2008 Seventh International Conference on Machine Learning and Applications

Does Wikipedia Information Help Netflix Predictions?

John Lees-Miller, Fraser Anderson, Bret Hoehn, Russell Greiner
University of Alberta
Department of Computing Science
{leesmill, frasera, hoehn, greiner}@cs.ualberta.ca

Abstract

We explore several ways to estimate movie similarity from the free encyclopedia Wikipedia with the goal of improving our predictions for the Netflix Prize. Our system first uses the content and hyperlink structure of Wikipedia articles to identify similarities between movies. We then predict a user's unknown ratings by using these similarities in conjunction with the user's known ratings to initialize matrix factorization and k-Nearest Neighbours algorithms. We blend these results with existing ratings-based predictors. Finally, we discuss our empirical results, which suggest that external Wikipedia data does not significantly improve the overall prediction accuracy.

1 Introduction

Netflix distributes movies via an internet site. Their service includes a recommender system that suggests movies to a user based on that user's past movie ratings. The Netflix Prize is a competition to inspire researchers to find ways to produce more accurate recommendations. In particular, the challenge is to predict how a user will rate a particular movie, seen on a specified date. To help, Netflix provides approximately 100 million ratings for 17 770 movies by 480 thousand users as training data for a collaborative filtering method. Of these ratings provided by Netflix, 1.4 million are designated as the probe set, which we use for testing; see [4].

Previous approaches have achieved considerable success using only this ratings data [3]. We begin with the hypothesis that external data from Wikipedia can be used to improve prediction accuracy. The use of such data has been shown to be beneficial in many collaborative filtering tasks, both in recommendation system and movie domains [7, 12, 16]. Balabanovic and Shoham [1] successfully use content extracted from an external source, the Internet Movie Database (IMDb), to complement collaborative filtering methods. Netflix allows contestants to use additional data, but it must be free for commercial use. Unfortunately, this eliminates IMDb, even though this is known to be useful for Netflix predictions [14]. Fortunately, this does not prevent us from using movie information from the free encyclopedia Wikipedia [1].

Data sources such as IMDb and Yahoo! Movies are highly structured, making it easy to find salient movie features. As Wikipedia articles are much less structured, it is more difficult to extract useful information from them. Our approach is as follows. First, we identify the Wikipedia articles corresponding to the Netflix Prize movies (Section 2). We then estimate movie similarity by computing article similarity based on both *article content* (term and document frequency of words in the article text; Section 3) and *hyperlink structure* (especially shared links; Section 4). We use this information to make predictions with k-Nearest Neighbors (k-NN) and stochastic gradient descent matrix factorization [3,6,11] (also known as Pseudo-SVD) methods. We then blend our predictions with others from the University of Alberta's Reel Ingenuity team (Section 5).

2 Page Matching

We use a Wikipedia snapshot[2] containing roughly 6.2 million articles, most of which are not related to movies. In order to use the Wikipedia data we must map each Netflix title to an appropriate Wikipedia article, if one exists. We use several methods to find such matches.

One method finds articles using longest common subsequence and keyword weighting in the article titles. We put more weight on words like "movie" and "film," and less weight on words like "Season" and "Volume," which tend to be present in the Netflix titles but not in Wikipedia article titles. This method matches 14 992 Netflix titles with Wikipedia articles, of which approximately 77% are appropriate; here and below, we estimate appropriateness with spot checks conducted by the authors.

[1]`http://en.wikipedia.org/wiki/Wikipedia:Copyrights`
[2]`http://download.wikimedia.org` (February 25, 2008)

Our second method uses the Yahoo! search API [18][3]. We use three types of queries of varying specificity. The first query is '$\langle title \rangle$ $\langle year \rangle$ Wikipedia (tv or film or movie or video or show)'; the second is '"$\langle title \rangle$" $\langle year \rangle$ Wikipedia'; and the third is '$\langle title \rangle$ Wikipedia'. These queries match 11 704, 15 014, and 17 201 Netflix titles, of which 73%, 65%, and 70% are appropriate, respectively.

Our third method uses Wikipedia's "Year in Film" pages[4]; each of these pages links to Wikipedia articles for many of the movies that were released in its year. We collect all of the "Year in Film" pages and extract all of the links to movie articles. Each Netflix title is then stripped of certain keywords such as "Bonus", "Special" and "Edition" and compared against these links. We take the link with the smallest Levenshtein [9] distance to a given Netflix title as the match for that title. While this method finds matches for all 17 770 titles, only about 28% are appropriate.

No single method finds a large number of accurate matches, so we combine our methods in a voting system. We compare two voting systems in this paper. The first system uses heuristic title matching and one search query, and it accepts a match when both methods agree; this matches 7 938 (43%) of the Netflix titles which accounts for 46% of the probe set titles (7 720 of 16 938 titles in the probe set) and 75% of the probe set ratings. The second system adds Levenshtein distance and two more search queries to the first system. It accepts the majority match for each title, where each method has one vote. If none of the methods agree, the title is marked as having no match; we then, by hand, select which of the candidate matches is most appropriate, or indicate that none are appropriate. This matches 14 681 (83%) of the Netflix titles, of which 93% appear to be appropriate. These account for 83% of the probe set titles and 96% of the probe set ratings.

The quality of a match is determined by manual inspection and personal judgment. For example, even though an article for *Twilight Zone: The Movie* is not about *The Twilight Zone Vol 27*, the article about the movie is related to the article about the television series. The overall quality of a matched article is judged by manually examining a random sample of 200 pairs of Netflix titles and their matching Wikipedia articles and counting the number of matches that appear correct.

3 Natural Language Processing

Wikipedia articles are unstructured documents whose information content is in large sections of plain English text. We aim to estimate the similarity of each pair of matched pages, assuming that this is also an estimate of the similarity of the corresponding movies. Figure 1 shows the process of extracting similarity measures from the article text for use in the predictors.

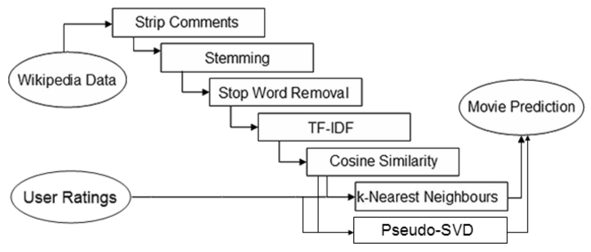

Figure 1. Processing flow for predicting movie ratings based on article content.

We begin by preprocessing the Wikipedia articles to remove the comments and markup. We also remove extra metadata relating to Wikipedia internals, formatting rules, and other unimportant information as well as removing all non-alphanumeric characters, which are abundant in the Wikipedia markup.

Next, we use the Porter stemmer [17] to reduce each word to its root word, making it easier to find shared words despite differences in form. For example the words "watched", "watching", and "watcher" all reduce to the root word "watch." This significantly reduces the size of the set of terms. For the smaller (7 938) set of Wikipedia articles about movies, the stemmer reduced the number of unique terms from nearly 200 000 to roughly 150 000.

After stemming, the text is stripped of stop words, which are words that do not add meaning to the sentence but are required in order to make it grammatically correct; these are words such as "a", "the" and "which". We remove all occurrences of words that are present in the Snowball list [5], which contains 174 stop words.

After stemming and stop word removal, we use *term frequency - inverse document frequency* (*TF-IDF*) to estimate the importance of each word in each article. This involves constructing a large vector for each article whose components each represent the number of times that a unique word is used in the article (the term frequency, or TF), scaled relative to the number of times that word was used throughout the set of articles about movies (the document frequency, or DF). We take the logarithm of each component (TF/DF), as the *TF-IDF*.

We define the similarity between two articles as the cosine of their *TF-IDF* vectors. For each pair of article vectors, a and b, the cosine similarity between the two articles

[3]Use for this purpose does not violate the terms of service; `http://info.yahoo.com/legal/us/yahoo/api`

[4]`http://en.wikipedia.org/wiki/List_of_years_in_film`

is computed as:

$$\cos(a,b) = \frac{a \cdot b}{\sqrt{a \cdot a} \cdot \sqrt{b \cdot b}}$$

This produces a value between -1 and 1 that corresponds to the similarity between each pair of articles. This is a common method for computing document similarity in the natural language processing domain; it is easy to compute, and it eliminates the effects of relative document length by including a normalization factor.

Many Wikipedia movie articles (14 603 of the 14 681 matched articles from Section 2) have an "InfoBox Film" side panel that lists movie features like genre, director and principal actors. As well as using the whole article text to find words for the *TF-IDF* approach, we also present results using only this side panel, with the aim of gauging the value of the whole article text relative to these commonly available movie features.

4 Link Similarity

When users write Wikipedia articles, they are encouraged to add links to other relevant pages [19]. This advice is not universally followed, but many articles are carefully linked to other articles. For example, in the article for the 1999 film *The Green Mile*, the first paragraph contains the following links (underlined):

> The Green Mile is an Academy Award-nominated 1999 American drama film, directed by Frank Darabont and adapted by him from the 1996 Stephen King novel The Green Mile. The film stars Tom Hanks as Paul Edgecombe and Michael Clarke Duncan as the 8 foot giant John Coffey.[5]

The links identify several salient features of the film, here its genre, director and principal actors. We can characterize each movie article based on its list of links, rather than its words. This is helpful because there are fewer links per article than words, and because this avoids having to perform stemming and stop word removal. Many pages also contain complete cast lists with links to all major and minor actors in the film. However, most pages also contain links that do not seem to be important in characterizing the movie. For example, the article for the 2000 film *Miss Congeniality* links to each of the 21 countries in which the film was released.

4.1 Weights from *TF-IDF* and Page Rank

Here we use *TF-IDF* on links and Page Rank [10] to estimate the importance of links. The simplest model for understanding Page Rank is the "random web surfer." Upon reaching a page, this surfer chooses one of the links on the page uniform-randomly and follows it; this process repeats many times. The rank of a page measures how often the the random surfer visits it. More formally, the rank $R(p)$ of page p is given by the recurrence

$$R(p) = c \left(\sum_{q \in B(p)} \frac{R(q)}{|F(q)|} + E(p) \right)$$

where $B(p)$ is the set of links from other pages to page p (back links), and $F(q)$ is the set of links that page q links to (forward links). The scalar c is a normalization constant that ensures that $\sum_p R(p) = 1$. The value $E(p)$ can be used to target the rankings toward a particular topic (movies, for example); pages with a high E value are given a higher apriori importance.

Page Rank was designed to improve search engine performance on vague queries [10]; the search engine should return more "important" (higher ranked) pages first. We use Page Rank to measure the importance of connections between movies. For example, two movies may have several minor actors in common, but we anticipate this will have little impact on how people relate the two movies. If two movies both star the same celebrity actor, however, the connection is likely to be much more meaningful. For example, even Star Trek fans must admit that Brad Pitt is better known than Jonathan Frakes, and sure enough, they rank 30 343rd and 149 330th, respectively [6]. Of course, not all links are meaningful in an obvious way. Many highly ranked articles are about countries (the United States, for example), and many movie pages list all countries of release; it is unlikely that these are meaningful connections.

To estimate the similarity of two pages based on their links, we apply cosine similarity to a modification of the *TF-IDF* vectors (see Section 3). We first find all pages q that are linked from at least one matched movie page. The modified *TF-IDF* vector for a matched movie page p has one component for each linked page q, whose value is

$$\frac{TF(p,q) \times R(q)^{\alpha}}{DF(q)} \tag{1}$$

where $TF(p,q)$ is the number of times that a link to q appears on page p, and $DF(q)$ is the number of matched movie pages that link to q. To motivate the inclusion of the Page Rank $R(q)$ in the numerator, we return to the example of Brad Pitt and Jonathan Frakes. There are 624 pages that link to the article on Brad Pitt and 146 pages that link to the article on Jonathan Frakes. So the document frequency

[5]Retrieved on May 28, 2008.

[6]Ranks are based on global (not biased toward movies) ranking, with $|E| = 0.1$, convergence 1×10^{-5}. Ranks range from 1 to 6.2 million. The corresponding $R(p)$ values are 4.2×10^{-6} and 1.5×10^{-10}.

of Brad Pitt is higher, and his links will have lower ***TF-IDF*** weights, even though he is a more popular actor and (we assume) has more influence on movie similarity. The Page Rank counteracts the effect of the document frequency, to account for this popularity effect.

The α parameter is introduced because $R(p)$ has a very large range (roughly 10^{-14} to 10^{-3}), and the $R(p)$ values of pages near the top of the ranking are several orders of magnitude larger than those farther down. When $\alpha = 0.1$, the range of the numerator is compressed to the range 10^{-2} to 1, which is comparable to the range of the denominator.

4.2 Weights from Rating Similarity

As another way to determine which links are relevant to predicting ratings, we use the ratings data for frequently-rated movies to weight links. We observe that k-NN methods using ratings-based similarity are quite successful when the query movie and neighbour movies have many raters in common, so we have high confidence in the ratings-based similarity estimates for pairs of frequently-rated movies. If a link appears on the matched pages for frequently-rated movies with similar ratings, we weight it heavily when computing the similarity between movies from Wikipedia content. Links that appear on the matched pages for movies with dissimilar ratings are taken to be less relevant and receive less weight.

We first identify the subset of movies M_r with at least r ratings in the training set. We then compute the ratings-based similarity $S(m, n)$ for all movies $m, n \in M_r$ using Pearson correlation on the ratings in the training set after removing global effects, as in [2]. For each Wikipedia page q, we form the set $B_r(q)$ of movie pages for movies in M_r that link to page q. The ratings-based similarity weight of a link to page q is then computed as

$$RSW(q) = \frac{|B_r(q)| \times \bar{S}(q)}{|B_r(q)| + \delta} \qquad (2)$$

where $\bar{S}(q)$ is the arithmetic mean of $S(m, n)$ over all $m, n \in M_r$ for which the matched movie pages both link to page q. Some pages q have no links from pages for movies in M_r, so $B_r(q)$ is empty, and in this case we set $\bar{S}(q) = 0$. For other pages, $B_r(q)$ is non-empty but small; in these cases it is helpful to introduce a prior link weight. We include δ to shrink the link weights toward zero (that is, links have no apriori importance), when $B_r(q)$ is small.

Having computed the similarity link weights, we then apply the cosine similarity measure as before. The vector for a matched movie page p has one component for each linked page q, whose value is $RSW(q)$ if p links to q and 0 otherwise.

Which of the links are most important in determining rating similarity is interesting in its own right. Eight of the ten most heavily weighted links pertain to the 1959 television series *The Twilight Zone*. Many other heavily weighted links pertain to the popular manga series *Dragon Ball* and American television actors. It appears that the most heavily weighted links correspond to "cult" items, which is intuitively appealing. We conjectured earlier that a link to *Brad Pitt* would be more important than one to *Jonathan Frakes*, and that both would be more important than a link to *United States*. In fact, these results indicate that Frakes (0.103) is a better similarity indicator than Pitt (0.001). Our assertion about the United States appears to be correct; it has weight -0.006. [7]

5 Experiments

For the experiments conducted, we train on the training set with the probe set removed, and we test our predictions on the probe set. Note that Netflix withholds a further 2.8 million ratings as the qualifying set, which they use to test predictions for the prize; we have not tested our predictions on this qualifying set.

5.1 Prediction

The content- and link-based methods discussed above each generate a similarity matrix containing the similarity of each movie-movie pair with matched Wikipedia articles (rows for unmatched movies are set to zero). To predict user ratings from this similarity matrix, we use a k-Nearest Neighbors (k-NN) algorithm and a Pseudo-SVD algorithm. Each of these methods combines the similarity estimates from Wikipedia with ratings from the training set to predict ratings in the probe set.

The successful Pseudo-SVD method [6,11] requires an initial set of k movie features for each movie, before the iterations begin. We choose these to be the first k principal components of a similarity matrix that is derived from Wikipedia content. We then run the Pseudo-SVD variant as described in formula 7 of [3]. We initialize movie feature values for unmatched movies to 0.1.

The k-NN method enumerates, for each user-movie query, all movies rated by the query user and then chooses the k movies most similar to the query movie. It predicts a rating from a similarity-weighted linear combination of the user's ratings of these k "neighbour" movies. We use the following definition of movie similarity to account for rating, viewing and Wikipedia data.

Let m and n be movies, and let $U(m, n)$ be the set of users who have rated both movies. Let $\cos(m, n)$ and Pearson(m, n) be the cosine similarity and Pearson correlation (see [2], for example) of the ratings

[7]Here $r = 0$ and $\delta = 10$. There are 201 378 links with weights ranging from -0.180 to 0.489 with mean 0.023.

over this set $U(m,n)$. We also define $V(m,n) = |U(m,n)|/(|U(m,n)| + \beta)$ as the "viewing similarity," where $\beta > 0$ is a user-defined parameter, here set to 500. This viewing similarity is high when there is a large group of users who have rated both movies (regardless of how they rated them). Denoting the similarity estimate from Wikipedia as $W(m,n)$, we then compute the similarity of movies m and n as

$$V(m,n)[w_0 \cos(m,n) + w_1 \text{Pearson}(m,n)] + w_2 \frac{W(m,n)}{|U(m,n)|^\gamma} \tag{3}$$

where w_0, w_1, w_2 and γ are free parameters. We typically set $\gamma = 0.9$. Because the cosine and Pearson terms are unreliable when the set of common users is small, we weight them by the viewing similarity $V(m,n)$. We divide the Wikipedia similarity $W(m,n)$ by $|U(m,n)|^\gamma$ because we assume that the need for Wikipedia similarity data is inversely related to the amount of ratings data available.

5.2 Blending

Bell et al. [4] (and many others) show that blending a large number of predictors can yield better predictions than any of the predictors alone. If mistakes in the predictions based on Wikipedia data are independent of those based only on ratings, the Wikipedia predictors should be valuable members of the blend.

We begin with a probe set prediction from each of our predictors, writing the predicted ratings in the same order as the probe set. These form a matrix M with one row for each probe set rating and one column for each predictor. We then solve the least-squares problem $M^{\mathrm{T}}Mx = M^{\mathrm{T}}y$, where y is the vector of true probe set ratings, and x is the blending weight vector with one component for each predictor. Note that we have used the true probe set ratings to compute these blending weights; we use the probe set RMSE for the blend as an indicator of how well the blend would perform on the qualifying set.

We can improve these blending results by dividing the probe set into categories and solving a separate least-squares problem for each category. We test two different categorization schemes, each based on the training set after removing the probe set.

1. *User categorization.* We divide the users into 9 categories based on how many movies each has rated. The category boundaries are 10, 30, 50, 100, 200, 400, 600 and 800 movies.

2. *Movie categorization.* We divide the movies into 11 categories based on how many users have rated them. The category boundaries are 200, 400, 800, 1 600, 3 200, 6 400, 12 800, 25 600, 51 200 and 102 400 users.

Table 1. Probe RMSEs on matched movies: $k = 24$-NN with Wikipedia similarity only.

Wiki Similarity	Parameters		RMSE[a]
text *TF-IDF*			0.97113
box *TF-IDF*			0.97648
	α		
link *TF-IDF*	0	(8k)	0.97508
link *TF-IDF*	0.1	(8k)	0.97521
link *TF-IDF*	0		0.96932
link *TF-IDF*	0.1		0.96938
	r	δ	
link *RSW*	0	0	0.96452
link *RSW*	0	10	0.96582
link *RSW*	500	0	0.96636
link *RSW*	500	10	0.96847

[a] To help calibrate these, note that Netflix's original recommendation system, Cinematch, achieves an RMSE of 0.9514; the best RMSE on the leaderboard, at the time of submission, is 0.8643.

We choose the category boundaries so that each category has a reasonably large number of probe set ratings; the smallest categories contain 48 521 and 19 969 ratings for user and movie categorization, respectively.

5.3 Results and Discussion

Table 1 lists the RMSEs obtained from the k-Nearest Neighbours method with $k = 24$ using our Wikipedia similarity measures, but no ratings-based similarity data; that is, $w_0 = w_1 = \gamma = 0$ and $w_2 = 1$ in (3). Here we make predictions only for queries about movies that we have matched in Wikipedia. The 7 938 ("8k") matches (Section 2) account for 75% of the probe set queries; the 14 681 ("15k") matches account for 96%. All of the results in Table 1 are for the "15k" matches, except for the two marked "8k." The "text" and "box" *TF-IDF* results are from Section 3, using the whole article text and only the side panel, respectively. The link *TF-IDF* and *RSW* results are from Sections 4.1 and 4.2, respectively.

The reported RMSEs are poor relative to similar ratings-based methods. They are similar to the results from a slope one predictor [8] (RMSE roughly 0.984), which is equivalent to our implementation of k-NN with all similarities set to 1. This suggests that the movie similarities computed using Wikipedia content are only weakly indicative of user rating patterns. Results based on links are consistently better than those based on text content; this supports our assertion that the page links contain salient features.

Table 2 lists the RMSEs obtained from the Pseudo-SVD and k-NN methods using both ratings and Wikipedia data

Table 2. Probe RMSEs: Wikipedia and ratings.

Wiki Similarity	Parameters		RMSE	
			PSVD	k-NN
none			0.90853	0.92162
text *TF-IDF*			0.91316	0.92928
box *TF-IDF*			0.91290	0.92830
	α			
link *TF-IDF*	0	(8k)	0.90855	0.92469
link *TF-IDF*	0.1	(8k)	0.90845	0.92470
link *TF-IDF*	0		0.90875	0.92463
link *TF-IDF*	0.1		0.90891	0.92463
	r	δ		
link *RSW*	0	0	0.90918	0.92456
link *RSW*	0	10	0.90938	0.92452
link *RSW*	500	0	0.90903	0.92456
link *RSW*	500	10	0.90945	0.92453

in (3). For k-NN, we set $w_0 = 0.688$, $w_1 = 0.112$ and $w_2 = 4.0$, which gives the best final blend results in our empirical tests. Here we make predictions for the whole probe set, including both matched and unmatched movies, using the similarity data from the "15k" matches (except for the two rows marked "8k"). The results when using Wikipedia data are consistently worse than those using only ratings data. However, if the mistakes in the Wikipedia predictions were relatively independent of those made by the ratings-based predictor, we would expect an improvement in the blend RMSEs.

In Table 3, we blend each Wikipedia similarity measure and method with its ratings-only counterpart. Here we use the "15k" matches, $\alpha = 0$ in (1) for link *TF-IDF*, and we set $r = 500$ and $\delta = 10$ in (2) for link *RSW*; the alternatives yield very similar results. Comparing the blend RMSEs with results before blending, from Table 2, we see that blending consistently improves the RMSE.

We then add all of our new results to the Reel Ingenuity team blend, which combines predictions from a variety of ratings-only methods.[8] These include several variants of k-NN, Pseudo-SVD and other matrix factorization methods [4], as well as Restricted Boltzmann Machines [15] and several simpler predictors such as constants, movie and user averages, k-means and slope one, for a total of 88 result sets. We achieve a modest overall improvement, with the RMSE decreasing from 0.88061 to 0.88033 on the probe set.

One of our main assumptions is that Wikipedia similarity data is most useful for movies that have few ratings.

[8] `http://www.aicml.ca` - Reel Ingenuity was ranked number 15 on the leaderboard at the time of submission.

Table 3. Probe RMSEs: blends.
Ratings Only Prediction + Wiki Similarity Prediction

Wiki Similarity	Categories	RMSE	
		PSVD	k-NN
(result before blending)		0.90853	0.92162
text *TF-IDF*	none	0.90714	0.92152
text *TF-IDF*	user	0.90687	0.92123
text *TF-IDF*	movie	0.90711	0.92098
link *TF-IDF*	none	0.90759	0.92158
link *TF-IDF*	user	0.90750	0.92109
link *TF-IDF*	movie	0.90754	0.92108
link *RSW*	none	0.90782	0.92157
link *RSW*	user	0.90759	0.92109
link *RSW*	movie	0.90776	0.92106

Reel Ingenuity Blend

Similarity	Categories	RMSE
none	none	0.88251
none	user	0.88061
none	movie	0.88122
all from Table 2	user	0.88033
all from Table 2	movie	0.88092

For this reason, we hypothesized that movie categorization would be more effective than user categorization, and that the weight of the Wikipedia-based predictions would be larger in the categories with movies with few ratings. However, the results in Table 3 are mixed. While the weight of the Wikipedia-based predictions in the blend is sometimes the highest in the category for least-rated movies, the category weights do not always follow the number of ratings.

Toward explaining this, we note that Wikipedia content is generally created by people with an interest in the topic, so it is reasonable to believe that more popular topics are more likely to have articles, and that these articles will be of higher quality. Thus a movie with fewer ratings (indicating low general interest) is less likely to have a high-quality Wikipedia page than a movie with many ratings.

We also note that the probe and qualifying sets contain more ratings from users with few known ratings than would be present in a random sample [13], but that this is not the case with movies; movies with few ratings in the training set also have few ratings in the probe and qualifying sets. So, we cannot have more categories for movies with few ratings, which would give more freedom in the blend, and still maintain a large number of ratings in each category, which is necessary to avoid overfitting. Moreover, because movies with few ratings in the training set also appear less frequently in the probe set, they do not contribute as much to the overall probe RMSE.

Another major question is, to what extent is the content of Wikipedia pages relevant to user ratings? The fairly poor results in Table 1 and the small overall improvement to the blend RMSE suggest that the Wikipedia content is only weakly related to user ratings, but we cannot yet answer this question quantitatively. There are clearly limits on how many aspects of movie similarity we can represent with a single number. For example, our text *TF-IDF* method finds that *Josh Groban: Live at the Greek* and *Outkast: The Videos* are both similar to *Ricky Martin: One Night Only*. These titles are all similar in that they are music-related DVDs, but it is not clear that a user who likes Ricky Martin's pop music will also like Josh Groban's opera, or Outkast's hip-hop. Adding more movie features, like genre or principal actors, is difficult in Wikipedia because that information is not available in an explicitly labelled form, as it is in the Internet Movie Database (IMDb) and other proprietary data sources.

Contributions

This paper has explored a number of ways to use Wikipedia data to determine movie similarity, hoping to make more accurate predictions for the Netflix Prize.

We introduced a novel ensemble system for matching movies with Wikipedia pages, which obtains a large number of pages that correspond to Netflix movies. We explored Page Rank as a measure of article popularity and incorporated it into the *TF-IDF* computations. We used ratings data to infer which links on Wikipedia are most relevant to predicting movie ratings. We restricted *TF-IDF* components to links rather than words in Wikipedia articles and produced similar results with less computation. We used the principal components of a similarity matrix to initialize the Pseudo-SVD method.

Unfortunately, these techniques did not improve significantly upon those of a mature ratings-based recommendation system; our best result improves the probe RMSE of the Reel Ingenuity blend by 0.00028. We conjecture that this is because the situation where external information can be most useful – dealing with movies with few ratings – is where Wikipedia provides the least help, as the associated pages are often either absent or impoverished.

Acknowledgments

We would like to thank Aaron Young, Eric Coulthard, and the whole Reel Ingenuity Netflix team for their help and feedback. We would also like to thank the Alberta Ingenuity Centre for Machine Learning for providing the computing power needed to run our experiments, and Nick Salyzyn for his work on graph clustering similarity measures.

References

[1] M. Balabanovic and Y. Shoham, "Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, pp. 66-72, 1997.

[2] R. M. Bell and Y. Koren, "Improved neighborhood-based collaborative filtering," in *Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[3] R. M. Bell and Y. Koren, "Lessons from the netflix prize challenge," *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 75-79, December 2007

[4] R. M. Bell, Y. Koren, and C. Volinsky, "Chasing $1,000,000: How we won the netflix progress prize," *Statistical Computing and Statistical Graphics Newsletter*, vol. 18, no. 2, pp. 4-12, 2007.

[5] R. Boulton, "Snowball," `http://snowball.tartarus.org/`.

[6] S. Funk, "Netflix update: Try this at home," 2006, `http://sifter.org/~simon/journal/20061211.html`.

[7] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999, pp. 688-693.

[8] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[9] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707+, February 1966.

[10] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., 1998.

[11] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, 2007, pp. 39-42.

[12] D. Pennock, E. Horvitz, S. Lawrence, and L. C. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach," in *Proc. 16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 473-480.

[13] Prizemaster, "Netflix Prize Forum: On the creation of the training and qualifying sets...," 2006, `http://www.netflixprize.com/community/viewtopic.php?pid=2242`.

[14] A. Rajaraman, "Datawocky: More data usually beats better algorithms," 2008, `http://anand.typepad.com/datawocky/2008/03/more-data-usual.html`.

[15] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *ICML '07: Proceedings of the 24th international conference on Machine learning*. New York, NY, USA: ACM Press, 2007, pp. 791-798.

[16] L. Si and R. Jin, "Flexible mixture model for collaborative filtering," in *Proc. Twentieth International Conference on Machine Learning (ICML-2003)*, 2003, pp. 704-711.

[17] C. J. van Rijsbergen, S. E. Robertson, and M. F. Porter, "New models in probabilistic information retrieval," 1980.

[18] Wikipedia:Build the web - Wikipedia `http://en.wikipedia.org/wiki/Wikipedia:Build_the_web`.

[19] Yahoo! Search APIs from Yahoo Search Web Service `http://developer.yahoo.com/search/web/`.