IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-7, July 2019 E-ISSN: 2347-2693

Recommendation System for Electronic Product

Dea Dania[1*], Lily Wulandari[2]

[1]Dept. of Computer Science and Information Technology, Gunadarma University, Depok, Indonesia
[2]Industrial Engineering, Gunadarma University, Depok, Indonesia

Corresponding Author: deadania27@student.gunadarma.ac.id

DOI: https://doi.org/10.26438/ijcse/v7i7.16 | Available online at: www.ijcseonline.org

Accepted: 12/Jul/2019, Published: 31/Jul/2019

Abstract**—** Recommendation System(RS) is one of machine that uses in many fields of application like music, book, shopping, and etc. With an RS, it makes users easier to find items that are very likely to be searched for. Not only star rating, but testimonials are also one of the data that affects buyers or connoisseurs of a product. The challenge is testimonial is not in numerical data type such as star rating. In this study, the researchers tried to build an architecture to combine the results of the testimonial through sentiment analysis and star rating which are processed separately in an RS. The dataset is reviews of few items in Amazon. The sentiment analysis uses Lexicon-based Approach, which RE use Collaborative filtering with PySpark library. The sentiment analysis has positive, negative, stop words, product-does corpora with double negative or positive words handling, cross negative-positive corpus words handling, and negative of product workless handling. The result is the architecture can be implemented with the testimonial and star rating dataset with giving recommendation items for every user.

Keywords**—**Recommendation System, Sentiment Analysis, Collaborative Filtering, PySpark

I. Introduction

A recommendation system (RS) is a subclass of information filtering systems that help to predict the rating or preference, based on the rating provided by users for an item. It is used in many areas, such as movie, music, news, books, research articles, search queries, social tags, products, jokes, restaurants, garments, financial services, life insurance, and online dating sites [1]. RS are able to neutralize the effect of information overload to a great extent by filtering vital information fragment from a large amount of dynamically generated information [2]. It is one of machine that helps users find what they might like, depends on review history users who have similarity personalization with saving their time.

The popular data use in an RS is the star rating of reviews. Star rating usually range between 1 and 5 which 5 stars is the best and 1 is the worst appraisement. Besides the star rating, there are other data that can affect other customer interests, like a testimonial. The testimonial is a spoken statement of product quality by users. These part of reviews is the most general things that considerable influence in the review. The difference is star rating is quantitative data while testimonial is qualitative. To combine the results of both in RS, testimonials must be processed into measurable data that can be done through sentiment analysis (SA). This study describes the recommendation system architecture using collaborative filtering. The data used are several testimonials and star ratings from several products. Testimonials in the form of qualitative data are converted into quantitative values through SA. This architecture is implemented using python programming. This section is followed by related work in section II, literature reviews in section III, methodology in section IV, results and discussions in section V, the conclusion in section VI.

II. Related Work

A work of an RS through SA has done with detects the opinions polarity score using the semi-supervised SVM [3]. They tested and evaluated the model using four datasets (Arabic, Dialect, French, and English). With user-based Collaborative Filtering algorithm, they set k nearest neighbors of a targeted user which identified by calculating the correlations of similarities between the users' ratings to associated their preference items. The SA has architecture; features extraction (17 features vector) continuous with S3VM classification (divided to positive, negative, and neutral score. The evaluation of the work use Mean Absolute Error (MAE), precision, and recall for each kind of the datasets.

Rita [4] considering adverb in SA with giving value in the Portuguese dictionary of adverbs. With the traditional sentiment metric, they doubling the final sentiment value of the sentence. Furthermore, the recommendation system has low complexity and presents low perceived impacts on the analysis of energy consumption according to benchmark software, it doesn't much explain enough.

RS for movies by Yibo, Mingming and Wei [5] using Spark Framework and Hybrid method. Which combination of content-based and collaborative screening methods result optimized by analyzing the effects of sentiment information. Spark is an open-source computational engine for analyzing data sets which cannot fit into memory (RAM and on disk) on one node. Different from disk-based storage of Hadoop, Spark is more inclined to save the intermediate results in memory in the calculation process, and the iterative calculation process has also been optimized. So Spark's processing efficiency is better than Hadoop in recommender systems [5]. Its dataset in Chinese words. For the sentiment, it using a vector space model. They use the term frequency-inverse document frequency (TF-iDF) value as feature weights. A difference with this research is the testimonial class and rating predictions would combine at the end of the process.

Twitter datasets have popular used in SA researches. One of the works is by Buntoro et, al. They did it using twitter dataset with lexicon based and double propagation in Bahasa. The work has low accuracy (23,43%), it might be happening because of too little data training[6]. The double propagation method extracts new sentiment words and features using the seed sentiment lexicon [7].

III. LITERATURE REVIEW

A. Matrix Factorization

Matrix factorization model is superior to classic nearest-neighbor techniques for producing product recommendations, allowing the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels [8]. The basic rule in matrix multiplication in the order it to be defined is that the number of columns in the first matrix must be equal to the number of rows in the second matrix. Means the product of $m \times n$ matrix and $n \times k$ matrix is $m \times k$ matrix.

In case, there are three matrices r, H, and W. Matrix r represent users by their ratings for different product in user-item rating matrix. In matrix H is user-latent factor with each row shows the user interest in the hidden factors. While matrix W is item latent vector, each column is the product influence by hidden factors. Matrix H and W are the decomposed from matrix r. In the sparse user-item interaction matrix, the predicted rating user will give item i.

$$\tilde{r}_{ui} = \sum_{f=0}^{nfactors} H_{u,f} W_{f,i} \qquad (1)$$

The main idea of the method is to figure out the prediction matrix based on unknown latent factors that projected from user-item interaction matrix. Increasing the number of latent factors will improve personalization, until the number of factors becomes too high, at which point the model starts to overfit. A common strategy to avoid overfitting is to add regularization terms to the objective function

$$\arg\min_{H,W} \left\| R - \tilde{R} \right\| F + \alpha \left\| H \right\| + \beta \left\| W \right\| \qquad (2)$$

Alternating-least-squares with weighted-λ -regularization (ALS-WR) algorithm managed to handle large-scale CF problems [9]. ALS is also used in matrix factorization algorithm and it runs Stochastic Gradient Descent in a parallel fashion. PySpark is a library has a class of Collaborative Filtering that use ALS. The implementation in spark.ml (spark machine learning) use several parameters, those are:

- rank is the number of latent factors in the model (defaults to 10).
- maxIter is the maximum number of iterations to run (defaults to 10).
- regParam specifies the regularization parameter in ALS (defaults to 1.0).
- nonnegative specifies whether or not to use nonnegative constraints for least squares (defaults to false).

Algorithm 1. SGD Algorithm for MF

```
Input: training matrix V , the number of features K , regularization parameter λ , learning rate ε;
Output: row related model matrix W and column related model matrix H
Steps:
1.  initialize $W$ , $H$ to Uniform Real $\left(0, \frac{1}{\sqrt{K}}\right)$
2.  Repeat until convergence
      ◦ for random $V_{ij} \in V$ do
          $error = W_{i*} H_{*j} - V_{ij}$
          $W_{i*} = W_{i*} - \varepsilon(error \cdot H_{*j}^{T} + \lambda W_{i*})$
          $H_{*j} = H_{i*} - \varepsilon(error \cdot W_{i*}^{T} + \lambda H_{*j})$
```

B. Sentiment Analysis

SA, also called opinion mining, is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes. There is two levels of sentiment classification, document, and sentence-level sentence classifications. There is no fundamental difference between both because sentences are just short documents [10].

Lexicon based sentiment analysis of a text is a data analysis task performed by employing opinion words and phrases with no prior knowledge opinion words are compiled and collected. Positive and negative words along with opinion phrases are collectively called Opinion Lexicon. Words in the text are evaluated based on opinion lexicon to determine their orientation and henceforth the sentiment of the text [11].

IV. SENTIMENT RULES

The SA check the testimonial by sentences with rules:

- Negation handling, if the word is in the positive or negative corpus, and the previous word is a negation, then the score give to the word is the opposite of polarity score. E.g. “good” has a positive polarity and “bad” has negative polarity, so “not good” score -1, and “not bad” score 1.
- Double negative or positive words handling. Double positive words like “well worth” score 1, not 2 and double negative words e.g. “my only complaint is the adapter is horrible” become [“complaint”, “horrible”] which has 1 negative meaning score -1 not -2.
- Cross negative-positive corpus words handling. Positive-negative words like “pretty bad” score -1 while negative-positive would be score like negation-positive e.g. “bad recommendation”.
- Negative of product workless like “doesn't work”, “not connect” etc score -1.

Algorithm 2. Sentiment Analysis

```
Input: review, corpus
Output: review score, and the score class.
Steps:
For each review
 1. Transform into lowercase.
 2. Remove numbers character and stop words
 3. Separate into sentences
 4. For each sentence
      • Separate to words
      • For each word
          o If it is not in the corpus, stem the word
          o Scoring the word by the rules
      • Summarize score in a sentence
 5. Summarize (sentences) review score
 6. Score classification
```

V. METHODOLOGY

Research method includes data analysis, data pre-processing, SA, a recommendation system and evaluation, see Figure 1. All processes are done by coding with Python programming in Jupyter Notebook offline application. The architecture of The RS in Figure 2. Testimonial data entered into SA with a classified testimonial as to the output. After that, the classified testimonial and star rating data get into RS and the results are evaluations and final recommendation list.

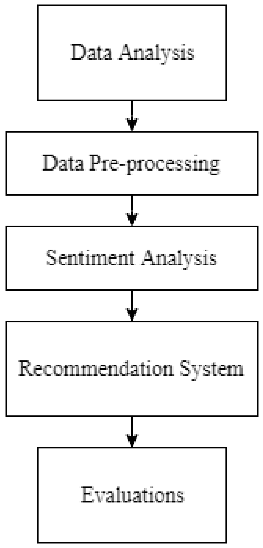


Figure 1. Research methodology

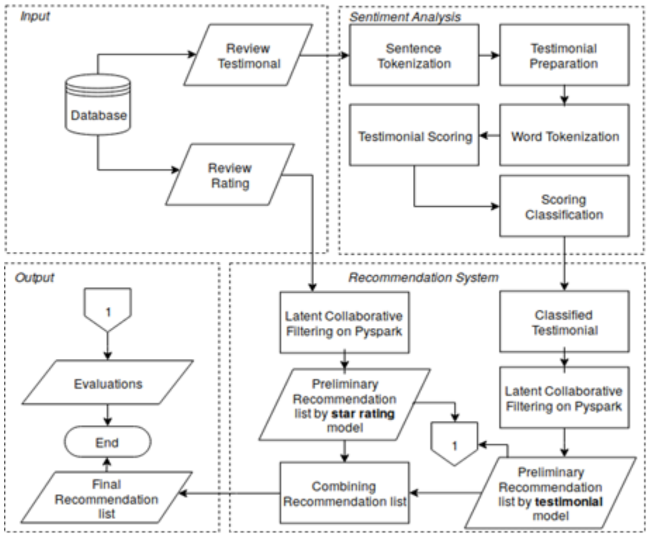

Input
Review Testimonial
Database
Review Rating
Sentiment Analysis
Sentence Tokenization
Testimonial Preparation
Testimonial Scoring
Word Tokenization
Scoring Classification
Output
1
Evaluations
End
Final Recommendation list
Recommendation System
Latent Collaborative Filtering on Pyspark
Preliminary Recommendation list by star rating model
1
Combining Recommendation list
Classified Testimonial
Latent Collaborative Filtering on Pyspark
Preliminary Recommendation list by testimonial model

Figure 2. Recommendation system architecture

VI. RESULTS AND DISCUSSION

A. Data Analysis

The reviews data from open data website kaggle.com. The data has 5000 records reviews from amazon.com (online retailer) in English includes 23 items which consists of a TV, an adapter, 3 speakers, 7 Kindles and 11 tablets where data distribution is uneven. Besides the reviews data, the data needed is data for input on SA, sentiment words and stop words. Sentiment word consists of positive, negative, negation, and product-does words. Positive words like “good”, “excellent”, “amazing” etc. Negative words like “bad”, “horrible” etc. Negation words “no”, “doesn’t” etc. The product-does words are “connect”, “play”, “work” etc. The stop words are “i”, “am”, “she”, “he”, “it”, “was” etc.

B. *Data Pre-processing*

The reviews data has to prepare before use in the system with data decomposition and integration, data reduction, and data cleaning. The data decompose with make product and user table, then integrate the tables with the Ids. After integrating with command merge, delete columns that have the same contents or unused columns (data reduction). Clean the data from duplication. The result is 4805 records reviews has prepared. Numpy and Pandas are libraries used in these stages.

To solve the problem of uneven data distribution, the data has added with 134 records consisting of 6 TVs, 6 speakers and 5 adapter items with 43-48 reviews per item type respectively. The total reviews become 4939, includes 40 items which consists of 7 TV, 6 adapters, 9 speakers, 7 Kindles and 11 tablets, which has a better distribution of items type.

C. *Sentiment Analysis*

The reviews testimonial must be numeric data type to process in RS. Sentiment score uses to reclassify the numeric class of each testimonial. In doing the analysis, there are several stages: testimonial tokenization, testimonial preparation, sentiment scoring, and score classification. Each review token by sentence and words. The preparation did remove numbers, and stop words, transform the reviews to lower case, and stem certain words.

The score classification stage defines where the testimonial score is classify. From the dataset, the most negative score is -5 and the most positive score is 21. The frequency of every score detail in Table 1. The Graphic can see in Figure 3.

Table 1. Frequencies of the scores

No.	Scores	Frequency	No.	Scores	Frequency
1.	-5	4	13.	7	24
2.	-4	6	14.	8	11
3.	-3	24	15.	9	9
4.	-2	64	16.	10.	4
5.	-1	16	17.	12	2
6.	0	615	18.	13	2
7.	1	1397	19.	14	1
8.	2	1383	20.	15	2
9.	3	728	21.	18	1
10.	4	290	22.	19	1
11.	5	152	23.	20	2
12.	6	56	24.	21	1

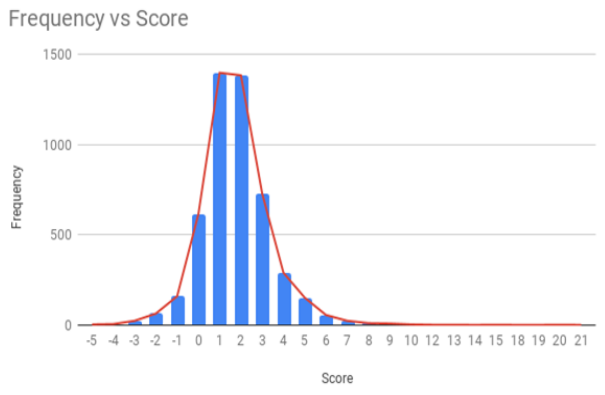


Figure 3. Frequency of the scores

Classification use frequency distribution with Larson rule in formula (3) to determine classes mount and Wand rule for knowing the class width (h) in formula (4). The range of data (biggest data minus smallest data) built from the scores which have frequency more than 50 as a classification requirement namely -2 to 6. The frequency of scores {3, 4, 5} added up but doesn’t have more than 50 frequency, so it determined as 2. For scores up to 6 was added up and has 60 frequency, therefore it determined as one value above score 6, which is 7. So, the range of data is 7-(-2) = 9. The score classification illustrated in Figure 4.

$$c = 1 + [2.2 \log(n)] \tag{3}$$

$$h = \frac{range}{c} \tag{4}$$

c : Number of bin/classes
n : Number of observed values
With n = 4939, range = 9, the values of c and h:

$$c = 1 + [2.2 \log(4939)] = 9.126 \approx 10$$

$$h = \frac{9}{10} = 0.9 \approx 1$$

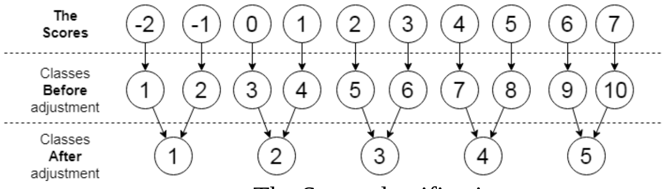
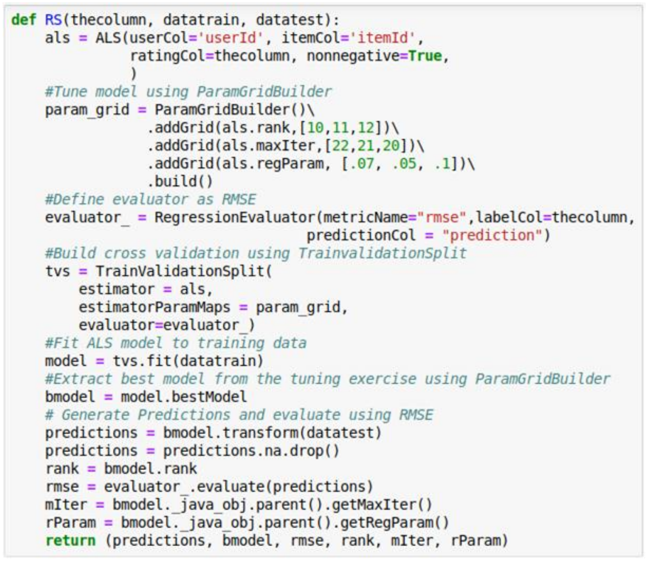


Figure 4. The Score classification

D. *Recommendation System*

The stages to get the prediction table are imported PySpark classes; get data train and testing data; create a model by fit the data train with ALS class, and Predict data testing result using the model. There are CrossValidator and TrainValidationSplit classes to hyper-parameter tuning. The difference is the TrainValidationSplit only evaluates each combination of parameters once, as opposed to k times in the case of CrossValidator. In this research using TrainValidationSplit which less expensive than CrossValidator, researchers believe it is good enough. ParamGridBuilder used to get the best parameter while using TrainValidationSplit.



```python
def RS(thecolumn, datatrain, datatest):
    als = ALS(userCol='userId', itemCol='itemId',
              ratingCol=thecolumn, nonnegative=True,
              )
    #Tune model using ParamGridBuilder
    param_grid = ParamGridBuilder()\
                .addGrid(als.rank,[10,11,12])\
                .addGrid(als.maxIter,[22,21,20])\
                .addGrid(als.regParam, [.07, .05, .1])\
                .build()
    #Define evaluator as RMSE
    evaluator_ = RegressionEvaluator(metricName="rmse",labelCol=thecolumn,
                                     predictionCol = "prediction")
    #Build cross validation using TrainvalidationSplit
    tvs = TrainValidationSplit(
        estimator = als,
        estimatorParamMaps = param_grid,
        evaluator=evaluator_)
    #Fit ALS model to training data
    model = tvs.fit(datatrain)
    #Extract best model from the tuning exercise using ParamGridBuilder
    bmodel = model.bestModel
    # Generate Predictions and evaluate using RMSE
    predictions = bmodel.transform(datatest)
    predictions = predictions.na.drop()
    rank = bmodel.rank
    rmse = evaluator_.evaluate(predictions)
    mIter = bmodel._java_obj.parent().getMaxIter()
    rParam = bmodel._java_obj.parent().getRegParam()
    return (predictions, bmodel, rmse, rank, mIter, rParam)
```

Figure 5. Recommendation System class

In the RS class, the recommendation model, predictions and evaluator build together. After the class is made, get the prediction, model and RMSE values by calling the class with input variables. The column in the input variable is for each star rating and testimonial separately. After getting the prediction matrix: get the top n products; give point for the products; mix the products; rank the products. For example here, get the top 5 product for each model and the recommendation to the user with Id 1580 result. In Table 2, there are lists of top 5 Item Ids each by star rating and testimonial. The items are given points from the highest to lowest order. The first sequence gets 5 points, the second sequence gets 4 points and so on. In table 3, items in table 2 are put together by matching the item id and averaging points by star rating and testimonials. For example, item 16 has 5 points star rating and 4 points testimonial, the final points (4+5)/2=4.5 and so on. The final top 5 recommendations for user 1580 based on item ids are 16, 26, 7, 35 and 3.

Table 2. Example of a recommendation list

Item Ids		**Points**
By star rating	*By testimonial*	
16	36	5
35	16	4
36	3	3
7	7	2
17	28	1

Table 3. Example of mix and rank recommendation list

Rank	**Item Ids**	**Points**		**Final Point**
		By star rating(s)	*By testimonial(t)*	
1	16	5	4	4.5
2	36	3	5	4
3	7	2	2	2
4	35	4	0	2
5	3	0	3	1.5
6	17	1	0	0.5
7	28	0	1	1

E. Evaluations

Evaluations were carried out on both the algorithms used, SA and RS. The sentiment evaluation using classification evaluator which is multi-class confusion matrix. The RS evaluation using regression evaluator, Root Mean Square Error (RMSE).

The classification evaluator used for discovery and proof observation. The sentiment evaluation using measuring tool based on the size of success with accuracy, precision, recall, and f1-score in the multi-class confusion matrix. The evaluation does not use all data, only 224 lines. The sentiment classes in the data are only 1, 2, and 3. The accuracy of the multi-class confusion matrix is 0.987. With "classification_report" class in Sklearn library, the values of evaluations can be seen in Table 4.

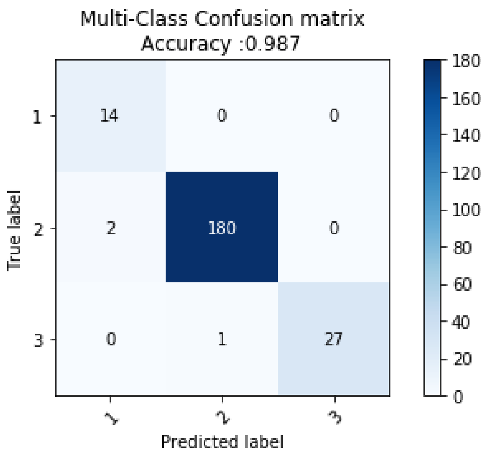


Figure 6. Multi-class confusion matrix diagram

Table 4. Classification report

Class	**Precision**	**Recall**	**F1-Score**	**Support**
1	0.88	1.00	0.93	14
2	0.99	0.99	0.99	182
3	1.00	0.96	0.98	28
Micro avg	0.99	0.99	0.99	224
Macro avg	0.96	0.98	0.97	224
Weight avg	0.99	0.99	0.99	224

The regression evaluator uses to measure the presence or absence of correlation between variables. PySpark provides evaluator class named Regression Evaluator. By importing this class, the evaluation for RS is 0.267≈0.346 with input training-testing (80:20)% best model parameters are in Table 5. The RMSE values got from the results of 10 experiments with the same data but took randomly 10 times.

Table 5. Parameters on the model

Parameters Name	**Value Selection**	**Best Value**
Rank	[10, 11, 12]	10
maxIter	[21, 22, 20]	22
regParam	[0.07, 0.05, 0.1]	0.07

VII. CONCLUSION AND FUTURE WORK

The architecture (Figure 2) could be implemented by combining SA and RS. The system gives recommendation for every user. The SA algorithm has high accuracy but still has limitations that can be overcome in subsequent studies. The limitations are the SA does not consider the subject of sentiment, idiom, and ambiguous word. The sentiment words can be developed again, considering that it was made based on existing reviews. The RS with more data training and experiment different PySpark collaborative filtering parameters could use to get the best errors. Components on the reviews data(star rating, testimonial, timestamp, etc) keep develop, therefore the architecture of recommendations system can be developed using different components of the review data and of course different methods.

REFERENCES

[1] G. Zaccone. and R. Karim, "*Deep Learning with TensorFlow*", 2nd ed. [S.l.]: Packt Publishing, 2018.

[2] C. Pan and W. Li, "*Research paper recommendation with topic analysis*", In the Proceedings of the 2010 International Conference On Computer Design and Applications (ICCDA, 2010)

[3] A. Ziani et al., "*Recommender System Through Sentiment Analysis*", in 2nd International Conference on Automatic Control, Telecommunications, and Signals, Annaba, 2017.

[4] R. Guimaraes, D. Rodriguez, R. Rosa, and G. Bressan, "*Recommendation system using sentiment analysis considering the polarity of the adverb*", In the Proceedings of the 2016 IEEE International Symposium on Consumer Electronics (ISCE), Sao Paulo, Brazil, 2016. ISSN 2159-1423.

[5] Y. Wang, M. Wang and W. Xu, *"A Sentiment-Enhanced Hybrid Recommender System for Movie Recommendation: A Big Data Analytics Framework"*, Wireless Communications and Mobile Computing, Vol. 2018, pp. 1-9, 2018.

[6] G. Asrofi Buntoro, T. Bharata Adji and A. Erna Purnamasari, "*Sentiment Analysis Twitter dengan Kombinasi Lexicon Based dan Double Propagation"*, Conference on Information Technology and Electrical Engineering, pp. 39-43, 2014.

[7] G. Qiu, B. Liu, J. Bu, and C. Chen, *"Expanding domain sentiment lexicon through double propagation"*, In the Proceedings of the 2009 International Joint Conference on Artificial Intelligence, Pasadena, California, USA, 2009, pp. 1199-1204.

[8] Y. Koren, R. Bell, and C. Volinsky, *"Matrix Factorization Techniques for Recommender Systems"*, Computer (IEEE, 2009), vol. 42, no. 8, pp. 30-37, 2009.

[9] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, *"Large-Scale Parallel Collaborative Filtering for the Netflix Prize"*, Algorithmic Aspects in Information and Management, pp. 337-348.

[10] B. Liu, *Sentiment Analysis and Opinion Mining*. Morgan & Claypool, 2012.

[11] B. Verma, R. Thakur, and S. Jaloree, *"Predicting Sentiment from Movie Reviews Using Lexicon Based Model"*, International Journal of Computer Sciences and Engineering, Vol. 6, No. 10, pp. 28-34, 2018.

Authors Profile

Dea Dania pursued Bachelor of Science from the University of Gunadarma, Indonesia in 2017. She is currently pursuing an information systems master's degree.

Lily Wulandari pursed Bachelor, Master, and Doctor of Information Technology from University of Gunadarma, Indonesia. She is head of Industrial Technology Development Institute and currently working as Senior Lecturer, Oracle course instructor and Oracle Workforce Development Program in Gunadarma University. Her main research work focuses on information interoperability which are semantic web, web services, ontology, query rewriting, query processing. She has 20 years of teaching experience and 15 years of Research Experience.