***
  - Overview: Computes positions of brackets (beginning of year in most cases) for IMDb movies. Adds auxiliary attributes.
  - Input: IMDb_Movies
  - Output: auxiliar_4_4 (creation)
  - Query:

```
CREATE TABLE auxiliar_4_4 AS
SELECT  IMDb_Movies.MovieTitle AS TitleIMDb,
   InStr(1,[IMDb_Movies].[MovieTitle],"(",1) AS position, "" AS prefix
FROM IMDb_Movies;
```

  - Result: 858961 tuples

- ***Aux_4_5***
  - Overview: Extracts the prefix ignoring year (before brackets)
  - Input: auxiliar_4_4
  - Output: auxiliar_4_4 (update)
  - Query:

```
UPDATE auxiliar_4_4
SET   auxiliar_4_4.prefix = Mid([TitleIMDb],1,[Position]-1);
```

  - Result: 858961 tuples updated (858961 tuples)

- ***Aux_4_6***
  - Overview: Joins both tables by prefix
  - Input: auxiliar_4_1, auxiliar_4_4
  - Output: auxiliar_4_6 (creation)
  - Query:

```
CREATE TABLE auxiliar_4_6 AS
SELECT auxiliar_4_1.MovieId, auxiliar_4_1.TitleMovieLens,
   auxiliar_4_4.TitleIMDb, auxiliar_4_1.prefix, "year" AS Type
FROM auxiliar_4_4 INNER JOIN auxiliar_4_1
   ON auxiliar_4_4.prefix = auxiliar_4_1.prefix;
```

  - Result: 930 tuples

- ***Manual_4***
  - Overview: Manual filtering of auxiliar_4_6 table, eliminating erroneous joins
  - Input: auxiliar_4_6
  - Output: manual_4 (creation)
  - Query: Manual filtering, copying result to table manual _4
  - Result: 295 tuples

- ***Join_4_year***
  - Overview: Adds new joined tuples to movies table
  - Input: manual_4
  - Output: movies (insert); copied to table join_4
  - Query:

```
INSERT INTO Movies ( MovieId, TitleMovieLens, TitleIMDb, Type )
SELECT  manual_4.MovieId, manual_4.TitleMovieLens, manual_4.TitleIMDB,
   manual_4.Type
FROM manual_4;
```

  - Result: 295 additional tuples (3556 tuples)

- ***Diff_4_year***
  - Overview: Computes the difference between movies from MovieLens and movies already joined
  - Input: diff3, manual_4
  - Output: diff_4 (creation)
  - Query:

```
CREATE TABLE diff_4 AS
SELECT  diff_3.MovieId, diff_3.TitleMovieLens
FROM diff_3
WHERE   diff_3.TitleMovieLens NOT IN
   (SELECT TitleMovieLens FROM manual_4);
```

  - Result: 327 tuples

3.5. Matching of 20 first characters

This strategy consists in truncating titles to 20 characters, joining them and manually verifying the obtained matches. It consists of six procedures: the two former extracts the first characters of the both, the unmatched MovieLens titles and IMDb titles, the following one computes the join among them, the following one is a manual procedure for eliminating erroneous matches, and the two latter register solved movies and compute unsolved movies, respectively.

- ❑ ***Aux_5_1***
  - – Overview: Extracts a 20-character prefix of movies not yet joined
  - – Input: diff_4
  - – Output: auxiliar_5_1 (creation)
  - – Query:

```
CREATE TABLE auxiliar_5_1 AS
SELECT diff_4.MovieId, diff_4.TitleMovieLens,
  Mid([TitleMovieLens],1,20) AS Extr20
FROM diff_4;
```

  - – Result: 327 tuples

- ❑ ***Aux_5_2***
  - – Overview: Extracts a 20-character prefix of IMDb movies
  - – Input: IMDb_Movies
  - – Output: auxiliar_5_2 (creation)
  - – Query:

```
CREATE TABLE auxiliar_5_2 AS
SELECT  IMDb_Movies.MovieTitle AS TitleIMDb,
  Mid([MovieTitle],1,20) AS Extr20
FROM IMDb_Movies;
```

  - – Result: 858961 tuples

- ❑ ***Aux_5_3***
  - – Overview: Joins both tables by prefix
  - – Input: auxiliar_5_1, auxiliar_5_2
  - – Output: auxiliar_5_3 (creation)
  - – Query:

```
CREATE TABLE auxiliar_5_3 AS
SELECT  auxiliar_5_1.MovieId, auxiliar_5_1.TitleMovieLens,
  auxiliar_5_2.TitleIMDb, auxiliar_5_1.Extr20
FROM auxiliar_5_1 INNER JOIN auxiliar_5_2
  ON auxiliar_5_1.Extr20 = auxiliar_5_2.Extr20;
```

  - – Result: 160 tuples

- ❑ ***Manual_5***
  - – Overview: Manual filtering of auxiliar_5_3 table, eliminating erroneous joins
  - – Input: auxiliar_5_3
  - – Output: manual_5 (creation)
  - – Query: Manual filtering, copying result to table manual _5
  - – Result: 34 tuples

- ***Join_5_extr20***
  - Overview: Adds new joined tuples to movies table
  - Input: manual_5
  - Output: movies (insert); copied to table join_5
  - Query:

```
INSERT INTO Movies ( MovieId, TitleMovieLens, TitleIMDb, Type )
SELECT  manual_5.MovieId, manual_5.TitleMovieLens, manual_5.TitleIMDB,
   manual_5.Type
FROM manual_5;
```

  - Result: 34 additional tuples (3590 tuples)

- ***Diff_5_extr20***
  - Overview: Computes the difference between movies from MovieLens and movies already joined
  - Input: diff4, manual_5
  - Output: diff_5 (creation)
  - Query:

```
CREATE TABLE diff_5 AS
SELECT  diff_4.MovieId, diff_4.TitleMovieLens
FROM diff_4
WHERE   diff_4.TitleMovieLens NOT IN
   (SELECT TitleMovieLens FROM manual_5);
```

  - Result: 293 tuples

3.6. Matching of 10 first characters

This strategy repeats the previous one, but truncating titles to 10 characters.

- ***Aux_6_1***
  - Overview: Extracts a 10-character prefix of movies not yet joined
  - Input: diff_5
  - Output: auxiliar_6_1 (creation)
  - Query:

```
CREATE TABLE auxiliar_6_1 AS
SELECT diff_5.MovieId, diff_5.TitleMovieLens,
   Mid([TitleMovieLens],1,10) AS Extr10
FROM diff_5;
```

  - Result: 293 tuples

- ***Aux_6_2***
  - Overview: Extracts a 10-character prefix of IMDb movies
  - Input: IMDb_Movies
  - Output: auxiliar_6_2 (creation)
  - Query:

```
CREATE TABLE auxiliar_6_2 AS
SELECT  IMDb_Movies.MovieTitle AS TitleIMDb,
   Mid([MovieTitle],1,10) AS Extr10
FROM IMDb_Movies;
```

  - Result: 858961 tuples

- ***Aux_6_3***
  - Overview: Joins both tables by prefix
  - Input: auxiliar_6_1, auxiliar_6_2
  - Output: auxiliar_6_3 (creation)
  - Query:

```
CREATE TABLE auxiliar_6_3 AS
SELECT  auxiliar_6_1.MovieId, auxiliar_6_1.TitleMovieLens,
  auxiliar_6_2.TitleIMDb, auxiliar_6_1.Extr10
FROM auxiliar_6_1 INNER JOIN auxiliar_6_2
  ON auxiliar_6_1.Extr10 = auxiliar_6_2.Extr10;
```

  - Result: 2488 tuples

- ***Manual_6***
  - Overview: Manual filtering of auxiliar_5_3 table, eliminating erroneous joins
  - Input: auxiliar_5_3
  - Output: manual_5 (creation)
  - Query: Manual filtering, copying result to table manual _5
  - Result: 46 tuples

- ***Join_6_extr10***
  - Overview: Adds new joined tuples to movies table
  - Input: manual_6
  - Output: movies (insert); copied to table join_6
  - Query:

```
INSERT INTO Movies ( MovieId, TitleMovieLens, TitleIMDb, Type )
SELECT  manual_6.MovieId, manual_6.TitleMovieLens, manual_6.TitleIMDB,
  manual_6.Type
FROM manual_6;
```

  - Result: 46 additional tuples (3636 tuples)

- ***Diff_6_extr10***
  - Overview: Computes the difference between movies from MovieLens and movies already joined
  - Input: diff5, manual_6
  - Output: diff_6 (creation)
  - Query:

```
CREATE TABLE diff_6 AS
SELECT  diff_5.MovieId, diff_5.TitleMovieLens
FROM diff_5
WHERE   diff_5.TitleMovieLens NOT IN
  (SELECT TitleMovieLens FROM manual_6);
```

  - Result: 247 tuples

3.7. Manual look-up

This strategy consists in manually examining unmatched MovieLens titles looking for possible matches in the IMDb_Movies table. It consists of three procedures: the former is a manual procedure for finding matches and the two latter register solved movies and compute unsolved movies, respectively.

- ***Manual_7***
  - Overview: Manual join
  - Input: diff_6, IMDb_Movies
  - Output: manual_7 (creation)
  - Query: Manual join, copying result to table manual _7
  - Result: 116 tuples

- ***Join_7_manual***
  - Overview: Adds new joined tuples to movies table
  - Input: manual_7
  - Output: movies (insert); copied to table join_7
  - Query:

```
INSERT INTO Movies ( MovieId, TitleMovieLens, TitleIMDb, Type )
SELECT  manual_7.MovieId, manual_7.TitleMovieLens, manual_7.TitleIMDB,
   "manual" AS Type
FROM manual_7;
```

  - Result: 116 additional tuples (3752 tuples)

- ***Diff_7_manual***
  - Overview: Computes the difference between movies from MovieLens and movies already joined
  - Input: diff6, manual_7
  - Output: diff_7 (creation)
  - Query:

```
CREATE TABLE diff_7 AS
SELECT  diff_6.MovieId, diff_6.TitleMovieLens
FROM diff_6
WHERE   diff_6.TitleMovieLens NOT IN
   (SELECT TitleMovieLens FROM manual_7);
```

  - Result: 131 tuples

3.8. Web look-up

This strategy uses the search engine of MovieLens web site to find unmatched movies, uses the link provided by MovieLens site to access the movie page at thus find the IMDb title. It consists of two procedures: the former is a manual procedure for finding matches through the web interface and the latter registers solved movies.

- ***Manual_8***
  - Overview: Manual join
  - Input: diff_7, IMDb_Movies
  - Output: manual_8 (creation)
  - Query: Manual join, copying result to table manual _8
  - Result: 131 tuples

- ***Join_8_manual***
  - Overview: Adds new joined tuples to movies table
  - Input: manual_8
  - Output: movies (insert); copied to table join_8
  - Query:

```
INSERT INTO Movies ( MovieId, TitleMovieLens, TitleIMDb, Type )
SELECT  manual_8.MovieId, manual_8.TitleMovieLens, manual_8.TitleIMDB,
   "web" AS Type
FROM manual_8;
```

  - Result: 131 additional tuples (3883 tuples)

3.9. Delete duplicate movies

After matching titles, we found that 2 movies had the same corresponding movie at IMDb, i.e. we detected 2 duplicates. The remaining two procedures compute duplicate movies and eliminate them.

- ***Dupl_9***
  - Overview: Computes duplicate movies (those referencing the same IMDb movie)
  - Input: movies
  - Output: duplicates_9 (creation)
  - Query:

```
CREATE TABLE [Movies-duplicates] AS
SELECT  Movies.TitleIMDb, Count(*) AS Quantity,
   Min(Movies.MovieId) AS MinDeMovieId, Max(Movies.MovieId) AS MaxDeMovieId
FROM Movies
GROUP BY Movies.TitleIMDb
HAVING Count(*)>1;
```

  - Result: 2 tuples

- ***Join_9_duplicates***
  - Overview: Eliminates duplicates deleting (given priority to movies joined by movietitle)
  - Input: movies, duplicates_9
  - Output: movies (deletion); copied to table join_9
  - Query:

```
INSERT INTO Movies ( MovieId, TitleMovieLens, TitleIMDb, Type )
SELECT  manual_8.MovieId, manual_8.TitleMovieLens, manual_8.TitleIMDB,
   "web" AS Type
FROM manual_8;

DELETE FROM Movies
WHERE Movies.MovieId IN (select MinDeMovieId from duplicates_9)
   AND Movies.Type)<>"title"
OR Movies.MovieId IN (select MaxDeMovieId from duplicates_9)
   AND Movies.Type)<>"title";
```

  - Result: 2 deleted tuples (3881 tuples)

The whole matching process was executed in a Microsoft Access database and then migrated to an Oracle database.

4. Conclusion

In this report we described the process followed for matching MovieLens and IMDb titles. It consisted in 8 strategies, which alternate automatic processing and manual verification. It was a very costly and time-consuming work, but we succeeded to match all MovieLens titles to IMDb titles.

This integration of MovieLens and IMDb data is not only useful for our testing purposes but can benefit to a wide database community working on query personalization.

5. References

[1] GroupLens Research: "movielens: helping you to find the right movies". Web site, ULR: http://movielens.umn.edu, last accessed on July 9th, 2006.

[2] Internet Movie Database, Inc.: "The Internet Movie Database", Web site, URL: http://www.imdb.com/, last accessed on July 9th, 2007.

[3] Peralta, V.: "Extraction and Integration of MovieLens and IMDb Data". Technical Report, Laboratoire PRiSM, Université de Versailles, Versailles, France, 2007.