Opinion Mining in Roman Urdu using Baseline Classifiers

Zareen Sharf, Husnain Ali Mansoor

PhD Scholar SZABIST Associate Professor SZABIST

1. Introduction

Sentiment Classification is a division of text mining that also incorporates information extraction, lexical analysis and other related techniques. A number of methods (used in text mining) are utilized in sentiment mining as well. But sentiment classification is different from the factual-based text analysis in a sense that there are special characters used in sentiment expressions in different languages. There are many applications of opinion mining and sentiment classification. And one of the significant applications is customer review mining. There have been many studies recorded on different review sites in this regard. In this paper, a comparison of nine popular classification methods is presented and is cross validated using ensemble approach that combines five classification algorithms into an ensemble classification algorithm. These standalone algorithms include Support Vector Machine (SVM), Random Forest Classifier, Decision Tree Classifier, Regression, Perceptron and K-Nearest Neighbor methods. The ensemble algorithm first takes the results of standalone classifiers and then Majority-Vote method is applied to determine the final sentiment class prediction. more, the experimental results with comparison of the mentioned standalone classifiers with the proposed one are discussed in a detailed manner.

2. Baseline Classifiers and Techniques used for Sentiment Analysis

The training of sentiment classifiers (from the sentiment labeled data) is one of the crucial phases of Sentiment Analysis. There are several supervised machine learning algorithms in order to carry out this training. One of the easiest approaches is Lexicon based approach that executes this process by calculating the sum of positive and negative sentiment words (by extracting them from the input source) and then overall sentiment is calculated. This approach is unable to identify the effect of negation element on the overall sentiment which is its weakness. Besides this, many other supervised machine learning methods are there that mainly depend upon human annotated samples. One of such methods is Naïve Bayes that is widely used in the field of text classification and sentiment analysis. The working principle of this approach is that the probability of an event (when the occurrence of the event is given) can be computed by the joint probability of two events. It makes as assumption that attributes in classification are independent of each other, thus resulting in the reduced computational complexity. Another machine learning method that is employed in sentient classification and analysis on a large-scale is K-Nearest Neighbors. As the name implies, this approach performs comparison of the similarity (of the given document) with its neighbors. This is what differentiates it from other approaches that it does not extract any features from the training dataset. Support Vector Machine (SVM) is not only a widely used machine learning method but considered the best one to perform classification. In this approach, the maximization of margin between instances and separation hyperplane is performed. [1] concluded in this research work that SVM is far better than Naïve Bayes. While [2] performed a research in which four feature selection methods and five machine learning approaches were compared. This research was performed on Chinese texts and concluded that Information Gain algorithm showed the best results among feature selection methods and SVM is the best among classification algorithms.

Several techniques and methods for sentiment classification have been employed in the course of previous years. These can be classified into the following three categories:

- Machine Learning Algorithms
- Link Analysis Methods
- Score-Based Approaches

There are various methods and approaches that have been utilizing for sentiment classification and opinion extraction. These techniques and methods can be categorized into three main classes: machine learning algorithms, link analysis methods and score-based approaches [3]Most of the research on sentiment classification focuses on the opinions presented in English. But usually people leave their suggestions or comments in their native languages. Due to the lack of resources for other languages, sentiment classification (in languages other than English) is not that much accomplished. But work on other languages is also growing [4]

In the previous years, several approaches and methodologies have been applied in text classification for extracting sentiments or opinions. All of this research

Manuscript received September 5, 2018
Manuscript revised September 20, 2018

revolves around machine learning approaches [5, 3], lexicon-based approaches [6, 7, 8] and hybrid approaches. [9] conducted a study that measured the effectiveness of machine learning techniques after applying them on sentiment classification. Many of the machine learning techniques and approaches have been utilized with Support Vector Machine (SVM) and Naïve Bayes (NB) as the most commonly used methods. Whereas SVM has been widely utilized for movie reviews while NB has been in use for web discourse and reviews. When these algorithms were compared, SVM showed better results and outperformed NB and other algorithms. Genetic Algorithms (GAs) are basically probabilistic search methods that are similar to the process of biological evaluation and natural selection as well as survival of the fittest. When they are applied for natural selection and genetics (in artificial intelligence), they give you a solution (from the set of feasible solutions) that is globally optimized [10] GAs perform their working with a huge set of syntactic, semantic and discourse level feature set. Then, the computational accuracy of subjectivity classifier is determined by applying the fitness function. The feature seat is actually identified through natural selection by the operations of crossover and mutation. A different technique (from the discussed ones) is ensemble technique. It basically takes outputs from different base classification models and integrate them into a single output. This is considered to be highly-effective technique for different domains [11, 12].The ensemble technique also played a vital role in making improvements and accuracy in the discipline of classification in topical text classification. In early research, a combination of different classification algorithms was utilized [13] and proved that a higher accuracy is achieved in results by utilizing the combination instead of using individual approaches. In actual, the ensemble technique acquires this accuracy by combining arrays of specialized learners. Bootstrapping was one of the initial ensemble techniques. This is also called as Bagging [14]. An approach called Inverse Document Frequency was proposed by [15] in which classification was performed by utilizing bagging algorithms. In this study, bagging was applied and evaluated on movie reviews in combination with NB, SVM and GA, these were used as the base learners. Then, the performance of these bagged algorithms was then compared with that of individual classifiers.

3. Baseline Classifiers Selected for Roman Urdu Classification

In this research we selected a variety of classifiers from different domains for establishing performance comparison metrics for the proposed model (discussed in Chapter 5). These are listed below:

Supervised Learning

Multinomial NB

[29] MultinomialNB basically practices the Naive Bayes algorithm for multinomially distributed data. It is one of the two classic Naive Bayes variants that is widely used in text classification in which data is typically represented in the form of word vector count.

SGD Classifier

[29] This is the Linear classifiers (SVM, logistic regression.) with SGD training. This estimator in actual implements the regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated for each sample at a time and the model is updated along the way with a decreasing strength schedule (learning rate).

Linear SVC

[29] The main objective of Linear SVC (Support Vector Classifier) is to fit the data (provided to it), returning the 'best fit' hyperplane that divides or categorizes the data.

Ridge Classifier

[29] Ridge Regression is a remedial measure taken to alleviate multi-collinearity amongst regression predictor variables in a model. Often predictor variables used in a regression are highly correlated.

Unsupervised Learning

KNeighbors Classifier

[29] K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute. KNN makes predictions using the training dataset directly. Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances.

Tree Based Model

Decision Tree Classifier

[29] Another supervised machine learning classification technique utilized in this study is Decision Tree Classifier. Decision Trees (DTs) refer to a non-parametric supervised learning method used for classification and regression. The major aim is to build a model that may predict the value of a target variable by learning simple decision rules inferred from the data features.

Neural Network Based Model

Perceptron

[29] A simple machine learning algorithm appropriate for large scale learning is Perceptron. By default: It does not require a learning rate, It is not regularized (penalized) and it updates its model only on mistakes.

Ensemble Models

RandomForest Classifier

[29] Random Forest is a supervised machine learning algorithm that works on the principle that it builds a forest - an ensemble of decision trees to make an accurate and stable prediction. In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.

Voting Ensemble (Equal Weighted) Model

[29] The concept of **Voting Classifier** is to combine various machine learning classifiers and use majority vote (or the average predicted probabilities referred as soft vote) to make a prediction about the class labels. This kind of classifier is quite useful in the situation where there is a set of equally well-performing model to balance out their individual weaknesses.

4. Data Collection

To train a sentiment classifier, a fairly large size of training dataset of records (already labeled with sentiment) is required. In this study, the irrelevant records (i.e. records other than those in Roman Urdu) were discarded and each relevant record in the dataset was manually labeled as positive, negative or neutral sentiment. As shown in Table 4.1 our data corpus comprises of approximately 22000 records. Out of which 4500 records are labeled positive, 4900 records are labeled negative, 13000 records were labeled neutral and approximately 5000 records were discarded for being irrelevant. A list of web sources and Facebook pages used for retrieval of records is presented in Table 2 and Table 3.

Table 4-1: Dataset Distribution

Total Records	Positive	Negative	Neutral
22000	4500	4900	13000

Table 2: Web Sources for Data Collection

Sources	
Biographies	Womens_Fashion
UrduSafha	Express Urdu
Khabees	Reddit
Others	Comments & Reviews
Bitchy Urdu Cards	Urdu Poetry
AryNewsAsia	Sentiment Analysis Interviewer
Social Workers	Careem
Phones_tablets	2LinePoetry
Blog Khuwaar	Mens_Fashion
Beauty_Health	

Table 3: Facebook Pages for Data Collection

Source			
Expressnews pk	SHUGAL	sarcasm123	MRSMofficial
UrduAdab	TheOlaadMovement	arhamsayss	IdaraTulMustafaInternational
NaziaH	urdughazal	halalhumour	girls.pk
LolWaalay	EdaTuD	Dostonkibateinn	english.eminem
HusseyOfficialClub	HappensInPk	Filmygyan7	jppk123
Entertainmenttrackerpage	WBloverss	Pakistanii Awaaaam	hadeed.khi
css	urdupoint	Sheikhspeareofficial	GHStrangers
Programmer KiBaatein	OfficialLahoreQalandars	AkbarSayss	Gulshan-E-Hadeed-Official
shahlylaB7	EmployeesUpGradation	Bfkbaatein	Hadeediansrocks
NVinez	UzairAltafPage	ChotiSiQayamat.seetv	yaradaynaalyari
TheMahiraKhanOfficial	ChupbeyTharki	Alif-on-See-TV	zindgigulzarha
ArifaSobhKhan	ItssAZee	Humorists2	Aghaz-Society
PyaareNabi KiBaatain			

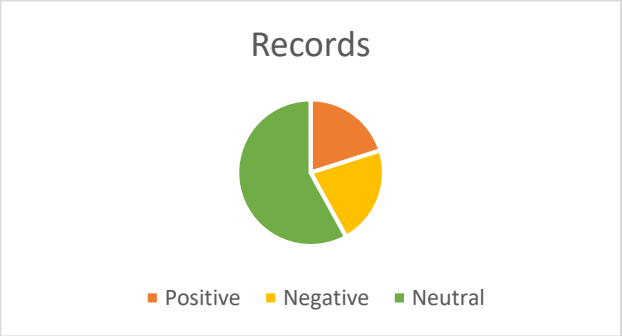

Fig. 1 Dataset Distribution based on Sentiment Polarity

5. Human Annotation

The large-scale sentiment analysis can be performed through any of the two main and prominent approaches:

i- Lexicon-Based Approach

ii- Machine Learning Approach

The first approach carries out sentiment analysis by computing sentiment (in the text) from the set of sentiment-bearing words (present in the given text). But the second approach builds a sentiment classification model from the set of sentiment labeled text and then applies it on the stream of unlabeled texts. The model constructed in this

approach has a form of function that easily maps features (extracted from the text) into sentiment labels (that have discrete values: positive, negative or neutral). It is important to be noted that a significant human involvement (at least at initial level) is required in both approaches of sentiment analysis because the perception of sentiment (expressed in words or short text) needs to be labeled by human beings. This automatically leads to their involvement. Moreover, the sentiment labeling is specific to domain, topic and language.

In this study, we have collected and analyzed a set of over 22000 reviews and comments in Roman Urdu (and these posts are labeled by human annotators). To train sentiment classifiers in Roman Urdu, these labeled tweets are utilized as training data.

6. Ordering of Sentiment Values

The polarity of a given text is one of the significant elements in sentiment analysis. So, the classification of polarity of given text (on document, sentence or feature level) is one of the basic tasks in sentiment or opinion extraction. This opinion or sentiment (expressed in that given document, sentence or feature) may be positive, negative or neutral.

7. F1 Score

The accuracy of a test is also very important and must be proved to show the reliability of that research. F1 score (also called as F-measure or F-score) is one of the performance measures to measure the accuracy of a test in statistical analysis of binary classification. Precision and Recall are other important measures that are required to be analyzed. Denoted by p, precision refers to the number of correct positive results divided by all positive results. Whereas recall is denoted by r and represents the number of correct positive results divided by the number of positive results needed to be returned in original, for computing the score, F1 makes use of both precision p and recall r of the test. To compute a score, F1 score contemplates both precision ‘p’ and recall ‘r’ of the test. It can be explained as the weighted average of precision and recall. When F1 score reaches around 1, it represents its best value while the value of 0 represents the worst one.

As per definition

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{F1 Score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

7.1 F1 Scores of Baseline Classifiers

The accuracy of the system can be measured through variant measures based on precision and recall by considering the two target categories comprising of positive and negative texts. This accuracy demonstrates that how much the results (produced by the proposed sentiment analysis approach) match with the human judgements.

Table 4 gives a comparison of the time it took for the classifiers to train themselves on a dataset of almost 18000 records and to test a dataset of 4000 records for sentiment prediction. Figure 2 also shows the same information as an illustration.

Table 4: Classifier wise Training and Testing Time

Classifiers	Training Time (Seconds)	Testing Time (Seconds)
Multinomial NB	0.124	0.0069
SGD Classifier	2.155	0.0069
Linear SVC	1.437	0.0039
Ridge Classifier	2.065	0.0190
KNeighbors Classifier	0.038	2.963
Decision Tree Classifier	0.599	0.0060
Perceptron	0.134	0.0069
RandomForest Classifier	426.577	1.691
Voting Ensemble (Equal Weighted) Model	0	0

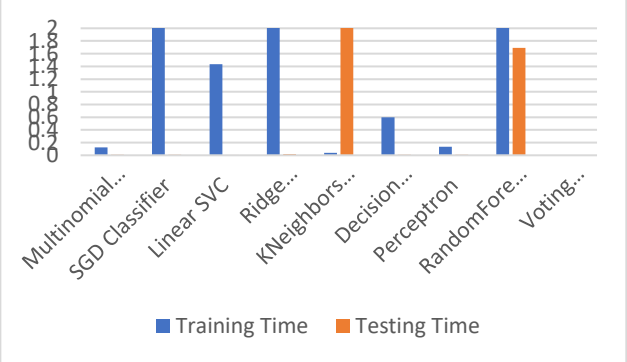

Fig. 2 Comparison of Training and Testing Time of Classifiers

For our study we divided the dataset into two parts: The Training Set comprised of 18083 records whereas the Test dataset comprised of 4521 records. As apparent from Table 4 and Figure 3 it takes RandomForest Classifier the longest to get trained as well as to process test data whereas KNeighbors model took the least time to train and Linear SVC model took the least time to predict results for the test data. Table 5 shows the values of precision, recall and F1-score when the baseline classifiers were trained on the given dataset. A detailed account of the calculations of individual classifiers as well as the confusion matrices are presented in Appendix A. The best results were produced by Linear SVC giving an F1-score of 81 percent whereas the worst performance was given by Decision Tree which gave an F1-score of 38 percent. All the other scores were between 40 to 80 percent.

Table 5: Classifier Wise F1 Score and Accuracy

Classifiers	Precision	Recall	F1-Score	Accuracy
Supervised Learning				
Multinomial NB	0.69	0.70	0.69	0.700
SGD Classifier	0.73	0.72	0.70	0.721
Linear SVC	0.81	0.81	0.81	0.807
Ridge Classifier	0.78	0.79	0.78	0.785
Unsupervised Learning				
KNeighbors Classifier	0.72	0.6	0.46	0.595
Tree Based Model				
Decision Tree Classifier	0.69	0.41	0.38	0.595
Neural Network Based Model				
Perceptron	0.76	0.76	0.76	0.763
Ensemble Models				
RandomForest Classifier	0.78	0.76	0.75	0.760
Voting Ensemble (Equal Weighted) Model	0.62	0.85	0.71	0.726

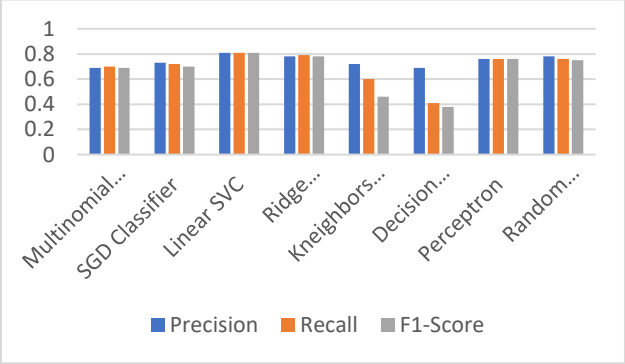

Fig. 3 Comparison of F1 Scores of Classifiers

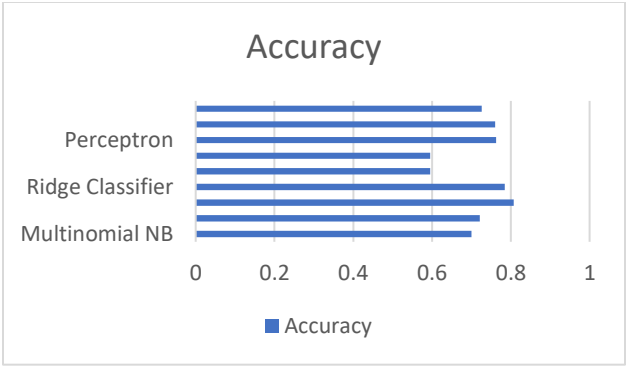

Fig. 4 Comparison of Accuracy of Classifiers

8. Optimization

Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. Typical examples include C, kernel and gamma for Support Vector Classifier, alpha for Lasso, etc. Two generic approaches to sampling search candidates are provided in scikit-learn: for given values, GridSearchCV exhaustively considers all parameter combinations, while RandomizedSearchCV can sample a given number of candidates from a parameter space with a specified distribution. Therefore, in order to optimize the algorithm parameters, cross-validation is performed on the training data using GridSearchCV. In this way the predictions of various single learning algorithms can be improved. There were three classifiers selected for this purpose. Linear SVC and Ridge Classifier were selected as they gave the best results and we wanted to observe if the results could be improved using optimization. Perceptron was selected to ascertain if neural network model could perform better after optimization.

8.1 Support Vector Machine

The default parameters used for SVC were:
Kernel='rbf'
Gamma='auto'
C= 1
Where:

Kernel represents the kernel type to be used in the algorithm. It can have one of these values 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. 'rbf' is used as default value.

Gamma is the Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. If gamma is 'auto' then 1/features are used instead.

C is the Penalty parameter of the error term. Its default value is 1.

SVC revealed the following values for Precision after its hyper-parameters were tuned as shown in Table 6.
Kernel = ['rbf'] gamma= [1e-3, 1e-4] C = [1, 10, 100, 1000]
Kernel'= ['linear'] C= [1, 10, 100, 1000]

Table 6: Optimization of SVC for Precision

Kernel	C	Gamma	Value	Std Dev
rbf	1	0.001	0.194	+/- 0.000
rbf	1	0.0001	0.194	+/- 0.000
rbf	10	0.001	0.194	+/- 0.000
rbf	10	0.0001	0.194	+/- 0.000
rbf	100	0.001	0.823	+/- 0.048
rbf	100	0.0001	0.194	+/- 0.000
rbf	1000	0.001	0.792	+/- 0.016
rbf	1000	0.0001	0.823	+/- 0.048
linear	1	-	0.807	+/- 0.012
linear	10	-	0.786	+/- 0.007
linear	100	-	0.781	+/- 0.011
linear	1000	-	0.775	+/- 0.010

SVC revealed the following values for Recall after its hyper-parameters were tuned as shown in Table 7.

Table 7: Optimization of SVC for Recall

Kernel	C	Gamma	Value	Std Dev
rbf	1	0.001	0.333	+/- 0.000
rbf	1	0.0001	0.333	+/- 0.000
rbf	10	0.001	0.333	+/- 0.000
rbf	10	0.0001	0.333	+/- 0.000
rbf	100	0.001	0.390	+/- 0.008
rbf	100	0.0001	0.333	+/- 0.000
rbf	1000	0.001	0.652	+/- 0.009
rbf	1000	0.0001	0.390	+/- 0.007
linear	1	-	0.607	+/- 0.010
linear	10	-	0.644	+/- 0.010
linear	100	-	0.642	+/- 0.010
linear	1000	-	0.639	+/- 0.009

8.2 Ridge Classifier

The default parameters used for Ridge Classifier were:
tol=1e-2
solver= ‘sag’
Where:

alpha is the regularization strength. It is a positive float and is used to improve the conditioning of the problem and reduces the variance of the estimates. Larger values specify stronger regularization.

tol is the Tolerance for stopping criteria having default value = 1e-4

solver could have any one of these values {‘auto’, ‘svd’, ‘cholesky’, ‘lsqr’, ‘sparse_cg’, ‘sag’, ‘saga’} default being ‘auto’ chooses solver based on the data type. ‘lsqr’ is used because it is the fastest.

Ridge Classifier revealed the following values for Precision after its hyper-parameters were tuned as shown in Table 8.
Alpha = [0,1,2,3,4,5]
tol = [1e-1, 1e-2, 1e-3]
solver = ['sag']

Table 8 Optimization of Ridge Classifier for Precision

Alpha	Tolerance	Solver	Value	Std Dev
0	0.1	sag	0.769	+/- 0.007
0	0.01	sag	0.765	+/- 0.007
0	0.001	sag	0.754	+/- 0.012
1	0.1	sag	0.791	+/- 0.011
1	0.01	sag	0.791	+/- 0.013
1	0.001	sag	0.791	+/- 0.013
2	0.1	sag	0.796	+/- 0.014
2	0.01	sag	0.799	+/- 0.015
2	0.001	sag	0.798	+/- 0.015
3	0.1	sag	0.804	+/- 0.018
3	0.01	sag	0.803	+/- 0.018
3	0.001	sag	0.803	+/- 0.018
4	0.1	sag	0.805	+/- 0.018
4	0.01	sag	0.804	+/- 0.022
4	0.001	sag	0.804	+/- 0.022
5	0.1	sag	0.804	+/- 0.022
5	0.01	sag	0.804	+/- 0.020
5	0.001	sag	0.804	+/- 0.021

Ridge Classifier revealed the following values for Recall after its hyper-parameters were tuned as shown in Table 9.

Table 9: Optimization of Ridge Classifier for Recall

Alpha	Tol	Solver	Value	Std Dev
0	0.1	sag	0.651	+/- 0.012
0	0.01	sag	0.644	+/- 0.010
0	0.001	sag	0.632	+/- 0.009
1	0.1	sag	0.599	+/- 0.005
1	0.01	sag	0.599	+/- 0.008
1	0.001	sag	0.599	+/- 0.008
2	0.1	sag	0.552	+/- 0.015
2	0.01	sag	0.552	+/- 0.014
2	0.001	sag	0.552	+/- 0.013
3	0.1	sag	0.519	+/- 0.017
3	0.01	sag	0.518	+/- 0.017
3	0.001	sag	0.518	+/- 0.016
4	0.1	sag	0.492	+/- 0.016
4	0.01	sag	0.489	+/- 0.018
4	0.001	sag	0.490	+/- 0.018
5	0.1	sag	0.467	+/- 0.012
5	0.01	sag	0.468	+/- 0.013
5	0.001	sag	0.468	+/- 0.012

8.3 Perceptron

The default parameters used for Perceptron were:
n_iter=50
Perceptron revealed the following values for Precision after its hyper-parameters were tuned as shown in Table 10.
alpha = [1e-3, 1e-4, 1e-5]
tol = [1e-2, 1e-3, 1e-4]
shuffle = [True, False]
n_iter = [50,100]
where:
shuffle is the decision whether to shuffle the training data after each epoch or not. Default value is True
n_iter is the number of passes over the training data.

Table 10: Optimization of Perceptron for Precision

Alpha	Shuffle	Tol	N_iter	Value	Std Dev
0.001	True	0.01	50	0.728	+/- 0.019
0.001	True	0.001	50	0.746	+/- 0.011
0.001	True	0.0001	50	0.750	+/- 0.015
0.001	False	0.01	50	0.735	+/- 0.024
0.001	False	0.001	50	0.747	+/- 0.026
0.001	False	0.0001	50	0.748	+/- 0.026
0.001	True	0.01	100	0.728	+/- 0.019
0.001	True	0.001	100	0.746	+/- 0.011
0.001	True	0.0001	100	0.750	+/- 0.015
0.001	False	0.01	100	0.735	+/- 0.024
0.001	False	0.001	100	0.747	+/- 0.026
0.001	False	0.0001	100	0.748	+/- 0.026
0.0001	True	0.01	50	0.728	+/- 0.019
0.0001	True	0.001	50	0.746	+/- 0.011
0.0001	True	0.0001	50	0.750	+/- 0.015
0.0001	False	0.01	50	0.735	+/- 0.024
0.0001	False	0.001	50	0.747	+/- 0.026
0.0001	False	0.0001	50	0.748	+/- 0.026
0.0001	True	0.01	100	0.728	+/- 0.019
0.0001	True	0.001	100	0.746	+/- 0.011
0.0001	True	0.0001	100	0.750	+/- 0.015
0.0001	False	0.01	100	0.735	+/- 0.024
0.0001	False	0.001	100	0.747	+/- 0.026
0.0001	False	0.0001	100	0.748	+/- 0.026
1e-05	True	0.01	50	0.728	+/- 0.019
1e-05	True	0.001	50	0.746	+/- 0.011
1e-05	True	0.0001	50	0.750	+/- 0.015
1e-05	False	0.01	50	0.735	+/- 0.024
1e-05	False	0.001	50	0.747	+/- 0.026
1e-05	False	0.0001	50	0.748	+/- 0.026
1e-05	True	0.01	100	0.728	+/- 0.019
1e-05	True	0.001	100	0.746	+/- 0.011
1e-05	True	0.0001	100	0.750	+/- 0.015
1e-05	False	0.01	100	0.735	+/- 0.024
1e-05	False	0.001	100	0.747	+/- 0.026
1e-05	False	0.0001	100	0.748	+/- 0.026

Perceptron revealed the following values for Recall after its hyper-parameters were tuned as shown in Table 11.

Table 11: Optimization of Perceptron for Recall

Alpha	Shuffle	Tol	n_iter	Value	Std Dev
0.001	True	0.01	50	0.677	+/- 0.029
0.001	True	0.001	50	0.672	+/- 0.018
0.001	True	0.0001	50	0.669	+/- 0.015
0.001	False	0.01	50	0.675	+/- 0.017
0.001	False	0.001	50	0.672	+/- 0.030
0.001	False	0.0001	50	0.670	+/- 0.025
0.001	True	0.01	100	0.677	+/- 0.029
0.001	True	0.001	100	0.672	+/- 0.018
0.001	True	0.0001	100	0.669	+/- 0.015
0.001	False	0.01	100	0.675	+/- 0.017
0.001	False	0.001	100	0.672	+/- 0.030
0.001	False	0.0001	100	0.670	+/- 0.025
0.0001	True	0.01	50	0.677	+/- 0.029
0.0001	True	0.001	50	0.672	+/- 0.018
0.0001	True	0.0001	50	0.669	+/- 0.015
0.0001	False	0.01	50	0.675	+/- 0.017
0.0001	False	0.001	50	0.672	+/- 0.030
0.0001	False	0.0001	50	0.670	+/- 0.025
0.0001	True	0.01	100	0.677	+/- 0.029
0.0001	True	0.001	100	0.672	+/- 0.018
0.0001	True	0.0001	100	0.669	+/- 0.015
0.0001	False	0.01	100	0.675	+/- 0.017
0.0001	False	0.001	100	0.672	+/- 0.030
0.0001	False	0.0001	100	0.670	+/- 0.025
1e-05	True	0.01	50	0.677	+/- 0.029
1e-05	True	0.001	50	0.672	+/- 0.018
1e-05	True	0.0001	50	0.669	+/- 0.015
1e-05	False	0.01	50	0.675	+/- 0.017
1e-05	False	0.001	50	0.672	+/- 0.030
1e-05	False	0.0001	50	0.670	+/- 0.025
1e-05	True	0.01	100	0.677	+/- 0.029
1e-05	True	0.001	100	0.672	+/- 0.018
1e-05	True	0.0001	100	0.669	+/- 0.015
1e-05	False	0.01	100	0.675	+/- 0.017
1e-05	False	0.001	100	0.672	+/- 0.030
1e-05	False	0.0001	100	0.670	+/- 0.025

A close analysis of the optimization process reveals that the precision and recall values do not differ very much for the different combination of input values.

9. Results

As can be determined from the results shown in Tables 6 to 11, the values of precision achieved after optimization by SVC, Ridge Classifier and Perceptron model is close to the precision achieved by the default models however a minor difference between these can be observed and that is probably because of overfitting. Overfitting takes place when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the model's ability to generalize. GridSearch utilizes Cross-Validation while default models don't use it hence cause them to overfit. In this case best result was given by SVC with F1 score and accuracy of 0.81 whereas the worst results were computed by Decision Tree classifier having F1 Score value 0.38 and accuracy of 0.595. All the other classifiers gave an accuracy between 0.6 and 0.8.

The values of Recall achieved after optimization were however significantly lower than the ones produced by default models. The Voting Ensemble model gave best results for recall of 0.85 with linear SVC giving a value of 0.81 whereas the best value recorded after optimization was of perceptron model with the value of 0.677 although Ridge Classifier 0.651 and SVC 0.652 follow closely. The main reason for this is the significant imbalance or skewness of the label class. In our data corpus more than half of the data is labelled neutral whereas the remaining are almost equally divided into positive and negative labelled records.

10. Conclusion

Extracting an individual's attitude or thoughts from text is not an easy task and therefore Sentiment Analysis (also called Opinion Extraction or Opinion Mining) brings a ton of challenges. This analysis does not only extract the overall sentiment of a paragraph or a document. But this analysis must be thorough and in-depth enough to extract sentiments or opinions on a very granular level. And then relate that sentiment to the aspect it corresponds to. The process can be more complicated on an advanced level when it goes beyond positive or negative attitude and may encounter difficult data types. We have conducted this study for performing sentiment analysis on Roman Urdu dataset (acquired from various resources) and also analyzed results yielded from nine classifiers. We targeted the datasets in Roman Urdu that were extracted from social media websites. The results demonstrated a fair percentage of success rate based on F1 score. In future, more accuracy can be achieved by increasing the size of the dataset, having stronger preprocessing algorithms and having data annotated by larger number of different human annotators.

References

[1] Yi, Jeonghee, et al., "Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques.," in Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. IEEE, 2003.

[2] Tan, Songbo, and Jin Zhang., "An empirical study of sentiment analysis for chinese documents.," Expert Systems with applications 34.4, pp. 2622--2629, 2008.

[3] Govindarajan, M., "Bagged Ensemble Classifiers for Sentiment Classification of Movie Reviews.," IJECS , pp. 3951-3961, 2014.

[4] Albared, Mohammed, et al., "PROBABILISTIC ARABIC PART OF SPEECH TAGGER WITH UNKNOWN WORDS HANDLING.," Journal of Theoretical & Applied Information Technology, 2016.

[5] Al-Moslmi, Tareq, et al., "Enhanced Malay sentiment analysis with an ensemble classification machine learning approach.," Journal of Engineering and Applied Sciences, pp. 5226-5232, 2017.

[6] Ba-Alwi et al., "Choosing the Optimal Segmentation Level for POS Tagging of the Quranic Arabic.," British Journal of Applied Science & Technology, p. 10, 2017.

[7] Amiri, Fatemeh et al., "Lexicon-based sentiment analysis for Persian Text.," in Proceedings of the International Conference Recent Advances in Natural Language Processing., 2015.

[8] Xianghua, Fu, et al., "Multi-aspect sentiment analysis for Chinese online social reviews based on topic modeling and HowNet lexicon.," Knowledge-Based Systems, pp. 186-195, 2013.

[9] Pang, Bo, et al., "Thumbs up?: sentiment classification using machine learning techniques.," in Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics,, 2002.

[10] ChandraKala, S., and C. Sindhu., "Opinion Mining and sentiment classification a survey.," ICTACT journal on soft computing 3.1, pp. 420-425, 2012.

[11] Ho, Tin Kam, et al., "Decision combination in multiple classifier systems," IEEE transactions on pattern analysis and machine intelligence 16.1 , pp. 66-75, 1994.

[12] Kittler, Josef. , "Combining classifiers: A theoretical framework.," Pattern analysis and Applications 1.1 , pp. 18-27, 1998.

[13] Larkey, Leah S., and W. Bruce Croft., "Combining classifiers in text categorization.," in Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1996.

[14] Anitha, N. et al., "Sentiment Classification Approaches.," International Journal of Innovation Engineering and Technology 3.1 , pp. 22-31, 2013.

[15] Saraswathi, K., and A. Tamilarasi., "A modified metaheuristic algorithm for opinion mining.," International Journal of Computer Applications 58.11, 2012.

[16] M.Govindarajan et al., "Bagged Ensemble Classifiers for Sentiment Classification of Movie Reviews," International Journal Of Engineering And Computer Science, 2014.

[17] Iqra Javed, Hammad Afzal, "Creation of bi-lingual Social Network Dataset using classifiers.," in International Workshop on Machine Learning and Data Mining in Pattern Recognition. , 2014.

[18] Daud, Misbah et al., "Roman Urdu opinion mining system (RUOMiS).," arXiv preprint arXiv:1501.01386 , 2015.

[19] Bilal, Muhammad, et al., "Sentiment classification of Roman-Urdu opinions using Naïve Bayesian, Decision Tree and KNN classification techniques.," Journal of King Saud University-Computer and Information Sciences , pp. 330--344, 2016.

[20] Hajmohammadi, Mohammad Sadegh, and Roliana Ibrahim, "A svm-based method for sentiment analysis in persian language.," in International Conference on Graphic and Image Processing (ICGIP 2012). , 2013.

[21] Ashari, Ahmad, et al., "Performance comparison between Naïve Bayes, decision tree and k-nearest neighbor in searching alternative design in an energy simulation tool.," International Journal of Advanced Computer Science and Applications (IJACSA), 2013.

[22] Jebaseel, A., and Dr E. Kirubakaran., "M-learning sentiment analysis with data mining techniques.," International Journal

of Computer Science and Telecommunications, pp. 45--48, 2012.
[23] Choi D, Kim J, et al., "A method for normalizing non-standard words in online social network services: A case study on twitter.," in Second International Conference Context-Aware Systems and Applications, ICCASA. , 2013.
[24] Chrupała, Grzegorz., "Normalizing tweets with edit scripts and recurrent neural embeddings.," in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)., 2014.
[25] Brocki, Łukas et al., "Multiple model text normalization for the polish language.," in International Symposium on Methodologies for Intelligent Systems., 2012.
[26] Irvine, Ann et al., "Processing informal, romanized Pakistani text messages," in Proceedings of the Second Workshop on Language in Social Media. Association for Computational Linguistic., 2012.
[27] Rafae, Abdul, et al., "An Unsupervised Method for Discovering Lexical Variations in Roman Urdu Informal Text.," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. , 2015.
[28] Pedregosa, Fabian, et al. , "Scikit-learn: Machine learning in Python.," Journal of machine learning research , pp. 2825-2830, 2011.
[29] Henrich, Verena, and Erhard Hinrichs, "Standardizing wordnets in the ISO standard LMF: Wordnet-LMF for GermaNet.," in Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics,, 2010.