Exploiting crowd sourced reviews to explain movie recommendation

Sara El Aouad, Christophe Dupuy, Renata Teixeira, Christophe Diot, Francis Bach

Exploiting crowd sourced reviews to explain movie recommendation

Sara El Aouad, Technicolor, Inria
Christophe Dupuy, Technicolor, Inria
Renata Teixeira, Inria
Christophe Diot, Technicolor
Francis Bach, Inria

ABSTRACT

Streaming services such as Netflix, M-Go, and Hulu use advanced recommender systems to help their customers identify relevant content quickly and easily. These recommenders display the list of recommended movies organized in sublists labeled with the genre or some more specific labels. Unfortunately, existing methods to extract these labeled sublists require human annotators to manually label movies, which is time-consuming and biased by the views of annotators. In this paper, we design a method that relies on crowd sourced reviews to automatically identify groups of similar movies and label these groups. Our method takes the content of movie reviews available online as input for an algorithm based on Latent Dirichlet Allocation (LDA) that identifies groups of similar movies. We separate the set of similar movies that share the same combination of genre in sublists and personalize the movies to show in each sublist using matrix factorization. The results of a side-by-side comparison of our method against Technicolor's M-Go VoD service are encouraging.

1. INTRODUCTION

According to a recent study [7], over 40% of households in the United States have access to VoD services. With the overwhelming number of videos offered per service, a key challenge is to help users decide which movie to watch. Sophisticated VoD services use recommender systems based on matrix factorization applied to movie ratings [1]. VoD services then display the long list of recommended movies ranked based on the predicted rating per movie. This list is often organized into labeled sublists that help users browse the recommendations. For example, Netflix presents movies in rows according to genres (which go from more traditional, coarse-grained labels such as "Action" or "Comedy" to more specific labels such as "Visually-striking Goofy Action & Adventure") [1]. Existing methods to group movies into sublists and to label sublists require people to manually label each movie [10]. This manual method has two main drawbacks. First, the labels of each sublist are subjective and biased toward the opinion and cultural background of annotators. Second, manual annotation is expensive and requires an extensive human effort especially because labels are often specific to a region and movie databases keep growing with new movie releases.

In this paper, we argue that instead of relying on a few people to tag movies, we can use crowd sourced reviews to automatically identify groups of similar movies and label these groups. Many moviegoers enter detailed reviews of movies they have watched on sites such as IMDb and Rotten Tomatoes. The corpus of online reviews represents a source of rich meta-data for each movie. We use a database of 2000 movies and 100 users extracted from IMDb by Diao et al. [4] as a proof of concept in this paper.

The challenge we face is to mine the noisy free-text reviews from a heterogeneous set of people to extract meaningful and personalized sublists of movies. We are tackling this challenge in three steps. First, we design an algorithm based on LDA (Latent Dirichlet Allocation) to estimate the similarity between movies based on the content of reviews (§2.1). The outcome of this step is a list of the most similar movies to the movie the user selected. Second, we split the list of similar movies into sublists of movies that share common characteristics (§2.2). Our initial version uses a combination of genres to generate these sublists. For each sublist, we provide a list of words that characterizes this sublist, extracted from the crowd sourced reviews. To give more insight about the recommended movies, users can click on these words to see in which movies of the sublist the word corresponds to. Finally, we personalize each sublist by filtering and ordering the movies using matrix factorization (§2.3). We only display to the user movies with a predicted rating of at least six (out of ten).

We implement a web service using our method with the same look as Technicolor's VoD service in the United States, M-Go (§3). This implementation allowed us to perform side-by-side comparisons of M-Go's existing system (which uses manual tagging of movies to extract sublists of similar movies and labels for each sublist) and our sublists. Our initial results are encouraging (§4).

We do not claim that our method gives better recommendation, but instead that the movies we recommend come with more insight about why they are recommended. Note that this is early results and that we believe that the method can be greatly improved, and extended to many other domains.

2. METHOD

We envision a scenario where a user will click on a movie, say m, and the recommender will display a list of movies similar to m recommended for the user organized in labeled sublists. We use a database of 2,000 movies and 100 users extracted from IMDb (a subset of the data described in [4]) for our method. Our method works in three steps. First,

we identify the list of similar movies. This step uses the content of the IMDb reviews to identify similarities among movies with no personalization. Second, we split the list of similar movies into sub-groups. Finally, we apply matrix factorization to personalize the sublists.

2.1 Movie similarity

We base our method to identify the list of most similar movies to a given movie on Latent Dirichlet allocation (LDA) [2]. LDA is a probabilistic topic model that extracts K latent topics from a corpus of documents. Each topic is a discrete distribution over the words of the vocabulary. Ideally, in our case, we would like each topic to be consistent around movie features such as genres, actors or directors. To apply LDA, we must define what constitutes a document, an appropriate vocabulary, and a value of K.

We create a corpus where each document is the concatenation of the reviews written for a single movie in the IMDb dataset. This concatenation has the effect of increasing the consistency of each document as users tend to employ the same vocabulary to describe the same movie.

We build the vocabulary by extracting relevant words from the reviews in our dataset. Some words (for example stop words or words such as movie, film, and plot) appear in many reviews, but are not significant. We eliminate such words using a dictionary that we create manually from our dataset. For the remaining words, we apply *term frequency-Inverse document frequency* (TF-IDF) score as a filter. TF-IDF gives a high score for words that are frequent in a document but rare in other documents. We compute TF-IDF for each word of each review in the corpus and select the 10,000 words with the highest score.[1]

We select K empirically. After multiple experiments with values of K between 8 and 260, we observe that for $K \leq 30$, the topics mix several features as there are not enough topics to separate the different aspects expressed in the reviews. For $K \geq 150$, individual features get split over multiple topics. We chose $K = 128$ as a good compromise for movies.

Then we assign a weight to each word in the documents as follows:

$$\hat{f}_w^d = \frac{f_w^d}{\sqrt{N_d}} \tag{1}$$

where f_w^d is the number of occurrences of the word w in the document d, and N_d the number of reviews in the same document. The goal of this normalization is to reduce the imbalance between the most popular movies and the least popular ones.

We apply LDA to this new corpus. The assumption behind the LDA model is that each document d is generated from a mixture of topics denoted θ_d. With LDA, we infer the topic distribution θ_d of each document d and the topics ϕ. To compute similarity between movies, we use their topic distribution θ_d and the *Kullback Leibler divergence* similarity metric (KL). For two topic distributions θ^1 and θ^2 we have:

$$KL(\theta^1 \parallel \theta^2) = \sum_{i=1}^{k} \theta_i^1 log \frac{\theta_i^1}{\theta_i^2} \tag{2}$$

2.2 Genre-based sub-grouping

[1]After empirical verification, this number of words seems to give the best results for the data bases that we are using.

This section presents the method to split the list of the most similar movies to a movie m (extracted using the method presented in the previous section) into sublists. We also describe our method for extracting a title and a subtitle for each sublist.

We inspire our design on M-Go's interface, which presents four sublists: the first with the most similar movies and the other three grouped around more specific labels. Similarly, we generate four sublists. The first contains the N most similar movies to m. The three remaining sublists groups movies based on a pair of genres. We use the movie genres because they are familiar to users and because genres help create consistent sublists.

First, we generate the list of all the possible pairs of movie genres that appear in our dataset. We repeat the following steps to extract our sublists: for each pair (g1, g2) in our dataset, we will extract the N most similar movies that have at least g1 and g2 as genres. We then compute the average distance between m and the extracted movies. We select the pair of genres and the corresponding sublist that have the smallest average distance to m. For the next sublist, we eliminate the already selected pairs and movies in the previous sublists, in order to avoid redundancy among sublists.

Each sublist has a title and a subtitle. The title corresponds to the pair of movie genres (e.g., Action and Thriller). The subtitle is the set of words that best describes the movies of the sublist. We generate these words with LDA parameters. We define the score of a word as follows:

$$s(w) = \sum_{i=1}^{K} \phi_i[w]\bar{\theta}[i] \tag{3}$$

where K is the number of topics, and $\phi_i[w]$ is the weight of the word w in topic i. We also set:

$$\bar{\theta} = \frac{1}{N} \sum_{j \in currentsublist} \theta_j. \tag{4}$$

where N is the number of movies in each sublist and θ_j is the topic distribution of movie j. We compute the score of each word of the vocabulary. We keep the first twenty words with the highest scores. Figure 1 illustrates our method to generate the subtitle for a sublist with two similar movies.

2.3 Rating prediction

This section explains how we personalize the movies displayed in each sublist. We order the movies in each sublist according to the predicted ratings of the user on each movie. We predict ratings for movies users have not rated using matrix factorization [6]. We map both the users and the movies to the same space of dimension p. Each user u has a feature vector U_u and each movie i has feature vector V_i. The rating prediction model can be written as follows:

$$r_{u,i} \sim \mathcal{N}(V_i^T U_u, 1) \tag{5}$$

We learn the feature vectors (V_i and U_u) by solving the following regularized minimization problem:

$$\min_{U,V} \sum_{(u,i) \in observations} (r_{u,i} - V_i^T U_u)^2 + \lambda(\|V\|^2 + \|U\|^2) \tag{6}$$

The goal of the regularization is to avoid over-fitting the observed data. The regularization parameter λ is chosen

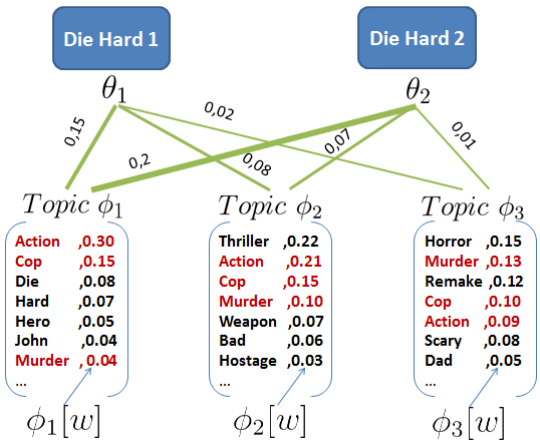

Figure 1: Example of our technique to extract subtitles for a sublist of recommended movies. The words in red are the selected words (Action, Murder, Cop) as they have the highest scores.

with cross-validation. We train our algorithm using gradient descent where we loop over all the ratings in our dataset.

3. PROTOTYPE

We create a web-based prototype, built using the IMDb dataset to demonstrate our approach. The user interface comes from Technicolor's VoD system, M-Go [2], to easy side-by-side comparisons. We apply the methods described in the previous section to identify groups of similar movies and subgroups. For each user in the IMDb dataset, we compute the predicted ratings using the matrix factorization model. We select pairs of genres to display to each user based on the preferred genres for the user. In our prototype we identify the preferred genres per user based on the most frequent movie genre pairs that the user has already seen. We then organize the recommended movies with a high rating prediction in sublists, according to the user most preferred genre pairs. When a user selects a movie from the sublists of recommended movies, our application suggests the similar movies presented under four sublists with the added list of words as described in §2.2. The sublists are personalized for each user by reordering the movies according to the users predicted ratings. Our implementation is available at: http://muse.inria.fr/tagit.

4. RESULTS

In this section, we first show an example of our subgrouping technique for a popular movie. Then, we discuss the feedback we got from comparing our prototype to Technicolor's M-Go service.

Example of subgrouping technique. Table 2 shows the sublists of similar movies we generate for the movie *Casino Royale* to illustrate the results of our method. We pick Casino Royal as the James Bond series is very popular, so we hope most readers will relate to it. The second row of the table presents the genre combination of two sublists; the third row presents the subtitle of each sublist (i.e., the words that describe the movies in each sublist); and the bottom part of the table presents the set of movies in each sublist.

Casino Royale (2006)	
Action and Thriller	Action and Adventure
shoot, blow, rambo, bullet, enemy, bruce, flick, hard, weak, gun, air, machine, pace, bad, kill, escape, impossible, hole, team, pack	cgi, superhero, hero, matrix, rescue, knight, cgus, trilogy, destroy, visual, terminator, batman, battle, earth, comic, blockbuster, super, original, special, superman
Taken 2	Indiana Jones and the Kingdom of the Crystal Skull
From Paris With Love	Captain America: The Winter Soldier
Lethal Weapon	Batman Begins
Mission Impossible III	X-Men Origins: Wolverine
Goldfinger	Transformers: Revenge of the Fallen
Live Free or Die Hard	Pacific Rim
The Dark Knight Rises	Captain America: The First Avenger
Rambo	The Scorpion King
The One	Pirates of the Caribbean: At World's End

Table 1: Sublists of movies similar to "Casino Royale".

We see that using our sub grouping method the movies in each sublist are consistent, which mean that they share common features. For example, in the first sublist, all the movies are action-packed movies. In the second list, almost all movies are sequel adventurous movies.

The words in the subtitle describe the movies in each sublist. These words contain descriptive words such as (visual and battle), and qualitative words for example (super and original), as users tend to use both qualitative and descriptive words while expressing their opinion about a movie. The subtitles help us make a distinction between sublists. For example, the first sublist is about action packed movies: *gun, shoot, kill, blow, escape*, whereas the second is about movies with visual effects (*cgi, visual*) and sequel movies (the fact that it contains *trilogy* movies such as: *Indiana Jones, Captain America, Batman, Transformers* and *The Pirates of the Caribbean*).

User feedback. During an internal event at Technicolor we demonstrated our prototype service side-by-side with the M-Go website. M-Go groups similar movies using manual labels provided by a third party service. We compare with M-Go because it is the existing service of Technicolor and most of the people attending the event were Technicolor employees. We had feedback from about 50 users who visited our stand; mainly Technicolor's employees expert in the movie domain.

We summarize the lessons learned from their feedback as follows. Almost all users agreed that it is hard to decide quickly on which movie to watch with existing systems. After seeing our sublists users said that they appreciated having the genre pair and the list of words per movies as this

[2]http://www.mgo.com

information gives them an extra description of the movies in this list; especially when they clicked on a word to see the most related movies to this word. Many users, however, considered the single-word tags hard to interpret because it required extra cognitive effort to combine single words into a meaningful concept; for example to go from *chase* and *car*, to *car chase*. Users also complained because of the presence of some generic words such as *person* or *etc*. These comments indicate that we must improve the method to filter words and that instead of presenting single words we should present short sentences that better capture comments in the reviews. Users also made a number of comments about how they interact with movie recommendation systems. They said that they usually look at the first or second sublist without scrolling down further, also they usually look just at the first displayed movies in each single sublist without looking at the rest of the sublist. This observation implies that we should aim at reducing the number of sublists and the number of movies displayed in each sublist.

5. RELATED WORK

Our work organizes the list of recommended movies into labeled sublists. Prior studies have also focused on organizing items in labeled lists, for example to group query results into labeled categories [11, 12]. We can see the title and subtitle of the sublists that our method outputs, as an explanation to movie recommendations. A number of studies have shown that adding explanation to the recommended movies improves the user satisfaction and loyalty to the system [5, 8, 9, 3]. Herlocker et al. [5] compared between multiple explanation interfaces and showed that using a simple interface outperformed complex explanation interfaces. The types of recommendation explanation include item-based, feature-based, and tag-based explanation. For example item-based explanation has the format: *We Recommend the movie X because you reviewed the movie Y*. Feature-based explanation has the format: *We Recommend the movie X because it contains the features Y and Z*. Finally, in tag-based explanation, the explanatory tags are the tags that characterize both the user profile and the movie profile. Chen et al. [3] showed that using tag-based and feature-based explanation performed better than item-based explanation. Therefore in our study we will combine feature-based and tag-based explanation. Chen et al. [3] also presented a method to explain recommendation using tags. The main difference with our approach is that we extract single-word tags automatically from user reviews rather than using the tags explicitly entered by users. We believe that the full reviews contain rich meta-data that is essential to group and label movies.

6. CONCLUSION AND FUTURE WORK

Movie recommenders generally give little insight on why a given movie is recommended. We propose to leverage crowd sourced reviews written by internet users to provide additional information about recommended movies. We use LDA applied to words found in reviews as a novel approach to content similarity that we combine to matrix factorization for personalization of recommendations. We attach to recommended movies a set of single-word tags that best describe these movies using reviewers' own vocabulary. This approach eliminates the manual tagging of movies used in most recommenders. Results are encouraging. We have shown the results of our method to around 50 people (experts in media creation and delivery). Most users found that the insight given by the title and subtitle was helpful as it helped them understand the common characteristics that describe the movies in each sublist. There are several interesting directions to improve this work. First, we plan to use meaningful expressions (e.g. *Complex story* or *Funny action*) rather than single-word tags in labeling the movie sublists. Our next objective is to personalize such vocabulary and use the most likely words a user would use to comment a movie. Last, we intend to evaluate our system with a large panel of real users, using an A/B testing for our system and M-Go in order to understand (1) if users prefer our automatic clustering and explanation over the manual one and (2) if they think our recommendation is more acute.

7. ACKNOWLEDGMENT

This project is supported by the European Community's Seventh Framework Programme (FP7/2007-2013) no. 611001 (User-Centric Networking).

8. REFERENCES

[1] X. Amatriain. Beyond data: from user information to business value through personalized recommendations and consumer science. In *Proc. of ACM CIKM*, 2013.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. volume 3. JMLR. org, 2003.

[3] W. Chen, W. Hsu, and M. L. Lee. Tagcloud-based explanation with feedback for recommender systems. In *Proc. of ACM SIGIR*, 2013.

[4] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proc. of ACM SIGKDD*, 2014.

[5] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proc. of ACM CSCW*, 2000.

[6] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[7] Nielsen. The total audience report: Q4 2014. http://www.nielsen.com/us/en/insights/reports/2015/the-total-audience-report-q4-2014.html.

[8] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Moviexplain: a recommender system with explanations. In *Proc. of ACM RecSys*, 2009.

[9] N. Tintarev and J. Masthoff. Effective explanations of recommendations: user-centered design. In *Proc. of ACM RecSys*, 2007.

[10] T. Vanderbilt. The science behind the netflix algorithms that decide what you'll watch next. http://www.wired.com/2013/08/qq_netflix-algorithm.

[11] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *Proc. of ACM SIGIR*, 2004.

[12] J. Zhao and J. He. Learning to generate labels for organizing search results from a domain-specified corpus. In *Proc. of IEEE/WIC/ACM Conference on Web Intelligence*, 2006.