Proposal

Jens Rembe
rembejens@aol.com

Robin Kreuzig
robin.kreuzig@hotmail.de

January 5, 2018

1 Motivation and goal

We all know the discussion what movie should I watch today? When you watch television, we see a lot of advertising about *good* movies, but sometimes there are just bad or mainstream trash. Our goal is to search for insider tipps in the movie industry. That means we want to research movies, which doesnt have the money to do publicity as much as hollywood productions. We are searching for insider movies, which doesnt have so much specialeffects as the mainstream movies and we want to help independent studios which doesnt have the money to do advertising as hollywood productions but produce good movies.

We use *The Movie Data* dataset of the Kaggle website. It is our metadata and has over 45,000 movies. There are 26 million ratings from over 270,000 users. The last update was two months ago.
In our project we are streaming the movies in chronological order day for day in the pipeline. Every month we refresh our cluster and build a new static model for analyzing.

2 Analytics

For our project we need the files *movies_metadata.csv* and *credits.csv* from *The Movies Dataset*. In table 1 you can see the Format of the files.
In the project we need the features *budget, production_companies, release_date, revenue, vote_avarge, vote_count, cast* and *crew*. The movies are sorted chronological and will be published every day. That means that we commit many movies, sometimes just one or no movie a day. At the end we will publish a list with insider tipps and parameters for our cluster.

adult	belongs_to_collection	**budget**	genres
boolean	string	numeric	string
hompage	id	imdb_id	original_language
string	numeric	string	string
original_title	overview	popularity	poster_path
string	string	numeric	string
production_companies	production_countries	**release_date**	**revenue**
string	string	datetime	numeric
runtime	spoken_languages	status	tagline
numeric	string	string	string
title	video	**vote_average**	**vote_count**
string	boolean	numeric	numeric
cast	**crew**		
string	string		

Table 1: Summarized metadata from *movies_metadata.csv* and *credits.csv*

We use clustering for seperating all movies. The reason is that we dont know which movies are already professional or insider tipps. That means we have to analyze all movies and determine a cluster, which contains all insider tipps. In figure 1 you can see our concept to identify insider tipps. For example we check the movies on budget, movie rating and so on.

One problem of clustering is that the evaluation is not that simple as classification. Because of that we had an idea for an unfair method that should ensure well rated movies in our cluster:

1. For the evaluation we use a subset of not used movies.
2. We check with the parameters of our model if the movies are inside the cluster, but we ignore the rating feature in this classification. In this moment we got the unfair part, because we are cutting one parameter off. This can lead to more famous movies in our cluster.
3. If a movie is in the cluster we know it must have a high rating.
4. Now we can check if the rating of the movie is really that good.

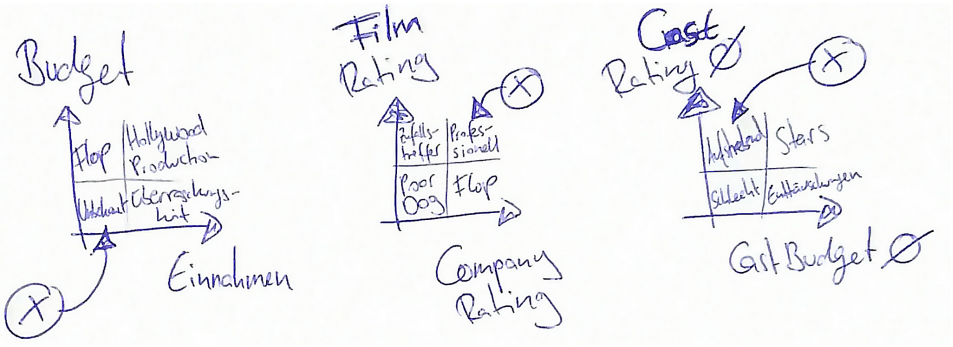

Figure 1: Cluster concept to identify insider tipps

3 Architectures

The following list shows the different parts of our architecture:

1. All movies are going to be streamed in CSV format of the particalar day with **Kafka**
2. The data is going to be customized in **Spark** and saved in **HDFS** with **Avro**-Format
3. Every month we refresh the cluster with **Spark** and the **scikit** library.
4. We use the parameters of the cluster as our model
5. With the parameters of the model and **Hive** as query language we produce a list of insider tipps.
6. When new movies are streamed with **Kafka**, **Spark** will determine with our model insider tipps.
7. The list of insider tipps and the parameters of the model will be safed in a **MySQL** database on fast storage.
8. We split the cluster areas in three two-dimensional graphs and plot them with **matplotlib**.
9. The data will be shown with **HTML/CSS/JS**.