Prediction of actor collaborations using IMDB data

Vassilis Polychronopoulos   Abhinav Venkateswar Venkataraman

June 9, 2014

Abstract

The Internet Movie Database (IMDB) is a well-known site that contains information on movies, shows and personalities. We aim to build a model that uses IMDB data from the past to predict actor collaborations in the future. We install a relational database derived from the IMDB publicly available data on a PostgreSQL 9.1 server using the IMDBPy script. We form 4 different sets of actors of increasing size by querying the database appropriately. We also extract a multitude of side information for each actor, such as, languages they have used, genres of movies they play in, collaborating directors and producers, ratings of movies and gender among others. For each set of actors we perform link prediction, using the common neighbors score on the actor network as baseline, and the Probabilistic Soft Logic (PSL) modeling language using a set of appropriate rules. We report the area under the receiving operating curve (AUC-ROC) to evaluate the output of both methods. Results show that the use of PSL for our datasets and set of features can enhance the classification results of the baseline.

1 Introduction

Link prediction has emerged as an interesting and challenging problem in the current era of big data. It can be used in a range of domains, including prediction of future events, recovery of incomplete information and thwarting of terrorist threats. For example, an advertising company can enhance its campaign by targeting persons with personalized ads. To do this, they need to establish links between consumers and products where each link represents a possible interest of the consumer for the product. For that, personal data can make the task easier, including demographics and profiles of individual consumers. Another example is link prediction in social networks. Social networking sites can enhance their growth by inferring missing links and suggesting connections to their users.

Link prediction can be viewed as a special case of a binary classification problem, with the challenge of extreme class imbalance. Each pair of nodes can be predicted as either positive or negative. An additional challenge is that the number of candidate comparisons is quadratic to the number of entities that can be linked to each other, which creates a problem of scalability for large graphs. There are several methods that have been proposed for link prediction. A common approach is to use an unsupervised score such as cardinality of common neighbors set for each pair of nodes, the Jaccard coefficient, preferential attachment and more sophisticated measures like *PropFlow* [4] among others. These methods are generally exploiting only the topological structure of the graph and do not make use of side features. Supervised approaches use side features (which can include unsupervised scores) [4][5]. Other proposed techniques are mixed membership stochastic blockmodels [2] and non-parametric latent features models [6].

Probabistic soft logic (PSL) [3] is a modeling language that can be used for the task of learning and predicting in relational domains. There is an open-source implementation of PSL available on github. PSL constitutes a template for hinge-loss Markov random fields (HL-MRFs). HL-MRFs is a class of probabilistic graphical models. They are extremely scalable because they are log-concave densities over continuous variables, a fact that renders their optimization tractable. Link prediction can be viewed as a probabilistic inference problem. PSL can be used for this task, where a set of weighted first logic rules constitute the features. The PSL output contains every candidate pair of linked entities with an assigned score which is the inferred probability that the particular link exists (or will exist).

We perform a link prediction task using actor collaboration data from IMDB. We use an open-source tool written in Python, IMDBPy [1], to extract the semi-structured downloadable data of IMDB into a PostgreSQL relational database. We use this database as a practically inexhaustible source of information on actors and movies. We constraint ourselves to 4 datasets of different size and perform the link prediction task using data in a specific time frame, deemed as the training data, to predict collaborations in a subsequent time frame. We compare the common neighbors method which we use as baseline with outputs of PSL models that use several side features extracted from the database. There has been previous work performing link prediction with datasets from IMDB [7] but to the best of our knowledge not on the task of the prediction of actor collaborations. Though IMDB is an abundant and reliable source of data, datasets from IMDB are not prominent in previous link prediction studies; bibliographical data are among the most common datasets used. The goal of the project is to familiarize ourselves with the use of Probabilistic Soft Logic to perform machine learning tasks and also to stress the usefulness of IMDB as an abundant source of data, since it appears overlooked by the community. The code of our project lies in this repository.

2 The Internet Movie Database

The Internet Movie Database (IMDB) provides information on films, TV shows, video games, actors and personalities involved in production and directing as well as information on fictional characters, summaries of plots and trivia. The site is the oldest of its kind, it was launched in 1990 by software developer Col Needman. It is currently a subsidiary of Amazon that uses it as an advertising resource.

The database contains more than 2.5 million titles (including episodes) and more than 4 million personalities. It currently has more than 50 million registered users.

The site provides a big chunk of its data to the public. The data are downloadable in semi-structured text format. We restricted ourselves to the snapshot of the data. There is additional data that records the evolution of the database in time, but this would require resources that are outside of the scope of the present class project. The site also hosts dozens of fora, which contain

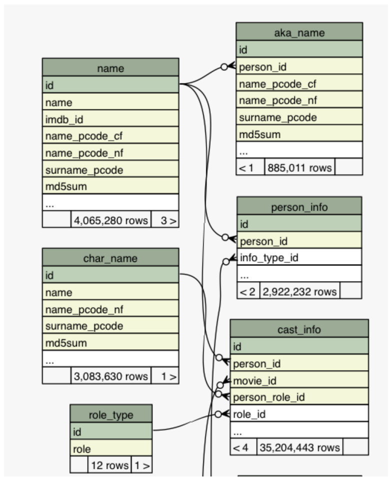

Figure 1: A snapshot of part of the PostgreSQL schema showing the 'name' relation

discussions among authenticated users. This data is probably suitable for sentiment analysis and could be used for several tasks in machine learning but sadly the site does not provide this data to the public domain.

2.1 IMDBPy

IMDbPY[1] is a Python package that can be used to retrieve and manage the data of the IMDb movie database. It is an open source tool, intended for use by developers and programmers. It can transform the unstructured downloadable data of IMDB into a traditional relational database schema and import it into a RDBMS. We extracted the data into a PostgreSQL 9.1 server.

The data are extracted into a complex database schema and the appropriate indices are created. The package lacks a documentation on the schema. There are entities for titles (movies, shows, games) and names (actors, directors, producers, fictional characters), cast info, movie info etc. The user has to navigate through the schema and record the proper codes that represent different types of information, person roles etc. so that the write-up of the appropriate SQL queries to the database is possible. In Figure 1 we can see a snapshot of portion of the database schema obtained through the use of the SchemaSpy tool.

2.2 The Datasets

Due to the convenience of having the entire database of IMDB in our disposal and since we did not know the challenges that we will encounter nor the execution time for performing the link prediction given our limited resources, we extracted four different sets of actors from the database. This also allowed us to experiment in datasets of different magnitude to confirm the generality of our approaches.

All four datasets contain actors and actresses that starred in movies in the 2001-2005 period. They are of increasing size, we designate them by *small, medium, large* and *extra large*. We use the number of votes that a movie receives as a knob. For a particular number of votes we obtain all movies that have at least that number of votes and thereafter extract all personalities that participated in those movies as actors. As the number of votes increases the number of movies and, hence, the number of associated actors becomes smaller. Thereafter, for the given set of actors we obtain the collaborations among actors for any movie in the 2001-2005 period, i.e. not solely for high voted movies but all movies where these actors have participated. For the same set of actors, we then obtain all collaborations of the 2006-2007 period in all movies of that period where these actors participated. We remove reoccurring collaborations, i.e. collaborations that occur both in the 2001-2005 and 2006-2007 period to obtain a set of new collaborations which is the target of link prediction. In Table 1 we show the size of each actor dataset, the corresponding number of observed collaborations used as the train set and the number of new collaborations.

Dataset	Actors	Observed Collaborations(2001-05)	New Collaborations (2006-07)
Small	219	8,208	134
Medium	530	22,722	1,760
Large	1,010	54,056	6,430
Extra Large	2,529	187,630	28,266

Table 1: Size of four datasets used in the actor collaboration prediction task

3 Common neighbors

We used the common neighbors as a baseline. For any two given nodes x and z, the score $c(x, z)$ assigned by the common neighbors method is:

$$c(x, z) = |\Gamma(x) \cap \Gamma(z)|$$

The intuition behind this measure is straightforward. The more common neighbors two nodes have, the more likely it is that a link between them exists or will exist. This is an observed fact in several real networks. This measure fails to incorporate side features and only relies on the structure of the graph. It does not distinguish among different neighbors based on node degree or other properties. It has similar performance in most applications to other unsupervized measures such as the Jaccard coefficient (which normalized the score with the cardinality of the union of neighbors of x and z), Adamic/Adar, preferential attachment etc.

4 Probabilistic Soft Logic for link prediction

We installed the open-source tool of the PSL modeling language to perform link prediction on the datasets. We experimented with several rules incorporating side features of each actor. In section 6, we describe in detail the side features and rules we included. The predicate we aim to predict is *COLLABORATION(X,Y)* where X, Y are unique identifiers. We also include all observed collaborations in a separate closed predicate *OBSERVEDCOLLABORATION(X,Y)*

4.1 Blocking

We implemented a blocking method that blocks together pair of actors that have a specific number of common neighbors and above. This created files of considerable size, because the number of blocked pairs are large in size. The blocked pairs are simply included as conjuncts within existing rules to reduce the number of pairs that the model will take into consideration during the inference, i.e.:

```
m.add rule: ACTOR(X) & ACTOR(Y) & BLOCK1(X,Y) & ..
```

This takes a toll on the loading and execution time, and requires a lot of memory because the blocked pairs have to be uploaded from the disk. However, it is the only way of obtaining results in reasonable time for some of the datasets.

4.2 Weight Learning

We initially set weights to each of the rules in the PSL model manually. The problem with this is that the manual weights are arbitrary and solely based on our intuition of the problem. PSL rules with manual weights often fail to provide the best results, although in some cases those rules with unlearned weights already achieved a superior performance to the baseline.

We then performed weight learning. We split the set of observed collaboration in two parts. One partition contains 70% of the observations and the other 30% of the observations. The large set was used as the train set and the small partition as the truth set. We perform weight learning using the method for maximum likelihood provided by the PSL implementation. The performance of the models with learned weights was generally superior to those with manual weights. The split was random, we could aim for further improvement in the classification performance by splitting the sets multiple times and learning with different train and truth sets picking the one with the best performance. We delegate this to future work.

5 Evaluation

We evaluate our methods using the AUC-ROC measure. AUC-ROC is a evaluation measure for binary classification problems, and has gained popularity recently as the classification measure of choice in machine learning studies due to several advantages. One of its main advantages over other evaluation measures, such as accuracy, or the F-1 measure is its insensitivity to unbalanced datasets, a problem which is rampant in link prediction tasks. Traditionally, an evaluation method over an output set of scored predictions has to use a threshold cut-off point in the output set. Outputs with score values over the threshold are considered positive whereas those below the threshold are collectively considered as negative. This carries the disadvantage that one must have an estimate of the size of the set of positives or consult the test set itself to estimate the size of the predicted positives. Moreover, this method is oblivious to the disriminaroty power of the technique within the set of predicted positives and negatives. AUC evaluates the output at all threshold points, and it allows for a better insight as to the classifier's ability to separate between positive and negative instances.

Formally, the AUC-ROC represents the area under the ‘receiver operating characteristic’ (ROC). This term has military origin. The ROC curve is a plot of the true positive rate and the false positive rate as the value of the threshold changes. The maximum value for the AUC is 1.0, in the case of a perfect classifier. An AUC value of 0.5 is the performance of a random classifier. One interpretation of the AUC-ROC measure is that it represents the probability that a randomly chosen positive instance will have higher score in the output set than a randomly chosen negative instance.

6 Features of our PSL model

6.1 Common neighbors with PSL

```
m.add rule : ( (X-Z) & observedcollaboration(X,Y) &
 observedcollaboration(Y,Z) ) >> collaboration(X,Z), weight : 1.0
```

We initially employed a single rule which derives common neighbors in PSL. Our goal was to match the performance of the Python common neighbors script and confirm the correctness of our code for AUC evaluation of the PSL output.

6.2 Recursive common neighbors

We used the same rule as above with intentional predicate (IDT) *collaboration* in the body of the rule to leverage more of the structure of the graph. It deteriotated performance for two of the datasets that we tested it with.

6.3 Negative common neighbors

```
m.add rule : ( observedcollaboration(X,Y) &
 ~observedcollaboration(Y,Z) ) >> ~collaboration(X,Z), weight : 0.5
```

This rule led to a slight increase in AUC performance with weight learning.

6.4 Gender

The sets of actors contain both actors and actresses. In fact, the schema assigns a different code to male and female actors in the ‘cast’ relation and not within the person entity itself. This appears conceptually wrong and based solely on the fact that English differentiates between actors and actresses using separate words; there is no distinction between male and female directors or other roles in the schema. Due to us failing to grasp this point initially, our first version of our extracted datasets contained only male actors. Once we realized the way gender is coded in the schema, we included the actresses.

The gender attribute could be a predictor for a collaboration since there may be a preferential attachment of males to males and of females to females. We thus extracted the gender information for each actor to an input file and included the following rule:

```
m.add rule: (ACTOR (X) & ACTOR(Y) & GENDER(X,G) & GENDER(Y,G))
 >> COLLABORATION(X,Y),  weight:1.0
```

This rule did not seem to provide any benefits, even with weight learning. It appears that two individuals sharing the same gender is not a predictor of collaboration.

6.5 Directors

The intuition we had is that actors who have collaborated with the same director in a movie, but have not yet collaborated with each other are more likely to collaborate in the future than actors who have never collaborated with the same director. This seems to be true in many cases; many directors choose specific actors multiple times in their careers so if two actors are generally chosen by the same director their trajectories are likely to meet. We extract the set of directors for all movies in the 2001-2005 period and for that set of directors we query the database appropriately to obtain the pairs of actors and directors for every instance of a collaboration between an actor and a director. We then assemble the following rule and include it in the model:

```
m.add rule: (actor(Y) & actor(Z) & directed(X,Y) & (X ^ Y) &
 directed(X,Z) & (X ^ Y ) & ~observedcollaboration(Y,Z) ) >>
  collaboration(Y,Z), weight : 10.0
```

The rule by itself leads to a noticeably higher performance for two of the datasets, the small and the medium.

6.6 Producers

Similarly to the directors collaboration, we suspect that two actors who have collaborated with the same producer are more likely to collaborate with each other if they have not yet. We extract producer information in the same way and assemble an analogous rule for producers. The rule enhances the separating ability of the classifier.

6.7 Distribution and production companies

In a similar line of thought, we extract indirect collaborations between actors and distribution and production companies, using the movies where an actor has participated and the companies involved in the production and distribution of the movie. We attempt to infer collaborations among actors based on the indirect collaboration of actor with specific companies through a movie. The rule is as follows:

```
m.add rule: (actor(Y) & actor(Z) & distributioncompany(X,Y) & (X ^ Y) &
 distributioncompany(X,Z) & (X ^ Y ) & ~observedcollaboration(Y,Z) ) >>
  collaboration(Y,Z), weight : 5.0
```

and likewise for production companies. These rules did not affect the results. A possible reason is the big number of companies (actually, IMDB contains more than 250,000 companies in total) involved in each movie and the indirect nature of the actor-company collaboration. To keep the number of companies for each movie (and subsequently, actor) low, we had to restrict ourselves to US-based companies (country code= '[us]'). Without it, the company roster included every worldwide distribution company and made the data too big. Restricting to American companies pruned the average number of companies per actor but still it was in the dozens on average.

6.8 Genres

Genres should be a good predictor for actor collaboration. Usually, particular actors are associated with particular genres, e.g. an actor who plays too often in comedies is a comedian. Two comedians are more likely to collaborate, while, inversely, an actor who generally stars in horror movies is less likely to collaborate with a comedian. We queried the database to obtain pairs of actors and genres based on the genres of movies where the actors have played. Unsurprisingly, each actor was generally associated with multiple genres. The rule is as follows:

```
m.add rule: (genre(A,B) & genre(C,D) & (A-C) &
 ~observedcollaboration(A,C) & sameGenre(B,D)) >>
  collaboration(A,C), weight : 0.1
```

We manually set the rule weight to a small number, since due to the large number of genres per actor there are significant overlaps so a single match in genre by itself should not be considered an important predictor. The weight learning confirmed this approach by applying small weights to this rule.

6.9 Languages

An actor or actress is linked to a language based on the languages spoken in the movies s/he has participated. We extracted that information from the database and obtained the appropriate files. We then loaded language predicates and assembled a PSL rule that infers collaboration between actors who have worked in the same language. This rule did not appear to improve results. This is likely because the overwhelming majority was only English, and for other languages there did not seem to be much variation in the 2001-2005 frame.

6.10 Actor quality based on ratings

The IMDB database includes movie ratings. The ratings are derived from audience evaluation. We obtain an indirect rating of an actor based on the rating of movies that s/he has participated in. In particular, we issue an aggregate SQL query computing the average rating of movies that each actor in our dataset has. This is the actors indirect rating. The intuition is that the higher or lower the quality of movies onr participates in, the higher the chance that one will collaborate with someone of similarly high or low quality. We use a boolean function of similarity for ratings that considers two ratings similar if they differ no more than 0.8 points (this is arbitrary and it would be interesting to tune it). This rule provided mild benefits to the performance for the small and medium datasets.

6.11 Death of actor

Some of the actors in our datasets passed away in the 2001-2005 period. IMDB contains the death dates for the deceased actors. We expect that a deceased actor will not collaborate with actors posthumously unless there is already recorded material released after the actor's death. We obtained the set of deceased actors in the 2001-2005 period. Some of the dates were incomplete (e.g. 'May 2005') and were not recognized by the PostgreSQL date parser yielding an error. We had to work around this by doing pattern matching:

```
CREATE TABLE Deceased20012005extralarge AS
SELECT A.person_id
FROM Actors20012005extralarge A, person_info P, info_type T
WHERE A.person_id=P.person_id AND P.info_type_id=T.id
AND T.info='death date' AND P.info SIMILAR TO '%200(0|1|2|3|4|5)%';
```

The negative rule was as follows:

```
(dead(A) & actor(B))>> ~collaboration(A,B), weight: 10
```

Unsurprisingly, the rule proved efficient in excluding unlikely collaborations.

7 Results

We experimented with several combinations of rules and attempted to manually fine-tune them. Once the set of rules became larger and took its full form, manually fine-tuning was less meaningful. We report the results for all 4 datasets including all the rules with automatic weight learning. The weight learning process determines the usefulness of each individual rule to some extent. For all the datasets but the extra large one, blocking was not necessary to make the execution scale, so the results are without blocking for them. For the extra large dataset we used the blocking strategy to make the program terminate in reasonable time using the limited computational resources of a personal laptop.

For all datasets we restricted the maximum number of iterations to 500 and the maximum number of rounds to 3 through setting the appropriate constants in the PSL configuration file. The extra large dataset is the only one out of the datasets we examined where the PSL model does not beat the baseline. This is probably due to the blocking, pairs of collaboration that do not fall in the same block are never examined as candidate collaborators and are assigned a zero score. This takes a toll on classification performance. We can see the results in Table 2.

Dataset	Common neighbors	PSL
Small	0.6015	**0.6270**
Medium	0.6984	**0.7022**
Large	0.6948	**0.6993**
Extra Large	**0.7013**	0.6878

Table 2: AUC-ROC for baseline and PSL for the 4 datasets

8 Future work

Overall, it was an exciting project. We made our hands dirty by exploring the IMDB data and familiarizing ourselves with the PSL modeling language. We witnessed in practice the immense difficulty of the task of link prediction.

Future work could focus on fine tuning the model, adding useful rules and enhancing the weight learning using different splits of the training data. Moreover, a good idea is to implement additional baselines, for instance, SVM or supervised methods proposed in [4].

Finally, it would be great to contribute to the project of PSL by writing code that will be committed to the repository, but also contributing to the general domain of probabilistic learning with PSL by researching on open topics.

References

[1] Imdbpy: Python script for retreiving data from imdb. `http://imdbpy.sourceforge.net/support.html#documentation/`.

[2] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, June 2008.

[3] A. Kimmig, S. H. Bach, M. Broecheler, B. Huang, and L. Getoor. A short introduction to probabilistic soft logic.

[4] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 243–252, New York, NY, USA, 2010. ACM.

[5] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2011.

[6] K. T. Miller, T. L. Griffiths, and M. I. Jordan. Nonparametric latent feature models for link prediction.

[7] G. Rossetti, M. Berlingerio, and F. Giannotti. Scalable link prediction on multidimensional networks. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ICDMW '11, pages 979–986, Washington, DC, USA, 2011. IEEE Computer Society.