Four different ways to predict reviews' rating through text analysis

Daniele Dellagiacoma and Davide Lissoni
Department of Information Engineering and Computer Science
University of Trento
via Sommarive 14, 38123, Trento, Italy
{daniele.dellagiacoma, davide.lissoni}@studenti.unitn.it

ABSTRACT
Nowadays, organizations are accumulating vast and growing amounts of data in different formats, storing them in huge databases. This data can provide useful information through patterns, associations or relationships. Therefore, data mining has emerged to analyse a huge amount of data in automatic or semi-automatic way, in order to discover as much information as possible about them. Moreover, also the amount of online reviews is growing rapidly and their role in e-commerce has become increasingly important. Both academics and practitioners is aiming at extracting meaningful information from these reviews through text mining techniques. In this paper we describe our implementation of an algorithm used to predict the rating of an input text review, based on a list of positive and negative opinion words.

Keywords
Data Mining; Rating Prediction; Sentiment Analysis

1. INTRODUCTION
In the last few years, progress in digital data acquisition and storage technology has led in the growth of huge datasets [1]. For this reason, data mining has recently emerged as an interdisciplinary subfield of computer science. Data mining has the goal of extracting as much information as possible from a dataset and transform it into an understandable and useful structure for further use [2]. The automatic or semi-automatic analysis of huge quantities of data aims to discover interesting patterns such as groups of data records, unusual records and dependencies. These patterns may be used in further analysis or in machine learning and predictive analytics.

Data mining can be used in a predictive way for a great variety of different applications. Indeed, it is widely used in business, science research and government security. Moreover, some of the major database vendors have already taken steps to ensure that their platform incorporate data mining techniques [3]. For example, Oracle's Data Mining Suite [4] and Microsoft's SQL Server [5] implement some algorithms and techniques of data mining.

One important step toward building a efficient data mining program is the gathering of data. This process of gathering data for analysis is critical to the eventual success of any data mining project and many companies already perform this task. Starting with a dataset it need to be used an algorithm to extract the desired information. The two most popular algorithms used in data mining are regression and classification.

Regression is used to measure the relation between one dependent variable and one or more other independent variables. Instead, classification aims to discover to which set of categories a new element belongs. They are both considered exponents of supervised learning, a subfield of machine learning.

Another relevant technique related to data mining is text mining. It has the purpose of extracting meaningful information from a text. Nowadays, more and more users can freely express their opinions about products or services on Internet. These reviews can be very useful both to other users for making informed decisions and to companies for upgrading their services [6]. Moreover, the amount of reviews grows rapidly and understanding their role in e-commerce has become an important topic both for academics and practitioners [7].

In the following sections is described our idea and the implementations of work that was done. We thought about an algorithm which could be used to predict the rating of an input text review, based on all others rating reviews in the dataset. We compare the input review with each other review in the dataset to determine which are the most similar to the input review. We determine three different similarities using a set of positive and negative opinion words. Finally, we use the computed similarities and linear regression for predicting the ratings of the input review.

This paper is organized as follows: section 2 describes related works which have been used as starting point for our implementation of the project Section 3 presents the problem we are trying to solve whereas section 4 provides our solution and implementation. Section 5 describes all the experiments we performed using different review datasets and input texts. Finally, we conclude in section 6.

2. RELATED WORK
Opinion words are words, which are often used to express positive and negative opinions. For example, marvelous, amazing, cool and excellent can be considered positive opinion words whereas terrible, inadequate and bad are negative opinion words. However, most opinion words are adjectives and adverbs but also nouns and verbs can be used to indicate opinions. Opinion words are used in many sentiment classification task [8].

We used a pre-existing list of positive and negative opinion words as starting point for our program. This list has been compiled over many years by Bing Liu and Minqing Hu, starting from their first paper in 2004 [9]. The list is composed by 2006 positive opinion words and 4782 negative opinion words, all of them concern the English language.

Determine which and how many opinion words are present in each review of the dataset is only the first step of our work. We also need to determine which are the review in the dataset most similar to the input review. Various similarity measures have been formulated throughout the years, each with its own strengths and weakness [10]. We decided to use three different similarity

measures in order to quantify the similarity between the vector of opinion words in the input review and the vector of opinion words in each other review.

We use as first measure cosine similarity, which measures the cosine of the angle between two vectors. Its outcome is bounded between 0 and 1. Moreover, it is commonly used in text mining to determine the similarity between two texts, counting the number of occurrence of each word in the texts. As second measure we use Jaccard similarity, which is commonly used to comparing the and the diversity of sample sets. It is defined as the size of intersection divided by the size of the union of the sample sets. The third measure that we use in our programs is Pearson correlation coefficient. It calculates the correlation between two variables (or two vector), returning a value between 1 and -1 where is positive correlation, is no correlation and -1 is negative correlation.

Finally, we try to predict the rating of the input review using a well-known technique of machine learning, the weighted k-nearest neighbor (WkNN) algorithm. It is useful to assign weight to each neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

In addition, we also try to predict the rating of the input review using linear regression. We took inspiration from the work of Max Woolf, which analyze an huge amount of review on Yelp through linear regression [11]. However, we handle this task in our own way, using the rating of the reviews as dependent variable y and the number of positive words divided by the sum of positive and negative words in each text review as independent variable x.

3. PROBLEM STATEMENT

Nowadays, online reviews are created every day, for a large variety of products and services, by customers who have bought these products or used these services. The number of reviews for a products can often be prohibitive for a potential customer who want to read all significant information, compare alternatives and make a conscious decision. Therefore, the ability to analyze a set of online reviews and produce a summary is a primary challenge for online sellers, review aggregators and local search services. Furthermore, opinion mining can be useful in other ways. For example, it can help marketers to evaluate the success of an advertising campaign or new product launch, determine which versions of a product or service are popular and identify which demographics like or dislike particular product features [12].

In the last few years, advanced techniques of opinion mining has been developed by some big companies such as Microsoft and Google [13]. In fact, there are many and many algorithms that try to guess the personal opinion based on a text review and categorize it, using machine learning systems Although these tool are very efficient, sentiment analysis is still one of the data mining barriers where nobody has been able to find a perfect solution in a general scenario yet.

With our work we try to provide a different way to predict a review rating, based only on a list of opinion words (positive and negative) and a dataset of reviews. Our idea is to solve this task in two different ways. Our first solution is to determine which and how many opinion words are present in the input review and in the reviews of the dataset. After that we compare the vector of opinion words with all the vectors of the other reviews, using three different similarity measures. The measures used are cosine similarity, Jaccard similarity and Pearson correlation coefficient. Finally, the program provide three different results of rating prediction through the WkNN algorithm. The second method is to guess the rating using the linear regression, which provide another result.

In addition, the program compute the root mean square error (RMSE) based on each previous result of the program in order to determine which method provide the most accurate results.

4. SOLUTION

The process development of the project has been split into 3 parts: The first one is the research of the existing recommender system solution and similarity measures that could be useful to achieve our goal; development of the solutions chosen; testing and studying of our results.

This chapter explains the first two parts. We talk about the testing part and the meaning of the results in the next chapter.

4.1 Recommender system and similarity methods in text analysis

As said before one study about text analysis in particular got our attention. In this study they tried to guess the rating score of a review using linear regression, taking as input-data the number of positive and negative opinion words present in the input review and the total words in the review as well. Our own opinion is that the study could be meaningful, especially because the solution seems very simple, easy to implement and very fast in a time complexity view. However, we thought that using only a basic linear regression, the ideas loses much of its potential. That's because with linear regression they could take in consideration only a positive or negative value (i.e. formula) by counting how many time the input review contains the opinion words and predict a possible rating based on the positive or negative values of the other comments in the dataset, but, in this way, they exclude which the positive and negative words written are, and if there is a possible connection between them.

4.2 Investigation of the solution

Hence, we extend the Bing Liu and Minqing Hu's study [9], trying to use other prediction system which use similar methods that can in our opinion express better sentiment similarity between two reviews than a linear model do i.e. collaborative filtering. Before starting explain how our algorithm works we also want to improve what collaborative filtering means.

Collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints and data sources [14] in order to recommend different kind of data. Collaborative filtering systems have many forms, but we decided to develop the most common systems used, which architecture can be reduced in two steps (the steps listed below have been adapted for our subject matter):

1. Look for reviews which use the same opinion words with the input review (the review whom the prediction is for);
2. Use the ratings from those like-minded reviews found in step 1 to calculate a prediction for the input review.

In order to discover which objects or items are similar and how much, collaborative filtering method usually use cosine similarity, Jaccard similarity, Pearson correlation or hybrid methods which are essentially composed by more than one of these abovementioned similarity methods combined. For handle predictions instead, one of the most common, simple and useful algorithm used in machine learning is the k-nearest neighbors

algorithm (kNN). kNN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification [15]. In our problem in particular, we decided to use a weighted k nearest neighbors, in fact, can be useful to assign weight to the contributions of the neighbors, that means that the nearer neighbors contribute more to the average than the more distant ones.

4.3 Our solutions

We developed three different collaborative filter method systems and a linear regression in order to compare our result with the Bing Liu and Minqing Hu's study and also to understand when and in which situations a method works better than others. The collaborative filter methods developed differ only by the different measures used to calculate the similarity between the input review and the other reviews in the dataset chosen (cosine similarity, Jaccard similarity and Person correlation). For the prediction task we use the same method in all the three collaborative filter methods implemented. For this task, we developed a weighted k nearest neighbors algorithm, using as weights the similarity calculated between the input review and the reviews compared.

For the linear regression instead, we needed to find a formula that could express as similar as possible the Bing Liu and Minqing Hu's study results, in fact, in the study report, they don't mention which formula has been used to achieve them. Having the number of positive and negative words and the number of words in the review itself as opinion indicators, we decided to use the Positive Predict Value and after to try the Negative Predict Value which gave us worse results. The Bing Liu and Minqing Hu's study took into consideration in addition that the amount of words in the review has a lesser, negative effect, but, we disagree with that idea. That is because in Amazon's datasets (dataset that we used to train and test this project) and we suppose in any other review dataset as well, an user who is using a high number of words in his/her review is actually describing the object rated, or a specific part of it, without give a personal opinion about the object. So, our idea is that a huge number of words in a review means nothing in a sentiment analysis view. In this way, we decided to not use the amount of words of the review as an opinion indicator.

4.4 Development

For the development we decided to use Java as programming language. We created a Java-Apache-Maven project using Apache Maven (version 4.0.0) and the Java Development Kit version 1.7. We wrote our project using Eclipse Mars 1 as development environment.

The main force of our algorithm is that it could be used for any reviews' dataset, irrespective of which kind of object, film or whatever the dataset is about. So we tried to adapt our project in order to use it in the maximum number of datasets. Our training set and dataset were composed with Amazon Product data [16] a dataset which contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014 [17] [18]. Amazon Product data is a set of 24 dataset categorized by object category (Books, electronics, Movies, Tv, etc.). Data in these datasets are represented in JSON format. So we decided to use gson 2.2.2 [19] a Java serialization/deserialization library that can convert Java Objects into JSON and back in order to achieve our input phase.

Now we are going explain the implementation of the software in its entirety and our project works , after that step by step, we are going to explain how it is written showing the program's classes and looking further a little bit in their functionality.

Basically our program requires as input a dataset, a list of positive and a list of negative words (opinion words), an input-review and its overall. The system starts scanning the input review and the reviews in the dataset looking for opinion words in the text, saving in a list (Figure 1) which and how many positive and negative words each review scanned contains (opinion list).

marvelous	amazing	inadequate	bad	excellent	terrible	....
0	2	0	1	1	0	

Figure 1: opinion list example.

We decided to let users choose which prediction method the software will use (i.e. using cosine similarity, Pearson correlation, Jaccard similarity, linear regression or all of them). According to the user's choice the system confronts the input-review-opinion list generated before with all the dataset review-opinion list. In the event that, the user, picked one of the collaborative filter methods, for each dataset's review will be calculated the similarity (using the similarity method chose) between the dataset-review-opinion list and the input-review-opinion list producing a set of similarity measures. As said before, for prediction we used a WkNN algorithm, using as weight 1/1-similarity (in this way, the more a review is similar to the input review the more will be its meaning). In order to select which k neighbors take into consideration, the system, using a quicksort algorithm, sorts the generated list of similarity measures in a descending order, picking then the firsts k elements of the sorted list. In order to illustrate how our WkNN algorithm handle the k nearest elements selected to achieve a prediction result, we think that the formula (Figure 2) can express the idea in the most meaningful, fastest and easiest way.

$$WkNN(y) = \frac{\sum_{j=1}^{k} \frac{y_{i_j}}{1-similarity(x_{i_j},\ x)+d_0}}{\sum_{j=1}^{k} \frac{1}{1-similarity(x_{i_j},\ x)+d_0}}$$

Figure 2: formula used to predict the rating of the input review. Y represents the rating, X the review text and d_0 is a small constant offset used to avoid division by zero.

For linear regression, we don't have to compare each review of the dataset with the input one, but to find the right values that modelling a relationship for independent variables denoted x and scalar dependent variables y. Of course this relationship must express in the best way possible our problem 's need. We decided to use as dependent variable the overalls (rating review) and as x the Positive Predict Value (PPV) of the reviews (Figure3).

$$PPV = \frac{n^\circ\ positive\ words}{n^\circ\ positive\ words \quad n^\circ\ negative\ words}$$

Figure 3: Positive Predict Value formula.

As linear model we used an ordinary least squares regression model with one independent variable developed using Apache Commons Math 3.6.1 API [20] and in particular the Class SimpleRegression.

The project has been built using three classes: Review. java, SimilarityOverall.java and Application.java. The first two classes mentioned are used mostly to store data useful during the whole run of the project. Application.java instead is the main classes and the core of our program.

4.4.1 *Review*

This class is used to store all the fields of each review in the dataset as private variables. We build the class according to

information about the reviews extrapolated in Amazon's dataset(i.e. ID of the reviewer, ID of the product reviewed, name of the reviewer, text of the review, rating of the product, summary of the review, time of the review (unix time and raw), but the class could be changed easily in order to use other review dataset with different structure and information.

The Review Class contains also the Get and Set method for the private class variable mentioned above. Data stored in this class are used in the first part of the Run system.

4.4.2 SimilarityOverall

This class is used to store each review overall of the dataset and the similarity measure between the reviews and the input review.

Get and Set methods for the data stored has been implemented in this class as well. SimilartyOverall class also contains quicksort recursive methods() used to sort in a descending direction, by a quicksort algorithm.

This class is used only in the event that the program has to execute one or more of the collaborative filter methods available.

4.4.3 Application

This is the main class of the project and provides all the methods used for the management of the program run. So now we're going to deepen the functionality of this class in order to understand how in practice the system works, following step by step the Application.java code.

At the beginning the program has to prepare and initialize input variables. First of all, the software scans all the files passed as input. Positive and negative words lists are filled by using a simple Scanner that read the words on the appropriate files then, save them. The dataset is read using a JSON parser and data are saved in a list of Review (Review.java class). Once saved all the reviews in the system, the program has to seeking, for each review, opinion words contained in the review text and then comparing it with those belonging to the input review.

Therefore, in order to count the opinion words occurrences in the review text we implemented the Counters() method which, taking as input a text, return a Map<key,value> which contains as keys all the opinion words and as value the opinion words occurrences founded in the text. Before scanning, the methods transform the whole text in lower case character and splits, from the words, punctuation marks and special characters in order to avoid some text analysis issues. Then Counters() method starts to scan checking, thanks to Java.lang.String.contains() method, which and how many times the user used opinion words in his/her review text.

The Counters() method has to iterate on each dataset review and on the input review as well. Once finished this process, according to the similarity user's choice, the system must establish the similarity measure between the Map (opinionWords Map), returned by the Counters() method, of the input review and all the maps calculated on the dataset reviews.

In order to do that, using collaborative filter predictions, we implemented three different methods, one method for each different similarity measure used(CosinSimilarity(), JaccardSimilarity(), PearsonCorrelationCoefficient()). These methods, taking as input the opinionWords map calculated on the input review and iteratively, the opinionWords map of the dataset review, quantify the similarity between these two input by using for each similarity measure the appropriate formula and return the result as Double. Then we will have three different arrays of Double containing the similarity results, obviously grouped according to the collaborative filter method used to quantify it.

Since linear regression uses a totally different approach in order to calculate its model and predict the value required, we need a different representation technique of our problem. In fact, the counters() method that we used for collaborative filter predictions, merges in a single map all the positive and negative opinionWords together (this is because the similarity is calculated taking into consideration a possible correlation between the words written itself and not their meaning). According to our linear model decision, we needed to count the opinion words while preserving their positive and negative division. So we developed counterLinearRegression () method, that counts the opinionWords presents in the review text passed as input and returns the Positive Predict Value of the review. CounterLineraRegression is called iteratively passing it as input every review in the dataset in order to build the linear regression function. Thanks to the SimpleRegression () class of the Apache Commons Math library, it was pretty easy to implement a linear regression predictor method. In fact, once computed all the Positive Predict Value of every dataset review, we only have to pass them to the API method addData() (Apache Commons Math) on a simpleRegression variable initialized before. AddData() adds the observation (x,y) to the regression data set where x values will be the Positive Predict Value while y values will be the overall of the review.

At this point, the system has prepared all the necessary data both for collaborative filter predictions and for a simple linear regression prediction. For collaborative filter prediction, as said before we handle the prediction by a WkNN algorithm. So the program has to sort the similarity measure lists in a descending direction using quicksort() method written in the SimilarityOverall class in order to peak the most similar reviews to the input review. Then the sorted lists will be passed at the weightedNearestNeighbor() method which is in charge of calculate a prediction using the formula (Figure 2). For linear regression prediction we simply pass the Positive Predict Value to the predict(double x) API method which returns the "predicted" y value associated with the supplied x value, based on the data that has been added to the model when this method is activated.

4.5 Root Mean Square Error

In the end the system prints the results measured on an output file text, plus stores all the runs results in four different txt files (each file cover only a single prediction method), saving the all the prediction given by the system in every run and the “real“ overall given by the user as input. These files (ErrorFiles) will be used by the program in order to calculate the RMSE for each prediction method. The RMSEs will be defined in the output file taking into account all the results written on the ErrorFiles.

RMSE of course will be calculated on each different prediction method separately, and the values used will be got by scanning the appropriate ErrorFile. To restart from scratch the calculation of the RMSE, forgetting the previous results, for example in order to train the algorithm or to understand the results, it will be needed to delete the ErrorFiles. Doing that the error calculation will start from 0.

5. EXPERIMENTS

This algorithm, at the beginning, was trained using small dataset (containing from 1000 to 50000 reviews), in this way we could

easily check the accuracy of the results. This allows us to determine if the system works properly without any algorithmic or development error. We also trained it, using for each test a different k value (k stay for the number of nearest neighbors similarity measures take into account to predict the result), looking for a value that could give results as accurately as possible and trying to minimize the RMSE. We know that the best solution in order to do that would be to use k-fold cross-validation technique, but since our subject matter of this project is data mining and not machine learning, plus a k-fold cross-validation algorithm would require too much time to run using such big dataset as we used, we thought that the best and fastest choice to solve the k problem is by training the k value manually, keeping the idea of a k-fold cross-validation for possible future optimizations. Nevertheless we discovered that the accuracy of our result are highly depending on the k value only if the input review has a low overall value (from 1 to 3 on a scale of 1 to 5). In fact, Amazon Product data dataset contains much more positive overall than low, for instance in a 1000 reviews dataset 875 reviews have an high overall, including 691 which their rating is 5, therefore is more difficult to find a similar review to the input one, which have a low overall. This also means that the best k value will change its amount based on the overall that has (or will have) the input review. In particular, in this case the k value has to increase in parallel to the overall, the higher the overall is the higher the k has to be, the opposite is true as well.

For tests we used a HP-Pavilion-dv6 laptop 64 bit (8Gb RAM memory, CPU: Intel® Core™ i7-2670QM CPU @ 2.20GHz octa-core).We used Ubuntu 15.10 as operating system.

We have always tested it using the input choice that execute all the predictions implemented. This because the velocity of the program to perform is very much more dependent on the number of reviews that it has to scan (i.e. on the dataset size) than on how many different prediction it has to do, giving us, in this way, more results.

5.1 Results

Now we are going to list the most important and meaningful experiments we did, we will discuss in the next sub-chapter about their meaning.

For the first trainings we used a part of the Amazon video dataset which is composed by 1000 reviews. Using this dataset we did 10 different test, as input review we took real reviews on the same dataset. The input reviews were choose preferring to diversify the review overall and the number of words that they contain. These tests has required 5 seconds each to be completed. Since the output produced by the prediction are double (producing numbers with decimal places) while the overalls are integer the system has never predicted a perfect result, but, by rounding results to integer, output seems to become better, in particular with Pearson correlation the system guessed correctly 4 rating reviews on 10. The results of these test are described in further detail in the table (Table 1).

We continue testing, with 15 new test runs using another slice of the same dataset composed by 50000 reviews. The time run of these tests is 3'18" each. These trials gave us more or less the same results of the firsts 10 tests, reconfirming for the moment the collaborative filter method with Pearson correlation similarity measure the most accurate method among those implemented.

Table 1: errors data of the first test set.

Reviews Amazon instant videos: 1000 reviews				
	cosine	**Jaccard**	**Pearson**	**linear regression**
Root Mean Square Error	1,594387149	1,709871424	1,579351083	1,740101966
Min Error	0,000000007	0,000000007	0,000000278	0,181920852
Max Error	2,869210814	2,869210814	3,233333333	3,098451770
Predicted perfectly	0	0	0	0
Right predicted (rounding)	3	4	3	2

Then, looking for better results, we decided to run our algorithm on the entire Amazon video dataset (583930 reviews) , performing in 31' 07". The accuracy of the results in general was little higher (the RMS value of the seconds 10 tests had a slightly decrease from 1,767267663 to 1,651526685), but the number of right rounded prediction decreased by 10%.

Obviously the results could change on the basis of the categories where the objects rated belong. Hence, we needed to change dataset in order to resolve as many doubt arising from our results as possible. Fortunately Amazon product data dataset contains 24 categorized dataset allowing us to solve this problem. We did the next 10 tests on the Homes and kitchen Amazon reviews dataset composed by 583930 reviews about homes and kitchen tools bought from Amazon.com. With this dataset the program ran in about 27 minutes time. Here the results were a little worse respect the previous, in fact the mean RMSE of this tests was 1,826756976 and the average of the rights rounded prediction is 3. This is because the trials have been done using reviews which have a low overall value. The dataset in fact, contains too few low rated review for expect accurate prediction using a low rated review as input. The system indeed, in the last results described, did not predict an overall lower than 3,5 even if, 7 of the input review on 10 used, had an overall which varied from 1 to 3.

We made the last tests using Amazon Health and Personal Care dataset (346355 reviews) showing results very similar to the previous done using dataset which contains a huge amount of reviews (Table 2).

Table 2: results of the last test set.

Reviews Amazon Health and Personal Care: 346355 reviews								
id	**number of words**	**overall**	**k used**	**cosine prediction**	**Jaccard prediction**	**Pearson prediction**	**linear regression pred.**	**running time**
50	71	5	100	4,5037572	4,4146105	4,0379242	4,3753850	26'31"
51	36	5	100	4,6010509	4,7100249	4,3746333	4,5847345	30'44"
52	50	4	50	4,6267381	4,8764492	4,3579874	4,0032033	30'19"
53	67	4	50	4,7485378	4,7832143	4,7140684	4,4351996	31'50"
54	92	3	20	3,8717913	3,6644004	4,1489428	4,3230508	28'50"
55	78	3	20	4,7949773	4,8217822	4,9504132	4,1660404	25'33"
56	136	2	10	4,4000000	4,9000000	4,5140049	3,7124717	26'27"
57	77	2	10	3,8588174	3,9100076	2,0000001	3,9305522	26'43"
58	142	1	10	1,6000000	3,5011400	2,1204819	3,7997481	26'06"
59	103	1	10	4,5295544	4,8965836	4,1762464	4,0614218	25'31"

5.2 Study of the result

To summarise, we reported 60 different tests on 5 different Amazon dataset (Figure 4). We achieved the best results using the cosine similarity collaborative filter methods that has had a RMSE equal to 1,66980667, predicting in total 13 right values. These results mean that, the similarity between two reviews expressed in order to predict their rating based on the opinion word that they contains, is more precise if it will be calculated on the occurrences of the equal words that are written in the two reviews (cosine similarity) and not on a correlation between them (Pearson correlation) or on how many equal words are present in the two reviews (Jaccard similarity).

However we noticed that Pearson correlation collaborative filter methods works better using smaller dataset, in fact, for the first two test sets, we obtained the best results from this method. Pearson correlation method in our tests also produced the most precisely predictions, generating the minimal error on three test sets of 5. In any case this method sometimes gave us no reliable output at all.

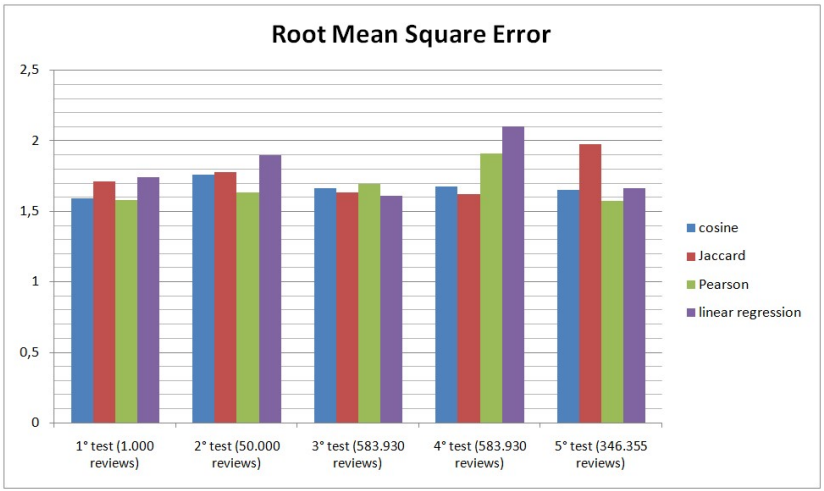

Figure 4: chart that show the RMSEs resulted during tests. Each x element corresponds to a different test set.

First thing to notice is, the results produced by using a small part of the dataset are more precise than by using the entire one. This is because, having a larger number of positive reviews in the dataset, the bigger the dataset is the easier is to find a similar review that has a positive overall, in fact, we saw that the system find hard to predict, using big dataset, a negative (1 to 3) overall. Furthermore, we realized that low rated reviews usually, contains more than one user's opinion. In fact, in most cases, negative reviews are divided in two different part, one that give the user's opinion about the product while the other part contains opinions about the service gave by the Service provider, where, at least one of these, express an user disappointment. The rest of the review instead, argue in a positive way whatever it described. Consequently, the program, scanning the whole review text, finds and analyses either positive a negative user's opinion contained in the review, and will predict a value based on both, that actually will be not the real overall gave by the user. We shall certainly obtain better results if users would rate what effectively they write in their reviews, reviewing only the service or the product in base at what is needed.

In some tests, the system produced quite far prediction values from the real overall value that would have to predict, making an error greater than 3. In these cases, the program make such predictions because the reviewer used a lot of negations (i.e. not bad, I don't like it) in his text. Our system can not be able to handle this kind of sentences since the opinion words list is only composed by singular words. The program, in fact, actually manage negations in a completely wrong way, counting as positive word a negations that means a negative opinion, and the opposite as well. For example, if the review text contains " I don't like", our algorithm will increment the occurrences of the word "like" as positive word, but the meaning of this sentence is negative. To avoid this problem, a possible future optimization could be to change a little bit the program in order to insert as opinion words also sentences or, at least, the most popular negations used by reviewers. For the same reasons the system can not handle any grammar error.

During the tests, we also stored the number of words used for each input review tried, in order to prove our theories that the words amount of a review is meaningless in sentiment analysis. For the moment our results have confirmed our theories producing no relations between the number of words used in the review and its rating.

5.3 Linear regression data

Our linear regression system is in general the method that produced the worst results between those that we developed, in particular for the set of tests made on the Amazon House and kitchen Dataset, where the root mean square deviation calculated on it is 2,1011398908171537. The SimpleRegression (Apache Commons Math 3.6.1 API) allowed us to have each test greater details of the regression model created in order to study the linear regression results. The library API in fact let us to get the slope of the estimated regression line, Pearson's product moment correlation coefficient, the sum of squared errors (SSE) associated with the regression model and the sum of squared deviations of the predicted y values about their mean (which equals the mean of y) [20].

Slope in tests resulted about 1,3. This means that when the variable x increase by 1, the value of y increase by 1,3. This results showed that review texts usually don't contains a big gap between the number of positive and negative words, and a weak correlation between the x values and y values.

In addition, Pearson's product moment correlation coefficient that is a measure of the linear correlation between two variables X and Y, has always resulted about 0,32, showing a no linear dependence between the two variables. In other words, linear regression data shows that the linear model does not describe the relationship between x and y in a good way, therefore, in our study view, there is definitively no strong relationship between the Positive Predict Value and its overall. We also tried to change linear model using the Negative Predict value as X, in order to check if the relationship between the variables would improve, but the results were even worse.

The SSE, measure of the discrepancy between the data and an estimation model, show a poor fit of the model to the data resulting sometimes over 5500. The sum of squared deviations of the predicted y values, reaching an average amount of 6000, has only strengthened our assumptions that linear regression method cannot be used to describe our model problem, or rather, exist models that can produce more reliable results.

6. CONCLUSION

We have built and developed this project and, by a correct formatting of the input files, it works without any bugs or errors. Performances, as we have seen, depend only on the dataset size passed as input, the bigger the dataset is the lower the system will be.

We reported that there is not a perfect k value valid for any run and that the best solutions for this problem would be train the algorithm every time that will be utilized a new input dataset. The results of our tests seems not to be good, but using only 6000 opinion word we expected to have not detailed results, in fact the

average of the number of opinion words contained in the reviews, was found only to be about 3. In any case we would like to try this program by using more opinion words and sentences either, being hopeful about our idea. However, this system has been created in order to predict a rating value in a easy way and quickly, but we now that exists other systems which, by using more complex algorithms can achieve much better results than ours.

Future optimizations for this project have already taken into consideration. Using machine leaning in order to train the algorithm and find a right k value for the run would be a greater method that will certainly give more accurate results. Another optimization, as said, could be improve the number of opinion words and develop some method that will avoid the negations issues reported. The best optimizations, in our opinion, will be to change the scan method, in the way that the system will check and handle not only singular word but sentences in their entirety. According to our study of the results, dividing sentences in topic in base at what the writer is evaluating, could also be an useful improvement.

For now we have created and studied, by expanding the Bing Liu and Minqing Hu's study, a general and alternative way to predict reviews ratings that with the right update could be work on any review dataset, which can be used in future text sentimental opinion studies.

7. REFERENCES

[1] D. J. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. MIT Press, 2001.

[2] "Data Mining Curriculum." http://www.kdd.org/curriculum/index.html.

[3] "A Beginner's Guide to 'Data Mining,'" *About.com Tech*. http://databases.about.com/od/datamining/a/datamining.htm.

[4] "Oracle Data Mining." http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/overview/index.html.

[5] "SQL Server 2014 | Microsoft." https://www.microsoft.com/en-us/server-cloud/products/sql-server/default.aspx.

[6] H. Wang, Y. Lu, and C. Zhai, "Latent Aspect Rating Analysis on Review Text Data: A Rating Regression Approach," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2010, pp. 783–792.

[7] W. Duan, B. Gu, and A. B. Whinston, "Do online reviews matter? — An empirical investigation of panel data," *Decis. Support Syst.*, vol. 45, no. 4, pp. 1007–1016, Nov. 2008.

[8] B. Liu, "Sentiment analysis and subjectivity," in *Handbook of Natural Language Processing, Second Edition. Taylor and Francis Group, Boca*, 2010.

[9] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2004, pp. 168–177.

[10] K. V. Kale, *Advances in Computer Vision and Information Technology*. New Delhi, India, India: I K International Publishing House, 2008.

[11] "The Statistical Difference Between 1-Star and 5-Star Reviews on Yelp," *minimaxir | Max Woolf's Blog*. http://minimaxir.com/2014/09/one-star-five-stars/.

[12] "Sentimate Analysis For Web Product Ranking | International Journal IJRITCC - Academia.edu." http://www.academia.edu/17633431/Sentimate_Analysis_For_Web_Product_Ranking.

[13] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar, "Building a Sentiment Summarizer for Local Service Reviews," *NLP Inf. Explos. Era*, 2008.

[14] L. Terveen and W. Hill, "Beyond Recommender Systems: Helping People Help Each Other," in *HCI in the New Millennium*, 2001, pp. 487–509.

[15] "k-nearest neighbors algorithm," *Wikipedia, the free encyclopedia*. 25-May-2016.

[16] J. McAuley, "Amazon review data." http://jmcauley.ucsd.edu/data/amazon/.

[17] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-Based Recommendations on Styles and Substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2015, pp. 43–52.

[18] J. McAuley, R. Pandey, and J. Leskovec, "Inferring Networks of Substitutable and Complementary Products," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2015, pp. 785–794.

[19] "google/gson," *GitHub*. https://github.com/google/gson.

[20] "SimpleRegression (Apache Commons Math 3.6.1 API)." http://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/stat/regression/SimpleRegression.html.