Sentiment Analysis of Roman Urdu/Hindi using supervised methods

Huniya Arif[1], Kinza Munir[1], Abdul Subbooh Danyal[1], Ahmad Salman[1] and Muhammad Moazam Fraz[1]

[1]School of Electrical Engineering and Computer Science
National University of Sciences and Technology (NUST)
Islamabad, Pakistan

***Abstract*— The usage of online platforms to receive feedback, opinion or remarks of the public about a particular subject has become very common. Sentiment analysis is used to understand the latest trends, summarize the general opinion and investigate the cognitive human behavior. This paper performs sentiment analysis on Roman Urdu dataset and reports on the experimental results produced by different classifiers using feature selection and representation. After translating a pre-existing English hotel reviews dataset to Roman Urdu, the resulting corpus is analyzed by employing machine learning approach for classification purposes. The result of social analytics can assist organizations in applying the proposed methodology to the collective sentiment intelligence embedded in customers' feedback in order to improve their product, services, and marketing strategies.**

Keywords—Sentiment Analysis, Machine Learning Techniques, Support Vector Machine, Ensemble Classifiers, Naive Bayes, k-Nearest Neighbor, Nearest Centroid, Decision tree, Linear Classifiers

I. Introduction

The use of the Internet around the world has created vast amounts of user-generated data. The rise in social media has changed the way people view, create and share information. Sites like Twitter and Facebook have become interactive platforms where users openly share their opinions. For organizations and consumers, these platforms are a great way of knowing the opinions, feedback, and mindset of a large number of people. It has become increasingly difficult to process this steady stream of content. Sentiment Analysis (also known as opinion mining) is a tool used to extract, analyze and summarize textual information and discover the cognitive behavior of humans. It determines whether the given sentence is positive or negative in the context of human emotional behavior. It opens up opportunities for marketing teams, celebrities, politicians, and anyone concerned with opinions and moods of the general public.

There have been namely two methods to find the sentiment polarity of textual data: corpus-based and lexicon-based. Corpus-based sentiment analysis trains a machine learning classifier on a labeled sentiment corpus. The performance of the classifier depends on the quantity and quality of the training data. Lexicon-based sentiment analysis finds the polarity of every word or phrase in a text document with the use of a sentiment lexicon. The two ways to perform sentiment analysis using lexicons is by either using a dictionary or a corpus. The dictionary-based approach uses an existing dictionary that contains a collection of words with their sentiment strength. A lexicon-based approach using corpus searches through vast amounts of data to find the probability of a term occurring in conjunction with the positive or negative set of words.

In order to extract market intelligence embedded in online comments, sentiment analysis needs to cater for a wider set of languages. The Urdu language is spoken and understood in major parts of Asia. In both script and morphology, it is a different language from English. Whereas Roman Urdu is the Urdu language written with the Roman alphabet. It is a universal, non-standard mode of communication among Urdu-speaking Internet users. This language is also used for poetry, marketing, blogs, online news, commercials, music and films in South Asia and the Middle East [1], [2].

Roman Urdu is a non-standard language used frequently on the Internet. It has many irregularities in spellings. For example, the word *khubsurat* (beautiful) in Roman Urdu has multiple spellings. It can also be written as *khoobsurat*, *khubsoorat*, and *khobsorat*. These words have the same meaning, but the spelling varies according to the user. Another problem arises when two words in Roman Urdu are spelled identically but are lexically different, for example, *aam* means both mango and common. In order to ensure quality in our training corpus, the spelling of a word needs to remain consistent throughout its use.

Research on sentiment analysis has been mostly conducted on a small number of languages. There is a scarcity of annotated corpora in other languages. Facebook and Twitter APIs are useful in extracting public comments posted on social media but the manual labeling of data is a tedious and time-consuming task. A solution to this problem is semi-supervised learning. Semi-supervised self-training tries to automatically label examples from unlabeled data and add them to the initial training set in each cycle. Alternatively, an existing labeled corpus can be translated to the target language for the generation of a dataset.

Support vector machine classifier (SVM), Ensemble Classifiers, K-Nearest Neighbors (kNN), Nearest Centroid, Linear Classifiers, Decision Tree and Naive Bayes (NB) Classifier are used to investigate the effectiveness of sentiment analysis on Roman Urdu with different term weighting and selection schemes. In order to facilitate research in Roman Urdu, the annotated corpus used for training and classification of text in this paper is available for download [3].

The paper is organized as follows: Section II gives a brief overview of related work in sentiment analysis, Section III illustrates the methodology followed in the paper, Section IV discusses the performance measures used in evaluating the classifiers, Section V uses figures and tables to describe the experimental results and Section VI concludes the paper along with a description of future works.

II. Related work

A. *Sentiment Analysis using machine learning*

In recent years, the field of sentiment analysis has encouraged a lot of research albeit in limited languages. Abinash et al. [4] have done sentiment classification by machine learning methods. The vectorization of features is done using two methods CountVectorizer and tf-idf. They have used NB and SVM classification algorithm to determine the polarity of English reviews and achieved an accuracy of 89.5% and 94.0% respectively. Another machine learning approach by Melody Moh et al. [5] makes use of a multi-tier classification architecture consisting of data collection, preprocessing, feature selection, classification and evaluation measures. The machine learning algorithms, SGD, SVM, NB and Random forest classifier, perform with accuracies above 80%. Geetika Gautam et al. [6] preprocessed the data collected from twitter, extracted sentiment words and used machine learning to classify the data. Semantic analysis is achieved with English WordNet that extracts the adjective from the sentence. It performs the highest accuracy of 89.9%, followed by NB with 88.2% accuracy, SVM with 85.5% accuracy and Maximum Entropy has the lowest classification prediction with 83.8%. For sentiment classification Vala Ali Rohani et al. [7] propose SentiRobo, a supervised machine learning approach and enhanced version of NB algorithm. P. Waila et al. [8] compare unsupervised semantic orientation based SO-PMI-IR algorithm with supervised classifiers NB and SVM to classify movie reviews. The SO-PMI-IR algorithm obtains the highest accuracy of 88% but it is time-consuming due to large computation of Pointwise Mutual Information (PMI) values. Further comparison by V.K Singh et al. [9] involves SentiWordNet and proves that NB performs better than SVM in classification. SentiWordNet has the minimum computation time as it needs no training but has the lowest performance with accuracy level around 65%.

Warih Maharani [10] compares sentiment analysis performance by the two approaches, lexical based and machine learning based, with Indonesian language using Twitter. Machine learning approach produces better accuracy with SVM i.e. 81.43%, as compared to lexical based approach having 74.59% accuracy without stemming. Neethu et al. [11] created their own feature vector by giving positive keywords the label ’1’ and negative as ’-1’. Emoticons and hashtags inside a tweet are also labeled. If the keyword doesn’t exist, then it is given as ’0’. The vector is used in the classification of the tweets with the help of SVM, NB, Maximum Entropy and Ensemble Classifiers. The accuracy is close to 90.0% for all the algorithms. Bin Lu et al [12] combined large sentiment lexicon and machine learning techniques and used this approach for subjectivity classification. Human-annotated lexicons and the annotated corpus were both used. This approach combined with SVM outperforms with an accuracy of 74-75% as compared to the machine learning techniques themselves alone. Haruna Isah et al. [13] compares lexicon and machine learning scores when analyzing sentiment in product reviews on the Web. Both methods give similar negative scores but their positive and neutral scores vary sharply. The NB classifier gave a total of 83% accuracy. Deepu S. Nair et al. [14] use SVM and Conditional Random Field (CRF) to check the polarity of Malayalam film reviews. After the collection of data, the corpus was manually tagged with respect to the context of the term in the sentence. SVM performs better than CRF and accuracy of 91% is achieved. A hybrid approach to sentiment analysis of news comments by Addlight Mukwazvure et al. [15] involve using sentiment lexicon for polarity detection and using its results to train machine learning algorithms, SVM, and kNN. Yu Huangfu et al. [16] introduce Improved Sentiment Analysis (ISA) which analyzes Chinese news with an algorithm of subjective sentences recognition and subject word recognition. General Sentiment Analysis (GSA) by Prashant Raina [17] is compared with ISA and ISA proves to be faster and more accurate. Serrano-Guerrero et al. [18] reviews and compares fifteen free web services that provide sentiment analysis by their ability to classify text as positive, negative or neutral, and measuring the intensity of each detected sentiment. Amandeep Kaur et al. [19] conduct a hybrid research approach consisting of N-grams model and NB for sentiment analysis of the Punjabi text. NB classifier trains on the features extracted from the N-grams model. When the results were compared with existing methods, the accuracy of hybrid approach appeared to be effective.

B. *Sentiment Analysis in the Urdu language*

The work on sentiment analysis is still in initial stages for the Urdu language, especially in the Roman version. In Urdu Script, Afraz Z. Syed et al. [20] used sentiment-annotated lexicon-based approach and identified and extracted SentiUnits (expressions containing sentiment information) from Urdu texts with the help of shallow parsing. They produced an accuracy of 72.0% on movies reviews and 78.0% on product reviews. Tafseer Ahmed [21] proposes a method to transliterate Urdu in the roman script to Urdu script. He identifies the different spelling rules used in roman Urdu and encodes each word according to an encoding scheme to match it with a word list containing frequently used terms in the language. Abdul et al. [22] discuss the lexical variations found in roman Urdu language written on the Internet. They devise a phonetic algorithm UrduPhone motivated from Soundex to map Urdu strings to their phonetic equivalents.

Iqra et al. [23] performed sentiment classification of bilingual (Roman Urdu and English) tweets related to Pakistan's general elections 2013. They filtered the relevant tweets and assigned sentiment strength to 3900 Roman Urdu words in order to create their own bilingual repository. Roman Urdu opinion mining system (RUOMiS) proposed by Daud et al. [24] used the key matching method to perform Roman Urdu opinion mining. Adjectives of the opinions were matched with a dictionary manually designed and were used to find polarity of that opinion. Due to the noise in data, the accuracy was low as RUOMiS categorized about 21.1% of the neutral comments falsely. Muhammad Bilal et al. [25] conducted opinion mining on Roman Urdu using three classification algorithms which are NB, Decision Tree, and kNN. NB performs the best in Roman Urdu in terms of accuracy, precision, recall, and F-measure.

Our paper contributes to the research in sentiment classification of Roman Urdu text by using supervised machine learning algorithms. The scarcity of labeled data in the desired language is solved with a series of translation tools. The choice of feature selection and extraction methods depends on the characteristics of the given dataset. The effectiveness of a set of classifiers is observed when they predict the polarity of the unseen samples after learning the training data.

III. Method

The labeled Roman Urdu documents are preprocessed to remove any irrelevant information. The important features are extracted and the textual data is converted to a numerical format. The stop words are removed and the features having the highest chi-squared test scores are selected. The training and testing datasets are formed from the original corpus. They are converted into a sparse matrix where the number of rows is the number of samples and the columns represent selected features. The matrix is an argument of the machine learning algorithm and is used in training and classifying the data. A diagrammatic view of the method is shown in fig. 1.

A. Material

The dataset used in our work contains 1600 total documents with an equal number of positive and negative reviews. The work on sentiment analysis has been very limited in Urdu and Roman Urdu languages. Roman Urdu is frequently written on blogging websites, chat rooms and advertisements and by users in the comment sections. The online text can be scraped and processed but requires annotation to train the classifier.

In order to create a large and labeled sentiment corpus, we performed the following steps. First, we downloaded a publicly-accessible Hotel Reviews labeled data [26] in English. We used Google Translate API to convert our data from English to Urdu Script. The Google Translate API allows translation for more than 50 languages. It performs better for short sentences and faces limitations in differentiating between imperfect and perfect tenses. For longer sentences, the structure of the resulting output becomes convoluted; therefore, the overall meaning is difficult to understand. We used an online Urdu Script to Roman Urdu Transliteration tool [27] to convert our labeled corpus in Urdu Script to Roman Urdu. We used a single tool in transliteration, to ensure minimum variation in the spellings of the words in Roman Urdu language. As a result, we have 800 documents each of positive and negative reviews available in Roman Urdu language [3].

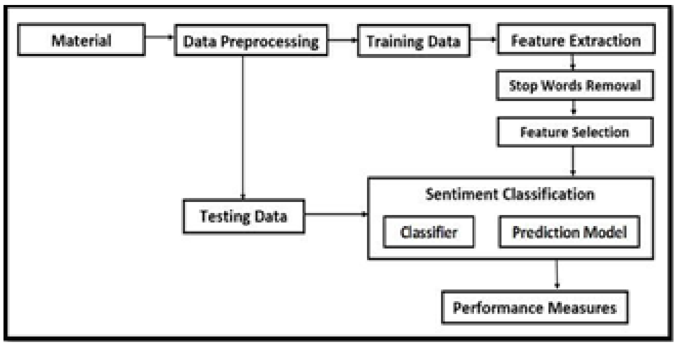

Fig. 1. Schematic view of the proposed methodology

B. Data Preprocessing

The transliteration tool is not able to convert all the Urdu words to Roman script. Hence, the documents have some unconverted Urdu texts in between the sentences. A preprocessing step before feature extraction removes all the numbers and special characters from the documents. This step is to ensure that only the relevant features are extracted and selected from the dataset.

C. Feature Extraction

An essential preprocessing step to machine learning problems is feature extraction. It involves building feature vectors from textual data to facilitate learning. A feature vector is an n-dimensional vector representing the dataset with attributes that can be binary ('male' or 'female' for gender), categorical ('A', 'B', 'AB' or 'O', for blood type), ordinal ('high', 'low' or 'medium') or numerical (for example, the number of occurrences of a word in a document).

An algebraic model for representing text documents as vectors of identifiers is called the vector space model. Two features are of main concern when we are indexing a term: statistical term weighting and semantic term weighting. In statistical term weighting, term weighting is based on the discriminative supremacy of a term that appears in a document or a group of documents. In semantic term weighting, term weighting is based on a term's meaning. Term Frequency-Inverse Document Frequency (tf-idf) as well as normalized tf-idf is considered as the most effective document weighting functions for information retrieval and text categorization tasks [28].

The vectorization of the corpus to a sparse matrix is performed by three methods:

1) Term Frequency-Inverse Document Frequency (tf-idf)

Tf-idf is a statistical measure of the importance of a term to a document in the corpus. The importance of the term is increased with its frequency in the document but it is reduced when there is an increase in the frequency of the term in the corpus. It is composed of two terms: term frequency (TF) and inverse document frequency (IDF) [29].

$$\mathbf{t\,f_{t,d} \times idf_t} \tag{1}$$

TF measures the frequency of a term within a document. For normalization, the frequency is divided by the length of the document. Normalization helps in preventing bias towards the longer document as longer document have a higher frequency of a term. Term frequency gives equal importance to every term in the document.

$$\boldsymbol{T\,F_{t,d}} = \frac{\textbf{frequency of a term } \boldsymbol{t}}{\textbf{length of the document}} \tag{2}$$

The terms which are not of much importance, such as ’as’, ’a’, ’the’, ’of’, usually have a higher frequency. Hence, such frequent terms are required to be weighed down. While the important terms, with a lower frequency, needs to be scaled up. IDF is used to increase the importance of the relevant terms that are less frequently used.

$$\boldsymbol{IDF_t} = \mathbf{log}\left(\frac{\textbf{number of documents}}{\textbf{number of documents with term } \boldsymbol{t}}\right) \tag{3}$$

2) CountVectorizer

CountVectorizer transforms the document content into the count matrix by tokenizing and counting the presence of the feature [29]. It extracts the features of length greater than 2 by tokenizing the sentences. It counts the presence of each feature and creates a sparse matrix. For example, if we have the following four sentences in Roman Urdu:

”Kamra ganda tha.”
”Kamra chota tha.”
”Kamra saaf tha.”
”Kamra pyaara tha.”

The features extracted from the sentences are ”kamra”, ”ganda”, ”tha”, ”chota”, ”saaf” and ”pyaara”. They will be assigned a unique index corresponding to the column of the matrix. These 4 sentences and 6 unique features will be represented in a sparse matrix of size 4 X 6. The matrix will look like Table I, where 1 represents the feature’s presence in the document and 0 indicates its absence.

TABLE I. SPARSE MATRIX OF COUNTVECTORIZER

	Feature 1	**Feature 2**	**Feature 3**	**Feature 4**	**Feature 5**	**Feature 6**
Sentence 1	1	1	1	0	0	0
Sentence 2	1	0	1	1	0	0
Sentence 3	1	0	1	0	1	0
Sentence 4	1	0	1	0	0	1

3) HashingVectorizer

HashingVectorizer combines hashing with the preprocessing and tokenization features of the CountVectorizer [29]. It is fast and stateless. The transformation performed by the HashingVectorizer is irreversible. Hence, the original representation of the features is impossible to access after the hashing is done. It is used in parallel pipelining and for performing vectorization of a vast text corpus. HashingVectorizer is flexible when an application is likely to receive new data. The larger the corpus is, the larger the vocabulary will grow. Hence, more memory is used. The fitting of data also requires memory for the intermediate data structures of size proportional to the original dataset. Collisions occur when two different tokens are mapped to the same feature index.

D. Stop words Removal

Non-semantic words like articles, prepositions, conjunctions and pronouns are usually described as stop words. They are removed from the features because they hold little or no information about the sentiment of the sentence [30], [31]. In Roman Urdu, stops words can be pronouns like ’mein’, ’tum’, ’hum’ etc. that can confuse the classifier. A list of Urdu stop words [32] is available on the Internet. By transliterating the stop words to Roman Urdu script, we can filter the resulting tokens from the feature extraction algorithms to make sure there are no stop words in the feature vector. The use of a stop word list reduces the features extracted from the training set from 1875 to 1770.

E. Feature Selection

Feature selection (also called as variable or attribute selection) reduces the dimensionality of the sample sets. It involves identifying and selecting relevant features from the data that contribute to the accuracy of the predictive model. It simplifies the predictive model and makes it easy to understand by eliminating the redundant and unneeded attributes. It boosts performance on very high-dimensional datasets by decreasing the training time, increases classification accuracy and reduces over-fitting. Feature extraction creates a set of features which are composites of the original feature set whereas feature selection forms a subset of the attributes occurring in the original training set.

The four approaches related to feature selection are Filter, Wrapper, Hybrid, and Embedded. Filter method uses mathematical techniques to evaluate and derive a subset from the entire set of features. It requires less computation and does not depend on learning methods used by classifiers [33]. Wrapper approach evaluates features that are optimal for classification with a predetermined learning algorithm. The hybrid method combines the techniques used in filter and wrapper approaches. It finds the best subset using an independent measure and a learning algorithm. Embedded method is a filter algorithm built with the classifiers and is more efficient as compared to Wrapper approach. Recursive Feature Elimination (RFE) is an example of Embedded method used in SVM classifier.

In this paper, Filter method is used for the selection of a good subset of the original features set. Methods used in Filter approach are the Chi-squared test, Document Frequency (DF), Information Gain (IG), Mutual Information (MI) and Term Strength (TS).

Chi-squared test is a statistical test used to compute the independence of two events, the occurrence of a term and occurrence of a document [34]. Features for which the chi-squared test gives the highest score are selected. A high score shows that the occurrence of the term and class are dependent. The features that have a low chi-squared score, i.e. they are independent of the class, are irrelevant for classification.

For aggressive term removal, Chi-squared test and IG are most effective and do not lose accuracy in text classification. Compared to other methods, MI has the most inferior performance due to its bias towards infrequent terms and sensitivity to probability estimation errors [35].

F. Sentiment Classification

The two types of sentiment classifications are binary classification and multi-class sentiment classification. In binary classification, a feature, sentence or document is classified according to the two polarities, positive or negative, favorable or unfavorable. On the other hand, multi-class sentiment classification can label the review to more than two classes, for example positive, neutral, negative. In this paper, machine learning algorithms are implemented on the selected features for binary classification.

Supervised machine learning techniques require a training set, containing feature vectors and their corresponding labels, and a testing set [36]. The classifiers learn a set of rules from the training corpus before they can be applied for testing. SVM, kNN, Ensemble Classifier, Decision Tree Classifier, Nearest Centroid, Perceptron, Passive Aggressive Classifier, Stochastic Gradient Classifier, Ridge Classifier and NB are used to learn and classify the Roman Urdu dataset.

IV. PERFORMANCE MEASURES

Confusion matrix visualizes the performance of a supervised learning algorithm. The matrix shows the number of data predicted correctly and incorrectly. In binary classification, any document identified as positive by the classifiers and also labeled as positive in the corpus is a true positive. Any document which is predicted as positive but is actually negative is a false positive.

TABLE II.CONFUSION MATRIX TABLE

	Actual Positive	**Actual Negative**
Predicted Positive	True Positive(TP)	False Positive(FP)
Predicted Negative	False Negative(FN)	True Negative(TN)

TABLE III. PERFORMANCE METRICS FOR SENTIMENT CLASSIFICATION

Performance Parameter	**Description**
Accuracy	$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
Sensitivity	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
Specificity	$\frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}}$
F1-score	$\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$

Accuracy, Sensitivity, Specificity, precision, and F1-score are the performance evaluation parameters calculated from the confusion matrix. Table III defines these metrics using the terms in Table II. Accuracy is the ratio of correctly predicted documents (sum of true positives and true negatives) by the total number of documents in the corpus. Sensitivity (also known as recall) shows the ability of the classifier to detect the positive documents. Specificity is the ability to detect negative documents. Sensitivity denotes true positive rate whereas specificity shows the true negative rate of the techniques. Precision gives the fraction of identified positive documents which are true positive documents. F1-score is the harmonic mean of precision and recall.

V. EXPERIMENTAL EVALUATION

The Roman Urdu labeled corpus has 1600 total reviews, containing an equal number of positive and negative documents. The dataset was split into training and testing sets in such a way that both had the same number of positive and negative reviews. For training, features were extracted from 1200 hotel reviews as an input to the classifier and the remaining 400 documents were allocated for determining the accuracy of the classifier.

After the preprocessing and vectorization of data, K features are selected for classification using the chi-squared test. K is also the number of columns in the sparse matrix. The performance of classifiers is analyzed by varying the value of K. Training time always reduces when K decreases, but the effect on accuracy is different for each classifier. Most of the classifiers show little change in their performance. However, kNN exhibits a different behavior from other classifiers as can be seen in the fig. 2. The value of K can change the accuracy of kNN classifier from 53.5% to 89.75%.

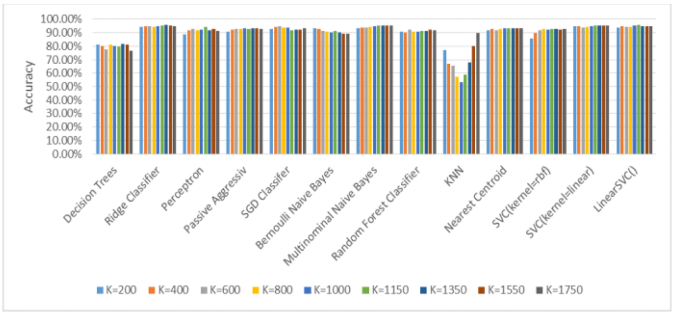

Fig. 2. Accuracy of classifiers with different values of K using tf-idf vectorizer

TABLE IV. CONFUSION MATRIX FOR CLASSIFIERS WITH TF-IDF VECTORIZER

Classifiers	Precision	Specificity	Accuracy	Sensitivity	F1 Score
Decision Trees	0.76	0.79	0.78	0.78	0.77
Ridge Classifier	0.98	0.93	0.95	0.93	0.95
Perceptron	0.96	0.93	0.95	0.93	0.95
Passive Aggressive	0.94	0.91	0.92	0.91	0.92
SGD classifier	0.94	0.90	0.92	0.90	0.92
Bernoulli NB	0.90	0.93	0.91	0.92	0.91
Multinomial NB	0.96	0.95	0.95	0.95	0.95
Random Forest	0.95	0.90	0.92	0.90	0.92
kNN	0.16	1.00	0.59	1.00	0.30
Nearest Centroid	0.98	0.89	0.93	0.90	0.94
SVC(kernel=rbf)	0.98	0.89	0.93	0.89	0.93
SVC(kernel=linear)	0.98	0.94	0.96	0.94	0.96
LinearSVC()	0.98	0.93	0.96	0.93	0.96

The number of selected features is 1150 when the behavior of classifiers with different vectorizers used for feature extraction is investigated. Table IV-VI shows the statistical measures of the performance of all the tested classifiers with tf-idf vectorizer, CountVectorizer, and HashingVectorizer. Most of the classifiers assign the test documents to correct classes since the values of sensitivity and specificity are quite high.

Accuracy, precision, specificity, recall and F1-score were calculated for all the classifiers using tf-idf, CountVectorizer, and HashingVectorizer. Fig. 3-6 shows the effect of different vectorizers on the classifiers. Most of the classifiers have little variation in accuracy for the vectorizers used. Nearest centroid, kNN and SVC with rbf kernel show the greatest variation, each performing best with different vectorizer. kNN performs best with HashingVectorizer at 81%, nearest centroid with tf-idf with 91% and SVC with tf-idf at 91%. kNN performs poorly with tf-idf vectorizer as opposed to other classifiers. Tf-idf vectorization gives the best overall accuracy, precision, recall and F1-score.

TABLE V. CONFUSION MATRIX FOR CLASSIFIERS WITH COUNTVECTORIZER

Classifiers	Precision	Specificity	Accuracy	Sensitivity	F1 Score
Decision Trees	0.79	0.80	0.80	0.80	0.79
Ridge Classifier	0.94	0.91	0.92	0.91	0.92
Perceptron	0.96	0.84	0.90	0.86	0.90
Passive Aggressive	0.93	0.91	0.92	0.91	0.92
SGD classifier	0.93	0.91	0.92	0.91	0.92
Bernoulli NB	0.88	0.91	0.89	0.90	0.89
Multinomial NB	0.97	0.93	0.95	0.93	0.95
Random Forest	0.96	0.87	0.91	0.88	0.92
kNN	0.63	0.95	0.79	0.92	0.75
Nearest Centroid	0.74	0.84	0.79	0.82	0.78
SVC(kernel=rbf)	0.90	0.92	0.91	0.91	0.91
SVC(kernel=linear)	0.93	0.88	0.90	0.89	0.90
LinearSVC()	0.93	0.90	0.92	0.90	0.92

TABLE VI. CONFUSION MATRIX FOR CLASSIFIERS WITH HASHINGVECTORIZER

Classifiers	Precision	Specificity	Accuracy	Sensitivity	F1 Score
Decision Trees	0.85	0.70	0.77	0.74	0.79
Ridge Classifier	0.98	0.90	0.94	0.91	0.94
Perceptron	0.95	0.88	0.91	0.88	0.91
Passive Aggressive	0.96	0.88	0.92	0.88	0.92
SGD classifier	0.95	0.93	0.94	0.93	0.94
Bernoulli NB	0.86	0.86	0.86	0.86	0.86
Multinomial NB	0.98	0.89	0.93	0.90	0.94
Random Forest	0.93	0.90	0.91	0.90	0.91
kNN	0.89	0.75	0.82	0.78	0.83
Nearest Centroid	0.91	0.82	0.86	0.83	0.87
SVC(kernel=rbf)	0.99	0.50	0.75	0.66	0.80
SVC(kernel=linear)	0.98	0.90	0.94	0.90	0.94
LinearSVC()	0.97	0.91	0.94	0.91	0.94

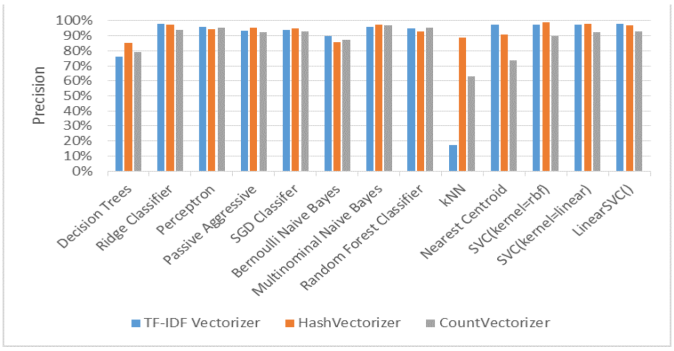

Fig. 3. Comparison of precision for tf-idf, CountVectorizer, and HashingVectorizer

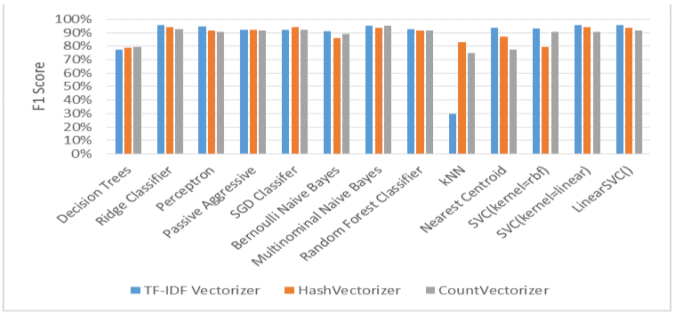

Fig. 4. Comparison of F1-score for tf-idf, CountVectorizer, and HashingVectorizer

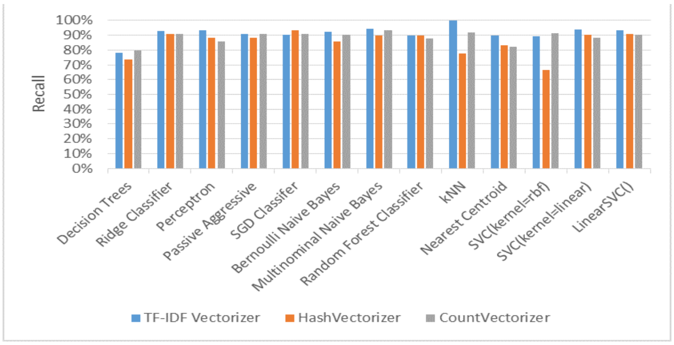

Fig. 5. Comparison of recall for tf-idf, CountVectorizer, and HashingVectorizer

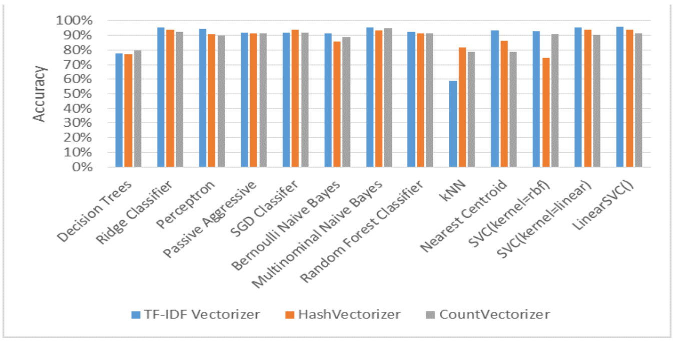

Fig. 6. Comparison of accuracy for tf-idf, CountVectorizer, and HashingVectorizer

Decision tree and kNN classifiers have the lowest performance out of all the other classifiers. The accuracy, precision, recall and F1-score of the decision tree in case of each vectorization method is approximately 78% which is less than the other classifiers. kNN performs poorly with an accuracy of 59% for tf-idf vectorizer but gives the best results with HashingVectorizer at 81%. kNN depends on the value of n, which is the number of neighbors of the test data used in the classification of the test point. The number of neighbors is selected as an odd number between 1 and 20 to study the effect on the performance of the kNN classifier when using tf-idf vectorizer. The accuracy of the kNN classifier is maximum when the value of k is 3.

Ridge classifier, multinomial NB, SVC with linear kernel and linear SVC have the overall best performance at the accuracy of around 96% with tf-idf vectorizer as compared to other classifiers.

VI. CONCLUSION

Social media has become a platform for the public to share their feelings and opinions about a product, brand or service. Sentiment analysis is a measurement tool that extracts the sentiment value of a sentence, paragraph or document. The value of the sentiment can be either positive, negative or neutral. The application of social analytics is founded in areas like social media monitoring, tracking survey responses and customer reviews, political sentiment determination and predicting market movements based on news, blogs and feedback.

Roman Urdu is a non-standard form of writing the Urdu language on the Internet. Each user has their own way of forming the Roman Urdu words, hence the spellings of the terms are not consistent. In this paper, we identified the possible issues in performing sentiment analysis on the Roman Urdu language and the measures needed to increase the accuracy of the classifiers.

The overall performance of the machine learning algorithms depends on the dataset used in training. An annotated corpus is not available in Roman Urdu, therefore a labeled English dataset of hotel reviews is translated to the Urdu language and then converted to Roman script. A single transliteration tool is used to make sure there are no irregularities in spelling. The longer sentences lose their structure during translation and some Urdu words cannot be converted by the tool to Roman script. The resulting corpus goes through preprocessing, vectorization, elimination of stop words and irrelevant attributes from its feature set before it is divided into training and testing data.

Tf-idf is the term weighting model which gives best overall accuracy with machine learning based classifiers. The number of selected features in the sparse matrix affects the training time and accuracy of the classifiers. SVM performs better than all of the classifiers by giving 96% accuracy at 1150 features selected from the original set initially extracted by tf-idf vectorization. The accuracy of KNN has the most significant change as the dimensionality of the matrix is reduced.

Machine learning techniques are simpler and often give better results than the lexicon-based approach. The scope and usability of this application can be increased by improving the quality and size of the dataset involved in classification of the hotel reviews. The work can be further extended by considering the context in which some terms are used to get a better idea about the sentiments of the review.

REFERENCES

[1] Shashca. (2012). Shashca: Youth ka newspaper, [Online]. Available: http://www.shashca.com.

[2] SongLyrics. (2016). Song lyrics, [Online]. Available: http://www.songlyrics.com/a-r-rahman/jai-ho-lyrics/.

[3] R. U. H. Dataset. (2016). Roman Urdu hotel dataset, [Online]. Available: http://vision.seecs.edu.pk/sentimentanalysis/.

[4] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentimental reviews using machine learning techniques," Procedia Computer Science, vol. 57, pp. 821– 829, 2015, ISSN: 18770509. DOI: 10.1016/j.procs.2015.07.523.

[5] M. Moh, A. Gajjala, S. C. R. Gangireddy, and T.-S. Moh, "On multi-tier sentiment analysis using supervised machine learning," 2015.

[6] G. Gautam and D. yadav, "Sentiment analysis of twitter data using machine learning approaches and semantic analysis," 2014.

[7] V. A. Rohani and S. Shayaa, "Utilizing machine learning in sentiment analysis: Sentirobo approach," 2015.

[8] P. Waila, Marisha, V. K. Singh, and M. K. Singh, "Evaluating machine learning and unsupervised semantic orientation approaches for sentiment analysis of textual reviews," 2012.

[9] V. K. Singh, R. Piryani, A. Uddin, P. Waila, and Marisha, "Sentiment analysis of textual reviews," 2013.

[10] W. Maharani, "Microblogging sentiment analysis with lexical based and machine learning approaches," 2013.

[11] N. M. S and R. R, "Sentiment analysis in twitter using machine learning techniques," 2013.

[12] B. LU and B. K. TSOU, "Combining a large sentiment lexicon and machine learning for subjectivity classification," 2010.

[13] H. Isah, P. Trundle, and D. Neagu, "Social media analysis for product safety using text mining and sentiment analysis," 2014.

[14] D. S. Nair, J. P. Jayan, R. R.R, and E. Sherly, "Sentiment analysis of malayalam film review using machine learning techniques," 2015.

[15] A. Mukwazvure and K. Supreethi, "A hybrid approach to sentiment analysis of news comments," 2015.

[16] Y. Huangfu, G. Wu, Y. Su, J. Li, P. Sun, and J. Hu, "An improved sentiment analysis algorithm for chinese news," 2015.

[17] P. Raina, "Sentiment analysis in news articles using sentic computing," 2013.

[18] J. Serrano-Guerrero, J. A. Olivas, F. P. Romero, and E. H. Viedma, "Sentiment analysis: A review and comparative analysis of web services," 2015.

[19] A. Kaur and V. Gupta, "N-gram based approach for opinion mining of punjabi text," 2014.

[20] A. Z. Syed, M. Aslam, and A. M. Martinez-Enriquez, "Lexicon based sentiment analysis of Urdu text using sentiunits," 2010.

[21] T. Ahmed, "Roman to urdu transliteration using word list," 2009.

[22] A. Rafae, A. Qayyum, M. Moeenuddin, A. Karim, H. Sajjad, and F. Kamiran, "An unsupervised method for discovering lexical variations in roman urdu informal text," 2015.

[23] I. Javed and H. Afzal, "Opinion analysis of bi-lingual event data from social networks," 2013.

[24] M. Daud, R. Khan, Mohibullah, and A. Daud, "Roman urdu opinion mining system," 2014.

[25] M. Bilal, H. Israr, M. Shahid, and A. Khan, "Sentiment classification of roman-urdu opinions using naive bayesian, decision tree and knn classification techniques," 2015.

[26] C. H. Dataset. (2016). Chinese hotel dataset, [Online]. Available: http://myleott.com/op_spam/.

[27] Ijunoon. (2016). Ijunoon transliteration, [Online]. Available: http://www.ijunoon.com/transliteration/urdu- to-roman.

[28] M. Abdel Fattah and F. Ren, "Ga, mr, ffnn, pnn and gmm based models for automatic text summarization," 2012.

[29] G. Moncecchi and R. Garreta, Learning Scikit-Learn: Machine Learning in Python. Packt Publishing Ltd., 2013.

[30] W. J. Wilbur and K. Sirotkin, "The automatic identification of stop words," Journal of information science, vol. 18, no. 1, pp. 45–55, 1992.

[31] C. Fox, "A stop list for general text," in ACM SIGIR Forum, ACM, vol. 24, 1989, pp. 19–21.

[32] R. NL. (2016). Ranks nl webmaster tools, [Online]. Available: http://www.ranks.nl/stopwords/urdu.

[33] M. A. Hall and L. A. Smith, "Feature selection for machine learning: Comparing a correlation-based fil-ter approach to the wrapper.," in FLAIRS conference, vol. 1999, 1999, pp. 235–239.

[34] M. G. Akritasa, "Pearson-type goodness-of-fit tests: The univariate case," 1988.

[35] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," 1997.

[36] B. Pang and L. Lee, "Thumbs up? sentiment classification using machine learning techniques," 2002.