Rule-based Complaint Detection using RapidMiner

Salma Tayel[1], Matthias Reif[2], Andreas Dengel[3]

[1]Department of Computer Science and Engineering
German University in Cairo, Egypt
[2,3]German Research Center for Artificial Intelligence (DFKI GmbH)
D-67663 Kaiserslautern, Germany

Abstract

Complaints are signs of user dissatisfaction from a service or product. Still, they are a good source of feedback for companies. Considering people reviews and complaints on the web can help them meet their customers' expectations. Manually processing web articles related to some business to find out what people think of it is very time and effort consuming. Therefore, this task should be automated. Rule-based classifiers are very suitable for complaint detection. The generated classifiers are formed of rules that are comprehensible. This makes it easy for an employee to understand the criteria for classifying text as complaint. Our work compare five rule-based algorithms, *OneR, ConjunctiveRule, Ridor, RIPPER*, and *PART* using *RapidMiner* against complaint detection data sets of different domains. Results show that *PART* algorithm is the most suitable for complaint detection task. It achieves average 75% accuracy compared with about 60% for the other algorithms.

1 Introduction

Every business is based on customers needs. Therefore, responsible people care a great deal about customer feedback. Big companies receive a huge amount of feedback daily. Moreover, many customers tend to use online review services to express their opinions about the business of companies. Among these opinions there can be many complaints and expressions of dissatisfaction. These services are available to anyone. So, customers, as well as potential customers, might be affected by negative reviews leading to affecting the business drastically. Besides, knowing why customers are not satisfied with a service or product help fixing the problems. This way, customers are satisfied with the enhanced service and they appreciate the response to their concerns. Manually processing all online reviews related to a specific product is a very time and effort consuming task. Therefore, there should be an automated system for online complaint detection. Complaint detection (*CD*) is special type of text categorization (*TC*) problem. *TC* is defined as the process of assigning a label or a class to some text. This label belongs to a set of predefined labels [13]. *CD* is a two-class *TC* problem. The class/label can be either complaint (c_c), as a main class, or non-complaint (c_n).

Nowadays, there are many types of classifiers that can handle *CD* task, e.g. *Support Vector Machine (SVM)*, *Neural Networks (NN)* ...etc. Such classifiers are called black-box classifiers. They do not provide any understandable information of the classification criteria for a normal clerk. Only a decision is given along with some weights that need an expert to understand [4]. Using such black-box classifiers don't allow employees to understand the used criteria for the classification. Hence, they won't be able to expect its output. This way a human would not be able to know whether this classifier agrees with their prediction or not. This is the main motivation to use rule-based classification. Rule-based classification technique is based on generating a model from training data. The model is composed of a set of rules, each has antecedents and a consequence. If the antecedents are satisfied by a document, consequently, it is labeled as given by the consequence. The rules clarify the reason for the classification decision and avoids the unpredictable conclusions drawn by black-box predictors [4].

In this work, we used *RapidMiner's WEKA* extension, as a tool that contains many rule-based algorithms, to test the suitability of rule-based classification algorithms for *CD*. First section 2 describes some previous related work. We use the data sets describes in 3.3 to compare the results of the five algorithms explained in section 3.2 according to the performance measurements discussed in section 4.1. Results are discussed in section 4.

2 Related Work

There have been previous trials to classify documents using rule classifiers. In [3], Dengel proposed a rule generation technique to build a classifier for office documents. In order to represent the documents for classification, the proposed way is based on *distance-separated-patterns*. First, all candidate words and terms are extracted from the text. Then, they are evaluated according to the hit rate to select pattern candidates. After that, patterns are generated according to the maximum distances between terms. For example, "good [*d*] service", where "good" and "service" are two anchor words that exist in one sentence, *d* is the maximum number of words separating them in all the documents. After that, the documents are learnt to construct conjunctive and disjunctive rules. This technique is meant for hierarchical classification of multi-classes.

Ebert [12] evaluated different machine learning techniques to build a complaint detection system. Among the techniques used are decision trees and one-rule. Decision trees are built from the training data based on separate-and-conquer strategy which recursively divides the data set into smaller data sets. Each of the small data sets is represented by a rule. One-rule classification technique builds one rule to represent the whole data set. Results show that decision trees have long training time and their performance is not promising. On the other hand, one-rule's results encourage further investigation in rule-based complaint detection.

3 Rule-based Complaint Detection

In order to evaluate how suitable rule-based algorithms are for *CD* we compared five algorithms. At first, features are selected to represent the instances of the data set. Then, the feature set is used to train a classifier using the rule-based algorithm.

3.1 Feature Selection

Text blocks cannot be interpreted by machines. Therefore, text is converted into a set of features that represents it [13]. The feature set should be sufficient to represent the data in a detailed way. Still, the more the features, the less the efficiency of the training. In this work we used *uni-grams* technique for feature selection. The features are the set-of-words occurring once or more in training data set documents. Each document is represented by an integer vector, representing the number of occurrences of the word in the document. In [12], Ebert showed that *uni-grams* performs at least as good as more complicated techniques. However. it was observed that the feature set generated for a data set of 200 reviews exceeds 4,000 features and this number increases rapidly by increasing the data set size. In order to decrease the size of the feature set, we removed stop words (i.e. conjunctions, articles, and quantifiers). These word do not have a meaning on their own. Therefore, removing them will decrease the feature set without affecting the accuracy. The values for the features are the counts of occurrences of the terms in the documents.

3.2 Rule Generation Algorithms

We selected five algorithms for our test, *One-rule, Conjunctive-rule, Ridor, RIPPER,* and *PART*. For the test, we used the *WEKA* extension for *Rapid-*

Miner. The user interface allows building processes for different algorithms and compare the results. Follows a brief description for the evaluated algorithms.

3.2.1 One-Rule (OneR)

Holte [9] proposed a simple rule learner. The algorithm is based on ranking all the attributes based on the error rate. Then, the algorithms builds exactly one rule consisting of one attribute. This rule is used to classify test data. Results in [12] show that one-rule generates promising results with *CD*.

3.2.2 Conjunctive-Rule

The idea of the algorithm is based on generating one rule from the whole training data set. This rule consists of antecedents, that are "ANDed" together, and the consequence is the class. If a test instance is not covered by the rule, the prediction depends on the distribution of the instances in the training data set that are not covered by the rule [8].

3.2.3 Ripple Down Rules (RIDOR)

The idea of the *Ripple Down Rules* is based on forming a general rule and then forming exceptions for this rule. The parent rule assigns the instance to a main class based on the premises. The exception rules assign the instances to other labels based on the exception premises. If an instance satisfies the premise of main rule, then, it is classified to the main class unless it matched one of the exceptions [7].The rules generated resemble a tree where each rule has exceptions that in turn have exceptions. *Incremental Reduced Error Prunning (IREP)* is used to prevent overfitting of the rules [1].

3.2.4 RIPPER (JRIP)

RIPPER stands for Repeated Incremental Pruning to Produce Error Reduction [2]. The idea of this algorithm is based on *Reduced Error Prunning (REP)* [1]. This is a pruning way where the training data set is divided into two parts. The first is *Grow Data* used by the algorithm to generate rules that overfit this data set. The other part is *Prune Data* which is used to trim the rules generated in the growing phase. Trimming is removing conditions from the rules to avoid overfitting. *RIPPER* algorithm follows the *separate-and-conquer* strategy [6]. This strategy divides the data set into subsets, generates a rule for one subset, removes this subset, and conquers the remaining subsets.

3.2.5 PART

PART algorithm is a combination of both *C4.5* [11] and *RIPPER* [2]. It is based upon building a partial tree from the full training data set. A partial tree is a regular tree that contains unexplored branches [5]. In order to build this partial tree, Subtree replacement as a pruning strategy is followed during building the tree. First, the algorithm expands the nodes based on minimum entropy until a node is found whose all children are leaves. Then, the pruning process starts. Subtree replacement checks if the node is better replaced by one of its leaf children. The algorithm then follows the *separate-and-conquer* strategy.

3.3 Data Sets

In our evaluation we used three data sets of reviews having different sizes and domains.

– *Movie* data set contains 1,000 complaint reviews and 1,000 non-complaint about movies. The data set consists of 2,000 processed down-cased text files used in [10]. The record's polarity is decided according to an automatic rating classifier [1].
– *Kindle* data contains 329 complaint and 329 non-complaint Amazon reviews [2].
– *Galaxy S3* data contains 100 complaint and 100 non-complaint Amazon reviews [2].

The Amazon data sets are arranged according to the star rating. All the reviews were read and manually classified as complaints or non-complaints. We removed incomplete reviews and reviews written in different languages [3]. The reviews vary in the size, they range between very short reviews that consist of few words to reviews that consist of three or more paragraphs.

In order to ensure the reliability of the results, 10-folds cross validation test was followed. The data set is divided into 10 equal subsets. Each of them is used once as testing data where the other 9 subsets are the training data. We used *RapidMiner* cross validation operator.

[1]Further description of the data set can be found at http://www.cs.cornell.edu/people/pabo/movie-review-data/poldata.README.2.0.txt

[2] We used an online script to crawl the data set given the ID of the item and the amazon domain http://www.esuli.it/software/amazon-reviews-downloader-and-parser/

[3]The data is available upon request

Table 1: Contingency table for a complaint classification problem

		PREDICTED CLASS	
		COMPL.	**NON-COMPL.**
TRUE	**COMPL.**	TP	FN
CLASS	**NON-COMPL.**	FP	TN

4 Evaluation

4.1 Performance Measurements

In order to evaluate classification results, they are represented in a contingency table as shown in table 1. True-Positive (TP) is the number of correctly classified complaints. True-Negative (TN) is the number of correctly classified non-complaints. False-Positive (FP) represents the misclassified non-complaints. False-Negative (FN) is for the misclassified complaints. In order to evaluate the classification, the *accuracy*, *precision*, and *recall* are calculated from the contingency table.

Accuracy (a): is the ratio between correctly classified documents and all documents (n)

$$a = \frac{TP + TN}{n} \tag{1}$$

Precision (p): is the ratio between correctly classified complaints and all classified complaints

$$p = \frac{TP}{TP + FP} \tag{2}$$

Recall (r): is the ratio between correctly classified complaints and all true complaints

$$r = \frac{TP}{TP + FN} \tag{3}$$

4.2 Results and Discussion

Table 2(a) Shows the results of evaluating the five algorithms using the movie data set. *PART* algorithm results in the best performance. The performance of the other four algorithms is very similar to each other around 60%. *RIPPER* algorithm results in low accuracy, precision, and recall of 63% but the

Table 2: Performance Measurements for the data sets

(a) Movie data set

	Accuracy	sigma	Precision	sigma	Recall	sigma
1 R	62.80	2.15	66.70	3.32	51.50	3.83
Con-R	58.75	2.99	74.24	7.42	27.80	9.83
Ridor	62.80	2.59	74.33	11.05	45.80	16.68
JRIP	63.40	3.27	63.86	3.83	63.50	6.93
PART	70.55	3.81	70.46	4.14	71.00	4.00

(b) Kindle data set

	Accuracy	sigma	Precision	sigma	Recall	sigma
1 R	63.09	6.19	58.24	4.79	95.75	3.09
Con-R	60.80	4.68	56.58	3.25	94.84	3.25
Ridor	74.94	6.78	72.46	8.28	84.22	13.73
JRIP	76.30	5.57	73.22	6.96	84.78	6.70
PART	79.03	5.12	80.98	7.06	76.62	7.86

(c) Galaxy S3 data set

	Accuracy	sigma	Precision	sigma	Recall	sigma
1 R	70.34	9.36	76.97	11.62	56.33	15.74
Con-R	63.84	6.88	69.29	10.66	58.33	22.32
Ridor	66.89	8.27	69.36	9.40	59.44	13.92
JRIP	70.84	9.98	77.11	11.75	57.33	16.85
PART	72.79	10.81	71.66	9.13	74.67	19.22

results have close precision and recall. These results indicate that the generated classifier does not randomly classify the instances to one label. On the other hand, *One-Rule, Conjunctive-Rule*, and *Ridor* classifiers have much higher precision than recall. This means that the generated classifiers classify most of the instances to the non-complaint class. So, only few instances are correctly classified as complaints resulting in the very low recall and higher precision.

Table 2(b) shows that the same results are repeated with the Kindle data set. *PART* algorithm results in the best performance with accuracy of 79%. The precision and recall have a difference of 4.36% which is not very significant to claim that the classifier randomly classifies the instances. As for *RIPPER* and *Ridor*, the accuracy is close to 75%. The precision is 10% less than the recall, meaning that the classifiers is more inclined to classify instances as complaints. This effect is more obvious with *One-Rule* and *Conjunctive-Rule*. There is a considerable difference of 37% between the precision and recall. The recall is much higher meaning that most of the complaints are detected. The significantly low precision means that most of the non-complaints are misclassified as complaints.

In table 2(c), *PART* gives best performance. Similar to the movies data set, the classifiers generated by the other algorithms tend to classify instances as non-complaints leading to a higher precision that recall.

There is only one rule generated by both *One-Rule* and *Conjunctive-Rule* algorithms. This rule is not enough to represent all cases. Hence, the resulting classifier has low performance. For *Ridor* and *RIPPER* algorithms, the generated classifier classifies randomly due to overpruning. This results from replacing a general rule by a more general one during the pruning process [5]. While the new rule might give the same result with the pruning data, it is too general for the test data leading to the misclassification of many instances. *PART* algorithm gives quite promising result with CD. The classifier generated by this algorithm has a high accuracy between 70% and 80%. However, by checking the generated rules from PART algorithm it was observed that the generated rules are very long. The reason is that the algorithm selects deep branches, long paths, to convert them to rules. Also the number of misclassified instances covered by the rules are significant. This results from ignoring the rule accuracy while concentrating on the maximum coverage.

5 Conclusion and Future Work

In our work we investigated the suitability of rule-based classification techniques for *complaint detection (CD)*. We tested five different rule-based algorithms on three different data sets represented as *uni-grams* after removing stop words. The tests show that *One-Rule* and *Conjunctive-Rule* algorithms generate semi random classifiers. *RIPPER* and *Ridor* algorithms generate classifiers that underfit the data due to overpruning. *PART* algorithm generates accurate classifiers that have high performance measurements.

A classifier generated by *PART* algorithm from *uni-grams* feature set can detect complaints up to an accuracy of 79%. Still, there is a possibility for enhancing the results by doing the following:

- Applying some filtering techniques to the *uni-grams* feature set. As the feature set size decrease, the training time decrease. This might also increase the accuracy due to removing insignificant features.
- Trying some different rule-selection-metrics for *PART* algorithm. These metrics should consider both precision and recall not just the coverage of the rules.

The previous changes have a great possibility of enhancing the performance of complaint detection systems built using rule-based classifiers.

References

[1] W.W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. In *Proceedings of the Thirteenth International Joint*

Conference on Artificial Intelligence, pages 988–994. Morgan Kaufmann, 1993.

[2] W.W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[3] A.R. Dengel. Learning of pattern-based rules for document classification. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 123–127, 2007.

[4] W. Duch. Rule-based methods. In W. Dubitzky, O. Wolkenhauer, H. Yokota, and K. Cho, editors, *Encyclopedia of Systems Biology*. Springer London, Limited, 2013. (in print, accepted 2011).

[5] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. pages 144–151. Morgan Kaufmann, 1998.

[6] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13:3–54, 1999.

[7] B.R. Gaines and P. Compton. Induction of ripple-down rules applied to modeling large databases. *Journal of Intelligent Information Systems*, 5(3):211–228, 1995.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[9] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[10] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, 2004.

[11] J. R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[12] S. Ebert. Separating the Good from The Bad: Investigation of Machine Learning Approaches for Detecting Documents with Complaint Character. In *Master Thesis, University of Kaiserslautern*, 2012.

[13] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.