Finding Similar Movies Using Topic Modeling

Aman Madaan, Ashish Mittal, *CS 728 Project Report, IIT Bombay*

April 2014

We propose a topic modeling based solution for finding movies *like* 2 given movies. We are after a much more complicated similarity, of the likes which the standard relations (by the same actor, by the same producer, etc) will fail to capture. The similarity we want to tap is plot based. We discuss the proposed solution, our implementation and the challenges involved.

1 Introduction

What is common between the Gran Torino, Scent of a Woman and Dead Poets Society? They all belong to the genre "Drama", but so do umpteen number of other movies. They have a disjoint set of actors, producers, directors and writers. They were not released in the same year. Yet, anyone who has watched all the three movies will instantly figure out what is similar; all these movies are about experienced people coaching young fellows, preparing them for the life ahead. Even though the crux of these movies, the basic line of thought, is the same, we cannot tell that they are similar by using the standard set of obvious relations. It is this similarity that we aim to model.

It is worth emphasizing that even though this report is based on finding similar movies, the problem is generic and is not limited to any particular domain.

We note that content based similarity will be the most amusing to the end user. Picking out similar objects based on facts about them is what the end user *expects* the computer to do; computers are supposed to be good at that. The similarity based on the very nature of objects will be indicative of intelligence.

1.1 Solution Outline

Movies are about a fixed set of topics, the problem is determining what these topics are and what do they mean. The problem of finding the cluster of words was solved by using LDA. Once the topic distributions were obtained, we used one of the two ranking algorithms to find the best solution, one based on KL divergence and another based on the top k words from dominating topics. We begin by describing the results of running LDA on the movies dataset by with flat alpha prior and 250 topics. We then describe the two ranking strategies in some detail. This is followed by a discussion on what went wrong and various fixes used to overcome the problem. We conclude with details about the front end and some anecdotal examples.

2 Getting the lines of thoughts

2.1 Movie Plots

We downloaded the Wikipedia dump and used Wikixmlj [5] , a Wikipedia scraping library to get the pages whose title satisfies the regular expression " .*film)". This is in accordance with the Wikipedia style manual [1] for movies.

We were able to extract a total of 16,139 movies. The distribution of words in the plot per movie is as shown.

2.2 Training

LDA was used to learn the words that belong to each of the clusters. Clearly, LDA does not know anything about the clusters we are looking for, and

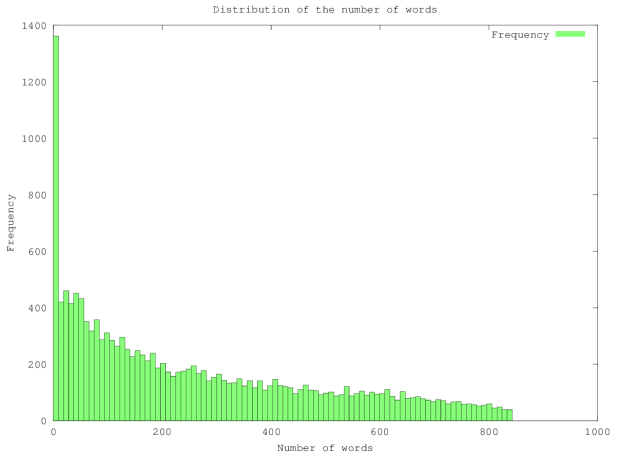

Figure 1: *Distribution of number of words in movie plots for 16139 movies*

depends purely on the occurrence and co occurrence of the words [2]. Table 1 shows some of the topics learned from the data.

3 Ranking

We used two different ranking strategies, each of which we describe in this section. Once the LDA model has been trained, we can calculate a topic posterior for any movie x. The topic posterior is essentially a probability distribution of topics for the given movie. Thus, we get, for each movie m, $P(\theta|m)$ where θ is the topic probability vector, with each term acting as a weight for that particular topic. We use this topic posterior in both the ranking strategies.

3.1 KL Divergence based ranking

Kullback - Leibler divergence[4] is an asymmetric measure of similarity between 2 probability distributions. Given 2 movies A and B, we first calculate their topic posterior, θ_A and θ_B and then combine and normalize the two. This is done by using one of the following strategies :

- **Max merge**
  $\theta_C = max(\theta_A, \theta_B)$
- **Avg merge**
  $\theta_C = (\theta_A + \theta_B)/2$

θ_C, the combined topic distribution, is finally normalized.

For all the movies, i, $KL\ Divergence(\theta_C, \theta_i)$ is calculated. The movies rankings in the results are

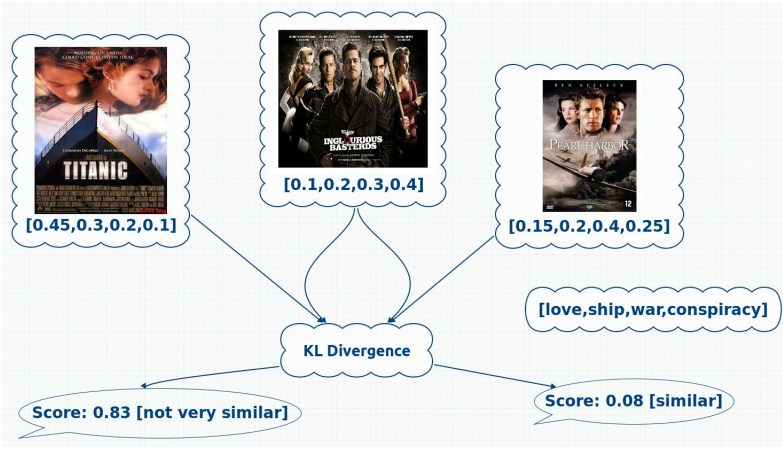

Figure 2: *Example of how KL Divergence yields similarity of two movies. Here [love, ship, war, conspiracy] are sample topics and the one with movies is the topic probability modeled by LDA for the movie.*

inversely proportional to this divergence score. Figure [2] shows a hypothetical ideal case.

3.2 Keyword based Ranking

The motivation for this method was that the movies similar to both the movies will have common top topics. For example, if the two movies are about [war, life struggle] and [life struggle, romance], (i.e. their θ posterior has high values corresponding to these topics) then returning movies that are about life struggle might be a good idea. For both the movies, the top common topics were chosen. The words were then given to an inverted index which returned the words corresponding to these movies. The returned set of movies were then ranked based on their topic posteriors, the movies for which the top common topics had high stakes were placed above those for which these numbers were lower.

4 Challenges

This section discusses some of the challenges we faced in implementation of the system.

- **Noisy Topics**

  Since we are using generative model for finding the topics, certain useless topics crop up after the training. These topics are mainly the names of characters, places etc. in the plot of the movie. Some sample noisy topics are shown in Table 2.

dinosaurs, dinosaur, valley, vineyard, brent, expedition, herd, plateau, yonggary, mammoth tyrannosaurus, prehistoric, rex, ape, volcano, napa, allosaurus, challenger, pack, pterodactyl, a
players, baseball, player, teams, league, basketball, hockey, sports, season, stadium, games, soccer, championship, teammates, victory, practice, goal, quarterback, minor, cup, c
jews, nazis, polish, holocaust, concentration, ss, poland, resistance, anti, jew, hitlers, germans, camps, gestapo, warsaw, auschwitz, czech, reich, occupied, propaganda, c

Table 1: *Some topics from the trained model*

bond, abby, buck, james, crystal, snake, reggie, raymond, omri, blofeld, python, ning, dolezal,kobayashi, xiaoqian, goldfinger, mi6, spectre, bonds, vishwanath
alex, sara, joanna, griffin, samantha, erin, alexs, breslin, mara, kemp, saras, clint, tyrone, friar, cory, tama, susanne, rain, ted, cortland

Table 2: *Some noisy topics from the trained model*

- **High Entropy Posteriors** There were some movies which had very even topic posteriors. Thus, their kl divergence was small from many movies and thus such movies kept showing up for many queries.

- **Insufficient Training Data** Although our corpora has 16139 movies, it is by no way representative of plethora of genres that define movies. We found that many movies were missing from our database. This was because not all the movie pages have (film) as part of their title. An immediate improvement is scanning through the document and looking for sufficient evidence to classify a document as a movie page. We plan to add this to the next version of this system.

- **How many topics?** Clearly, 250 was pulled out of no where. How many topics are there is a difficult figure to guess. Total number of genres perhaps can take us closer to the right solution.

5 Attempts at Solving Some Of the Challenges

We discuss some measure taken to tackle few of the aforementioned challenges.

5.1 Handling Noisy Topics

- **Stemming[6]** We stemmed the training corpora in order to reduce sparsity and to prevent shadowing of topics by a dominant word appearing in many forms.

- **Removal of common topics** Some topics that were present in the distribution of most movies were removed

- **Named Entity Removal** Since many of the noisy topic had names, we removed them using a Named Entity Recognition library [7].

6 Results

The system did give interesting results for some movies. Alas, for many movies, the problems listed above damped the better movies. We have added some screen shots of the web interface we added to the system. The output was a list of movies ranked by using one of the 2 measures above, along with short plot and keywords. We also display why this set of movies were chosen.

7 Conclusions

We expected the Named Entity removal would give better results, but then there were lot many noisy topics still left, which hints at using supervised approaches for better topic modeling. With the supervised LDA, we can model better topics related to the domain and thus perhaps get results which concur with the original idea that motivated this project.

Currently our ranking either checks for distribution similarity or key word similarity, but some mixed form of similarity measure should yield better results. Because that will capture the intuitiveness in probability distributions and *IR*ish keyword match to get best of both worlds.

References

[1] Wikipedia Style manual for movies http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Film

[2] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." the Journal of machine Learning research 3 (2003): 993-1022.

[3] http://mallet.cs.umass.edu/

[4] http://en.wikipedia.org/wiki/Kullback\%E2\%80\%93Leibler_divergence

[5] http://code.google.com/p/wikixmlj/

[6] http://snowball.tartarus.org/algorithms/porter/stemmer.html

[7] http://nlp.stanford.edu/software/CRF-NER.shtml

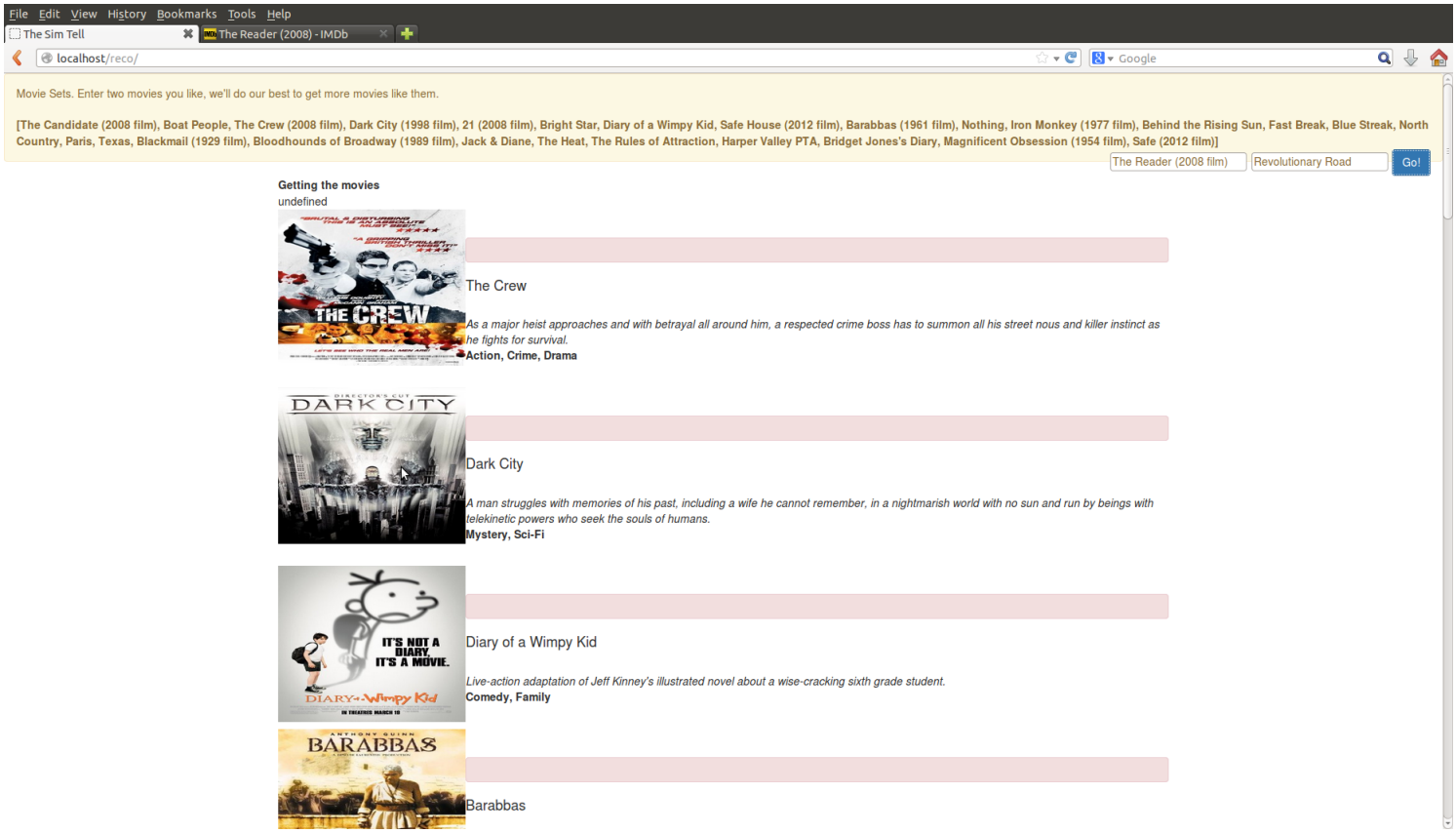

Figure 3: *Web interface to moviesim*

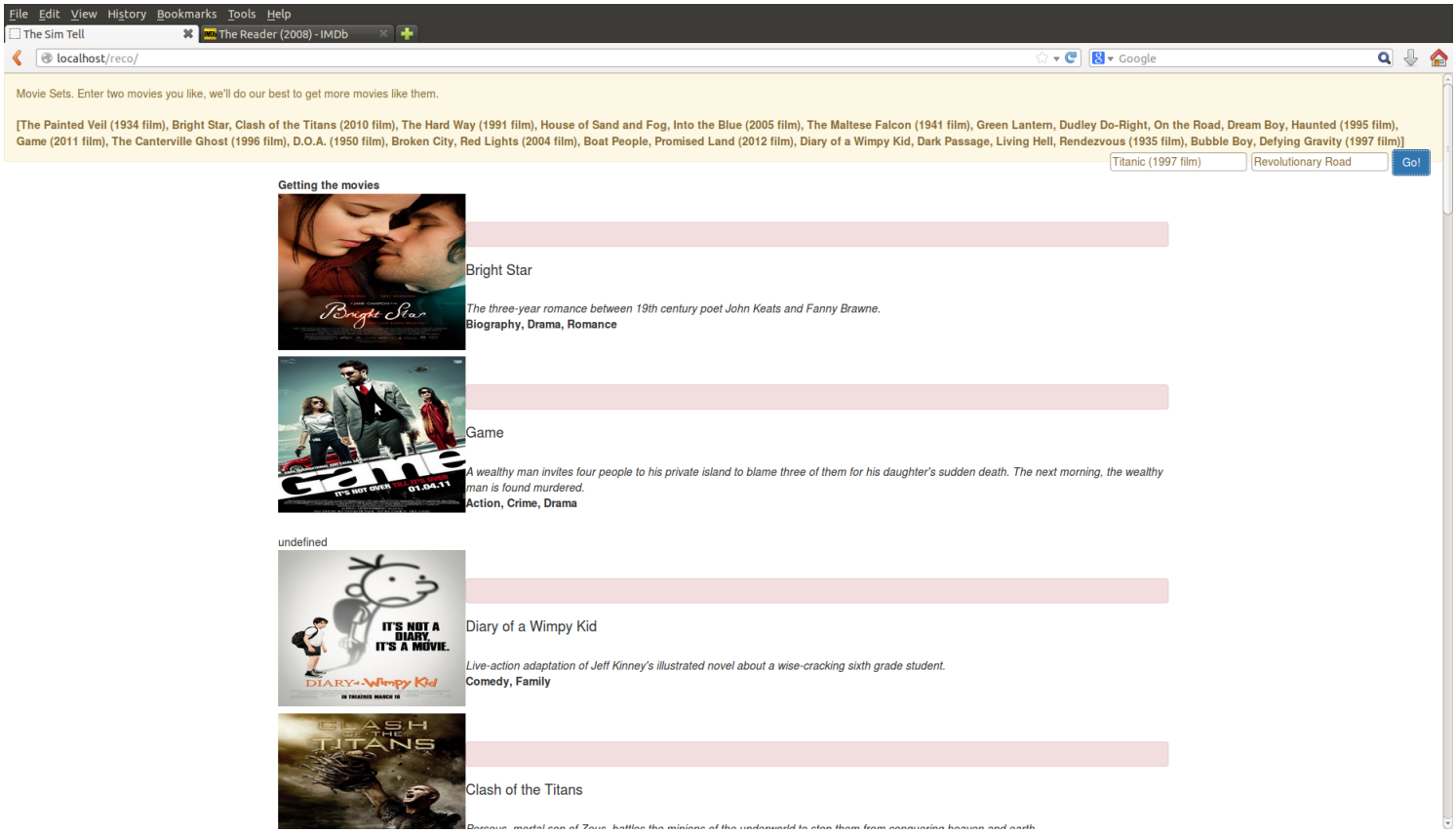

Figure 4: *Web interface to moviesim*

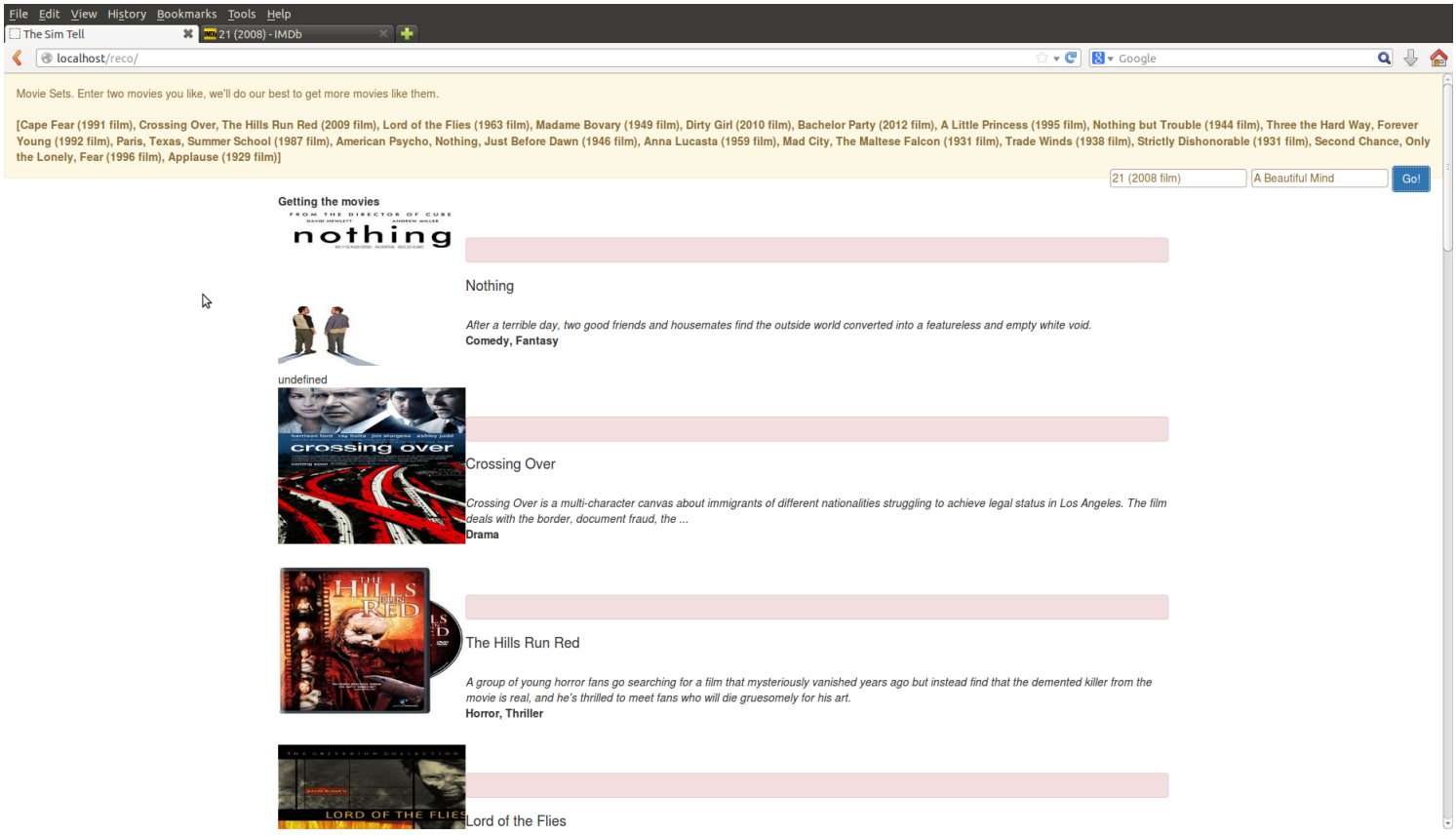

Figure 5: *Web interface to moviesim*