Using Big Data to Get Us from Where We Think We Are to Where We Might Want to Go

Tidewater Big Data Enthusiasts
Chuck Cartledge
Developer

April 23, 2020 at 12:14 Noon

Contents

List of Figures

List of Tables

1 Introduction

When we wander through the Internet, we leave behind digital traces of our passage. Some of our traces are embedded in our digital pictures[5], in our cell phone usage patterns[2], and from the cookies that we leave on our personal computers[3]. We are going to explore how some of these systems work as "recommenders" by using the Internet Movie Database(IMDb) to recommend movies that we might enjoy based on "misinformation" that we provide.

Our exploration will be to design a recommendation system that will understand and correct our misinformation, use the corrected information to identify specific relevant attributes, and using those attributes, recommend other new items we may be interested in. With data from the IMDb, we will:

1. Start with a misspelled movie title,
2. Have the database recommend a title that is close to our misspelled one,
3. Have the database identify which genre, or genres the movie belongs to,
4. Have the database combine the genres with a subset of the actors from the movie, and

to recommend other movies that we might be interested in viewing. These recommended movies will have some of the same actors from our original movie, and will belong to the same genre. We will be doing this using a venerable piece of structured query language (SQL) (see Section D on page 38) software called PostgreSQL (see Section A on page 22).

The idea being; if we liked the original movie, then we should like some of the recommended ones. This idea is behind many of the recommender systems that we use and observe everyday (see Figure 1 on the following page).

2 Approach

Now we will start delving into the details of how our movie recommendations system works, and conclude with some examples.

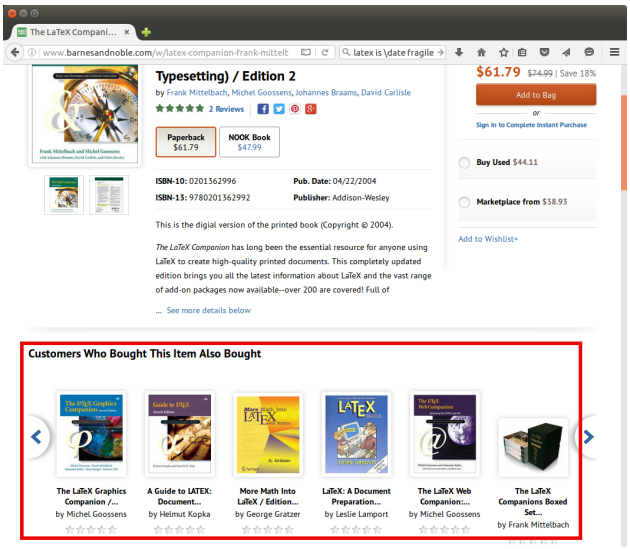
2.1 Data sources

Our raw movie database information is available for download from the Internet Movie Database (IMDb) website[1]. Rather than download individual entries from the IMDb, we will download large files specific to data types of interest from the IMDb Alternative Interfaces website[2]. The IMDb Alternative Interfaces in turn points to an FTP site where we can download data types of interest[3]. The three types of data we need are:

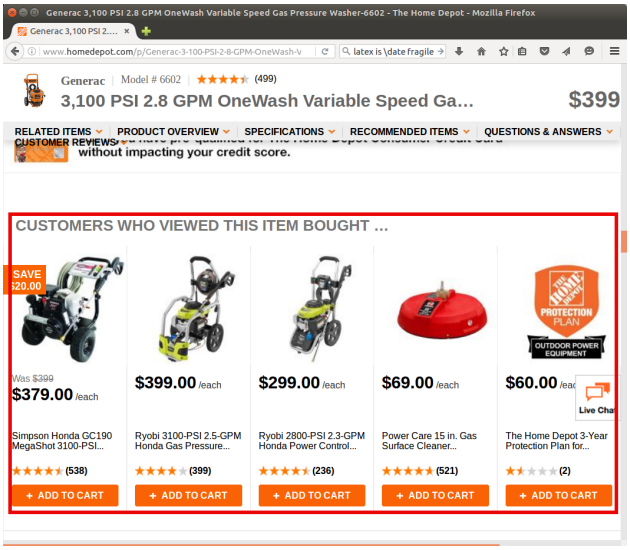
[1] `http://www.imdb.com/`

[2] `http://www.imdb.com/interfaces`

[3] `ftp://ftp.fu-berlin.de/pub/misc/movies/database/`



(a) Barnes & Noble



(b) Home Depot

Figure 1: Barnes & Noble and Home Depot web pages. These websites recommend additional items based on the user's past activities and the activities of other users. The recommendation areas are outlined in red.

1. actors (a list of actors from each movie) in the file actors.list.gz (approximately 295 GB compressed)
2. genres (which assigns each movie to one or more genres) in the file genres.list.gz (approximaltely 17 GB compressed)
3. movies (in the IMDb general sense) in the file movies.list.gz (approximately 37 GB compressed)

These files will allow us to go from a movie title to actors in that movie to which genres that movie is part of, and then to find other movies in the same genres that had some of the same actors as the original movie. Conceptually; if we liked movie A then we should mike movie B because it as some of the same actors, and is the same type as A.

Simplifying assumptions

This report is a "proof of concept" level effort, vice a production level effort. As such, there are a number of simplifying assumptions and design decisions that were made. These include:

1. Only deal with ASCII characters. The IMDb has movies (and other data) about productions made around the world. In many cases, the movie names contain characters that are outside the ASCII character set. Those characters are silently removed during the database creation.
2. Only deal with movies. The IMDb contains things other than traditional theater movies, including:
   - Shorts,
   - Videos,
   - TV shows,
   - Games,
   - Animations, and
   - other forms of electronic entertainment.
3. The number of common actors is fixed at 2. When the number of actors in a movie is small (< 10), it is likely that they will be together in another movie. When the number of actors is large (> 100), it is unlikely that they will be together in another movie. In a production system, the number of common actors could be a user entered value.
4. The Levenshtein distance (see Section 2.3 on page 6) is fixed at 4. In a production system, this could be a user entered value.

Table 1: Number of records downloaded from IMDb The gross number of records include a small number (< 300) lines of header and other information that is not loaded into the database.

File name	Gross number of records	Records in db.
actors.list.gz	17,855,640	6,117,750
actresses.list.gz	10,882,460	3,264,670
genres.list.gz	2,069,761	1,531,462
movies.list.gz	3,439,898	794,234

Table 2: PostgreSQL tables used for this demonstration. The name data type in the table is evident from the field name. Each table has another field (titleBase64) whose purpose may not be self evident. Because the IMDb is an international database, movie titles may contain characters that have special meaning to PostgreSQL, and would be difficult to load and process. Therefore the titles are all base 64 encoded so that special characters are preserved, but the encoding results in pure ASCII representation.

Table name	Column name	Characteristics
actors	actor	varchar(1000)
	titleBase64	varchar(1000)
genre	genre	varchar(1000)
	titleBase64	varchar(1000)
movies	title	varchar(1000)
	titleBase64	varchar(1000)

The simplifications resulted in fewer records being added to the recommendations database (see Table 1).

2.2 Database organization

The database consists of three tables, one for each type of data downloaded from the IMDb (see Table 2). A Pareto chart of the genres in the database is was created (see Figure 2 on page 7). From the chart it is easy to see that almost 50% of the entries are classified as "short" or "drama" or both. An textual distribution is provided as well (see Table 3 on the next page).

Table 3: Number of database entries by genre.

Genre	**Count**
Action	44,836
Adult	7,781
Adventure	25,769
Animation	37,826
Biography	18,083
Comedy	177,313
Crime	32,598
Documentary	140,445
Drama	264,613
Family	30,753
Fantasy	25,100
Film-Noir	612
Game-Show	51
History	15,337
Horror	37,999
Musical	12,521
Music	18,441
Mystery	22,041
News	7,140
Reality-TV	228
Romance	51,450
Sci-Fi	20,959
Short	452,549
Sport	9,015
Talk-Show	116
Thriller	51,766
War	12,455
Western	13,665

With so many different genres, and so many movies categorized into each genre; a natural question is: how many movies share common categories? For instance, how many movies are categorized as Biography, Crime, and Documentary? While the question is easily stated, and easily computed there are nuances that contribute to making the question difficult to answer as the number of categorizes increases. For the three representative categories, there are 8 possible ways that movies could be categorized. These are

1. In Biography only.
2. In Crime only.
3. In Documentary only.
4. In Biography and Crime, but not in Documentary.
5. In Crime and Documentary, but not in Biography.
6. In Biography and Documentary, but not in Crime.
7. In Biography and Documentary and Crime.
8. Not in Biography, nor Documentary, nor Crime.

It terms out that the number of possible combinations is 2^n. Our proof of concept has 28 genres:

2^{28} =268,435,456 possible combinations (see Section B on page 23)

.

2.3 Levenshtein distance algorithm

> *"..., the Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other."*
>
> Wikipedia Staff [12]

Levenshtein distance (LD) may also be referred to as edit distance. The LD can be viewed as a metric for comparing two strings[6]. The greater the LD is, the more different the two strings are.

LD has been used in:

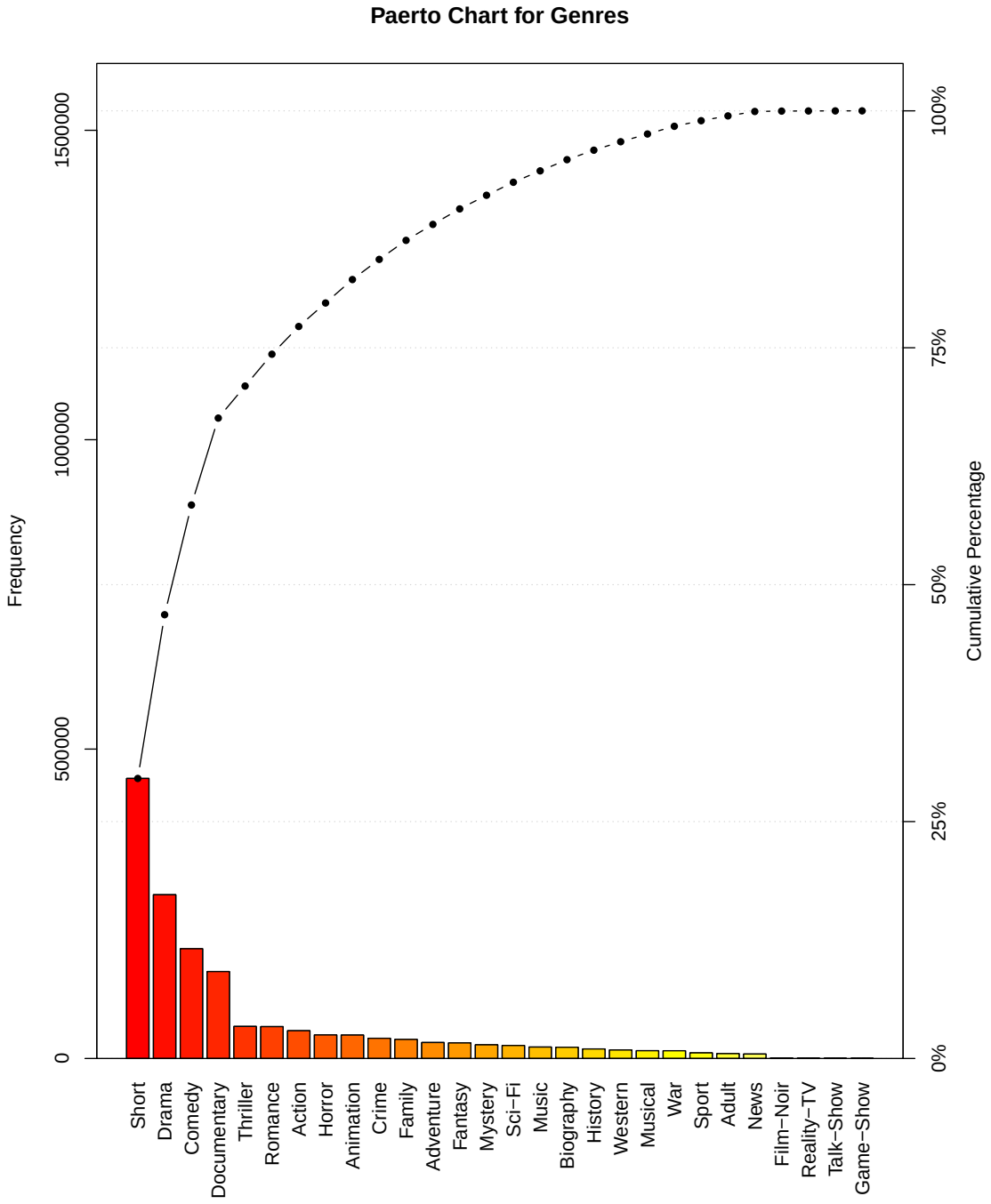


Figure 2: Pareto chart of genres. The three most common categorizations (Short, Drama, and Comedy) account for almost 60% of all categorizations.

- Spell checking
- Speech recognition
- DNA analysis
- Plagiarism detection

The following algorithm and example are taken from[4]. Assumptions:

1. The source string is called: `s`
2. The target string is called: `t`
3. The `cost` of all edit operations (insertions, deletions or substitutions) is 1. (Note: This may not be the real case in a real system.)

The goal is to compute the LD from `s` to `t`.

Mathematically[12]; LD between strings `s,t` (of length $\mid s \mid$ and $\mid t \mid$ respectively) is given by $lev_{s,t}(\mid s \mid, \mid t \mid)$. Where:

$$lev_{s,t}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{s,t}(i-1,j) + cost \\ lev_{s,t}(i,j-1) + cost \\ lev_{s,t}(i-1,j-1) + cost_{(s_i \neq t_j)} \end{cases} & \text{otherwise.} \end{cases}$$

where $1_{(s_i \neq t_j)}$ is the indicator function equal to 0 when $s_i = t_j$ and equal to 1 otherwise. An algorithmic explanation can be found in (see Table 4).

Table 4: Levenshtein algorithm.

Step	**Description**
1	• Set n to be the length of `s`. • Set m to be the length of `t`. • If $n = 0$, return m and exit. • If $m = 0$, return n and exit. • Construct a matrix containing $0 \ldots m$ rows and $0 \ldots n$ columns.

(Continued on the next page.)

Table 4. (Continued from the previous page.)

Step	Description
2	• Initialize the first row to $0 \ldots n$. • Initialize the first column to $0 \ldots m$.
3	Examine each character of `s` (i from 1 to n).
4	Examine each character of `t` (j from 1 to m).
5	• If $s[i]$ equals $t[j]$, the `cost` is 0. • If $s[i]$ doesn't equal $t[j]$, the `cost` is 1.
6	Set cell $lev(i, j)$ of the matrix equal to the minimum of: • The cell immediately above plus 1: $lev(i-1, j)+1$. • The cell immediately to the left plus 1: $lev(i, j-1)+1$. • The cell diagonally above and to the left plus the cost: $lev(i-1, j-1)+$ `cost`
7	After the iteration steps (3, 4, 5, 6) are complete, LD is found in cell $lev(n, m)$..

An example showing the *lev* matrix for changing the start string `s` GUMBO to the target string `t` GAMBOL provided (see Table 5 on the following page).

We will use LD to find the closest (lowest LD) between the a movie title entered by the user, and all movie titles in our database.

Table 5: Levenshtein converting GUMBO to GAMBOL. The LD (number of insertions, deletions or substitutions) is shown in the lower left hand cell of the last table as 2.

Steps 1 and 2

		G	U	M	B	O
	0	1	2	3	4	5
G	1					
A	2					
M	3					
B	4					
O	5					
L	6					

Steps 3 to 6 (i = 1)

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0				
A	2	1				
M	3	2				
B	4	3				
O	5	4				
L	6	5				

Steps 3 to 6 (i = 2)

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1			
A	2	1	1			
M	3	2	2			
B	4	3	3			
O	5	4	4			
L	6	5	5			

Steps 3 to 6 (i = 3)

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2		
A	2	1	1	2		
M	3	2	2	1		
B	4	3	3	2		
O	5	4	4	3		
L	6	5	5	4		

Steps 3 to 6 (i = 4)

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	
A	2	1	1	2	3	
M	3	2	2	1	2	
B	4	3	3	2	1	
O	5	4	4	3	2	
L	6	5	5	4	3	

Steps 3 to 6 (i = 5)

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	1	2	3	4
M	3	2	2	1	2	3
B	4	3	3	2	1	2
O	5	4	4	3	2	1
L	6	5	5	4	3	**2**

3 Results

Interactions with the database are initiated by the user. Some misspelled movie names will result in recommendations (small number of actors, lots of movies being made at roughly the same time, small industry, etc.), while others won't (large cast, belonging to multiple conflicting genres, etc.). We will got though a couple of example iterations, showing the "normal" output, and the more detailed SQL queries and responses (see Section E on page 43).

First we will show a "normal" interaction where the user enters the misspelled title: Copid rides the range. Normal interactions are listed (see Table 6), and then detailed ones are listed (see Table 7 on page 13). Normal interactions are listed (see Table 8 on page 17) for "The Dark Knight" identified as the intended movie when "The Dirk Knight" (the default movie title) is entered.

Table 6: Normal output for "Copid rides the range"

Line	Output	Explanation
1	You entered 'Copid rides the range' and we chose 'Cupid Rides the Range' as our best match. Out of: 1. Cupid Rides the Range	A repeat of what the user entered to provide feedback.
2	'Cupid Rides the Range' belongs to 3 genres including: 1. Music 2. Western 3. Short	The genres that the movie was classified as part of.
3	These 10 actors appeared in 'Cupid Rides the Range': Phelps, Norman ; Strange, Glenn (I) ; McKenzie, Robert (I) Worden, Hank ; Ros, Elvira ; Phelps, Willie Whitley, Ray ; Card, Ken ; Phelps Brothers, The Phelps, Earl ;	A list of actors and actresses.
4	We will pick 2 of them.	Actors are selected at random and may vary between runs.

(Continued on the next page.)

Table 6. (Continued from the previous page.)

Line	Output	Explanation
	We have selected these actors: Phelps, Norman Phelps, Willie	
5	There are 21 recommendations. They are (in no particular order): 1. Border G-Man 2. Prairie Papas 3. Ranch House Romeo 4. A Buckaroo Broadcast 5. Bandits and Ballads 6. Trouble in Texas 7. Molly Cures a Cowboy 8. Rawhide 9. Sagebrush Serenade 10. Painted Desert 11. The Renegade Ranger 12. Trouble in Sundown 13. Gun Law 14. Rhythm Wranglers 15. Six-Gun Gold 16. A Western Welcome 17. Hittin' the Trail 18. Where the West Begins 19. Cactus Capers 20. Tex Rides with the Boy Scouts 21. The Utah Trail	Recommendations were found. Sometimes there won't be any.

Table 7: Detailed output for “Copid rides the range”

Line	Output	Explanation
1	\connect chuck; select distinct title, titlebase64, levenshtein(title, 'Copid rides the range') from movies where levenshtein (title, 'Copid rides the range') < 4 order by levenshtein limit 10 ; \q	Commands sent to the database.
2	You are now connected to database 'chuck' as user 'chuck'. title — titlebase64 — levenshtein ————————+————————————————————+————- Cupid Rides the Range — Q3VwaWQgUmlkZXMgdGhlIFJhbmdl — 3 (1 row)	Response from the database.
3	You entered 'Copid rides the range' and we chose 'Cupid Rides the Range' as our best match. Out of: 1. Cupid Rides the Range	Feedback to the user.
4	\connect chuck; select distinct genre from genre where titlebase64 = 'Q3VwaWQgUmlkZXMgdGhlIFJhbmdl';	Commands sent to the database.
5	You are now connected to database 'chuck' as user 'chuck'. genre ——— Music Western Short (3 rows)	Response from the database.

(Continued on the next page.)

Table 7. (Continued from the previous page.)

Line	Output	Explanation
6	'Cupid Rides the Range' belongs to 3 genres including: 1. Music 2. Western 3. Short	Feedback to the user.
7	\connect chuck select distinct actor from actors where titlebase64 = 'Q3VwaWQgUmlkZXMgdGhlIFJhbmdl'; \q	Commands sent to the database.
8	You are now connected to database 'chuck' as user 'chuck'. actor ————————- Phelps, Norman Strange, Glenn (I) McKenzie, Robert (I) Worden, Hank Ros, Elvira Phelps, Willie Whitley, Ray Card, Ken Phelps Brothers, The Phelps, Earl (10 rows)	Response from the database.
9	These 10 actors appeared in 'Cupid Rides the Range': Phelps, Norman Strange, Glenn (I) McKenzie, Robert (I) Worden, Hank Ros, Elvira Phelps, Willie Whitley, Ray Card, Ken Phelps Brothers, The	Feedback to the user.

(Continued on the next page.)

Table 7. (Continued from the previous page.)

Line	Output	Explanation
	Phelps, Earl We will pick 2 of them. We have selected these actors: Phelps, Norman Phelps, Willie	
10	\connect chuck select distinct titlebase64 from genre where titlebase64 in (select distinct a1.titlebase64 from actors as a1 join actors as a2 on a1.titlebase64 = a2.titlebase64 and a2.actor = 'Phelps, Willie' and a1.actor ='Phelps, Norman'); \q	Commands sent to the database.
11	You are now connected to database 'chuck' as user 'chuck'. titlebase64 ——————————— Qm9yZGVyIEctTWFu Q3VwaWQgUmlkZXMgdGhlIFJhbmdl UHJhaXJpZSBQYXBhcw== UmFuY2ggSG91c2UgUm9tZW8= QSBCdWNrYXJvbyBCcm9hZGNhc3Q= QmFuZGl0cyBhbmQgQmFsbGFkcw== VHJvdWJsZSBpbiBUZXhhcw== TW9sbHkgQ3VyZXMgYSBDb3dib3k= UmF3aGlkZQ== U2FnZWJydXNoIFNlcmVuYWRl UGFpbnRlZCBEZXNlcnQ= VGhlIFJlbmVnYWRlIFJhbmdlcg== VHJvdWJsZSBpbiBTdW5kb3du R3VuIExhdw== Umh5dGhtIFdyYW5nbGVycw==	Response from the database.

(Continued on the next page.)

Table 7. (Continued from the previous page.)

Line	Output	Explanation
	U2l4LUd1biBHb2xk	
	QSBXZXN0ZXJuIFdlbGNvbWU=	
	SGl0dGluJyB0aGUgVHJhaWw=	
	V2hlcmUgdGhlIFdlc3QgQmVnaW5z	
	Q2FjdHVzIENhcGVycw==	
	VGV4IFJpZGVzIHdpdGggdGhlIEJveSBTY291dHM=	
	VGhlIFV0YWggVHJhaWw=	
	(22 rows)	
	There are 21 recommendations.	
	They are (in no particular order):	
	1. Border G-Man	
	2. Prairie Papas	
	3. Ranch House Romeo	
	4. A Buckaroo Broadcast	
	5. Bandits and Ballads	
	6. Trouble in Texas	
	7. Molly Cures a Cowboy	
	8. Rawhide	
	9. Sagebrush Serenade	
12	10. Painted Desert	Final data presented to the user.
	11. The Renegade Ranger	
	12. Trouble in Sundown	
	13. Gun Law	
	14. Rhythm Wranglers	
	15. Six-Gun Gold	
	16. A Western Welcome	
	17. Hittin' the Trail	
	18. Where the West Begins	
	19. Cactus Capers	
	20. Tex Rides with the Boy Scouts	

(Continued on the next page.)

Table 7. (Continued from the previous page.)

Line	Output	Explanation
	21. The Utah Trail	

(Last page.)

Table 8: Normal output for "The Dark Knight"

Line	Output	Explanation
1	You entered 'The Dirk Knight' and we chose 'The Dark Knight' as our best match. Out of: 1. The Dark Knight 2. The Dim Knight 3. The Dark Light	Feedback to the user.
2	'The Dark Knight' belongs to 5 genres including: 1. Action 2. Fantasy 3. Short 4. Drama 5. Crime	Feedback to the user.
3	These 261 actors appeared in 'The Dark Knight': Lesley, David White, Michael Jai List of actors redacted (see Section C on page 34) Ritz, Sara Campbell, Tommy Pierce, Ernest Malkin, Simon Webb, Bronson Hoving, John McAllister, Lisa (I) Chin Han List of actors redacted (see Section C on page 34)	Feedback to the user.

(Continued on the next page.)

Table 8. (Continued from the previous page.)

Line	Output	Explanation
	Kalesperis, Adam Shah, Vivek (II)	
4	We will pick 2 of them. We have selected these actors: Campbell, Tommy Chin Han	Feedback to the user.
5	Sorry no recommendations possible.	Feedback to the user.

4 Conclusion

We created a movie recommendation system that mimics recommendation systems in that we encounter all over the Internet. We demonstrated how the system would respond when given misspelled movie titles. Recommendations were based on having at least two actors from the original movie in another movie that was in same genres as the original. In some cases, recommendations were found, and in others none were found.

One misspelled movie title, "Copid rides the range" (see Figure 3 on the next page) resulted in 21 recommendations. Another misspelled movie title "Dirk Knight" (see Figure 4 on page 21) resulted in 0 recommendations.

The proof of concept recommendation system worked. It is not a production class system, and could be improved by:

1. Selecting the common actors based on their placement in the credits, vice random selection.

2. Supporting the full Unicode character set, vice only ASCII.

3. Incorporation the user's past genre history when limiting searches, vice using the complete set of genre assignments from the original movie.

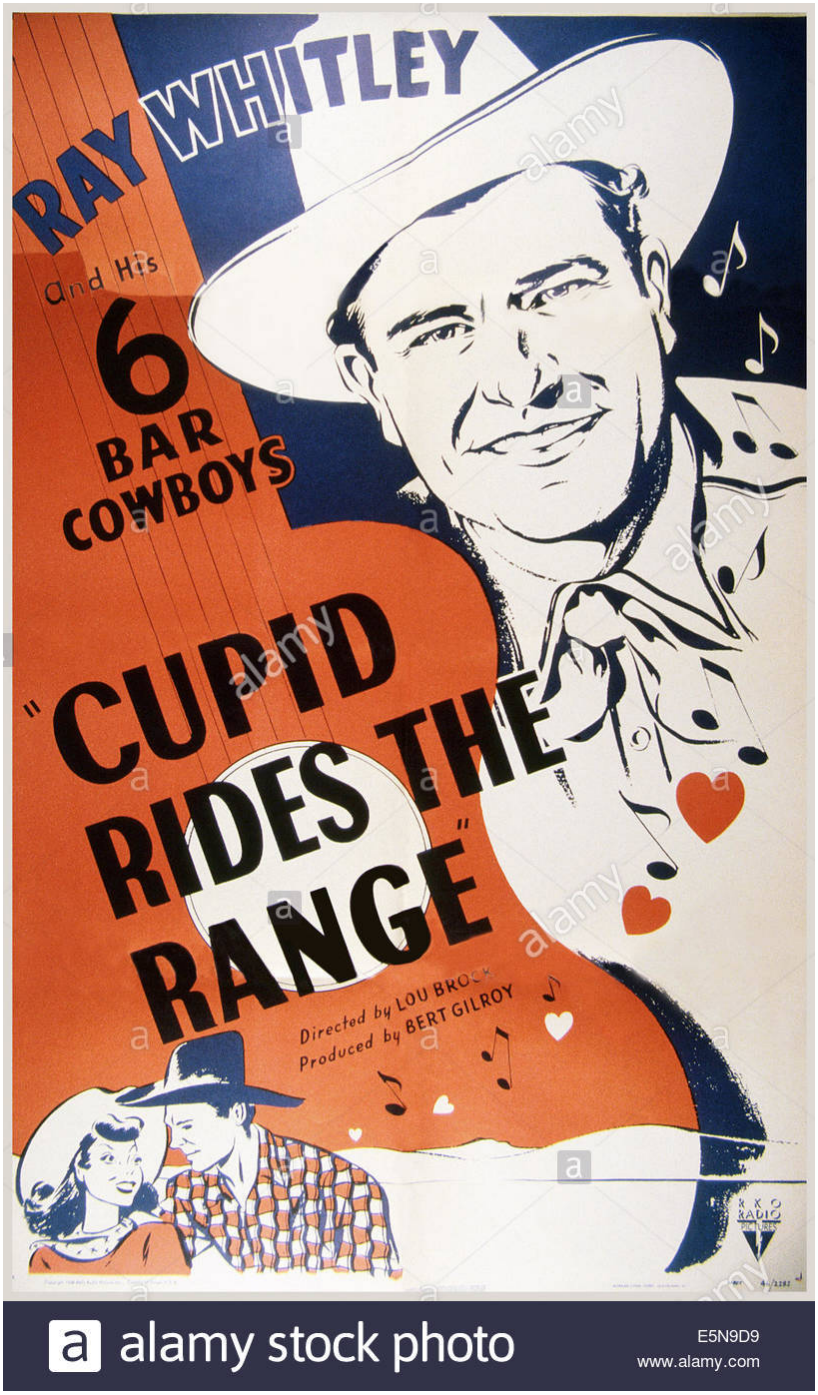


Figure 3: Movie poster for "Cupid Rides the Range" Movie staring Ray Whitley (1939).Image from [7]

Figure 4: Movie poster for "Dark Knight" Movie starring Christian Bale, and Heath Ledger (2008). Image from [8].

A A PostgreSQL timeline

PostgreSQL is old[9].

- Started at UC Berkley as POSTGRES in 1986.
- DARPA and Army Research Office (ARO) project
- Demoed in 1987, presented in 1988, released in 1989
- Evolved into Postgres95
- Name changed in 1985 to PostgreSQL to reflect origin and new SQL capability
- Open source software available 1 Aug. 1996 [10]
- Online presence at PostgreSQL.org since 22 Oct. 1996
- Under active development, major releases approx. yearly
- Official name PostgreSQL, nicknamed Postgres
- Postgres is Post(In)gres (yet another SQL database)[11]

An old and on going project. PostgreSQL is very widely available.

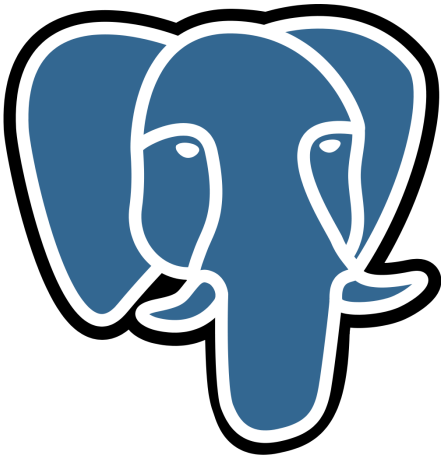


Figure 5: PostgreSQL Elephant.

B Venn diagrams

"A Venn diagram (also known as a set diagram or logic diagram) is a diagram that shows all possible logical relations between a finite collection of different sets."

Wikipedia Staff [13]

Venn diagrams are used to visualize how different sets of data intersect (or don't intersect). We'll use three different sets of data A, B, and U for our general discussion before delving into questions about our proof of concept genre database.

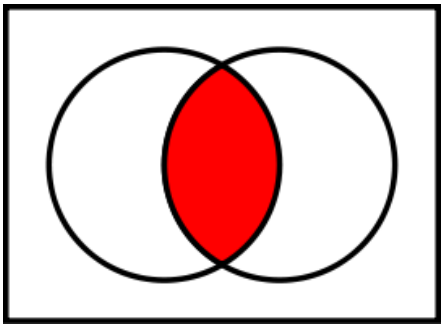
A and B represent to sets of data, that have some members in their respective sets in common. These sets could be animals, vegetables, minerals, numbers, books, or anything that we want. U is the universe of all things, including all of A and all of B. Using these three sets (see Figure 6 on the next page), we can say somethings about them.

A series of Venn diagrams were created to visualize the question of how many movies were in the data sets Biography and Documentary and Crime. The Western and Adventure genres were added to the data sets of interest in order to show how complex Venn diagrams can get with only a small number of sets.

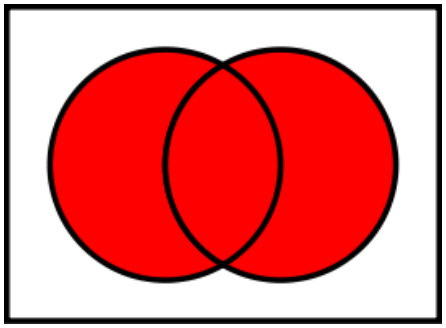
Monochromatic Venn diagrams can convey the overlap of the data sets (genres) (see Figures 7 on page 25 through 11 on page 29), but there are times when a polychromatic presentation helps (see Figure 12 on page 30).

The number of ways of picking unordered subsets from a larger set is known as the binomial coefficient. If the larger set contains the elements $Biography, Crime, Documentary, Adventure$ and we were interested in counting the number of subsets that had exactly 2 elements, then the subsets would be:

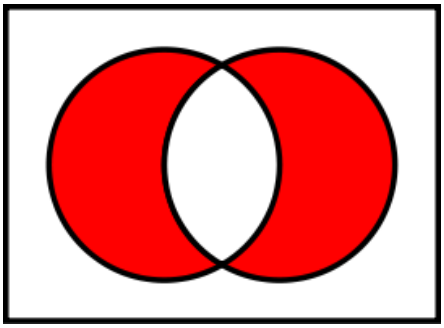
1. $Biography, Crime$
2. $Biography, Documentary$
3. $Biography, Adventure$
4. $Crime, Documentary$
5. $Crime, Adventure$
6. $Documentary, Adventure$



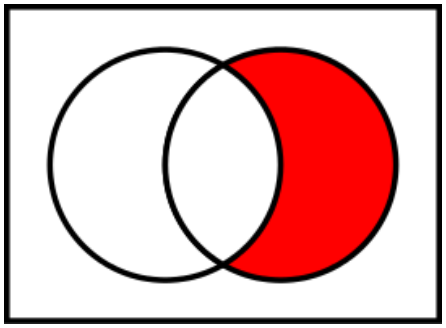
(a) Intersection of two sets: $A \cap B$



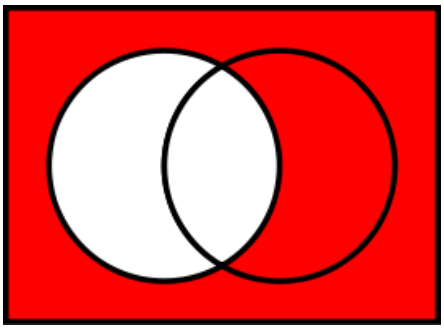
(b) Union of two sets: $A \cup B$



(c) Symmetric difference of two sets: $A \Delta B$



(d) Relative complement of A (left) in B (right): $A^c \cap B = B \backslash A$



(e) Absolute complement of A in U: $A^c = U \backslash A$

Figure 6: Representative Venn diagrams. The corresponding mathematical notation is in the caption of each subfigure.

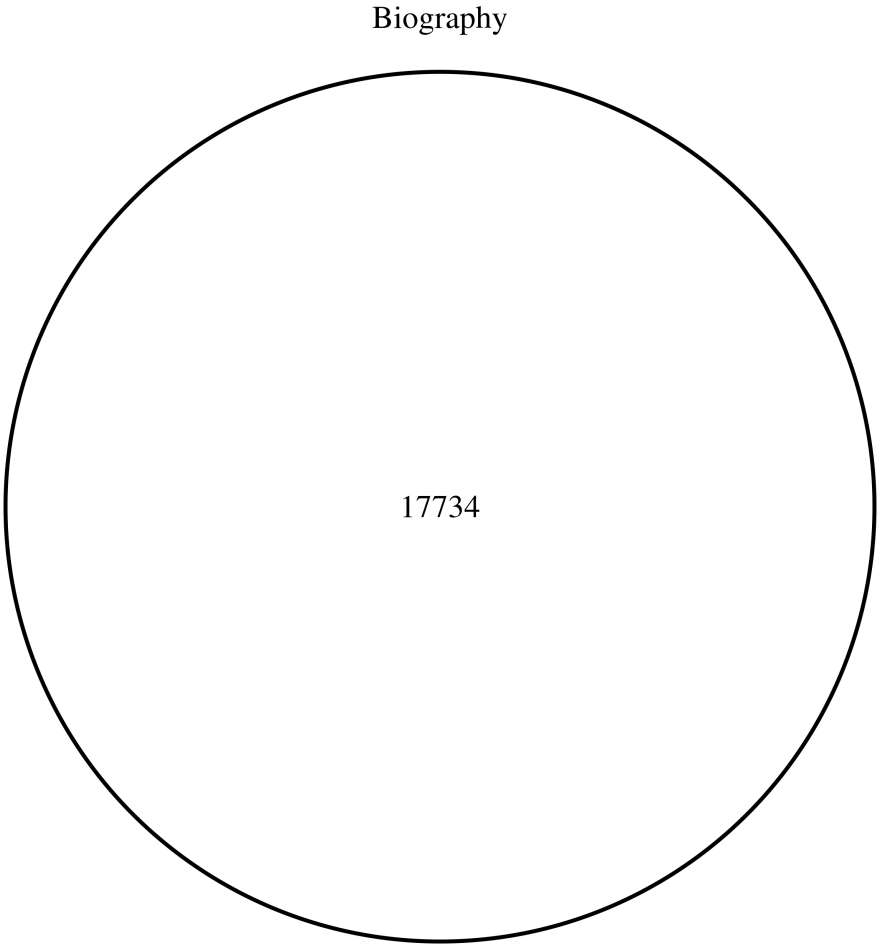


Figure 7: Venn diagram of the Biography genre.

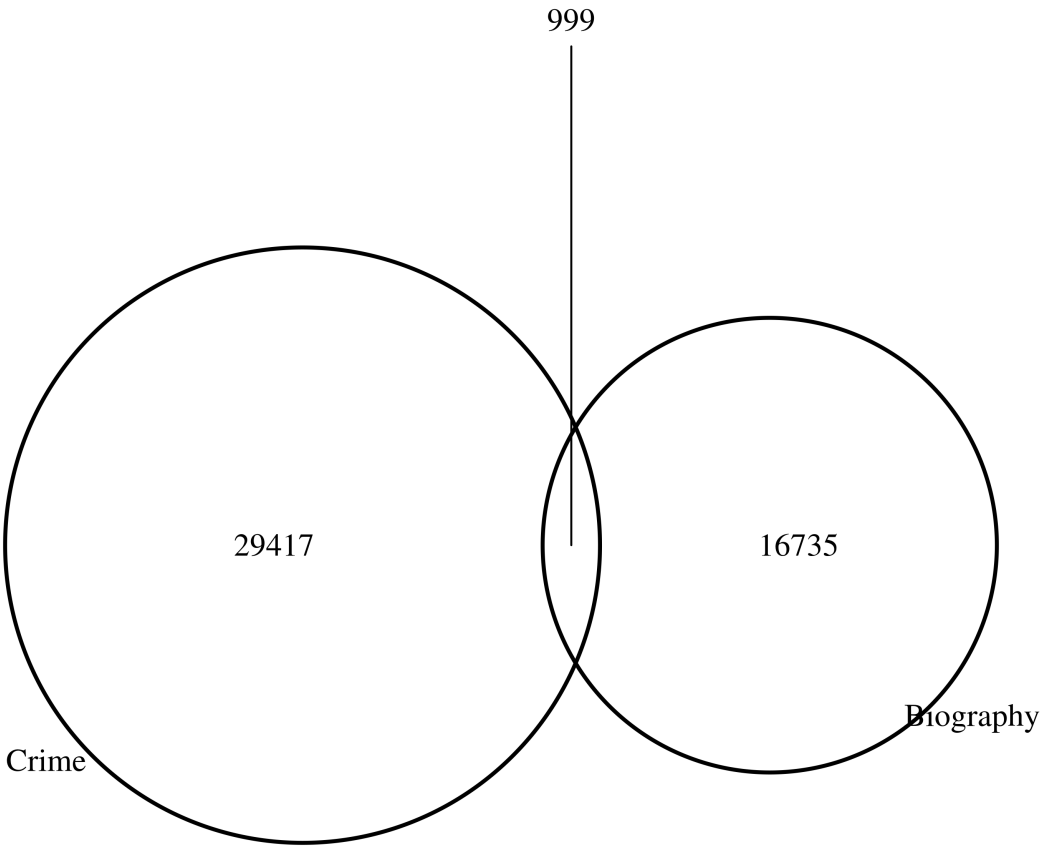


Figure 8: Venn diagram of the Biography and Crime genres ($Biography \cup Crime$). Showing there are 999 common movies in both genres.

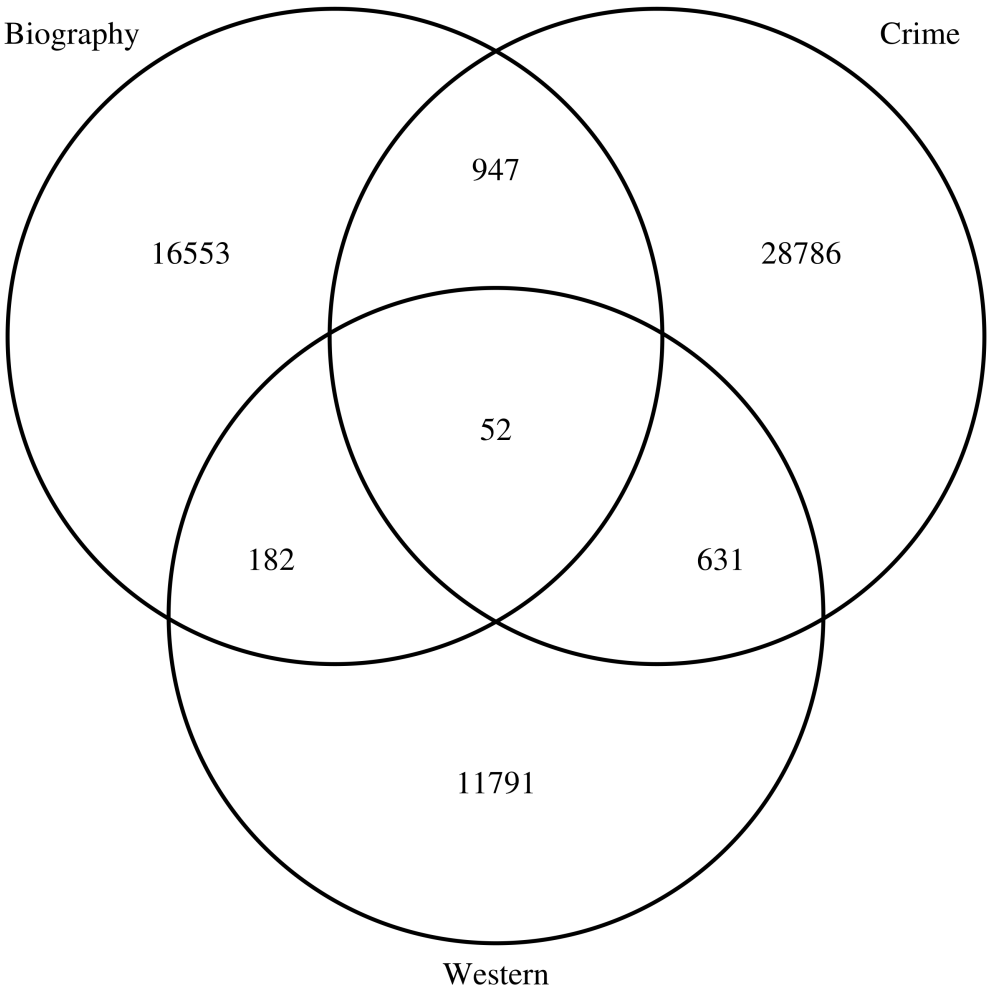


Figure 9: Venn diagram of the Biography and Crime Western genres ($Biography \cup Crime \cup Western$). Showing there are 52 common movies in all genres.

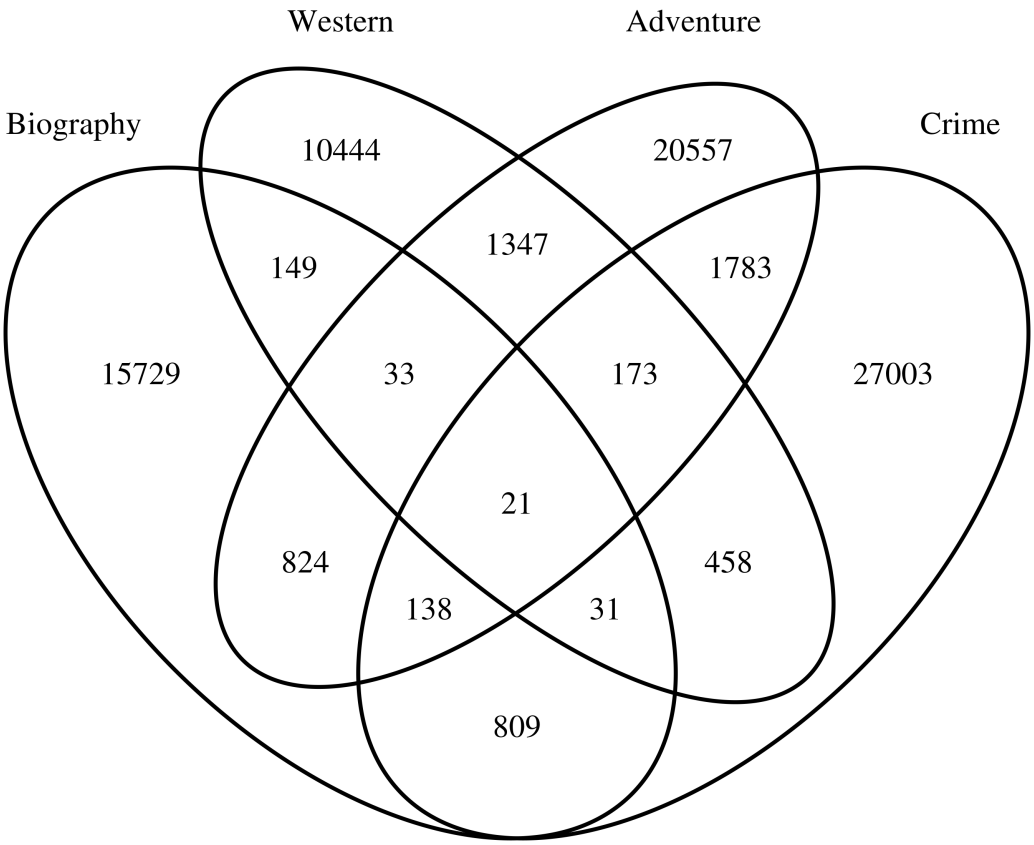


Figure 10: Venn diagram of the Biography and Crime Western and Adventure genres ($Biography \cup Crime \cup Western \cup Adventure$). Showing there are 21 common movies in all genres.

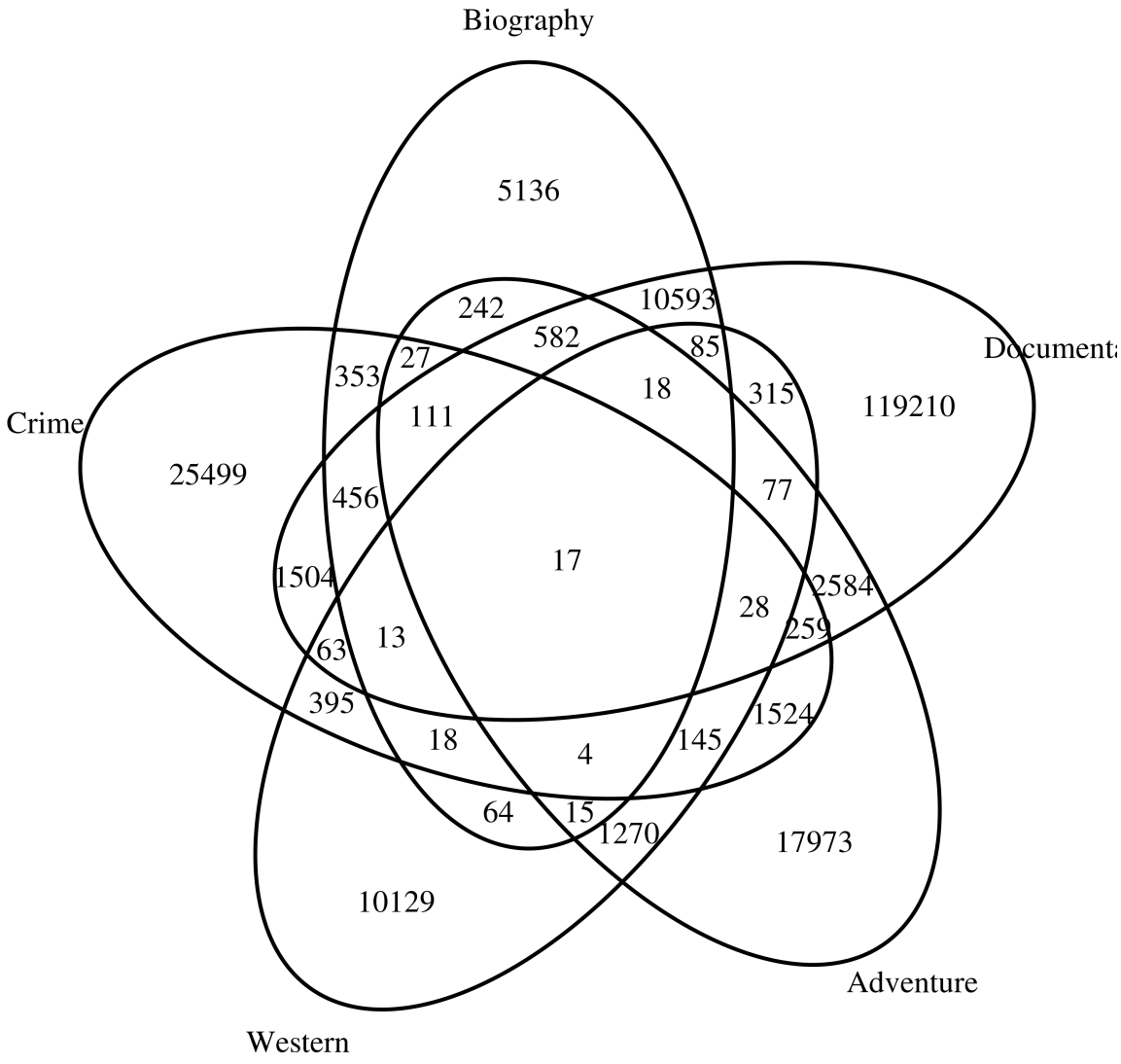


Figure 11: Venn diagram of the Biography and Crime Western and Adventure and Documentary genres ($Biography \cup Crime \cup Western \cup Adventure \cup Documentary$). Showing there are 17 common movies in all genres.

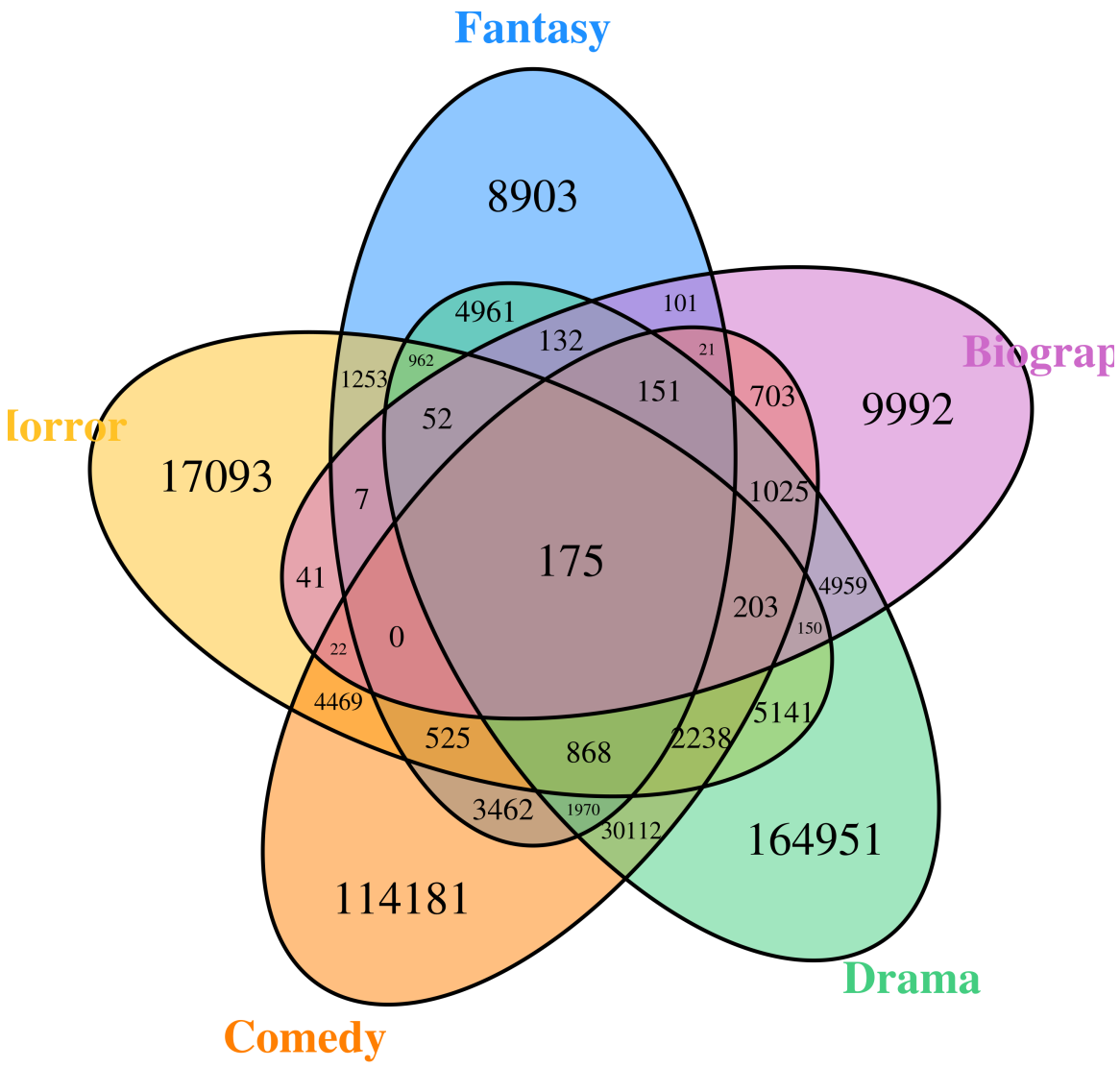


Figure 12: Colorful Venn diagram of Horror and Fantasy and Biography and Drama and Comedy. It is interesting to see that there are 0 movies common to all genres. This diagram was made possible by modifying the code from `http://www.inside-r.org/packages/cran/VennDiagram/docs/venn.diagram`. The modified code is attached to this report.

If we vary the number of elements in the subsets from 0 to 4 (because our example has 4 elements in the complete set), we end up with an interesting pattern of number of subsets based on the desired size of the subset (see Table 9 on the next page).

If we say that n is the number of elements in the larger set, and k is the number of elements in the subset, then we can say $n$choose$k$ which is often written $\binom{n}{k}$ (the binomial coefficient) which expands to $\frac{n!}{k!*(n-k)!}$.

By looking at the pattern of numbers in the **Count** column (see Table 9 on the following page), and varying k (all the while keeping n fixed and larger than k), we start to see a pattern. This pattern is known as Pascal's Triangle. Pascal's Triangle

> *"[T]he triangular array of integers, with 1 at the apex, in which each number is the sum of the two numbers above it in the preceding row . . .*
>
> Borowski and Borwein [1]

A Pascal Triangle for $n = 5$ has been created (see Table 10 on page 33). While the coefficients are patently obvious afterwards, it is helpful to see they represent (see Table 11 on page 33). A slightly different presentation of Pascal's Triangle makes it easy to understand and compute the number of coefficients[4] (see Table 12 on page 33).

Looking at enough of these cases, and counting the number of subsets for each $n$choose$k$ case where we vary k from 0 to n, we can arrive at the final formula:

$$\begin{aligned} subsets &= \sum_{k=0}^{k} \binom{n}{k} \\ &= \binom{n}{0} + \binom{n}{1} + \dots \binom{n}{n} \\ &= \frac{n!}{0! * (n-0)!} + \frac{n!}{1! * (n-1)!} + \dots \frac{n!}{n! * (n-n)!} \\ &= 2^n \end{aligned}$$

[4] `www.cs.columbia.edu/~cs4205/files/CM4.pdf`

Table 9: Choosing different genre combinations.

Choose	**Chosen**	**Subsets**
0	1. (None.)	1
1	1. *Biography* 2. *Crime* 3. *Documentary* 4. *Adventure*	4
2	1. *Biography*, *Crime* 2. *Biography*, *Documentary* 3. *Biography*, *Adventure* 4. *Crime*, *Documentary* 5. *Crime*, *Adventure* 6. *Documentary*, *Adventure*	6
3	1. *Biography*, *Crime*, *Documentary* 2. *Biography*, *Crime*, *Adventure* 3. *Biography*, *Documentary*, *Adventure* 4. *Crime*, *Documentary*, *Adventure*	4
4	1. *Biography*, *Crime*, *Documentary*, *Adventure*	1
	Total	**16**

Table 10: Pascal's Triangle up to $n = 5$. Each coefficient is the sum of the two above to the left and to the right (assuming that blanks are treated as 0s). Python source code to create table from `http://www.bedroomlan.org/coding/pascals-triangle-latex`. Python code is attached to this report.

$n = 0$:						1					
$n = 1$:					1		1				
$n = 2$:				1		2		1			
$n = 3$:			1		3		3		1		
$n = 4$:		1		4		6		4		1	
$n = 5$:	1		5		10		10		5		1

Table 11: Pascal's Triangle up to $n = 5$ as binomials.

$n = 0$:						$\binom{0}{0}$					
$n = 1$:					$\binom{1}{0}$		$\binom{2}{0}$				
$n = 2$:				$\binom{2}{0}$		$\binom{2}{1}$		$\binom{2}{2}$			
$n = 3$:			$\binom{3}{0}$		$\binom{3}{1}$		$\binom{3}{2}$		$\binom{2}{3}$		
$n = 4$:		$\binom{4}{0}$		$\binom{4}{1}$		$\binom{4}{2}$		$\binom{4}{3}$		$\binom{4}{4}$	
$n = 5$:	$\binom{5}{0}$		$\binom{5}{1}$		$\binom{5}{2}$		$\binom{5}{3}$		$\binom{5}{4}$		$\binom{5}{5}$

Table 12: Pascal's Triangle up to $n = 5$ reformatted coefficients.

n	$\binom{n}{0}$	$\binom{n}{1}$	$\binom{n}{2}$	$\binom{n}{3}$	$\binom{n}{4}$	$\binom{n}{5}$	Σ
0	1						1
1	1	1					2
2	1	2	1				3
3	1	3	3	1			8
4	1	4	6	4	1		16
5	1	5	10	10	5	1	32

C "The Dark Knight" actors

Actors and actresses can have the same name as other actors and actresses. The IMDb resolves these by appending a Roman numeral in parentheses after the name.

These 261 actors (in alphabetical order) appeared in 'The Dark Knight':

1. Ajala, David
2. Albertson, Jeff
3. Aldaz, Tracy L.
4. Allen, Matthew W.
5. Anderson, Christopher James
6. Armourae, Stephen
7. Armstrong, William (I)
8. Azpeitia, Alisa
9. Bach, Mike (I)
10. Bakare, Ariyon
11. Baker, Wayne (I)
12. Bale, Christian
13. Ballantyne, Martin (I)
14. Ballard, Doug
15. Barcelos, Alberto
16. Bartlett, Tommy (I)
17. Beam, Greg
18. Bennett, Blayne
19. Berard, Ryan
20. Bicknell, Andrew
21. Birchard, Paul (I)
22. Borchardt, Amanda (II)
23. Brindowski, Adam
24. Brody, Jon Lee
25. Brooke, Peter (I)
26. Bucholz, Adam
27. Bulcock, Philip
28. Burns, Debbi
29. Caballero, Joseph Luis
30. Cabrera, Maritza
31. Caine, Michael (I)
32. Caiola, Shirin
33. Calmels, Fabrice
34. Campbell, Tommy
35. Carbonell, Nestor
36. Carrington, Nigel
37. Chadwick, David (II)
38. Chapman, Josh (I)
39. Chen, Edison (I)
40. Chernicky, Laura
41. Chin Han
42. Cho, Matt (I)
43. Chu, Henry Milton
44. Clear, Patrick
45. Clevenger, Kelli
46. Cosey, David
47. Coster, Ritchie
48. Crane, Nancy
49. Crocker, Martin
50. Curnen, Monique Gabriela
51. Dastmalchian, David
52. Daugherty, Rachel
53. Dawson, Bruce Allen
54. Day, Danielle (II)
55. Dean, Ron (I)
56. DeFaria, Peter
57. Derence, Sam
58. Dillane, Richard
59. di Makena, Matteo
60. Divizio, Richard

61. Domino, Tony
62. Doyle, Jessica (V)
63. Dunn, Sarah Jayne
64. Eckhart, Aaron
65. Edwards, Grahame (I)
66. Edwards, Laine
67. Egan, R. Michael
68. Elliott, David William James
69. Ellis, Winston
70. Farb, Aaron
71. Farruggio, James
72. Feore, Aidan
73. Fichtner, William
74. Fierro, James (I)
75. Fojtik, Gene
76. Foster, Michael Corey
77. Foster, Reese
78. Fowler, Jim (IV)
79. Frederick, Jason (VI)
80. Freeman, K. Todd
81. Freeman, Morgan (I)
82. Fuller, Jason
83. Fulsher, Darren Elliot
84. Fultz, David
85. Gaitsch, Thomas
86. Gamble, Nathan
87. Ganyo, Scott
88. Gayle, Lorna
89. Geer, Daniel Richard
90. Gerard, Bill (I)
91. Giraud, Marisol
92. Glanfield, Tim
93. Goldring, Danny
94. Gordon, Tom (XII)
95. Gorman, Michael Andrew
96. Gossen, Dan
97. Griffin, Gordon (I)
98. Grover, Sharlene
99. Gull, R. Michael (I)
100. Gunn, Hannah
101. Gyllenhaal, Maggie
102. Hallam, Natalie
103. Hall, Anthony Michael
104. Harper, Terrell
105. Hartley, David (II)
106. Hartmann, Thomas (VII)
107. Harto, Joshua
108. Hathaway, Alexander
109. Healy, Brendan P.
110. Heaney, Craig
111. Hellman, Erik
112. Hepple, Victoria
113. Hinshelwood, Sophia
114. Hodges, Jordon
115. Hoving, John
116. Huebsch, Matthew
117. Ibrahim, Bill
118. Jamroz, Gerard
119. Jay, Erron
120. Jefferson, Daniel
121. Jimenez, Ramses
122. Jones, Will (VI)
123. Kalesperis, Adam
124. Kaliebe, Bob
125. Katt, Nicky
126. Keiser, Mark (I)
127. Kierscht, Charlie
128. Knox, Jennifer
129. Kolasa, Keryaki
130. Kosik, Thomas
131. Kress, Don
132. Kross, Ryan
133. Krueger, Tim (I)

134. Kupferer, Keith
135. Kuster, Michael
136. Lambdin, Brandon
137. Latham, Dan (II)
138. Lazicki, Joseph
139. Leahy, Patrick (I)
140. Ledger, Heath
141. Lee, Carl (V)
142. Leitch, Matthew
143. Lesley, David
144. Lewis, Walter (I)
145. Lineham, Emily
146. Lister, Tommy 'Tiny'
147. Lopez, Debra (II)
148. Lovegren, Ivan
149. Lovejoy, Lateef
150. Luther, Andy (II)
151. Luther, Scott
152. Lutz, Lanny
153. Lynn, Deborah (II)
154. Lynn, Noelle (I)
155. Macchi, Jonathan
156. Makepeace, Richard
157. Malkin, Simon
158. Marchesi, Al
159. Marshall, Peter (V)
160. Marston, Rob
161. Martino, J.R.
162. Mawell, Xander
163. Mazurk, Joseph
164. McAllister, Lisa (I)
165. McComas, Tom (I)
166. McElroy, Tom (I)
167. McEnany, Krista
168. McFarlane, Colin (I)
169. McGonagle, Ryan (I)
170. McGraw, Melinda
171. Mellor, James (II)
172. Monk, Roger (II)
173. Mosley, Gertrude
174. Murphy, Cillian
175. Nadolski, David J.
176. Nelson, Joseph (XII)
177. Nicoli, Vincenzo
178. Olawumi, Olumiji
179. Oldman, Gary
180. Oliveira, Joseph (III)
181. O'Neill, Matthew (I)
182. Ozbay, Sal
183. Patti, Chelsey
184. Pearson, Adam (II)
185. Pedersen, Libby (II)
186. Petschler, Chris
187. Pierce, Ernest
188. Pirie, Ian
189. Plante, Rory
190. Poe, Matthew (II)
191. Query, Charles
192. Radz, Marc
193. Rapkin, Brian
194. Rasque, Mitch
195. Reeves, Buster
196. Riotta, Vincent
197. Rippy, Matt
198. Ritz, Sara
199. Rivera, Dale
200. Rnic, Peter
201. Roberts, Eric (I)
202. Rollins, Joshua (I)
203. Rosen, Beatrice
204. Rowell, Harry
205. Ryder, Gary (II)

206. Ryland, Jonathan
207. Saindon, Kelly
208. Satcher, Daryl
209. Scales, James
210. Schleef, Elisa
211. Schweiner, Greg
212. Seybold, Jan
213. Shah, Amit (III)
214. Shah, Vivek (II)
215. Shalhoub, Amanda
216. Shallenberger, Matt
217. Sharp, Colin
218. Shields, Michelle (I)
219. Smillie, William
220. Smirnova, Sofiya
221. Snowden, John (I)
222. Solabarrieta, Lorea
223. Sora, Dwight
224. Spielbauer, Bruce
225. Stern, January
226. Stern, Robert Patrick
227. Stevens, Willie
228. Stewart, Douglas (VII)
229. Stone, Jordan (IV)
230. Stone, Robert (IX)
231. Stoyanov, Michael (I)
232. Strobel, Richard (I)
233. Summers, Ronan
234. Szarabajka, Keith
235. Tait, Sheila
236. Tait, Tristan (I)
237. Terracina, Nydia Rodriguez
238. Thomas, Chris D.
239. Thurner, John
240. Townsend, Tom (II)
241. Tsou, Chuen
242. Turk, John (I)
243. Tyrell, Michael (II)
244. Ukena, Matt
245. Venn, Charles
246. Venus, Alisa
247. Vieau, Michael
248. Walsh, Kyle (II)
249. Warfield, James
250. Warman, John
251. Webb, Bronson
252. White, Michael Jai
253. Whyte, Mike
254. Williams, Erik A.
255. Wilson, Chris (XXIX)
256. Wilson, Helene
257. Wolf, Lisa (IV)
258. Wong, Wai (III)
259. Zahir, Essa
260. Zahrn, Will
261. Zaideman, Kevin

D Explaining SQL database concepts using Excel

The world of structured query language (SQL) can be and sound very confusing, but you've probably been using most of SQL's concepts and ideas for a while, without realizing it. I will attempt to connect SQL ideas of tables, columns, rows, and selects using Excel. I'll use a table and a few figures to help make the connection (see Table 13).

Table 13: Cross referencing Excel terms to PostgreSQL terms.

Excel term	**PostgreSQL**
Workbook	Database
Worksheet	Table
Row	Row
Column	Column
	Variable type. All entries in a column must be of the same type (characters, numbers, undefined, etc.).
Filter	Select. SQL is built around the idea of a SELECT statement. Excel's filter command is not nearly as powerful.
1,048,576 rows	Unlimited
16,384 cols	250-1600 depending on column types

Our PostgreSQL database has three tables (see Figures 13 on the next page through 15 on page 41). Excel provides a way to filter data on a worksheet (see Figure 16 on page 42). The Excel filter command works on one worksheet, not across multiple worksheets. SQL support SELECT command on one, two, or as many tables as are available.

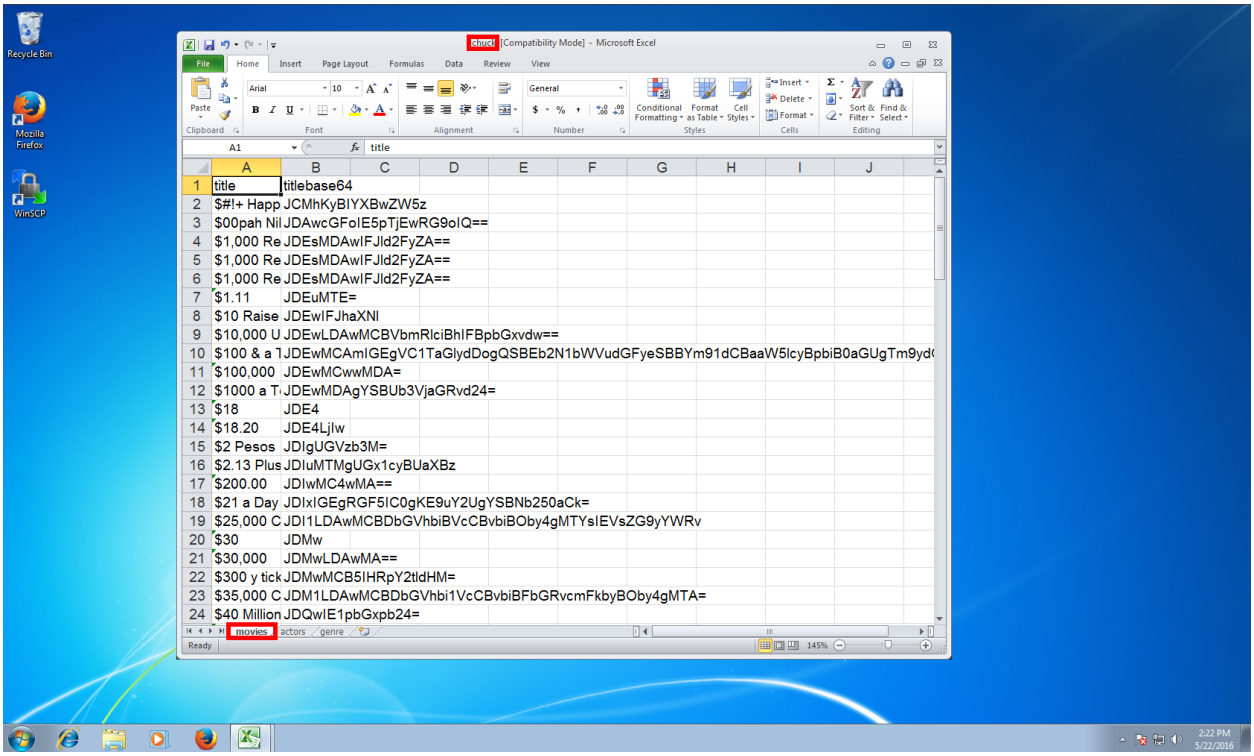


Figure 13: An Excel workbook with PostgreSQL data (movies worksheet). The workbook/database name and worksheet/table are outlined in red.

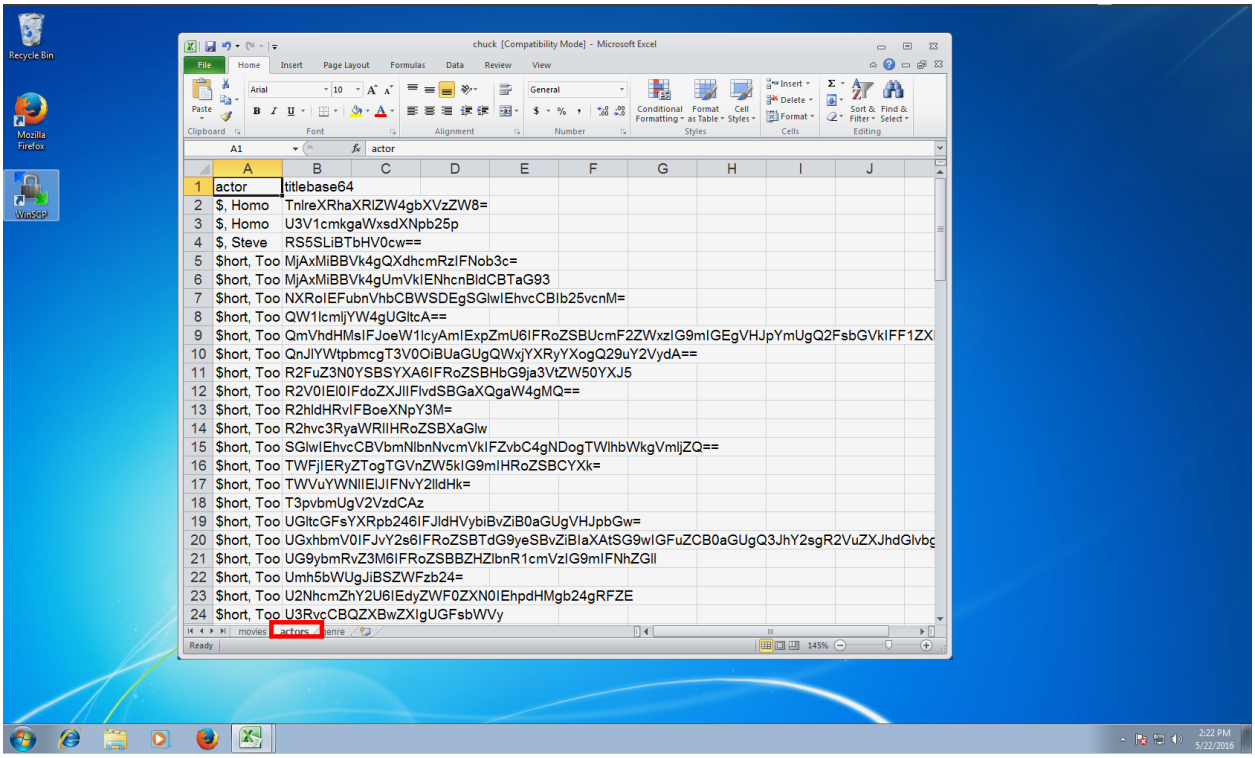


Figure 14: An Excel workbook with PostgreSQL data (actors worksheet). The worksheet/table are outlined in red.

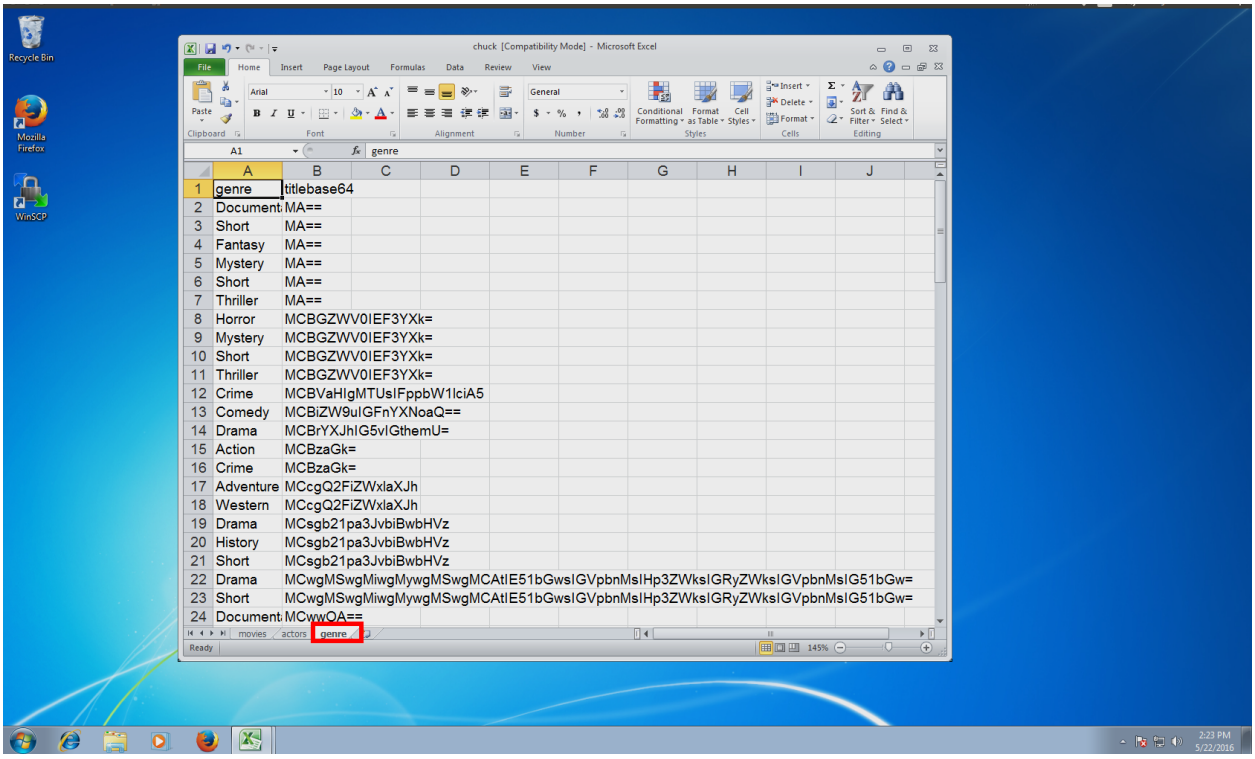


Figure 15: An Excel workbook with PostgreSQL data (genre worksheet). The worksheet/table are outlined in red.

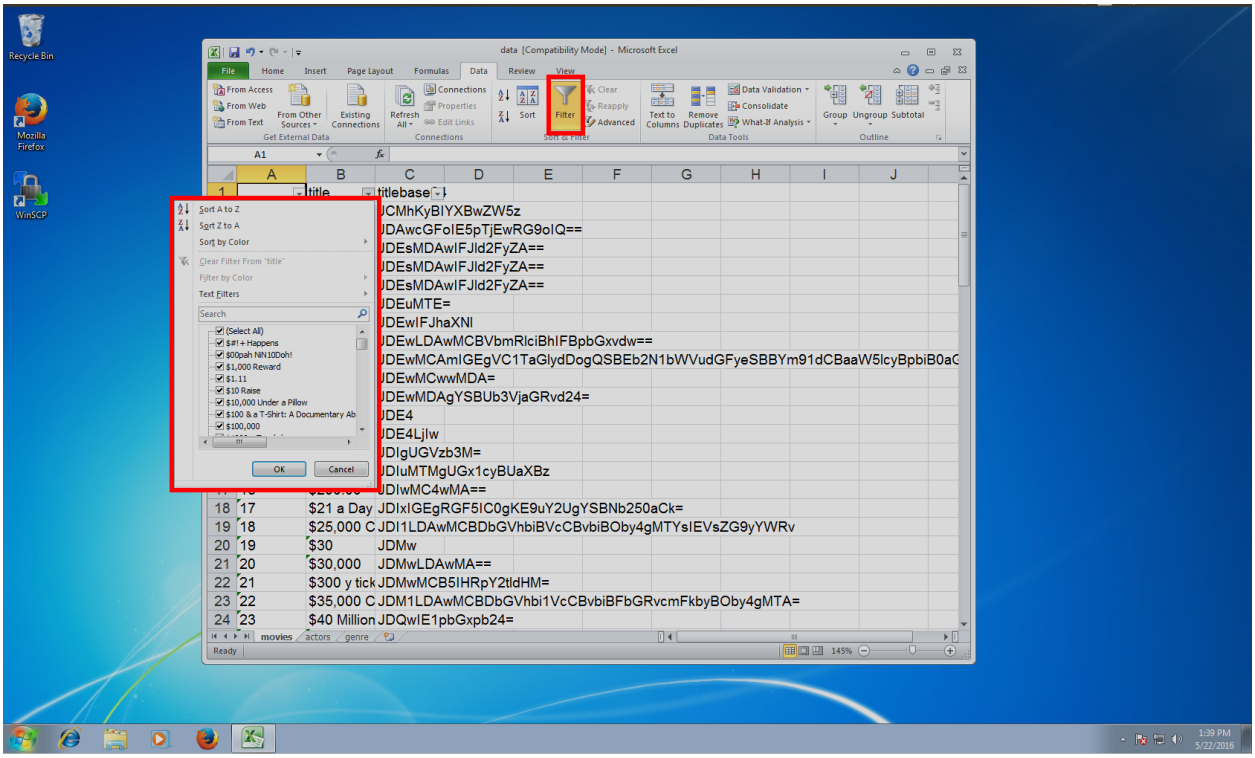


Figure 16: An Excel worksheet showing a the "Filter" interface.

E Misc. files

The files used to create all these figures are attached to this report. They are:

1. chuck.xls - an Excel spreadsheet with 100 entries from the postgress movies, actors, and genre tables.
2. demo.R - an R program used to discover and make recommendations using the Internet Movie Database[5]. The PostgreSQL database used by demo.R can be created and populated by setting the appropriate command line arguments.
3. pascalTriangle.py - a Python program to generate Pascal's Triangle suitable for use in a LaTeXfile.
4. vennDiagrams.R - an R program used to create Venn diagrams based on the genres.
5. writeExcel.R - an R program used to extract data from the database and make it available for Excel.

The demo.R supports a small set of command line arguments (see Table 14).

Table 14: demo.R command line arguments. Arguments can be entered in any order, but can only be used once per program execution.

Argument	**Usage**	**Default value**
m	The misspelled movie title. If the title is more than one word long then it should be enclosed in quotes (e.g., `-m ''The Terminator''`).	"The Dirk Knight"
p	Print SQL commands and results.	FALSE
r	Reset and restart the PostgreSQL database. This will drop and create all tables, and reload all data from source gzipped files in "well known" locations.	FALSE

(Continued on the next page.)

[5] `http://www.imdb.com/`

Table 14. (Continued from the previous page.)

Argument	Usage	Default value
t	The tex file for collecting data used in automated reports.	(NULL)
u	Select the user for the PostgreSQL database. Combining this option with the `r` option would create a unique database for a user.	chuck

F References

[1] E.J. Borowski and J.M. Borwein, *Web-Linked Dictionary, Mathematics*, Harper Collins, 2005.

[2] Francesco Calabrese, Giusy Di Lorenzo, Liang Liu, and Carlo Ratti, *Estimating origin-destination flows using mobile phone location data*, IEEE Pervasive Computing **10** (2011), no. 4, 0036–44.

[3] David C Edelman, *Four ways to get more value from digital marketing*, McKinsey Quarterly **6** (2010).

[4] Michael Gilleland, *Levenshtein distance, in three flavors*, `http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm`, 2006.

[5] Fabien Girardin, Francesco Calabrese, Filippo Dal Fiore, Carlo Ratti, and Josep Blat, *Digital footprinting: Uncovering tourists with user-generated content*, Pervasive Computing, IEEE **7** (2008), no. 4, 36–43.

[6] Vladimir I Levenshtein, *Binary codes capable of correcting deletions, insertions, and reversals*, Soviet Physics Doklady, vol. 10, 1966, pp. 707–710.

[7] Alamy Staff, *Stock photo - cupid rides the range*, `http://www.alamy.com/stock-photo-cupid-rides-the-range-ray-whitley-1939-72383173.html`, 2016.

[8] IMDb Staff, *The dark knight*, `http://www.imdb.com/title/tt0468569/?ref_=ttmd_ph_tt1`, 2016.

[9] PostgreSQL Staff, *A Brief History of PostgreSQL*, `http://www.postgresql.org/docs/9.0/static/history.html`, 2015.

[10] Wikipedia Staff, *PosgreSQL*, `https://en.wikipedia.org/wiki/PostgreSQL`, 2015.

[11] ———, *Ingres (database)*, `https://en.wikipedia.org/wiki/Ingres_(database)`, 2016.

[12] ———, *Levenshtein distance*, `https://en.wikipedia.org/wiki/Levenshtein_distance`, 2016.

[13] ———, *Venn diagram*, `https://en.wikipedia.org/wiki/Venn_diagram`, 2016.