EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, *SVERIGE 2017*

A sentiment analysis approach to manage the new item problem of Slope One

JONAS JOHANSSON

KENNETH RUNNMAN

KTH
SKOLAN FÖR DATAVETENSKAP OCH KOMMUNIKATION

A sentiment analysis approach to manage the new item problem of Slope One

JONAS JOHANSSON JONASJO3@KTH.SE

KENNETH RUNNMAN KRUNNMAN@KTH.SE

Bachelor in computer Science
Date: June 5, 2017
Supervisor: Kevin Smith
Examiner: Örjan Ekeberg
Swedish title: En ansats att använda attitydsanalys för att hantera problemet med nya föremål i Slope one
School of Computer Science and Communication

Abstract

This report targets a specific problem for recommender algorithms which is the new item problem and propose a method with sentiment analysis as the main tool. Collaborative filtering algorithms base their predictions on a database with users and their corresponding ratings to items. The new item problem occurs when a new item is introduced in the database because the item has no ratings. The item will therefore be unavailable as a recommendation for the users until it has gathered some ratings.

Products that can be rated by users in the online community often has experts that get access to these products before its release date for the consumers, this can be taken advantage of in recommender systems. The experts can be used as initial guides for predictions. The method that is used in this report relies on sentiment analysis to translate written reviews by experts into a rating based on the sentiment of the text. This way when a new item is added it is also added with the ratings of experts in the field.

The result from this study shows that the recommender algorithm slope one can generate more reliable recommendations with a group of expert users than without when a new item is added to the database. The expert users that is added must have ratings for other items as well as the ratings for the new item to get more accurate recommendations.

Sammanfattning

Denna rapport studerar påverkan av problemet med nya objekt i rekommendationsalgoritmen Slope One och en metod föreslås i rapporten för att lösa det specifika problemet. Problemet uppstår när ett nytt objekt läggs till i en databas då det inte finns några betyg som getts till objektet/produkten. Då rekommendationsalgoritmer som Slope One baserar sina rekommendationer på relationerna mellan användares betyg av filmer så blir träffsäkerheten låg för en rekommendation av en film med få betyg. Metoden som föreslås i rapporten involverar attitydanalys som det huvudsakliga verktyget för att få information som kan ersätta faktiska betyg som användare gett en produkt.

När produkter kan bli betygsatta av användare på olika forum på internet så finns det ofta experter får tillgång till produkten innan den släpps till omvärlden, den information som dessa experter har kan användas för att fylla det informationsgap som finns när ett nytt objekt läggs till. Dessa experter kommer då initiellt att användas som guide för rekomendationssystemet. Så när ett nytt objekt läggs till så görs det tillsammans med betyg från experter för att få mer träffsäkra rekomendationer.

Resultatet från denna studie visar att Slope One genererar mer träffsäkra rekommendationer då en ny produkt läggs till i databasen med ett antal betyg som genererats genom attitydanalysanalys på experters textrecensioner. Det är värt att notera att ett betyg enbart för dessa expertanvändare inte håller utan experterna måste ha betyg av andra produkter inom samma område för kunna influera rekommendationer för den nya produkten.

Contents

Chapter 1

Introduction

The last decade has seen a growing trend towards a more personalised web experience. The rise of social media and search engines during the world wide web's short lifespan has contributed to the trend. Personalised web experience refers to customising websites according to the individual who is visiting the website. Predictive analytics is one of the tools for making the web more personal for individual users, specifically it is the study of analysing both real-time and historical data to make predictions [9]. This field has been identified as one of the fastest growing trends in technology and it is believed that more businesses will focus less on the past and instead focus on predicting the future [2].

There is also a strong economic incentive for businesses to advance and make use of the technology. This is shown by the fact that a major income among the companies is through advertised searches and that advertisers are only charged if a user clicks on one of the advertised search items. One example is Google, which had a total revenue of $6 billion in 2005 and 98% of the revenue were from advertised searches [1]. In 2015 the figure shows a tenfold increase with 67 million dollars in ad revenue which was 90 % of its total revenue [12].

The usage for a more personalised web is immense and therefore many businesses are trying to create more efficient and accurate algorithms for recommendations. This is shown by *the Netflix challenge* in 2009, which was a competition where the participants were competing for $1M by creating the best algorithm in terms of accurately predicting films based on previous ratings [3].

One of the techniques used for recommendations is collaborative

filtering (CF) which uses information about users to create automatic predictions. The method finds users with similar interests, it therefore requires large number of existing data to find users with a similar preference in order to generate accurate recommendations. When sufficient information does not exist, a problem called cold start will arise. This problem can occur in a few different situations. For example a new user usually may not have enough information for the CF algorithm to be able to make predictions for that user [10].

In this report we will examine if sentiment analysis (SA) could be used in order to fill to information gap during a cold start. SA which is a method to determine the attitude of a person on a specific topic based on data such as text written by the person [8].

1.1 Problem Statement

The lack of available ratings for a specific product, which is common before its release, is a cause for unreliable recommendations by Slope One for that product. We propose to use SA on early text reviews that is available before the release of the product in order to generate ratings. These ratings can therefore act as a stand in while a product does not have the number of ratings needed in order to generate reliable recommendations.

Before user ratings are available, can we use text reviews to generate more reliable recommendations?

We intend to compare generated ratings for a specific product across a database with and without SA by studying the error of the generated ratings.

1.2 Scope

Due to the wide availability of data, this study looks at films and their corresponding reviews. The reviews vary in complexity where some of them are professionally put together and some are written in a more simple fashion. In order to get more accurate results, reviews with too simple of a structure, e.g. with few characters or a simple quote, are not included in the database. Another reason to use databases that in-

cludes ratings for films is that it is prevalent in the film industry that professional reviewers get access to films before the public. We therefore also restrict the written texts that we will perform SA on to have been written by either professional reviewers or popular reviewers on the web.

Chapter 2

Background

In this chapter we will explain the key concepts of the report. We divide the chapter into four sections. Firstly how CF works and the challenges that comes with it that is important for the reader to understand. SA is also explained further and how other methods can be used to increase the accuracy of SA. We will thereafter shortly introduce and explain the origin of the data that is later used. And lastly explain a estimator used to determine error.

2.1 Collaborative filtering

The original implementation of CF is that recommendations to the active user is based on what other users with similar tastes have liked in the past. This is the reason why CF also is referred to as "people-to-people correlation" [11].

Item-based collaborative filtering

The item-centric approach to CF which follows the idea that if users shares interest in an item and many of them has interest in another, it will likely be a good recommendation for the uses who has not shown any activity for the item. The approach can be reduced to two steps [11]:

1. It uses a user-item matrix which show similarities between different items.

2. Compare the preference of the current user with the matrix and matching the user's data.

2.1.1 Slope one

Slope One, first proposed by Lemire and Maclachlan, is a familiy of algorithms used for item-based CF. It uses a simpler form of regression where $\alpha = 1$

$$F(x) = x + \beta$$

hence the name, Slope One. Slope One has proved, in some cases, to outperform it's linear regression counterparts [5]. The simpler form uses fewer regressors and require less memory and also results in being computed faster. It is also easier to implement due to being less complex.

Since Slope One uses linear regression it also reduces the effect of overfitting. While other types of regression may represent the current data better than Slope One, it does not always represent new data as good and is thus more prone to error.

Slope One generates recommendations as the following example shows. If we want to predict what a user B would rate item J given that B gave item I a rating of 2, and some other user A gave item I a rating of 1 and item J a rating of 1.5.

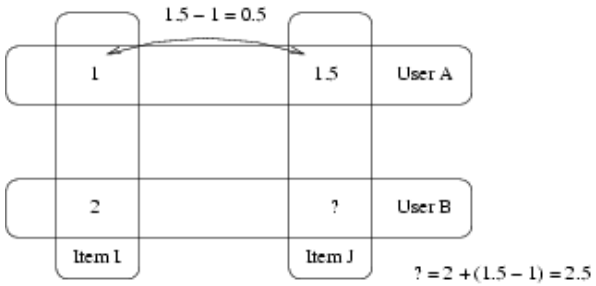

Figure 2.1: Predicting user B's ratings based on user A [5]

1. $\triangle A_{I \to J} = A_J - A_I = 0.5$
2. $\triangle B_{I \to J} = \triangle A_{I \to J} \Rightarrow B_J = B_I + \triangle B_{I \to J} = 2.5$

Thus, according to Slope One with the given information, user B would give item J a rating of 2.5.

Weighted Slope one

One of the disadvantages with slope one is that does not take the number of ratings into consideration. If for example there are a lot more users who have rated a pair of items which we can call A than the pair of items B. And a user have rated an item that is included in both pair A and B. It is more likely that the item in pair A is the better recommendation for the user. The weighted slope one takes this into consideration and it is defined as [5]:

2.1.2 Cold start

One of the biggest challenges in CF is cold start which is caused by data sparsity. Insufficient data for an item or an user when a CF algorithms is trying to predict recommendation will prove to inaccurate in most cases. Therefore it is important that users and item has sufficient information in order for CF to work [10].

New item problem

A particular problem for cold start is when there is an item with little to no ratings as CF will be unable to recommend other items based on the item with few ratings.

2.2 Sentiment analysis

SA is the method to identify the subjectivity in a text with computers. The subjectivity in the text can be described as the author's opinion or emotion towards a subject. SA involves estimation of the sentiment in texts, for example a text could be classified as positive, neutral and negative [8].

2.2.1 Semantic analysis

Semantic Analysis is the study of understanding linguistic input [4]. The use of semantic analysis is important in SA, as texts must be processed before performing the SA [7]. As the text sometimes contain

words or phrases that the SA algorithm will interpret as positive or negative even though they don't affect the sentiment of the text. This happens since the algorithm does not have any form of context regarding the text. The text must thus be tokenised, meaning that parts of it must be replaced by different generic and neutral tokens instead. Another use of semantic analysis in SA is negation of words.

For example

"Calling this film bad would be insane"

Where using only SA would generate a more negative result, due to interpreting both 'bad' and 'insane' as something negative, whereas combining both methods would result in a more positive result. When looking at the sentence as a whole, 'bad' and 'insane' become neutral.

2.2.2 TextBlob

There are a few libraries available for performing SA, one of them is the TextBlob library found in Python. Most libraries are provided as paid online APIs. TextBlob, however, is free and easily available.

2.3 film data

2.3.1 Letterboxd

Letterboxd describes themselves as a social network for sharing taste in films. Users can use the website as a diary to share their opinion or just keep track of the films that they watch through ratings and reviews [6].

2.3.2 MovieLens

MovieLens is a website used to get personalised film recommendations. They also host a number of different data sets that are free for the public. This report will be using the MovieLens 20M data set.

2.4 Error estimator

There are a few ways of estimating the error of a set of data. Two of the most common methods for doing this is using the mean absolute

error (MAE) or the root-mean-square error (RMSE). Regarding studies on recommender systems, RMSE tends to be the more common, but is sometimes combined with MAE.

RMSE

The formula for RMSE is,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2},$$

where n is the number of measurements and e_i the error of a specific measurement.

Chapter 3

Method

There are 3 parts needed to determine if the proposed method is viable.

1. One data set with reviews and one with normal users.
2. SA of the reviews to convert them to a rating.
3. An experiment to test the idea.

3.1 Database

The database consists of entries from two different data sets, those from Letterboxd will simulate experts, and those from MovieLens will be the normal users.
The experts were chosen to be the 25 most popular reviewers on Letterboxd at the time (March 2017). A web scraper was then used to gather all of their reviews and corresponding ratings.
The normal users were represented by the entire MovieLens data set.

All data is kept in a MySQL database.

3.2 The usage of Sentiment Analysis

Before the SA is performed, the review have to be tokenised. The following review:

"Bad Santa may be one of my favourite Christmas films of all time."

will receive a more negative result than it should since part of the title ("Bad") will be interpreted as something negative. To counteract this, the title "Bad Santa" would be replaced with the token "FILM", resulting in:

"film may be one of my favourite Christmas films of all time."

Other potential pitfalls, such as negators ("not good", where "not" change the sentiment of "good"), are taken into account by default in the used algorithm.

When performing the SA of the reviews, the following code-snippet written in Python was used

```
blob = TextBlob( review )
totalPolarity = 0
n = 0
for sentence in blob.sentences:
    if sentence.sentiment.subjectivity > 0.5:
        totalPolarity += sentence.sentiment.polarity
        n += 1
rating = totalPolarity / n
```

We used the TextBlob library, which analysed the text. The rating was then calculated as the average polarity of the text.
The code checks if the subjectivity of each sentence is greater than 0.5, the reasoning for this is that we wanted to get a rating to better match the reviewers subjective thoughts.
The generated rating is in the range $[-1, 1]$, and is transformed to be in the range of $[1, 10]$ to match the ratings of the normal users.

3.3 Limitations

Due to our lack of computing power, some restrictions had to be made to limit the size of the database.
To limit the number of films, only films with 12 or more ratings were kept.
To limit the number of users, only those who had rated at least one of the chosen films were kept. The normal users were also limited to a

maximum of 1000
To limit the number of ratings, normal users were limited to a maximum of 60 ratings each.

The resulting database contained 17 films with $91K$ ratings from 1000 normal users and 21 experts. The density of the database was 1.505%.

3.4 Experiment

To answer the question we have to run two experiments, one control test without experts, and one with experts. We ran each experiment 200 times with randomised parameters.

We have 3 groups: A, B and C. We also have 2 inputs, a film and a number t (how many ratings it should have).

Control test

1. All normal users who have rated the given film are put into group A, those who have not rated the film are put into group B.
2. t random users from group A are moved to group B, and the rest are moved to group C.
3. All users in group C loose their rating for the given film.
4. Predictions, based on group B, are then generated for every user in group C with weighted Slope one.

RMSE is then used as a measurement to compare the error between the predicted ratings and the actual ones.

Expert test

The exact same experiment as the control test, but with the experts added to group B.

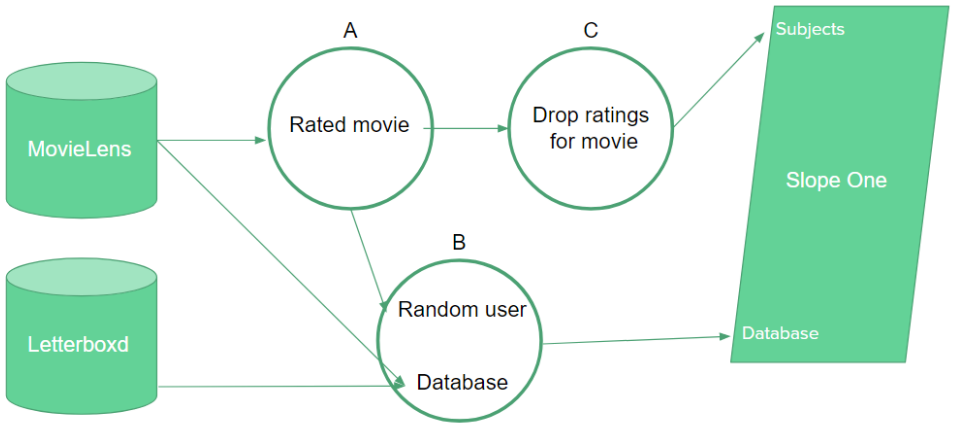

Figure 3.1: Experiment overview

Chapter 4

Results

4.1 SA rating difference

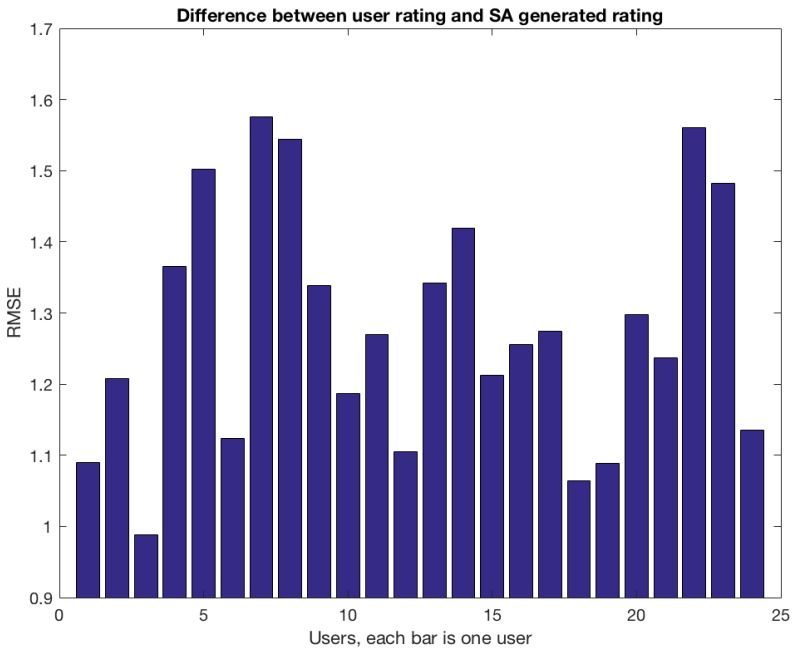

Figure 4.1: Difference between user rating and SA generated rating

The x-axis in figure 4.1 represents each user and the y-axis represents the error (RMSE) between the sentiment rating and the actual rating. The mean RMSE value is 1.2777. Lower is better.

Ex. Reviews from Letterboxd user 2 has an average RMSE value of approximately 1.2 when performing SA and comparing the result to their actual ratings.

This is to show how the SA affect the final result with our approach.

4.2 Recommendation accuracy

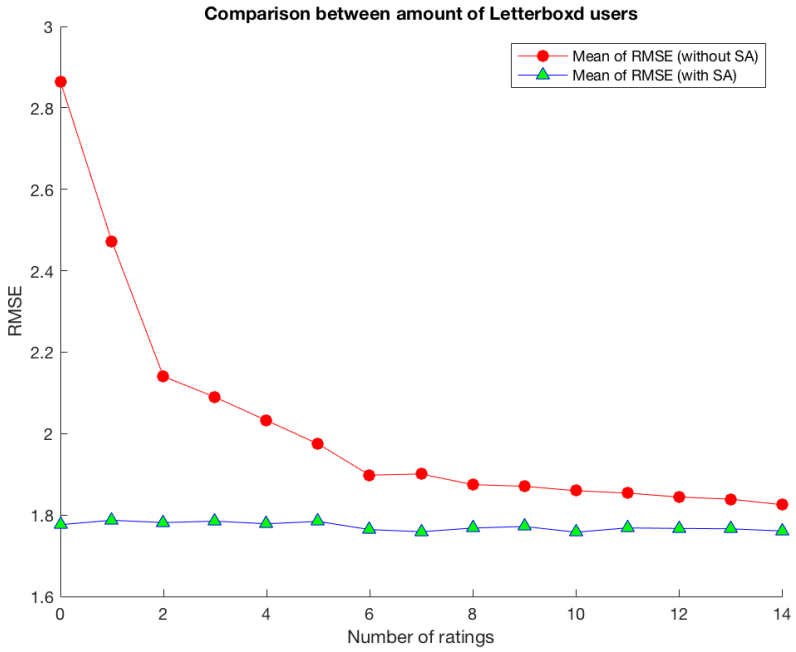

Figure 4.2: Comparison of error in cold start with and without SA

The graph 4.2 illustrates the mean RMSE error for all films tested when generating recommendations with, and without, SA. The x-axis represents the number of previous ratings, for the film being predicted, stored in the database. Lower is better.

4.2 also shows reliable predictions when there are no users in the database that has rated the film, except for the imported SA generated ratings from experts who influence the recommendations. The difference is not noticeable between 0 and 14 users in the database with SA.

But the new item problem is noticeable without the SA, as the value of RMSE decreases as more users have rated the film.

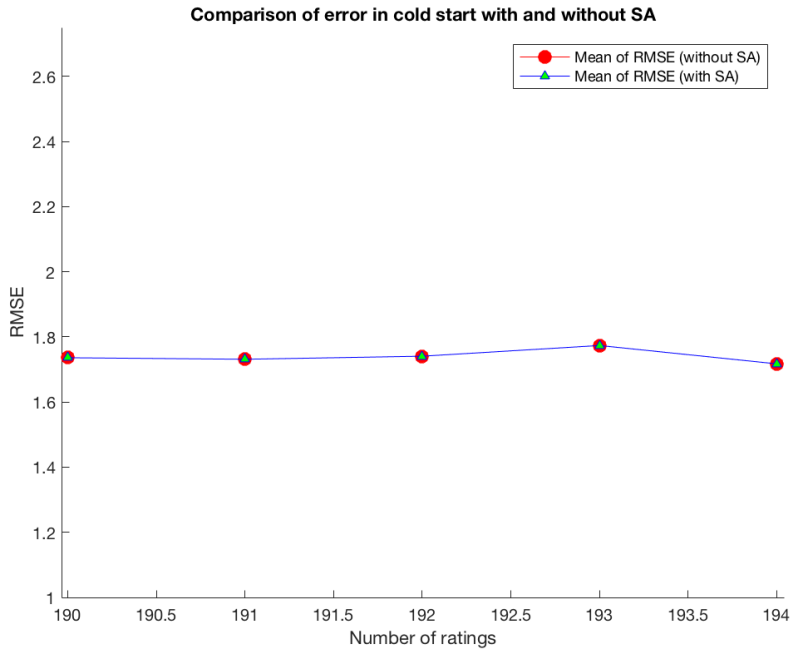

Figure 4.3: Comparison of error in cold start with and without SA

The graph 4.3 is an extension of the previous graph 4.2, with the x-axis shifted. With this graph we want to acknowledge that both values from the method with SA and without SA converge when many users have rated the film.

4.2.1 Number of expert users

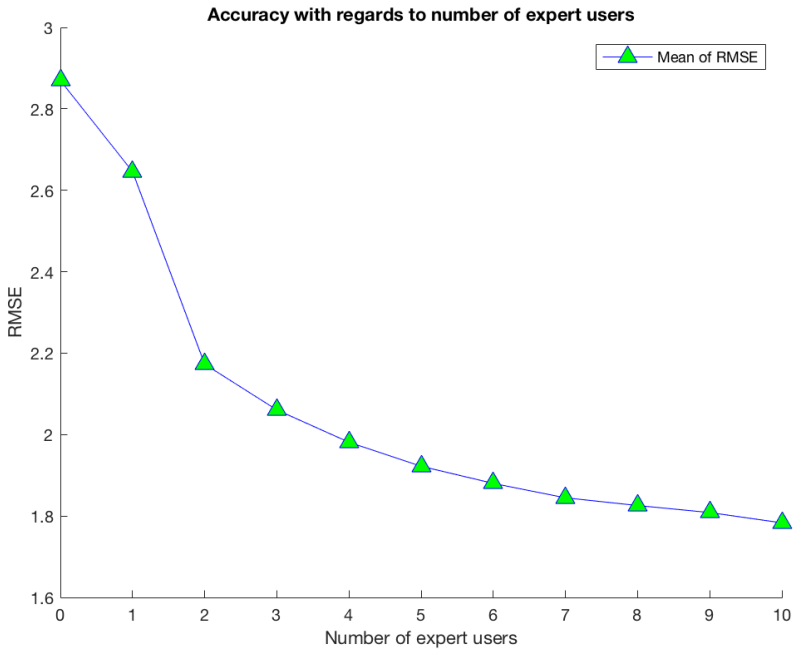

Figure 4.4: Accuracy with regards to number of expert users

The main experiment was performed with as many experts as possible. The above graph (4.4) shows the accuracy depending on the number of experts used. No normal users where used.
The graph clearly shows that more experts result in better accuracy. At around 8 experts, the graph starts to stabilise, thus meaning that about 8 experts are needed for this method to be effective.

Chapter 5

Discussion

Cold start is one of the major challenges with CF. In this report we proposed that SA could be used in order to fill the information gap during cold start. With the proposed method we then examined if we can use text reviews to generate more reliable recommendations for new items.

5.1 Results

Our hypothesis before we conducted the experiments was that we would find a cross section between the the tests with SA and the test without. This did not occur during the experiments and instead they converged together. The reason we thought that there would be a cross section is that SA is far from perfect, and while it would help in the beginning because of the lack of information, we thought that it would just worsen the predictions after a good chunk of actual ratings were introduced to the database.

SA could be used to standardise every rating, for example one users idea of a rating 7 might be different from another users. By analysing the polarity in each text for a user and comparing it to another it could be decided if a user is more positive or negative than another user when having the same rating. By taking this into account improvement could be possible.

The results indicate that we need about 8 experts for the method to be effective. While this is true for our conditions, we cannot make any general conclusion as to how many experts are needed nor how many ratings they need. As the tests did not vary the number of ratings per experts or data density.

5.2 Method

The results are affected by several factors, mostly due to time constraints. Regarding the reviews, it was previously mentioned that certain words and/or whole sentences taken out of context will affect the SA of the review. Some of these problems were taken care of but a few still remain. It should be mentioned, however, that these problems were not solved completely. If a film "Bad Santa" has a sequel (or a prequel), eg. "Bad Santa 2", this will not be converted into any token and will thus affect the results. This also applies if another film than the reviewed one was referenced. Titles with common words, such as "Absurd", could result in words that are not referencing the title to be replaced with a token, although this is very rare in the data set used during this study and will thus not have any significant impact on the results presented. It was observed that several of the reviews used in this study simply contained a quote from the film. Performing of SA on these reviews do not reflect the actual sentiments the user would have of the film should a quote not have been used.
It was also observed that several reviews were very short. If a review would consist solely of "Perfection.", the SA would most likely generate a rating lower than the actual rating.

5.2.1 Subjects

This entire study was performed on users that either write film reviews for a living, or do it often and well enough to be considered professional reviewers. It should be noted that SA of their reviews could better represent reality than performing SA on amateur writers would. This is because of the professional reviewers possibly being better at conveying their thoughts more accurately. However, it could also be that professional reviewers tend to be more formal than the amateur one. This would result in more positive results when performing SA and thus resulting in ratings of around 7, which approximately is the

average film rating for all users. The closer a rating is to the average, the more likely it is that a recommendation will be successful. This could be one reason that the method ends up with such a constant RMSE value.

5.2.2 Sentiment analysis

Sentiment, and semantic, analysis is still in it's infancy and subject of research. Current methods of performing these types of analysis are not that accurate. Sometimes the analysis will produce a result far from reality and thus affect the results. The average error in SA during the experiments were 1.2777, which certainly affect the outcome.

5.2.3 Limitations

One of the source errors of this study is the small database and that normal users had a restriction of a maximum of 60 ratings each while experts had an average of 1600 ratings each. This will clearly affect the results as experts, on average, share more films with normal users than normal users do. It is, however, not far fetched to think that experts rate more films than normal users.

Random chance

One of the parts most affected by random chance is the SA. Because of the somewhat small database used, the chosen reviews may be more positive or negative than the average review.

Chapter 6

Conclusion

From the graphs in the results section it is clear that the proposed method is viable. While the approach was successful with the currently available tools, it is possible that an improved analysis of the reviews, better parameters, or more data will generate different results. While the results suggest the idea works, more research is needed before any general rules can be made on how many expert users, as well as ratings, are needed.

6.1 Future work

Should any future study build on this one, we suggest to focus on following points:

- Bigger database.
- Focus on better sentiment analysis.
- Compare different collaborative filtering.

Bibliography

[1] B. Edelman, M. Ostrovsky, and M. Schwarz. *Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords*. Stanford University, 2007.

[2] IEEE. *IEEE-CS Unveils Top 10 Technology Trends for 2015*. IEEE, 2014.

[3] Netflix Inc. *Netflix Prize*. http://www.netflixprize.com/index.html. Accessed: 2017-05-12. 2009.

[4] A. Klapuri. *Semantic analysis of text and speech*. Institute of Signal Processing, Tampere University of Technology, Finland, 2007. URL: https://www.cs.tut.fi/sgn/arg/klap/introduction-semantics.pdf.

[5] D. Lemire and A. Maclachlan. *Slope One Predictors for Online Rating-Based Collaborative Filtering*. Cornell University, 2007.

[6] Letterboxd. *About Letterboxd*. https://letterboxd.com/about/frequent-questions/. Accessed: 2017-06-05. 2017.

[7] C. W. Leung, S. C. Chan, and F. Chung. *Integrating Collaborative Filtering and Sentiment Analysis: A Rating Inference Approach.* Singapore Management University, 2006.

[8] W. Medhat, A. Hassan, and A. Korashy. "Sentiment analysis algorithms and applications: A survey". In: *Ain Shams Engineering Journal* 5.4 (2014), pp. 1093–1113. ISSN: 2090-4479. DOI: http://dx.doi.org/10.1016/j.asej.2014.04.011. URL: http://www.sciencedirect.com/science/article/pii/S2090447914000550.

[9] C. Nyce. *Predictive Analytics White Paper*. Tech. rep. American Institute for Chartered Property Casualty Underwriters/Insurance Institute of America, 2007.

[10] F. Ricci, L. Rokach, and B. Shapira. "Introduction to Recommender Systems Handbook." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Springer, 2011, pp. 1–35. ISBN: 978-0-387-85819-7. URL: http://dblp.uni-trier.de/db/reference/rsh/rsh2011.html#RicciRS11.

[11] B. Sarwar et al. "Item-based Collaborative Filtering Recommendation Algorithms". In: 2001, pp. 285–295.

[12] United States Securities and Exchange Comission. *Alphabet Inc. and Google Inc. Form 10-K For the Fiscal Year Ended December 31, 2015*. 2016. URL: https://abc.xyz/investor/pdf/20151231_alphabet_10K.pdf.

Appendix A

slopeone.py

```
#!/usr/bin/env python

# this program implements slope one predictors
#
# D. Lemire and A. Maclachlan, "Slope One Predictors for
# Online Rating-Based
# Collaborative Filtering", In SIAM Data Mining (SDM'05),
# Newport Beach,
# California, April 21-23, 2005.

import functools

def cmp(a, b):
    if a < b:
        return -1
    if a == b:
        return 0
    return 1

# slope one scheme
class slopeone:
    def __init__(self, users, items):
        self._users = users
        self._items = items

    def _ave(self):
```

```
        self._ave = {}
        for i in range(len(self._users)):
            user = self._users[i]
            if len(user) == 0:
                self._ave[i] = 0
            else:
                self._ave[i] = sum(user.values()) / float(len(user))

    def _avedev(self):
        self._dev = {}

        num = len(self._items)
        for i in range(num):
            for j in range(i + 1, num):
                item1 = self._items[i]
                item2 = self._items[j]

                r = 0.0
                n = 0
                for k in range(len(self._users)):
                    user = self._users[k]
                    if item1 in user and item2 in user:
                        r += user[item2] - user[item1]
                        n += 1

                if n > 0:
                    r /= float(n)

                self._dev[(item1, item2)] = (r, n)
                self._dev[(item2, item1)] = (-r, n)

    def _predict(self, user, item):
        if item in user:
            return user[item]

        if len(user) == 0:
            return 0

        r = 0.0
```

```
            for key in list(user.keys()):
                dev = self._dev[(key, item)][0]
                r += dev + user[key]

            return r / len(user)

        def _doFirst(self, user):
            pass

        def recommends(self, user):
            self._doFirst(user)

            self._ave()
            self._avedev()

            items = [item for item in self._items if item not in user

            result = []
            for item in items:
                p = self._predict(user, item)
                result.append((item, p))

            sorted(result, key=functools.cmp_to_key( cmp ))
            return result


    # wheighted slope one scheme
    class wslopeone(slopeone):
        def _predict(self, user, item):
            if item in user:
                return user[item]

            if len(user) == 0:
                return 0

            r1 = 0.0
            r2 = 0.0
            for key in list(user.keys()):
                dev, n = self._dev[(key, item)]
```

```
            r1 += (dev + user[key]) * n
            r2 += n

        try:
            return r1 / r2
        except:
            return 0
```