Neural Computing and Applications (2021) 33:5437–5469
https://doi.org/10.1007/s00521-020-05321-8

ORIGINAL ARTICLE

Benchmarking performance of machine and deep learning-based methodologies for Urdu text document classification

Muhammad Nabeel Asim[1,2,4] · Muhammad Usman Ghani[3,4] · Muhammad Ali Ibrahim[2,4] · Waqar Mahmood[4] · Andreas Dengel[1,2] · Sheraz Ahmed[1]

Received: 24 February 2020 / Accepted: 2 September 2020 / Published online: 24 September 2020

Abstract

In order to provide benchmark performance for Urdu text document classification, the contribution of this paper is manifold. First, it provides a publicly available benchmark dataset manually tagged against 6 classes. Second, it investigates the performance impact of traditional machine learning-based Urdu text document classification methodologies by embedding 10 filter-based feature selection algorithms which have been widely used for other languages. Third, for the very first time, it assesses the performance of various deep learning-based methodologies for Urdu text document classification. In this regard, for experimentation, we adapt 10 deep learning classification methodologies which have produced best performance figures for English text classification. Fourth, it also investigates the performance impact of transfer learning by utilizing Bidirectional Encoder Representations from Transformers approach for Urdu language. Fifth, it evaluates the integrity of a hybrid approach which combines traditional machine learning-based feature engineering and deep learning-based automated feature engineering. Experimental results show that feature selection approach named as normalized difference measure along with support vector machine outshines state-of-the-art performance on two closed source benchmark datasets CLE Urdu Digest 1000k, and CLE Urdu Digest 1Million with a significant margin of 32% and 13%, respectively. Across all three datasets, normalized difference measure outperforms other filter-based feature selection algorithms as it significantly uplifts the performance of all adopted machine learning, deep learning, and hybrid approaches. The source code and presented dataset are available at Github repository https://github.com/minixain/Urdu-Text-Classification.

Keywords Urdu text document classification · Urdu news classification · Urdu news genre categorization · Multi-class Urdu text categorization computational methodologies · Deep neural networks · BERT

✉ Muhammad Nabeel Asim
Muhammad_Nabeel.Asim@dfki.de

Muhammad Usman Ghani
usman.ghani@uet.edu.pk

Muhammad Ali Ibrahim
ali.ibrahim@kics.edu.pk

Waqar Mahmood
director@kics.edu.pk

Andreas Dengel
Andreas.Dengel@dfki.de

Sheraz Ahmed
Sheraz.Ahmed@dfki.de

[1] German Research Center for Artificial Intelligence (DFKI) GmbH, 67663 Kaiserslautern, Germany

[2] Technische Universität Kaiserslautern, 67663 Kaiserslautern, Germany

[3] Department of Computer Science, University of Engineering and Technology (UET), Lahore, Pakistan

[4] Intelligent Criminology Research Lab, National Center of Artificial Intelligence, Al-Khawarizmi Institute of Computer Science, UET, Lahore, Pakistan

1 Introduction

Textual resources of diverse domains such as academia and industries are growing enormously over the web due to the rapid growth of technology [1, 2]. According to a recent survey of data facts, users have only utilized 0.5% of all electronic textual data [3]. The amount of electronic textual data which has been created in last two years is way more than the data created by entire human race previously [3]. This marks the desperate need of classifying or categorizing such humongous electronic textual data in order to enable the processing of text at large scale and for the extraction of useful insights. With the emergence of computational methodologies for text classification, multifarious applications have been developed such as email spam detection [4], gender identification [5], product review analysis [6], news categorization [7–9] and fake news detection [10–12] for various languages like English, Arabic, and Chinese. However, despite crossing the landmark of 100 million speakers [13], Urdu language is still lacking in the development of such applications. The primary reason behind this limited progress is the lack of publicly available datasets for Urdu language. Urdu text document classification datasets used in the previous works are private [14–19] which further restricts the research and fair comparison of new methodologies. In order to overcome this limitation, the paper in hand provides a new publicly available dataset in which news documents are manually tagged against six different classes.

On the other hand, regarding the improvement in performance of traditional machine learning-based text document classification methodologies, feature selection has played a significant role in various languages such as English, Arabic, and Chinese [20, 21]. The ultimate aim of feature selection is to eliminate irrelevant and redundant features [22]. Feature selection alleviates the burden on classifier which leads to faster training [23, 24]. It also assists the classifier to draw better decision boundary which eventually results in accurate predictions [23, 24]. State-of-the-art machine learning-based Urdu text document classification methodologies lack discriminative feature selection techniques [19]. In this paper, we embed ten most anticipated filter-based feature ranking metrics in traditional machine learning pipeline to extrapolate the impact created by the set of selected top k features over the performance of support vector machine (SVM) [25] and Naive Bayes (NB) [26] classifiers.

Although feature selection techniques reduce the dimensionality of textual data up to great extent, traditional machine learning-based text document classification methodologies still face the sparsity problem in bag-of-words-based feature representation techniques [27, 28]. Bag-of-words-based feature representation techniques consider unigrams, n-grams or specific patterns as features [27, 28]. These algorithms do not capture the complete contextual information of data and also face the problem of data sparsity [27, 28]. These problems are solved by word embeddings which do not only capture syntactic but semantic information of textual data as well [29]. Deep learning-based text document classification methodologies provide end-to-end system for text classification by automating the process of feature engineering and are outperforming state-of-the-art machine learning-based classification approaches [30–35].

Although there exists some work on the development of pre-trained neural word embeddings (Haider et. al [36], and FastText[1]) for Urdu language, no researcher has utilized any deep learning-based methodology or pre-trained neural word embeddings for Urdu text document classification. Here, we thoroughly investigate the performance impact of 10 state-of-the-art deep learning methodologies using pre-trained neural word embeddings. Among all, 4 methodologies are based on a convolutional neural network (CNN), 3 on a recurrent neural network (RNN), and 3 of them are based on a hybrid approach (CNN+RNN). Pre-trained neural word embeddings are just shallow representations as they fuse learned knowledge only in the very first layer of deep learning model, whereas rest of the layers still require to be trained using randomly initialized weights of various filters [37]. Moreover, although pre-trained neural word embeddings manage to capture semantic information of words but fail to acquire high-level information including long-range dependencies, anaphora, negation, and agreement for different domains [37–39]. Considering the recent trend of utilizing pre-trained language models to overcome the downfalls of pre-trained neural word embeddings [40, 41], we also explore the impact of language modelling for the task of Urdu text document classification.

However, due to the lack of extensive research, finding an optimal way to acquire maximal results on diverse natural language processing tasks through the use of Bidirectional Encoder Representations from Transformers (BERT) [42] is not straightforward at all [43–45]. For instance, whether pre-training BERT [42] on domain-specific data will produce good results, or fine-tuning BERT [42] for target tasks or multitask learning would be an optimal option [43–45]. In this paper, we thoroughly investigate multifarious ways to fine-tune pre-trained multilingual BERT [42] language models and provide key insights to make the best use of BERT [42] for Urdu text document classification.

[1] https://fasttext.cc/docs/en/crawl-vectors.html.

Previously, we proposed a robust machine and deep learning-based hybrid approach [46] for English text document classification. The proposed hybrid methodology reaped the benefits of both machine learning-based feature engineering and automated engineering performed by deep learning models which eventually helped the model to better classify text documents into predefined classes [46]. Hybrid approach significantly improved the performance of text document classification on two publicly available benchmark English datasets 20-Newsgroup[2], and BBC[3] [46]. This paper investigates whether utilization of both machine and deep learning-based feature engineering is versatile and effective enough to replicate promising performance figures with a variety of deep learning models for Urdu text document classification. Extensive experimentation with all machine and deep learning-based methodologies is performed on two closed source datasets, namely CLE Urdu Digest 1000*k*, CLE Urdu Digest 1Million, and one newly developed dataset, namely DSL Urdu news.

Among all machine learning-based methodologies, Naive Bayes [26] with Normalized Difference Measure [47] marks the highest performance of 94% over newly developed DSL Urdu news dataset, whereas SVM [25] proves dominant over both close source datasets CLE Urdu Digest 1000*k*, CLE Urdu Digest 1Million by marking the performance of 92% with normalized difference measure [47], and 83% with Chi-squared (CHISQ) [48]. On the other hand, trivial adopted deep learning-based methodologies manage to outshine state-of-the-art performance by the margin of 6% on CLE Urdu Digest 1000*k*, and 1% on CLE Urdu Digest 1Million. Contrarily, hybrid methodology which leverages machine and deep learning-based feature engineering [46], and BERT [42] mark similar performance across all three datasets. These methodologies outperform state-of-the-art performance with the figure of 18% on CLE Urdu Digest 1000*k*, 10% on CLE Urdu Digest 1*M* datasets, and almost equalize the promising performance figures of machine learning-based methodology over DSL Urdu news dataset.

The remaining paper is distributed into following sections. First section deep dives into contributions along with their anticipated impacts, followed by previous work solely related to Urdu Text Document Classification followed by a detail explanation of text document classification methodologies used in this paper. Then, all datasets are elaborated comprehensively. Afterwards, experimental setup and results are revealed in subsequent sections. Finally, we summarize the key points and give future directions.

[2] http://archive.ics.uci.edu/ml/datasets/twenty+newsgroups.

[3] http://mlg.ucd.ie/datasets/bbc.html.

2 Contributions: a review in a nutshell with anticipated impacts

Researchers have employed variety of ways to improve the classification performance for multifarious textual data ranging from trivial documents to convoluted genomic sequences [49]. To name a few, while some researchers have tried different data re-sampling approaches and machine learning classifiers (e.g. generative, probabilistic, and ensemble classifiers (bagging, boosting)) [50], others have employed more effective feature engineering approaches (feature representation and selection) and less biased evaluation matrices [21, 51, 52]. On the other hand, with the revolutionary success brought by deep learning-based classification approaches, in recent times, some researchers have utilized variety of neural word embeddings, activation and loss functions, precisely deep (MLP) and reasonably deep standalone (CNN, RNN) or hybrid neural networks (CNN+RNN) and model parameters optimization approaches to achieve optimal classification performance [53]. Contrarily, other researchers have achieved promising performance figures through transfer learning [54] (using pre-trained language models or training language model from scratch in an un-supervised manner and then fine-tuning over target task).

However, a comprehensive review of variety of machine and deep learning-based classification approaches with every minor detail of model parameters is quite scarce especially for under-resourced languages like Nastaleeq Urdu. In addition, despite the fact that different filter-based feature selection approaches have significantly raised the performance of machine learning classifier, no work has been performed to assess the impact of variety of filter-based feature selection approaches for deep learning models.

Building on above discussion, the current study attempts to perform a comprehensive comparative analysis of machine and deep learning-based classification approaches using variety of feature representation and feature selection approaches. Contributions of this work are summarized as below:

1. Development of a publicly available dataset that contains 662 documents of six different classes (health-science, sports, business, agriculture, world, and entertainment) containing 130 K words for Urdu text document classification.
2. For machine learning-based text document classification, optimal combination of feature selection approach and classification algorithm is found through rigorous experimentation with 10 filter-based feature selection algorithms such as balanced accuracy measure (ACC2) [55], normalized difference measure (NDM) [47],

max–min ratio (MMR) [21], relative discrimination criterion (RDC) [56], information gain (IG) [57], Chi-squared (CHISQ) [48], odds ratio (OR) [58], bi-normal separation (BNS) [55], Gini index (GINI) [59], Poisson's ratio (POISON) [60] and predefined benchmark test points. This will save a significant amount of time and effort of application developers.

3. For deep learning-based text document classification, impact of 10 filter-based feature selection algorithms is assessed over 10 state-of-the-art standalone and hybrid neural networks to find most optimal filter-based feature selection algorithm for deep learning models. This will largely assist deep learning researchers in the selection of most discriminative features from very high-dimensional feature vectors.
4. Considering the lack of research to optimize most widely used pre-trained multilingual language model Bidirectional Encoder Representations from Transformers (BERT [42]) for acquiring better performance over target tasks, performance of BERT is assessed with base vocabulary and using the vocabulary generated by top filter-based feature selection algorithm. Also, key steps to better fine-tune BERT for text classification are also provided. This facilitates how effective is the vocabulary generated by top filter-based feature selection algorithm for a language model and which parameters with what values play more crucial role in raising the classification performance.
5. The effectiveness of a previously proposed hybrid approach [46] which reaps the benefits of traditional feature engineering and deep learning-based automated feature engineering is thoroughly investigated for Urdu text document classification using variety of classifiers and Urdu datasets. This alleviates the sole reliance of researchers on automated feature engineering performed by deep learning models and will encourage them to employ different approaches to further improve the feature engineering of deep learning models.

3 Related work

Text document classification methodologies can be categorized into rule-based and statistical approaches. Rule-based approaches utilize manually written linguistic rules, whereas, statistical approaches learn the association among multifarious features and class labels in order to classify text documents into predefined classes [61]. This section briefly illustrates state-of-the-art statistical work on Urdu text document classification.

Ali et al. [14] compared the performance of two classifiers, namely Naïve Bayes (NB) [26], and Support Vector Machine (SVM) [25] for the task of Urdu text document classification. They prepared a dataset by scrapping various Urdu news websites and manually classified them into six categories (news, sports, finance, culture, consumer information and personal information). Based on their experimental results, they summarized that SVM [25] significantly outperformed Naive Bayes [26]. Their experiments also revealed that stemming decreased the overall performance of classification.

Usman et al. [15] utilized maximum voting approach in quest of classifying Urdu news documents. The news corpus was divided into seven categories, namely business, entertainment, culture, health, sports, and weird. After tokenization, stop words removal, and stemming, they extracted 93400 terms and fed them to six machine learning classifiers, namely Naïve Bayes (NB), linear stochastic gradient descent (SGD) [62], multinomial Naïve Bayes (MNB), Bernoulli Naïve Bayes (BNB) [63], linear SVM [25], and random forest classifier [64]. Then, they applied max voting approach in such a way that the class selected by majority of the classification algorithms was chosen as final class. Experimentally, they proved that linear SVM [25] and linear SGD [62] showed better performance on their developed corpora.

Sattar et al. [17] performed Urdu editorials classification using Naïve Bayes classifier. Moreover, most frequent terms of the corpus were removed to alleviate the dimensionality of data. Their experimental results showed that Naïve Bayes classifier performs well when it is fed with frequent terms as compared to feeding all unique terms of the corpus.

Ahmed et al. [16] performed Urdu news headlines classification using support vector machine (SVM) [25] classifier. They utilized a TF-IDF-based feature selection approach which removed less important domain-specific terms from underlay corpus. This was done by utilizing the threshold paradigm on TF-IDF score which enabled the extraction of those terms that had higher TFIDF than defined threshold value. After preprocessing and threshold-based term filtration, they used SVM [25] classifier to make predictions.

Zia et al. [18] evaluated the performance of Urdu text document classification by adopting four state-of-the-art feature selection techniques, namely information gain (IG) [57], Chi-square (CS) [48], gain ratio (GR) [65], and symmetrical uncertainty [66] with four classification algorithms (K-nearest neighbours (KNN) [67], Naïve Bayes (NB), decision tree (DT), and support vector machines (SVM) [25]. They found that for larger datasets, performance of SVM [25] with any of the above-mentioned feature selection technique was better as compared to

Naïve Bayes [26] which was more inclined towards small corpora.

Adeeba et al. [19] presented an automatic Urdu text-based genre identification system that classified Urdu text documents into one of the eight predefined categories, namely culture, science, religion, press, health, sports, letters, and interviews. They investigated the effects of employing both lexical and structural features on the performance of support vector machine [25], Naïve Bayes [26], and decision tree algorithms. For lexical features, the authors extracted word unigrams, bigrams, along with their term frequency and inverse document frequency. To extract structural features, part of speech tags and word sense information were utilized. Moreover, they reduced the dimensionality of corpora by eliminating low-frequency terms. For the experimentation, CLE Urdu Digest 100K[4] and CLE Urdu Digest 1 Million[5] corpora were used. Their experiments revealed that SVM [25] was better than other classifiers irrespective of feature types.

State-of-the-art work on Urdu text document classification is summarized in Table 1 by author name, benchmark dataset, exploited feature representation and selection techniques, classifiers, evaluation metrics, and their respective performances.

After thoroughly examining the literature, it can be summarized that SVM [25] and Naive Bayes [26] perform better than other classifiers for the task of Urdu text document classification.

For English text document classification, recent experimentation on public benchmark datasets also proves that performance of SVM [25] and Naive Bayes [26] significantly improves with the use of filter-based feature selection algorithms [21, 47]. Filter-based feature selection algorithms not only improve the performance of machine learning-based methodologies, but it has also substantially raised the performance of deep learning-based text document classification approaches [46].

However, Urdu text document classification methodologies are lacking to produce promising performance due to the lack of research in this direction as only Ahmed et al. [16], and Zia et al. [18] utilized some feature selection approaches in order to reduce the dimensionality of data. While Ahmed et al [16] only experimented with TF-IDF-based feature selection approach, Zia et al. [18] assessed the integrity of just four feature selection algorithms (information gain (IG) [57], Chi-Square (CS), gain ratio (GR), and symmetrical uncertainty) in domain of Urdu text document classification. However, the performance impact of more recent filter-based feature selection algorithms has never been explored specifically for Urdu text document classification.

In addition, despite the promising performance produced by deep learning methodologies for diverse NLP tasks [68, 69], no researcher has utilized any deep learning-based methodology for the task of Urdu text document classification.

4 Adopted methodologies for Urdu text document classification

This section comprehensively illustrates machine learning, deep learning, and hybrid methodologies which we have used for the task of Urdu text document classification.

4.1 Traditional machine learning-based Urdu text document classification with filter-based feature selection algorithm

This section elaborates the machine learning-based Urdu text document classification methodology. Primarily, our main focus is to investigate the performance boost in traditional machine learning-based Urdu text document classification methodologies through the embedding of filter-based feature selection algorithms. Figure 1 provides graphical illustration of machine learning-based Urdu text classification methodology which utilizes filter-based feature engineering. All phases of this methodology are discussed below.

4.2 Preprocessing

Preprocessing of text is considered as preliminary step in almost all natural language processing tasks as better tokenization, and stemming or lemmatization eventually leads to better performance in various machine learning tasks such as text classification [70, 71], information retrieval [72], and text summarization [73].

Stemming undoubtedly plays an important role to alleviate sparsity problems through dimensionality reduction; however, there are very few rule-based stemmers available for Urdu language which lack to showcase quality performance. Ali et al. [14] claimed that stemming degrades the performance of Urdu text document classification. We analysed that the stemmer utilized by Ali et al [14] was of poor quality which eventually caused the decline in performance as it has been proved by many researchers that stemming often improves the performance of text document classification for various languages (e.g. English) [74, 75]. Urdu language lacks better stemming algorithms; therefore, instead of stemming, we perform lemmatization through a manually prepared Urdu lexicon containing 9743

4 http://www.cle.org.pk/clestore/urdudigestcorpus100k.htm.

5 http://www.cle.org.pk/clestore/urdudigestcorpus1M.htm.

Table 1 State-of-the-art work on Urdu text document classification

Authors	Datasets	Feature representation techniques	Feature selection techniques	Classifier	Evaluation metric
Ali et al. [14]	Manually classified news corpus	Normalized term frequency	–	NB , SVM	Accuracy
Usman et al. [15]	News Corpus	Term Frequency (TF)	–	NB, BNB, LSVM, LSGB, RF	Precision, Recall, F1-score
Sattar et al. [17]	Urdu News Editorials	Term Frequency (TF)	–	NB	Precision, Recall, F1-score
Ahmed et al. [16]	Urdu News Headlines	TF-IDF	TF-IDF (Thresholding)	SVM	Accuracy
Zia et al. [18]	EMILLE, Self Collected Naive corpus (News)	TF-IDF	Information Gain, Chi Square, Gain Ratio, Symmetrical Uncertainty	KNN, DT, NB.	F1-score
Adeeba et al. [19]	CLE Urdu Digest (1000*K*, 1 Million)	Term Frequency (TF), TF-IDF	Pruning	NB, SVM (linear, radial, polynomial)	Precision, Recall, F1-score

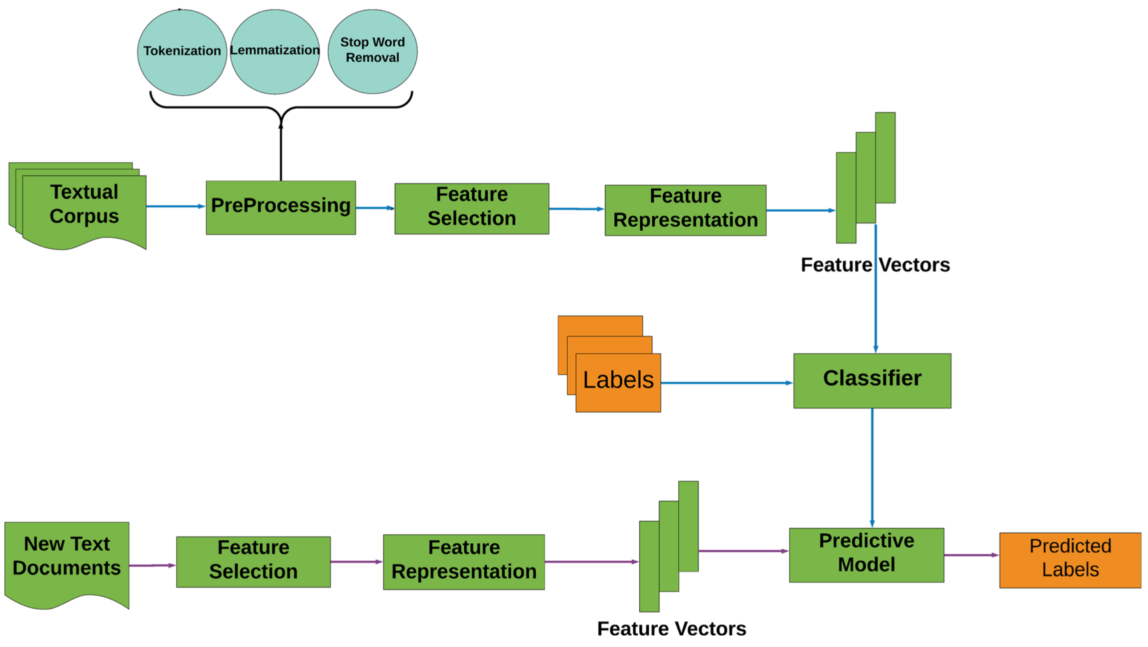


Fig. 1 Machine learning-based Urdu text document classification methodology

possible values of 4162 base terms. In Sect. 6, Tables 3, 4, 5 reveal the impact of lemmatization on the size reduction of three datasets used in our experimentation. We believe public access to the developed lexicon will enable the researchers to perform lemmatization in several different Urdu processing tasks. In addition, all non significant words of corpus are eliminated through a stop words list. The list of 1000 stop words is formed by manually analyzing the most frequent 1500 words of underlay corpora.

4.3 Feature selection

Feature selection is being widely used to reduce the dimensionality of feature space in different applications like text classification [47], plagiarism detection [76], and for query expansion in pseudo-relevance feedback-based information retrieval [77], which eventually assists to produce better results.

Feature selection approaches can be categorized into three classes wrapper [78], embedded [79], and filter [80]. In wrapper methods, classifier is trained and tested over several subsets of features and only one subset of features is selected which has produced the minimum error [78]. Similarly, embedded feature selection approaches also work like wrapper-based methods; however, wrapper-based methods can exploit one classifier (e.g. SVM [25]) to train over subset of features and other classifier (e.g. Naive Bayes) to test optimal set of features, but embedded feature selection approaches are bound to use the same classifier throughout the classification process [79].

On the other hand, filter-based feature selection algorithms do not take into account the error value of a classifier; however, they rank the features and pick top *k* features based on certain threshold [81]. In this way, a highly discriminative user specified subset of features is acquired by utilizing the statistics of data samples.

Wrapper and embedded feature selection methods are computationally far more expensive as compared to filter-based feature selection algorithms. While both former approaches assess the usefulness of features by cross-validating classifier performance, latter approaches operates over the intrinsic properties (e.g. relevance) of features computed through univariate statistics.

In our work, considering the efficiency of filter-based feature selection algorithm, we have adapted ten most anticipated filter-based feature selection algorithms. These algorithms are extensively being utilized for English text document classification such as balanced accuracy measure (ACC2) [55], normalized difference measure (NDM) [47], max–min ratio (MMR) [21], relative discrimination criterion (RDC) [56], information gain (IG) [57], Chi-squared (CHISQ) [48], odds ratio (OR) [58], bi-normal separation (BNS) [55], Gini index (GINI) [59], Poisson's ratio (POISON [60]).

Filter-based feature ranking algorithms utilize confusion matrix (shown in Table 2) to compute the scores of corpus features.

In confusion matrix, positive and negative classes are two predefined classes in a typical binary text document classification problem, whereas in a multi-class text document classification problem, iteratively, one class is considered positive and rest are combined to form a negative class. t_j and $\bar{t_j}$ represent the presence and absence of terms, respectively, in corresponding classes.

Table 2 Confusion matrix, where t_p refers to the number of documents in positive class having term t (true positives), f_p refers to the number of documents in negative class having term t (false positives), t_n implies the number of documents in negative class not having term t (true negatives), and f_n implies the number of documents in positive class not having term t (false negatives)

	t_j	$\bar{t_j}$
Positive class	t_p	f_n
Negative class	f_p	t_n

Here, we only refer these feature selection algorithms, and interested readers can explore these algorithms deeply by studying their respective papers.

4.3.1 Balanced accuracy measure (ACC2)

Accuracy measure (ACC) is the predecessor of the balanced accuracy measure (ACC2) [47]. ACC is evaluated as a difference between true positives and false positives of a feature. It is the most simplest filter-based feature ranking algorithm as it computes the difference between (t_p) total number of positive class documents having feature f and (f_p) total number of negative class documents having feature f. As in case of multi-class machine learning problem, ACC is biased towards true positives; therefore, it performs well only on balanced data.

$$\text{AccuracyMeasure} = ACC = t_p - f_p \tag{1}$$

To overcome the t_p biasedness, ACC2 was proposed. It is an absolute difference between true positive rate $\left(t_{pr}\right)$ and false positive rate$\left(f_{pr}\right)$.

$$\text{Balanced Accuracy Measure} = ACC2 = |t_{pr} - f_{pr}| \tag{2}$$

In Eq. 2, values of t_{pr} and f_{pr} are defined in Eqs. 3 and 4.

$$t_{\text{pr}} = \frac{t_p}{t_p + f_n} \tag{3}$$

$$f_{\text{pr}} = \frac{t_n}{t_n + f_p} \tag{4}$$

4.3.2 Normalized difference measure (NDM)

ACC2 treats all terms alike which have the same $|t_{pr} - f_{pr}|$ value even if the t_{pr} and f_{pr} values of terms are different from each other. According to Rehman et al. [47], the terms located at the bottom right and top left corners of the contour plot are more important than the ones located around the diagonals. Although ACC2 assigns higher values to the terms located at bottom right and top left corners of contour plot, it treats the terms alike which are located around the diagonal. In order to overcome this problem, normalized difference measure (NDM) treats the terms at corners and at diagonals differently.

According to NDM, a term is important if:

- it has high $|t_{pr} - f_{pr}|$ value.
- Either t_{pr} or f_{pr} is closer to zero.

- If any two terms have same $|t_{pr} - f_{pr}|$ values, then a higher rank must be assigned to that term which has smaller (t_{pr}, f_{pr}) value.

The mathematical representation of NDM is as follows:

$$\mathrm{NDM} = \frac{|t_{\mathrm{pr}} - f_{\mathrm{pr}}|}{min(t_{\mathrm{pr}}, f_{\mathrm{pr}})} \tag{5}$$

4.3.3 Max–Min ratio (MMR)

Max-min ratio is an improved version of ACC2 and NDM as it addresses the downfalls of both feature ranking algorithms. NDM assigns pretty high scores to all sparse terms ($t_{\mathrm{pr}} \approx 0$, $\mathrm{f}_{\mathrm{pr}} \approx 0$ or t_{pr}, $\mathrm{f}_{\mathrm{pr}} \approx 0$) and denominator factor $(\min(t_{\mathrm{pr}}, f_{\mathrm{pr}}))$ obliterate numerator $(|t_{pr} - f_{pr}|)$ in case of largely skewed data. However, MMR is highly capable of estimating true term relevance especially for those datasets in which predefined classes are highly skew in nature. The mathematical representation of MMR is as follows:

$$\mathrm{MMR} = \max(tpr, fpr) * \frac{|t_{\mathrm{pr}} - f_{\mathrm{pr}}|}{\min(t_{\mathrm{pr}}, f_{\mathrm{pr}})} \tag{6}$$

It is clear from the equation that the factor max(tpr, fpr) stops the NDM scores from getting too large. It significantly helps especially for those terms having tpr, fpr approximately equal to 0. Likewise, MMR and NDM assign same score when the max of tpr, and fpr are exactly 1. In case of having tpr exactly equal to fpr, MMR imitates as ACC2. Although MMR faces the problem of determining denominator factor $\min(\mathrm{t}_{\mathrm{pr}}, \mathrm{f}_{\mathrm{pr}})$ for a particular set of terms which do not exist in one of the predefined classes, it is still considered to be an improved version because of its capability to capture true relevance of corpus terms especially for highly skew datasets.

4.3.4 Relative discrimination criterion (RDC)

Relative discrimination criterion (RDC) [56] computes document frequencies of entire corpus terms counts and calculates the difference between document frequencies of each term count by taking into account the presence of term in positives and negatives classes. Moreover, in order to tackle the problem of assigning exactly same score to terms which have different discriminative powers, it divides the computed difference by the minimum of two document frequencies. Therefore, term having least minimum document frequency in one of the predefined classes would eventually get a higher score. This is because there is a widely accepted criteria that a term which is frequent in only one specific class shall get a higher score. In addition, in order to assign higher weight to the differences having smaller term counts, the difference is also divided by term count, thus alleviating the bias for higher term counts. Mathematical expression of RDC can be written as:

$$\mathrm{RDC} = \frac{|tpr_{\mathrm{tc}} - fpr\mathrm{tc}|}{min(tpr_{\mathrm{tc}}, fpr_{\mathrm{tc}}) * (tc)} \tag{7}$$

4.3.5 Information gain (IG)

Information gain (IG) is widely used in text data. It is a measure of how likely a term is to occur in a particular class as compared to other classes. For instance, a word 'mesmerizing' is more likely to occur in a positive review and less likely to occur in a negative review. Because the presence of word 'mesmerizing' is a strong indication of positive emotion, therefore it can be classified as a highly informative word.

Information gain (IG) also calculates whether the information is increased or decreased after adding or removing a term from feature subset. Information gain for a term t can be calculated as:

$$IG_t = e(p, n)[P_w e(tp, fp) + P_w^- e(fn, tn)] \tag{8}$$

where p and n represent the number of positive and negative instances; further, $e(p, n)$ can be calculated as:

$$e(p, n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n} \tag{9}$$

and p_w, p_w^- can be calculated as:

$$P_w = \frac{(tp + fp)}{N} \tag{10}$$

$$P_w^- = 1 - P_{\mathrm{term}} \tag{11}$$

4.3.6 Chi-squared (CHISQ)

It is another widely used feature selection algorithm. In statistics, Chi-squared (CHI) is used to measure the dependency of two events, whereas in text document classification, it is used to check the dependency of a term to a class [55]. High scores of *CHISQ* demonstrate the high dependency between a term and a class.

Moreover, it is a two-sided metric because it takes only positive value in consideration and ignores the sign [82]. On the basis of discriminating power of positive and negative classes, two-sided metric assigns positive values to both classes. One of the downsides of this feature ranking metric is that it performs poorly when the dataset contains infrequent terms. It exaggerates such terms and pays no attention to term distribution. However, performance of the this algorithm can be improved by applying pruning with a certain threshold on the dataset. The score of CHISQ for *ith* term of *kth* class is given as:

$$\mathrm{CHI} = \frac{\left(t_p \times t_n - f_n \times f_p\right)^2}{\left(t_p + f_p\right)\left(f_n + t_n\right)\left(t_p + f_n\right)\left(f_p + t_n\right)} \tag{12}$$

4.3.7 Odds ratio (OR)

Odds ratio (OR) measures the odds of the presence of a feature in a positive class normalized by that of negative class. It is based on the idea that the distribution of feature in positive documents is not the same as distribution in the negative documents. It takes only those features into consideration that repeatedly occur in a particular class and totally ignores the features of same scope in the other classes [58].

Also, OR does not prioritize any redundant and irrelevant features. It is good in handling a smaller number of features. Its mathematical formulation is given as:

$$\mathrm{OR} = \frac{t_p \times t_n}{f_p \times f_n} \tag{13}$$

4.3.8 Bi-normal separation (BNS)

Bi-normal separation (BNS) which is first introduced by Forman [55] is defined as follows:

$$\mathrm{BNS} = |F_c^{-1}(t_{pr}) - F_c^{-1}(f_{pr})| \tag{14}$$

Here, F_c^{-1} is inverse cumulative distribution function of normal distribution. Highest weights are assigned to those features that are strongly connected with positive class or negative class. Lowest weights are assigned to those features that are evenly distributed among all the classes. BNS method is not biased towards document frequency and helpful for extracting useful features in highly skewed datasets.

4.3.9 Gini index (GINI)

Gini index (GINI) is used to measure the purity of an attribute. The purity of a feature can be used to calculate its importance. A feature is pure if all the documents show that the feature belongs to the same class. Therefore, GINI produces useful results when applied to features. Bigger value of GINI depicts better purity of a feature.

The score of Gini index can be calculated for a feature *t* using the following formula.

$$GI(t) = \sum_{j=1}^{M} P(t|C_j)^2 P(C_j|t)^2 \tag{15}$$

4.3.10 Poisson ratio (POISON)

Initially, Poisson's ratio (POISON) is only used to extract query words in information retrieval. Later, Ogura et al [83] modified POISON for feature selection. It measures the deviation of a feature from the Poisson distribution. If the feature is far away from the Poisson distribution, then it is more effective. Conversely, if a feature lies near to or within the range of distribution, then it is poor. Mathematically, POISON is defined as follows:

$$\mathrm{POIS} = \frac{(a_p - \hat{a_p})^2}{\hat{a_p}} + \frac{(b_{np} - \hat{b_{np}})^2}{\hat{b_{np}}} + \frac{(c_{fp} - \hat{c_{fp}})^2}{\hat{c_{fp}}} + \frac{(d_{tn} - \hat{d_{tn}})^2}{\hat{d_{tn}}} \tag{16}$$

$$\hat{a_p} = N(C)(1 - e^{(-\lambda)}), \hat{b_{np}} = N(C)e^{(-\lambda)}, \tag{17}$$

$$\hat{c_{fp}} = N(\bar{C})(1 - e^{(-\lambda)}) \tag{18}$$

$$\hat{d_{tn}} = N(\bar{C})e^{(-\lambda)} \tag{19}$$

$$\lambda = F/N \tag{20}$$

where a_p represents the presence and b_{np} refers to the absence of a term in a particular class. If a term is present but do not belong to class C, it is represented as c_{fp}, and d_{tn} refers that both t and C are absent from the documents.

4.4 Feature representation

Diverse domains (e.g. textual, non-textual) have different stacks of features; for example, if we want to classify iris data, then the set of useful features would be sepal length, sepal width, petal length and petal width [84]. However, the set of textual features for certain domain is not fixed at all. Representation of features plays a vital role to raise the performance of diverse classification methodologies [29, 68, 69]. Machine learning methodologies utilize bag of words-based feature representation approaches. Term frequency [85] is the simplest and widely used feature representation technique for various natural language processing tasks such as text classification and information retrieval [86–88]. Term frequency (TF) [85] of a term in a document is defined as the number of times a term occur in that document. One of the most significant problems of TF is that it does not capture the actual importance and usefulness of a term. This downfall is well addressed by term frequency-inverse document frequency (TF-IDF) [85] which is a modified version of term frequency [85] as it declines the weight specifically for the words which are commonly used and raises the weight for less commonly used words of underlay corpus. It gives more importance to less frequent terms and vice versa. It is calculated by taking dot product of term frequency (TF) and inverse document frequency (IDF).

IDF assigns weights to all the terms on corpus level. According to IDF, a term is more important if it occurs in less documents. When IDF weighting scheme is used standalone, it can allocate same weights to many terms which have the same DF_t score. IDF is defined as follows:

$$IDF_t = \log\frac{N}{DF_t} + 1 \tag{21}$$

where N is the total number of documents in the corpus and DF_t is the document frequency of term t.

A higher TF-IDF score implies that the term is rare and vice versa. Its value for term t in a document d can be calculated as

$$TF-IDF_{t,d} = TF_{t,d} \cdot IDF_t \tag{22}$$

Thus, by using both TF and IDF, TF-IDF captures the actual importance of terms on both document and corpus level.

4.5 Classifiers

In order to assess the impact of filter-based feature selection algorithms on the performance of trivial machine learning-based Urdu text document classification methodologies, we utilize support vector machine (SVM) [25], and Naive Bayes (NB) [26] classifiers. This is because, in state-of-the-art Urdu text document classification work, we have found that only these two classifiers mark promising performance [14–19].

Naive Bayes [26] uses Bayes' theorem and probability theory in order to make predictions. Naive Bayes [26] classifiers are usually categorized as *generative classifiers* and are highly useful for applications like document classification [89], and email spam detection [90], whereas SVM [91] classifier is categorized as *discriminative classifier* and mostly used for anomaly detection [92], and classification problems [93]. It is a non-probabilistic linear classifier which plots each data sample as a coordinate point in multi-dimensional space and finds an optimal hyperplane which eventually helps to differentiate the class boundaries effectively.

5 Deep learning methodologies

To better understand multifarious deep learning methodologies adapted for Urdu text document classification, learning of convolutional neural network(CNN), recurrent neural network(RNN), long short-term memory network(LSTM), and gated recurrent unit (GRU) is illustrated. Also, their differences are explained through mathematical expressions in following subsections.

5.1 Input layer

In our work, we have either utilized randomly initialized or 300 dimensional pre-trained neural word embeddings to create numeric representation of corpus words. In other words, given a document of n words $\mathrm{w}_1, \mathrm{w}_2, \mathrm{w}_3, \ldots, \mathrm{w}_\mathrm{n}$, for each corpus word w_i, embedding vector e_i is generated by computing matrix vector product using embedding matrix $W\epsilon R^{\mathrm{d}} * |V|$ where |V| represents size of vocabulary and d associates to the dimension of real valued word embedding vector.

$$e_\mathrm{i} = Wv_\mathrm{i} \tag{23}$$

In this way, each corpus document is represented in terms of several word vectors containing real values in between 0 and 1 $e = e_1, e_2, e_3, \ldots, e_n$. These features are then fed to variety of feature extraction layers.

5.2 Convolutional neural network (CNN)

Typically, convolutional neural network (CNN) is composed of convolution and pooling layers proceeded by one/multiple fully connected layers which at times is replaced by global average pooling layer. Moreover, researchers have also experimented with dropout and batch normalization approaches to improve the performance of CNN [94]. Depth and components of CNN play a pivotal role in enhancing the task performance. Various components with their respective roles within CNN are briefly discussed below.

5.2.1 Convolution layer

Convolutional layer has collection of kernels that act as feature extractors. In case of symmetrical kernel, convolution operation certainly turns into a correlation operation [95]. Each kernel $w \in R^{kk}$ is applied upon a window containing h words with certain stride size in order to generate fresh feature. For instance, a fresh feature c_i is produced from the window of words $x_{i:i+h-1}$

$$c_i = f(w.x_{i:i+h-1} + b) \tag{24}$$

In Eq. 24, $b \in R$ represents bias and f acts as a nonlinear function like hyperbolic tangent. This kernel is executed over every possible window of words present in a sentence $w_{1:h}, w_{2:h+1} \cdots w_{n-h+1:n}$ to generate a feature map that can be represented as:

$$c = [c_1, c_2, c_3 \ldots c_{n-h+1}] \tag{25}$$

where $c \in R^{n-h+1}$. Model actually makes use of multiple kernels with diverse window and stride size to get collection of features. Convolution operation can be classified into distinct different types considering the size and type of filters, padding type, and convolution direction [94].

5.2.2 Pooling layer

After extracting features from documents, capturing their relative positions become an important task which is achieved by down-sampling or pooling. Pooling is a local operation that captures the dominant response of local region by aggregating similar neighbourhood information [96]. Pooling operation can be expressed as:

$$Z_l^k = g_p(F_l^k) \tag{26}$$

Here, Z_l^k refers to down-sampled feature map of l^th layer for k^th give feature map (F_l^k), g_p represents the kind of pooling operation. Pooling mainly assists to acquire feature combinations that are invariant to small distortions and translational shifts [97, 98]. Minimization of feature map alleviates the complexity of neural network and also assists in raise the generalization through reducing over-fitting. Variety of pooling operations are applied like average, min, max overlapping, L2, spatial pyramid, etc. [94, 99].

5.2.3 Activation function

It is a decision function and helps the network for learning complex patterns. Appropriate selection of activation function enhances the process of learning. For a feature map acquired through convolution operation, activation function can be written as:

$$T_l^k = g_a(F_l^k) \tag{27}$$

Here, (F_l^k) is produced by convolution, that is given to activation function represented as g_a. Activation function embeds nonlinearity and yields output T_l^k for lth layer.

Critical analysis of literature shows that multifarious activation functions have been used like tanh, sigmoid, maxout, SWISH, ReLU, and its variants (LeakyReLu, PReLU, ELU) [99–103]. But, ReLU and variants of ReLU are mostly preferred by the researchers as they greatly assist in dealing with gradient vanishing issue [104, 105].

5.2.4 Batch normalization

In order to resolve the issues related to covariance shift inside feature maps, batch normalization is widely used. Covariance shift refers to the change in distribution of network hidden units/values that significantly decreases convergence rate (by pushing learning rate to lower value) and demands watchful initialization of model parameters. For transformed feature map, batch normalization is given as follows:

$$N_l^k = F_l^k - u_b / \sqrt{\sigma_b^2 + \epsilon} \tag{28}$$

Here, 28, for mini batch, N_l^k and F_l^k refer to normalized and input feature map, σ_b^2 and u_b correspond to variance and mean of a given feature map. To avoid zero division, ϵ is injected to add numerical stability.

Batch normalization standardizes distribution of values of feature map through setting them into unit variance and zero mean [106]. Also, it greatly flatten gradient flow and serve as a regularizing factor, through which network generalization is improved up to great extent.

5.2.5 Dropout

Dropout layer serves as a regularizer in neural network that eventually alleviates overfitting and improves generalization through randomly neglecting few connections or units with particular probability [107]. In neural networks, as several connections based on nonlinear relation are co-adapted at times, random dropping of few units creates multiple thinned deep architectures and afterward one optimal representative network architectures is opted with quite small weights. Then, this opted architecture is considered an approximation of all proposed networks [108].

5.2.6 Fully connected layer

The layer which is used at the end of neural network is called fully connected layer. Unlike convolution and pooling, it can be classified as a global extraction. It takes the input from all feature extraction phases and analyses the result of former layers [109]. As a result, it creates a nonlinear association of selected features that are used for text classification [110].

5.3 Recurrent neural network (RNN) and its variants (LSTM, GRU)

Researchers have extensively utilized recurrent neural network (RNN) for text classification [111, 112]. As RNN allocates more weights to former points and takes into account information of former nodes, it analyses the structure of dataset in a more effective manner. Mostly, RNN makes use of long short-term memory (LSTM) or gated recurrent unit (GRU) that consists of embedding layer, hidden layers, and output layer. This methodology can be expressed as:

$$x_t = f(x_{t-1}, u_t, \theta) \tag{29}$$

At time step t, here x_t represents state and u_t refers to the input. Using weights, it can be expressed as:

$$x_t = W_{rec}\sigma(x_{t-1} + W_{in}u_t + b) \tag{30}$$

Here, W_rec represents recurrent weight matrix w_in is input weights, b refers to bias and σ implies the element-wise operation.

RNN is highly vulnerable to exploding and vanishing gradient problems [113] when error of neural network is back propagated in the network. Due to these reasons, its variants LSTM and GRU are used mostly in experimentation, details of which are given below.

5.3.1 Long short-term memory (LSTM)

Long short-term memory (LSTM), a special type of RNN given by Hochreiter et al. [114], addressed the downfalls of RNN such as vanishing gradient issue through preserving long-range dependencies in a very effective manner [114]. Although due to having chain-like architecture, it is quite similar to RNN, LSTM makes use of several gates in order to efficiently regulate information which is allowed for the state of each node.

$$i_l = \sigma(W_i[x_t, h_{t-1}] + b_i) \tag{31}$$

$$C_t = \tanh(W_c[x_t, h_{t-1}] + b_c) \tag{32}$$

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f) \tag{33}$$

$$C_t = i_t * C_t^{f_t C_{t-1}} \tag{34}$$

$$o_t = \sigma(W_o[x_t, h_t - 1] + b_o) \tag{35}$$

$$h_t = o_t(tanh(C_t)) \tag{36}$$

Here, Eq. 31 refers to input gate, Eq. 32 refers to the value of candid memory cell, Eq. 33 represents forget gate activation, Eq. 34 computes value of fresh memory cell, and Eqs. 35, 36 describe the final yield of gate value. Moreover, b refers to bias, w represents the weight matrix, x_t represents the input at timestamp t, and indies i, c, f, o refer to input, memory of cell, forget, and final output gates in turn.

5.3.2 Gated recurrent unit (GRU)

Gated recurrent unit (GRU) is a more simplified version of LSTM architecture [115]. But unlike LSTM, it has two gates and does not have internal memory. In addition, it does not apply second nonlinearity [116].

$$z_t = \sigma_g(W_z * x_t + U_z * h_{t-1} + b_z) \tag{37}$$

Here, z_t is the update gate representation of t, x_t is input vector, and W, b, U are parameter vectors. Activation function is either ReLu or sigmoid that can be formulated as:

$$r_t = \sigma_g(W_r * x_t + U_r * h_{t-1} + b_r) \tag{38}$$

where z_t is the update gate representation of t and r_t is reset gate representation of t.

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \sigma_h(W_h * x_t + U_h * (r_t \cdot h_t - 1) + b_h) \tag{39}$$

For t, h_t is the final output vector where σ_h represents hyperbolic tangent operation.

5.4 Selection and optimization of model parameters

To achieve optimal performance in multifarious natural language processing (NLP), tasks like classification, selection, and tuning of hyperparameters are quite crucial in deep learning approaches. Inappropriate selection or tuning does not only badly hit the generalization of neural networks which eventually leads to significant decline in performance, but it may also cause endless training and ineffective consumption of valuable resources. Due to humongous number of hyperparameters, selecting most crucial ones based on time and resource complexity and performance impact is not a straightforward task at all. In addition, optimization of hyperparameters is considered as black box research of x, in a way that for a defined function $f : S \subset R^d \Rightarrow R$, $f(x)$ values is quite small and function f is stochastic by nature. This infers the scenario where one is searching for best setting of hyperparameters for certain model by trying multiple values of such parameters and opting the value that yields best performance on validation data.

Existing hyperparameter search approaches can be classified into pattern search [117], Gaussian processes [118, 119], evolution strategies [120], random sampling [121], and grid sampling [121]. Building on the critical findings related to most crucial hyperparameters acquired by Yin et al. [122] after performing a thorough comparative analysis of deep neural networks for diverse NLP tasks, in our work, we have also only selected most influential hyperparameters. More specifically, we tweak hidden size, batch size, optimizer, learning rate, momentum, loss criterion, activation function, dropout, number of kernels, and their sizes. To find optimal values of selected parameters, instead of functional or manual evaluation that proves extremely expensive, we have employed most widely used hyperparameter optimization approach, namely grid search. In grid search, a set of hyperparameter are opted beforehand (random manner or on the grid) and model training is performed in parallel. Grid search is highly scalable and quite easy to execute. In our experimentation, we have tried different batch sizes [10, 20, 30, 40, 50, 60, 70, 80, 90, 100], optimizers [SGD', RMSprop, Adadelta, Adagrad, Adam, Nadam, Adamax], learning rate [0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5], momentum [0.0, 0.2, 0.4, 0.6, 0.8, 0.9] dropout rate [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9] hidden units [1, 5, 10, 15, 20, 25, 30], number of kernels [2^3–2^8], kernel sizes [25, 50, 75, 100, 125, 150, 200], epochs [10, 20,30, 40, 50], and loss criterion [categorical cross entropy, binary cross entropy]. For pre-trained language model BERT-based transfer learning, we find multilingual cased model containing 12 heads, 12 layers, 110 parameters, and 768 hidden units and pre-trained over corpora of 104 languages to be the most optimal among all available BERT variants. In terms of parameters, we experiment with sequence length ranging from 2^4–2^9, batch size 2^3–2^9, learning rate [1e-1, 1e-2, 1e-3, 1e-4 or 1e-5, 2e-1, 2e-2, 2e-3, 2e-4 or 2e-5], buffer size [100, 200, 300, 400, 500], and epochs [10, 20, 30, 40, 50]. On the other hand, for machine learning-based Urdu text classification , we have wrapped support vector machine (SVM) into one against rest classification paradigm with linear and rbf kernels and balanced class weights. In order to find optimize gamma and cost values, we utilize grid search to compute the accuracy of every parameter combination (log2g range(3,− 15), step=− 2, whereas log2c range(− 5, 15), step=2>> log). Contrarily, Naive Bayes is used with default parameters.

In our work, SVM marks better performance with linear kernel, whereas all deep learning models mark optimal performance with root mean square propagation (RMSprop) optimizer, learning rate of 0.001, categorical cross-entropy loss criterion, and batch size of 50 when executed for 20 epochs. Using BERT, we achieve optimal performance with buffer size of 400, sequence length of 512, batch size of 16, and learning rate of 1e-5 by training the model up to 50 epochs. Experimentation with a variety of deep learning model indicates that changes in batch size, hidden size significantly influence model performance. To sum up, for diverse deep learning models, we consider batch size and hidden size are really crucial parameters, tweaking of which leads to optimal or sub-optimal performance.

5.5 Adopted deep learning methodologies for Urdu text document classification

This section summarizes state-of-the-art deep learning-based methodologies adapted for the task of Urdu text document classification. In order to provide a bird's eye view on adopted deep learning methodologies, generalized architecture is drawn in Fig. 2.

We adapt a multi-channel CNN model presented by Yoon Kim [123] for the task of sentiment, question, and sentence classification. In order to reap the benefits of distinct pre-trained word vectors, for the very first time, they made few channels dynamic, and others static throughout training in order to prevent overfitting. Several researchers (e.g. Nabeel et al. [46]) utilized this model for English text document classification and achieved state-of-the-art performance. In our experimentation, we have fed FastText neural word embeddings at one channel and pre-trained neural word embeddings provided by Haider et al. [36] at the second channel. At third channel, we have used randomly initialized word embeddings. In order to avoid overfitting, we keep the FastText embeddings static, and fine-tuned other embeddings during training.

Embedding layer of this model is followed by 3 convolution layers with 128 filters of size 3, 4, and 5, respectively. After that, extracted features of all convolution layers are concatenated and fed to another convolution layer having 128 filters of size 5. After applying max-pooling of size 100, the extracted features are then passed to a flatten layer which flattens the features. These flattened features are then passed to a dense layer with 128 output units which are followed by a dropout layer of rate 0.5. Finally, a last dense layer acts as a classifier.

Another CNN-based approach adapted for Urdu text document classification was presented by Nal Kalchbrenner et al. [124]. A distinct aspect of this model was the use of wide convolutions. The authors claimed that the words at edges of a document do not actively participate in convolution and get neglected especially when the filter

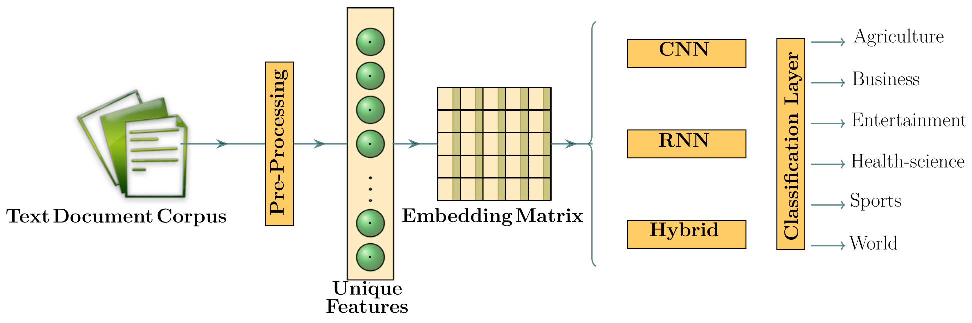


Fig. 2 Generalized methodology of adopted deep learning models

size is large. An important term can occur anywhere in the document so by using wide convolution every term take equal part while convolving. Although, originally, authors did not use any pre-trained word embeddings in the proposed CNN architecture, we have utilized pre-trained word embeddings.

This model begins with an embedding layer, followed by convolution layer with 64 filters of size 50. Top five features are extracted from the convolution layer by using a K-max-pooling layer of value 5. Zero padding is utilized to maintain the wide convolution structure. After that, there is another convolution layer with 64 filters of size 25. This layer is followed by a K-max-pooling layer of value 5. Finally, the extracted features are flattened and passed to a dense layer which classifies the documents.

Yin et al. [125] proposed a CNN model for the task of binary or multi-class sentiment analysis, and subjectivity-based question classification for the English language. The significance of the multi-channel input layer was deeply explored by the author by using five different pre-trained word vectors. This model has outperformed eighteen baseline machine and deep learning methodologies [125] for sentiment and question classification tasks. While adopting this model, we have utilized two embedding layers, two convolution layers along with wide convolutions.

The model starts with two embedding layers, and each embedding layer is followed by two wide convolution layers with 128 filters of size 3 and 5, respectively. Each convolution layer is followed by a K-max-pooling layer of size 30. After that, both convolution layers are followed by two other convolution layers of the same architecture except the value of *k* which is 4 in K-max-pooling layers. All the features from all convolution layers are then concatenated and flattened by using a flatten layer. These flattened features are then passed to two dense layers from which the first dense layer has 128 output units and the last dense layer acts as a classifier.

Just like Yin et al. [125] CNN-based approach, Zhang et al. [126] also proposed a CNN-based approach for text classification. In proposed approach, they not only experimented with three different pre-trained neural word embeddings but also applied l2 norm regularization before and after concatenating all features of different channels. While adopting this model in our experimentation, three embedding layers, l2 norm regularization after features concatenation, and wide convolutions are utilized.

The model starts with three embedding layers, and each embedding layer is then followed by two convolution layers. Both convolution layers have 16 filters of size 3 and 5, respectively, which are followed by a global max-pooling layer. After that, features of all layers are concatenated and l2 norm regularization is applied using a dense layer with 128 output units. These features are then passed to a dense layer which acts as a classifier.

Dani Yogatama et al. [127] proposed an LSTM-based neural network model for classifying news articles, questions, and sentiments. Two different versions of the model, namely generative and discriminative LSTM model, were proposed. Both models were the same except that the discriminative model tried to maximize the conditional probability, while the generative model maximized the joint probability. We adopt discriminative version of the model. This model begins with an embedding layer, and output of the previous layer is fed to an LSTM layer which has 32 units. The features extracted by LSTM are then flattened and passed to a dense layer for classification.

Another LSTM-based model was proposed by Hamid Palangi et al. [128] to generate the sentence neural embeddings for raising the performance of document retrieval task. This model was not used for any sort of text classification but as its architecture is pretty similar to Yogatama et al. [127] proposed model that is why we have adopted this model for our experimentation. The output of the first embedding layer is fed to an LSTM layer which has 64 output units. The output of the LSTM layer is then flattened and feed into a dense layer that acts as a classifier.

As discussed before, both CNN and RNN have their own benefits and drawbacks [122]. In order to reap the benefits of both architectures CNN, and RNN, researchers proposed hybrid models [122, 129–132] in which usually a CNN architecture is followed by RNN. CNN extracts global features [129, 133, 134], while RNN learns long-term dependencies for the extracted features [116, 135–141].

A hybrid model was presented by Siwei Lai et al. [142] for the task of text classification. The author claimed that RNN was a biased model in which later words were more dominant than earlier words. To tackle this problem, a hybrid model was suggested that consists of bidirectional LSTM, followed by a max-pooling layer. The bidirectional nature of the model reduces the words dominance, whereas max-pooling layer captures more discriminative features. This model has outperformed twelve machine and deep learning-based models for the task of text classification.

The model begins with three embedding layers, first one is passed to forward LSTM layer, and the second one is fed to backward LSTM layer. Both LSTM layers have 100 output units. The yielded features from both LSTMs are concatenated along with third embedding layer and pass to a dense layer which has 200 output units. Dense layer is followed by a max-pooling layer, and the output of max-pooling layer is then passed to another dense layer which acts as a classifier.

Guibin Chen et al. [143] proposed another hybrid model that consists of CNN and LSTM and used for multi-label text classification. Pre-trained word embeddings were used to feed the CNN, and then, features were extracted to feed LSTM. The author claimed that the pre-trained word vectors contain the local features of each word, whereas CNN captured the global features of the input document. Both local and global features were then used by LSTM to predict the sequence of labels. We have adopted this model for multi-class classification instead of multi-label classification.

The model starts with an embedding layer which is followed by five convolution layers with 128 filters of sizes 10, 20, 30, 40, and 50, respectively. Each convolution layer is followed by a max-pooling layer of the same filter size. The output features from all five max-pooling layers are concatenated and flattened using a flatten layer. These flattened features are then passed to a dense layer which has 128 output units. The output from the dense layer along with the output of the embedding layer is then passed to an LSTM layer. This LSTM layer is followed by another dense layer that acts as a classifier.

Another hybrid model based on CNN and LSTM was proposed by Chunting Zhou et al. [144] for sentiment analysis and question classification. CNN was used to capture the high-level word features, whereas LSTM extracted the long-term dependencies. Different types of max-pooling layers were applied to the features extracted from CNN. However, the authors suggested that max-pooling layer must be avoided if the features needed to be passed to LSTM. Because LSTM was used for sequential input and a max-pooling layer would break the sequential architecture.

The output of the first embedding layer is passed to five convolution layers which have 64 filters of size 10, 20, 30, 40, and 50, respectively. The extracted features of these five convolution layers are then concatenated and fed to an LSTM layer which has 64 output units. This layer is followed by two dense layers from which the first dense layer has 128 units and the last dense layer eventually acts as a classifier.

The last chosen model in our research is also a hybrid model presented by Xingyou Wang et al. [145] for sentiment classification. The theory behind this model is the same as Chunting Zhou et al. [144] model except it used both LSTM and GRU along with max-pooling layers after CNN. Based on experimental results, authors claimed that both LSTM and GRU produced the same results, that is why we have adopted this model only with LSTM for our experimentation.

This model begins with an embedding layer, followed by three convolution layers which have 64 filters of size 3, 4 and 5, respectively. Each convolution layer is followed by a max polling layer of same filter sizes. After that, all the output features of the max-pooling layers are concatenated and passed to an LSTM layer which has 64 units. The features yielded by LSTM layer is then passed to a dense layer which has 128 units. This layer is followed by another dense layer that finally acts as a classifier.

5.6 Transfer learning using BERT

This section discusses the fruitfulness of transfer learning using pre-trained language model “BERT [42]” for the task of Urdu text document classification. Pre-training language model has proven extremely useful to learn generic language representations. In previous section, all discussed deep learning-based classification methodologies utilized pre-trained neural word embeddings including Word2vec [146], FastText [147], and Glove [148]

Traditional neural word embeddings are classified as static contextualized embeddings. These embeddings are prepared by training a model on a gigantic corpus in an unsupervised manner to acquire the syntactic and semantic properties of the words up to certain extent. However, these embeddings fail to grasp polysemy which is all about generating distinct embeddings for the same word on account of different contexts [37, 37–39]. For instance, consider two sentences like “Saim, I ’ll get late as I have to deposit some cash in Bank” and the other one is “My house is located in canal Bank”. In both sentences, word bank has a different meaning. However, models build on top of neural word embeddings do not consider the context of words in which they appear; thus, in both sentences, the word “Bank” will get a same vector representation which is not correct.

These downfalls are resolved by pre-trained language models which learn the vector representation of words based on the context in which they appear, and this is why embeddings of pre-trained language models such as Bidirectional Encoder Representations from Transformers (BERT [42]) are categorized as dynamic contextualized embeddings. Dynamic contextualized embeddings capture word semantics in dissimilar contexts to tackle the problem of polysemous, and context-dependent essence of words. In this way, language models such as BERT [42] manage to create different embeddings for the same word which appear in multiple contexts. Traditional language models are trained from left to right; thus, they are framed to predict next word. Contrarily, there exist few approaches such as Universal Language Model Fine-Tuning (UMLFit) [41] and Embeddings for Language Models (ELMo) [40] based on Bi-LSTM. Bi-LSTM is trained from left to right in order to predict next word, and from right to left to predict previous word, however not both at the same time, whereas BERT [42] utilizes entire sentence to learn from all words located at different positions. It randomly masks the words in certain context before making prediction. In

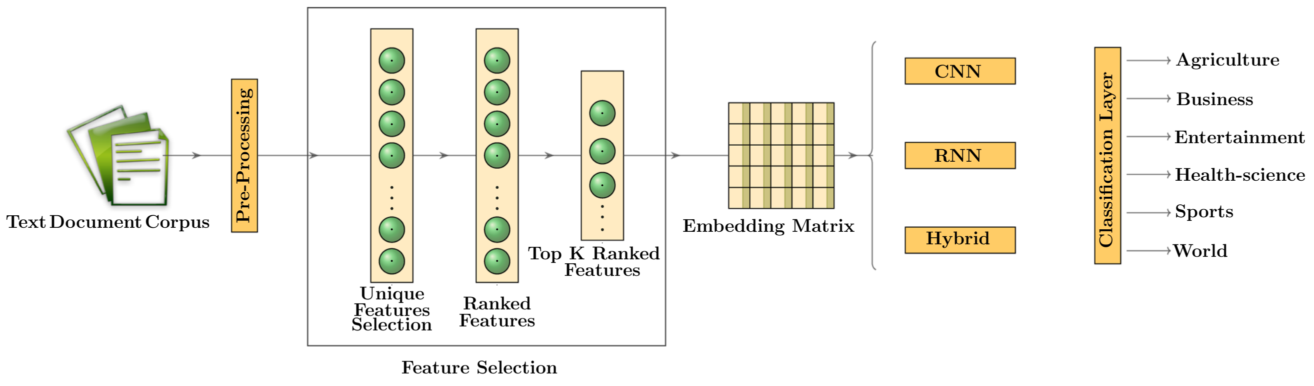


Fig. 3 Machine and deep learning-based hybrid methodology [46]

addition, it uses transformers which further make it accurate.

To summarize, due to masked language modelling, BERT [42] supersedes the performance of other language modelling approaches such as UMLFiT [41], and ELMO [40]. Moreover, training the transformed architecture bidirectionally in language modelling has proved extremely effective as it has deeper understanding of language context than unidirectional language models. Although BERT [42] has marked promising results in several natural language processing (NLP) tasks, there exists a limited research to optimize BERT [42] for the improvement of target NLP tasks. In this paper, we thoroughly investigate how to make the best use of BERT [42] for the task of text document classification. We explore multifarious methods to fine-tune BERT [42] in order to maximize its performance for Urdu text document classification. We perform pre-processing in a same manner as discussed in detail in Sect. 4.2.

5.7 Hybrid methodology for Urdu text document classification

This section explains the hybrid methodology for the task of Urdu text document classification. It is considered that deep learning-based methodologies automate the process of feature engineering; however, recent research in computer vision [149] and natural language processing (NLP) [46] extrapolates that these methodologies also extract some irrelevant and redundant features too which eventually derail the performance of underlay methodologies. In NLP, to remove irrelevant and redundant features, we [46] proposed a hybrid methodology which harvested the benefits of both trivial machine learning-based feature engineering, and deep learning-based automated feature engineering. In proposed hybrid methodology, first, a vocabulary of discriminative features was developed by utilizing a filter-based feature selection algorithm, namely normalized difference measure (NDM) [47], and then, the constructed vocabulary was fed to the embedding layer of CNN. Hybrid methodology managed to produce the promising figures on two benchmark English datasets 20-Newsgroup[6], and BBC[7], when compared against the performance figures of traditional machine, and deep learning methodology. To evaluate that the proposed hybrid approach is extremely versatile and its effectiveness is neither biased towards the size of training data nor towards specific language or deep learning model, we assess the integrity of hybrid methodology by performing experimentation on different datasets and language with a variety of deep learning models. We adopt 4 CNN, 2 RNN, and 4 hybrid models (CNN+RNN) which were previously used for text document or sentence classification (discussed in Sect. 5.5). Hybrid approach is evaluated on three Urdu datasets (CLE Urdu Digest 1000*k*, CLE Urdu Digest 1*M*, DSL Urdu News) (Fig. 3).

We perform pre-processing in the same manner as discussed in detail in Sect. 4.2.

6 Datasets

To evaluate the integrity of all variety of methodologies based on machine learning, deep learning, hybrid approach, and language modelling, we use two state-of-the-art closed source corpora CLE Urdu Digest 1000*k*, CLE Urdu Digest 1*M*, and one publicly available presented corpus namely DSL Urdu news. All textual documents of DSL Urdu news dataset are crawled from following web sites Daily Jang[8], Urdu Point[9], HmariWeb[10], BBC Urdu[11], and parsed

6 http://archive.ics.uci.edu/ml/datasets/twenty+newsgroups.
7 http://mlg.ucd.ie/datasets/bbc.html.
8 https://jang.com.pk/.
9 https://www.urdupoint.com/.
10 http://hamariweb.com/.
11 https://www.bbc.com/urdu.

Table 3 DSL Urdu news dataset statistics

Class	No. of documents	No. of sentences	No. of tokens	No. of tokens after lemmatization
Agriculture	102	669	17,967	9856
Business	120	672	20,349	9967
Entertainment	101	685	19,671	10,915
World	111	631	18,589	12,812
Health-sciences	108	823	27,409	12,190
Sports	120	744	24,212	9992

Table 4 CLE Urdu Digest 1000*k* dataset statistics before and after Lemmatization

Class	No. of documents	No. of sentences	No. of tokens	No. of tokens after lemmatization
Culture	28	488	8767	8767
Health	29	608	9895	9895
Letter	35	777	11,794	11,794
Interviews	36	597	12,129	12,129
Press	29	466	10,007	10,007
Religion	29	620	9839	9839
Science	55	468	8700	8700
Sports	29	588	10,030	10,030

Table 5 CLE Urdu Digest 1M dataset statistics before and after lemmatization

Class	No. of documents	No. of sentences	No. of tokens	No. of tokens after lemmatization
Culture	133	8784	145,228	145,228
Health	153	11,542	169,549	169,549
Letter	105	8565	115,177	115,177
Interviews	38	2481	41,058	41,058
Press	118	6106	125,896	125,896
Religion	100	6416	107,071	107,071
Science	109	6966	117,344	117,344
Sports	31	2051	33,143	33,143

through Beautiful Soup[12]. Table 3 illustrates the characteristics of newly developed corpus having 300*K* words, 4224 sentences, and a total 662 documents which belong to following six categories health-science, sports, business, agriculture, world, and entertainment. Average length of a document is approximately 193 words in the developed corpus.

State-of-the-art corpora CLE Urdu Digest 1000K contains 270 news documents, and CLE Urdu Digest 1*M* contains 787 news documents belonging to 8 classes. Former one is a precise corpus and average length of a document is nearly 140 words; however, latter one is a large corpus with an average document length of 900 words. Statistics of both corpora with respect to each class are reported in Tables 4 and 5, respectively.

In order to apply machine learning-based text document classification methodologies, for underlay corpus, textual documents of each class need to be asymmetric when compared with the documents of other classes. Here, in our work, to perform distribution analysis of experimental datasets with respect to unique classes, we have employed most widely used Kullback–Leibler (KL) divergence approach [150] following the work of Stehlik et al. [151]. It assists to deduce whether samples of distinct classes of one or more datasets are symmetrical or asymmetrical by nature. We receive multifarious divergences which empirically reveal that samples belonging to different classes for each dataset are asymmetrical. This asymmetry fully supports the fact that all experimental datasets are not biased towards samples of one particular class.

[12] https://pypi.org/project/beautifulsoup4/.

7 Experimental setup and results

This section summarizes different APIs that are used to perform Urdu text document classification. It also discusses the results produced by methodologies based on machine learning, deep learning, and hybrid approach on three datasets (DSL Urdu news, CLE Urdu Digest 1000*k*, CLE Urdu Digest 1*M*) used in our experimentation. In order to process Urdu text for the task of Urdu text document classification, we develop a rule base sentence splitter and tokenizer. To evaluate the integrity of machine learning-based Urdu text document classification methodology, all three datasets are split into train and test sets containing 70%, and 30% documents from each class, respectively. The parameters of Naive Bayes [26] classifier are alpha=1.0, fit_prior=True, class_prior=None, and SVM [25] classifier is used with linear kernel and balanced class weight.

On the other hand, in order to evaluate the performance of adopted deep learning methodologies and to perform a fair comparison with machine learning-based approaches for all three datasets, we use 30% data for test set and remaining 70% data is further split into train and validation sets having 60% and 10% data, respectively. We use Keras API to implement the methodologies of ten adopted neural network-based models. Pre-trained Urdu word embeddings provided by Haider et. al [36], and FastText[13] are used to feed all embedding layers except the second layer in Yin et al. [125] model and both second and third layers in Zhang et al. [126] model which are randomly initialized. To evaluate and compare the performance of filter-based feature selection algorithms, first we rank the features of training corpus against all classes. Then, at different pre-defined test points, we take top *k* features from all classes and feed these features to two different classifiers SVM [25], and Naive Bayes [26]. For adopted deep learning-based Urdu text document classification methodologies, we perform experimentation in two different ways. In first case, after pre-processing, we select entire set of unique terms of each corpus and fed to the embedding layer of all adopted models (discuss with detail in Sect. 5.5), whereas in second case, we select 1000 most frequent terms for DSL Urdu News, and CLE Urdu Digest 1000*k* datasets, and 10, 000 most frequent terms for CLE Urdu Digest 1*M* dataset.

Likewise, to evaluate the performance of hybrid approach which reaps the benefits of both machine and deep learning-based feature engineering, as similar to machine learning-based classification, for each dataset, we first rank the features of training corpus using NDM [47] feature selection algorithm. Then, top *k* features of each class are fed to 10 different deep learning models. Rather than performing extensive experimentation with all feature selection algorithms once again, considering the promising performance produced by NDM [47] with all machine learning-based methodologies, we only explore the impact of NDM [47] feature selection algorithm for 10 different deep learning-based classification methodologies.

To assess the effectiveness of transfer learning using BERT [42], we fine-tune multilingual cased language model (BERT-Base [42]) having 12-layers, 12, heads, 768 hidden units, 110M parameters and pre-trained on 104 languages. We utilize multilingual cased model as it resolves normalization problems in several languages. We fine-tune multilingual model with the buffer size of 400, sequence length of 512, batch size of 16, and learning rate of 1e-5 for 50 epochs.

As two close source experimental datasets (CLE Urdu Digest 1000*k*, 1*M*) are highly unbalanced, thus instead of using accuracy, or an other evaluation measure, we have performed evaluation using F1 measure as it is widely considered more appropriate evaluation measure for unbalanced datasets.

7.1 Results of traditional machine learning-based text document classification methodology

This section summarizes and compares the performance of ten feature selection algorithms (RDC [56], MMR [21], NDM [47], POISON [60], GINIINDEX [59], ACC2 [55], ODDS [58], IG [57], CHISQ [48], BNS [55]) on two closed source corpora (CLE Urdu Digest 1000*k*, CLE Urdu Digest 1*M* Benchmark dataset) and one newly developed corpus (DSL Urdu News) using Naive Bayes [26], and SVM [25] classifiers. We compare the performance of feature selection algorithms over predefined set of features {10, 20, 50, 100, 200, 500, 1000, 1500} in terms of F1 score.

7.1.1 DSL Urdu news dataset

Tables 6 and 7 summarize the performance of ten feature selection algorithms produced against 8 different benchmark test points over DSL Urdu news dataset using Naive Bayes [26] and SVM [25] classifiers, respectively.

It can be summarized from Tables 6 and 7 that NDM [47] marks the lowest performance at top 10 features; however, with the increase in number of features, its performance gets rocketed for both classifiers. NDM [47] outperforms rest of the feature selection algorithms with a huge margin after the induction of 150 or more number of features. Although MMR [21] does not perform well with Naive Bayes [26], it outshines other nine feature selection

[13] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md.

Table 6 Performance of ten feature selection algorithms against 8 different benchmark test points over DSL Urdu news dataset using Naive Bayes classifier

Feature selection algorithms	Benchmark test points							
	10	20	50	100	200	500	1000	1500
RDC [56]	0.83	0.85	0.85	0.88	0.90	**0.91**	0.90	0.89
NDM [47]	0.70	0.76	0.87	0.90	0.93	**0.94**	0.94	0.88
MMR [21]	0.71	0.82	0.88	**0.91**	0.91	0.91	0.91	0.89
POISON [60]	0.82	0.86	0.90	0.89	0.91	**0.92**	0.92	0.89
GINI [59]	0.77	0.81	0.88	0.87	0.88	**0.90**	0.90	0.90
ACC2 [55]	0.82	0.88	0.87	0.88	0.89	**0.90**	0.90	0.90
ODDS [58]	0.70	0.82	0.88	**0.91**	0.91	0.91	0.90	0.89
IG [57]	0.81	0.86	0.90	0.91	0.91	**0.92**	0.91	0.89
CHISQ [48]	0.79	0.87	0.90	0.91	0.91	**0.92**	0.91	0.89
BNS [55]	0.81	0.88	0.87	0.88	0.89	0.90	**0.91**	0.90

Peak performance of every classifier is highlighted in bold for each experimental dataset

Table 7 Performance of ten feature selection algorithms against 8 different benchmark test points over DSL Urdu news dataset using SVM classifier

Feature Selection Algorithms	Benchmark Test Points							
	10	20	50	100	200	500	1000	1500
RDC [56]	0.80	0.79	0.80	0.84	0.86	**0.88**	0.88	0.88
NDM [47]	0.74	0.78	0.86	0.88	0.90	**0.91**	0.91	0.90
MMR [21]	0.77	0.83	0.87	0.89	0.89	0.88	0.89	**0.90**
POISON [60]	0.80	0.83	0.85	0.88	0.87	0.89	0.88	**0.90**
GINI [59]	0.76	0.77	0.83	0.83	0.83	0.86	**0.88**	0.88
ACC2 [55]	0.80	0.84	0.82	0.85	0.86	0.87	0.88	**0.89**
ODDS [58]	0.74	0.83	0.86	0.88	0.88	**0.89**	0.89	0.89
IG [57]	0.83	0.85	0.84	0.88	0.89	0.89	0.89	**0.90**
CHISQ [48]	0.81	0.85	0.86	0.88	0.88	0.88	**0.90**	0.90
BNS [55]	0.81	0.83	0.82	0.85	0.87	0.87	**0.88**	0.88

Peak performance of every classifier is highlighted in bold for each experimental dataset

algorithms on 50, and 100 number of features using SVM [25] classifier. BNS [55] only manages to beat other feature selection algorithms at 20, and 1500 number of features with Naive Bayes [26] classifier compared to SVM [25] where it is badly beaten by seven feature selection algorithms as it marks the second lowest performance of 89%. While GINI [59] and RDC [56] show the worst performance for both classifiers, all other feature selection algorithms show a mix trend across all test points.

In a nutshell, Naive Bayes [26] outperforms SVM [25] by revealing a better performance. Moreover, the performance of Naive Bayes [26] reaches the peak of 94% than SVM [25] which manages to produce the performance of only 91%.

7.1.2 CLE Urdu Digest 1*M* dataset

Ten feature selection algorithms performance figures against 8 different benchmark test points over CLE Urdu Digest 1*M* dataset using Naive Bayes, and SVM classifiers are shown in Tables 8 and 9.

For CLE Urdu Digest 1*M* dataset, ODDS [58] performance begins at low of just 53%, and 59% with Naive Bayes [26] and SVM [25], but it shows an upward trend until 200 number of features with both classifiers considering the trends depicted by Tables 8 and 9. Although ODDS [58] outperforms nine feature selection algorithms at four benchmark test points (no. of features= 100, 200, 500, 1000) using Naive Bayes [26], it fails to produce highest performance with SVM [25] classifier. Contrarily, CHISQ [48] does not produce good performance with Naive Bayes [26] classifier, but it manages to reveal best performance with SVM [25] classifier. CHISQ [48] either equalizes or surpass the performance of nine feature selection algorithms at most test points. Although NDM [47] performance rises almost gradually until 200 number of features, afterwards its performance fluctuates and fails to surpass best performance figures. Likewise, feeding MMR [21] ranked features to both classifiers, performance

Table 8 Performance of ten feature selection algorithms against 8 different benchmark test points over CLE Urdu Digest 1*M* dataset using Naive Bayes classifier

Feature Selection Algorithm	Benchmark Test Points							
	10	20	50	100	200	500	1000	1500
RDC [56]	0.65	0.64	**0.66**	0.63	0.62	0.61	0.59	0.55
NDM [47]	0.51	0.58	0.63	0.64	**0.65**	0.61	0.57	0.60
MMR [21]	0.52	0.54	0.58	0.60	**0.62**	0.59	0.51	0.46
POISON [60]	0.50	0.60	0.61	0.61	**0.62**	0.56	0.48	0.45
GINI [59]	0.13	0.14	0.46	0.59	**0.62**	0.62	0.62	0.60
ACC2 [55]	0.65	**0.66**	0.65	0.65	0.64	0.62	0.57	0.53
ODDS [58]	0.53	0.56	0.62	0.66	**0.68**	0.65	0.64	0.56
IG [57]	0.62	0.62	0.63	**0.64**	0.63	0.60	0.49	0.45
CHISQ [48]	0.57	0.59	**0.64**	0.63	0.62	0.57	0.48	0.48
BNS [55]	**0.65**	0.64	0.64	0.64	0.64	0.63	0.56	0.53

Peak performance of every classifier is highlighted in bold for each experimental dataset

Table 9 Performance of ten feature selection algorithms against 8 different benchmark test points over CLE Urdu Digest 1*M* dataset using SVM classifier

Feature Selection Algorithm	Benchmark Test Points							
	10	20	50	100	200	500	1000	1500
RDC [56]	0.69	0.70	0.73	0.76	0.79	0.77	**0.78**	0.78
NDM [47]	0.55	0.67	0.76	**0.81**	0.81	0.79	0.76	0.80
MMR [21]	0.62	0.68	0.71	0.77	0.78	0.79	**0.80**	0.78
POISON [60]	0.53	0.64	0.75	0.76	**0.83**	0.82	0.80	0.78
GINI [59]	0.27	0.34	0.60	0.67	0.70	0.70	0.78	**0.79**
ACC2 [55]	0.67	0.69	0.77	**0.79**	0.79	0.79	0.78	0.78
ODDS [58]	0.59	0.69	0.74	0.77	0.80	0.76	0.80	**0.82**
IG [57]	0.66	0.69	0.75	0.79	0.78	**0.82**	0.81	0.78
CHISQ [48]	0.62	0.70	0.77	0.77	**0.83**	0.82	0.81	0.79
BNS [55]	0.67	0.71	0.74	0.78	**0.80**	0.79	0.78	0.78

Peak performance of every classifier is highlighted in bold for each experimental dataset

kept increasing until 200 number of features. After 200 features, its performance declines almost gradually with Naive Bayes [26] and shows mixed trend with SVM [25] classifier. While POISON [60] marks the worst performance with Naive Bayes [26], GINI [59] shows the lowest performance with SVM [25] classifier. The rest of the feature selection algorithms show both upward and downward trends across test points for both classifiers, but they produce better performance figures with SVM [25] classifier.

As a whole, SVM [25] outshines Naive Bayes [26] for CLE Urdu Digest 1*M* dataset. Moreover, the performance of SVM [25] reaches the peak of 83% than Naive Bayes [26] which only manages to reach at 68%.

7.1.3 CLE Urdu Digest 1000*K* dataset

Tables 10 and 11 elaborate the performance of ten feature selection algorithms on CLE Urdu Digest 1000*k* dataset using Naive Bayes [26] and SVM [25] classifiers, respectively.

It can be seen from Tables 10 and 11, with Naive Bayes [26] classifier, CHISQ [48] produces the highest performance until 100 number of features but its performance dips sharply with the induction of more features. With SVM [25] classifier, CHISQ [48] reveals a similar trend until 200 number of features but decreases slightly afterwards. CHISQ [48] manages to beat nine feature selection algorithms with Naive Bayes [26] as the performance of most feature selection algorithms start declining after 100 number of features. While NDM [47] manages to beat other feature selection algorithms at two benchmark test points (number of features= {1000, 1500}), MMR [21] reveals better performance at 200, and 500 number of features with Naive Bayes [26] classifier, whereas NDM [47] outshines rest of the feature selection algorithms at four test points {20, 50, 100, 200} with SVM [25] as compared to POISON [60] which manages to surpass the

Table 10 Performance of ten feature selection algorithms against 8 different benchmark test points over CLE Urdu Digest 1000*K* dataset using Naive Bayes classifier

Feature selection algorithm	Benchmark test points							
	10	20	50	100	200	500	1000	1500
RDC [56]	0.57	**0.58**	0.56	0.55	0.51	0.31	0.25	0.17
NDM [47]	0.63	0.70	**0.71**	0.55	0.40	0.27	0.36	0.28
MMR [21]	0.64	0.65	**0.81**	0.71	0.67	0.51	0.31	0.21
POISON [60]	0.50	0.50	0.55	**0.57**	0.43	0.36	0.21	0.22
GINI [59]	0.35	0.35	0.46	0.50	**0.52**	0.39	0.21	0.10
ACC2 [55]	0.60	0.63	**0.67**	0.60	0.44	0.37	0.17	0.17
ODDS [58]	0.64	**0.73**	0.70	0.70	0.62	0.41	0.31	0.20
IG [57]	0.69	**0.76**	0.74	0.71	0.46	0.32	0.21	0.19
CHISQ [48]	0.73	0.72	**0.81**	0.79	0.61	0.46	0.31	0.22
BNS [55]	0.61	0.63	**0.67**	0.61	0.50	0.37	0.12	0.17

Peak performance of every classifier is highlighted in bold for each experimental dataset

Table 11 Performance of ten feature selection algorithms against 8 different benchmark test points over CLE Urdu Digest 1000*K* dataset using SVM classifier

Feature Selection Algorithms	Benchmark Test Points							
	10	20	50	100	200	500	1000	1500
RDC [56]	0.63	0.64	0.67	0.65	**0.74**	0.73	0.72	0.65
NDM [47]	0.70	0.81	**0.92**	0.90	0.87	0.73	0.66	0.69
MMR [21]	0.66	0.74	0.85	**0.86**	0.81	0.82	0.72	0.68
POISON [60]	0.61	0.70	0.84	0.85	**0.86**	0.84	0.75	0.66
GINI [59]	0.50	0.47	0.54	0.59	0.64	**0.77**	0.75	0.71
ACC2 [55]	0.61	0.65	0.67	0.73	0.72	**0.75**	0.74	0.67
ODDS [58]	0.71	0.79	0.74	**0.86**	0.85	0.78	0.68	0.64
IG [57]	0.63	0.68	0.80	**0.86**	0.81	0.80	0.75	0.66
CHISQ [48]	0.69	0.72	0.77	0.83	**0.86**	0.82	0.71	0.66
BNS [55]	0.61	0.64	0.67	0.70	**0.77**	0.75	0.74	0.67

Peak performance of every classifier is highlighted in bold for each experimental dataset

performance of other feature selection algorithms at only one test point (number of features=500). Rest of the feature selection algorithms mark a mixed trend for both classifier, but produce better performance figures with SVM [25] classifier. Among all, GINI [59] shows the worst performance with both classifiers.

In a nutshell, once again SVM [25] outshines Naive Bayes [26] by revealing a better performance. Furthermore, the performance of SVM [25] reach the peak of 92% than Naive Bayes [26] which manages to produce the performance of only 81%.

7.1.4 Discussion

In order to provide a bird's eye view over the performance of each filter-based feature selection algorithm, this section reports the average performance of ten feature selection algorithms across three datasets. To assess the discriminative power of features ranked by ten feature selection algorithms, we select subset (10, 20, 50, 100, 200, 500, 1000, 1500) of top k features to feed Naive Bayes [26] and SVM [25] classifiers. Based on predefined subset of features, Tables 12, 13 and 14 show the best performing feature selection algorithm at each benchmark test point using Naive Bayes [26] and SVM [25] classifiers for two closed source (CLE Urdu Digest 1000*K*, CLE Urdu Digest 1*M*) and one presented dataset (DSL Urdu News).

For DSL Urdu news dataset, Table 12 illustrates that NDM [47] produces the best performance with both classifiers on three same benchmark test points (number of features= 200, 500, 1000), whereas MMR [21] reveals the top performance at 50, 100, and 1500 number of features with SVM [25] only. While BNS [55] manages to produce promising performance at two test points, ODDS [58] and RDC [56] outperform rest of the feature selection algorithms at only one test point with Naive Bayes [26] classifier. On the other hand, for SVM [25] classifier, IG [57] attains best performance at two test points compared to

Table 12 Best performing feature selection algorithm against each benchmark test point on DSL Urdu News dataset

Classifier	Number of Features							
	10	20	50	100	200	500	1000	1500
NB [26]	RDC [56]	BNS [55]	IG [57]	ODDS [58]	NDM [47]	NDM [47]	NDM [47]	BNS [55]
SVM [25]	IG [57]	IG [57]	MMR [21]	MMR [21]	NDM [47]	NDM [47]	NDM [47]	MMR [21]

Table 13 Best performing feature selection algorithm against each benchmark test point on CLE Urdu Digest 1000*k* dataset

Classifier	Number of Features							
	10	20	50	100	200	500	1000	1500
NB [26]	CHISQ [48]	IG [57]	CHISQ [48]	CHISQ [48]	MMR [21]	MMR [21]	NDM [47]	NDM [47]
SVM [25]	ODDS [58]	NDM [47]	NDM [47]	NDM [47]	NDM [47]	POISON [60]	IG [57]	GINI [59]

Table 14 Best performing feature selection algorithm against each benchmark test point on CLE Urdu Digest 1M dataset

Classifier	Number of Features							
	10	20	50	100	200	500	1000	1500
NB [26]	ACC2 [55]	ACC2 [55]	RDC [56]	ODDS [58]	ODDS [58]	ODDS [58]	ODDS [58]	NDM [47]
SVM [25]	RDC [56]	BNS [55]	ACC2 [55]	NDM [47]	CHISQ [48]	POISON	CHISQ [48]	ODDS [58]

Naive Bayes [26] classifier where it manages to mark best performance at only one test point. All other feature selection algorithms such as ACC2 [55], GINI [59], POISON [60], and CHISQ [48] have failed to outshine the peak performance of other feature selection algorithms with SVM [25] or Naive Bayes [26] classifier at any test point.

Turning towards the CLE Urdu Digest 1000*k* dataset, Table 13 depicts that NDM [47] once again reveals promising performance with both classifiers as it outshines other feature selection algorithms at four test points using SVM [25] (no. of features=20, 50, 100, 200) and at two test points using Naive Bayes [26] (no. of features=1000, 1500), whereas CHISQ [48] and MMR [21] show good performance with only Naive Bayes [26] classifier at two test points. While ODDS [58] and POISON [60] manage to produce best performance at one test point each using SVM [25] classifier, IG [57] once again shows mixed trend as it produces good performance with both classifiers. All other feature selection algorithms such as RDC [56], BNS [55], and ACC2 [55] fail to outperform other on any test point with any classifier.

Contrarily, in CLE Urdu Digest *IM* dataset, ODDS [58] reveals promising performance at four test points with Naive Bayes [26] and at one test point with SVM [25] as shown by Table 14. While ACC2 [55] shows highest performance on the induction of 10, and 20 number of features with Naive Bayes [26], and at 50 number of features with SVM [25] classifier, NDM [47] and RDC [56] mark best performance at one test point with each classifier. On the other hand, CHISQ [48] performs well with SVM [25] classifier only. Likewise, BNS [55] and POISON [60] also manage to beat other feature selection algorithms at one test point with SVM [25] classifier, whereas MMR [21], GINI, and IG [57] have failed to beat other feature selection algorithms with any classifier and test point.

Furthermore, Tables 15, 16 and 17 highlight the highest performance attain by each feature selection algorithm against particular test point using any classifier for all three corpora.

It can be clearly seen from Table 15 that NDM [47] outperforms rest of the feature ranking metrics by revealing the highest performance figure of 94% at 1000 number of features on DSL Urdu news dataset. Three algorithms {POISON [60], IG [57], CHISQ [48]} produce second highest performance of 92% on the induction of top 500 features. While MMR [21], ODDS [58], RDC [56], and

Table 15 Peak performance figures of ten feature selection algorithms on DSL Urdu news dataset using Naive Bayes and SVM classifiers

FR Metric	RDC [56]	NDM [47]	MMR [21]	POISON [60]	GINI [59]	ACC2 [55]	ODDS [58]	IG [57]	CHISQ [48]	BNS [55]
Test Point	500	1000	100	500	500	500	100	500	500	1000
F1 Score	0.91	0.94	0.91	0.92	0.90	0.90	0.91	0.92	0.92	0.91

Table 16 Peak performance figures of ten feature selection algorithms on CLE Urdu Digest 1000K dataset using Naive Bayes and SVM classifiers

FR Metric	RDC [56]	NDM [47]	MMR [21]	POISON [60]	GINI [59]	ACC2 [55]	ODDS [58]	IG [57]	CHISQ [48]	BNS [55]
Test Point	200	50	100	200	500	500	100	100	200	200
F1 Score	0.74	0.92	0.86	0.86	0.77	0.75	0.86	0.86	0.86	0.77

Table 17 Peak performance figures of ten feature selection algorithms on CLE Urdu Digest 1M dataset using Naive Bayes and SVM classifiers

FR Metric	RDC [56]	NDM [47]	MMR [21]	POISON [60]	GINI [59]	ACC2 [55]	ODDS [58]	IG [57]	CHISQ [48]	BNS [55]
Test Point	200	100	1000	200	1500	100	1500	500	200	200
F1 Score	0.79	0.81	0.80	0.83	0.79	0.79	0.82	0.82	0.83	0.80

BNS [55] attain third highest performance of 91% at different test points, GINI [59] and ACC2 [55] both mark the performance of 90% at same test point (number of features = 500).

On the basis of the figures reported in Table 16, for CLE 1000K dataset, NDM [47] reveals the highest performance figure of 92% at 50 number of features. Majority of the algorithms {MMR [21], ODDS [58], IG [57], CHISQ [48], POISON [60]} reveal second best performance of 86% on the account of different number of features. In addition, BNS [55] and GINI [59] produce third highest performance of 77% at different test points, whereas ACC2 [55] and RDC [56] show least best performance figures of 75%, and 74% on the induction of 500, and 200 number of features respectively.

However, POISON [60] and CHISQ [48] produce the highest performance figure of 83% at same test point (number of features = 200) for CLE Urdu Digest 1M dataset as illustrated by Table 17. IG [57] and ODDS manage to produce the second best performance of 82% at different number of features. Likewise, NDM [47] marks third highest performance figure of 81%, whereas BNS [55] and MMR [21] manage to produce fourth highest performance figure of 80% on the induction of different features. Remaining three feature selection algorithms {ACC2 [55], RDC [56], GINI [59]} reveal lowest best performance figure of 79% at different number of features.

Moreover, we compare the average performance of all feature selection algorithms against each corpus. Average performance is calculated by taking the ratio between number of outperforming test points and total number of test points with both classifiers which are 16. For each algorithm, outperforming test points are those test points over which an algorithm beats all other feature selection algorithms on certain dataset but regardless of classifier. For instance, as IG [57] outperforms nine feature ranking metrics at three test points (no. of features =10, 20, 50) in DSL Urdu news dataset, so IG [57] will get the score of 18.75 which is computed as below:

$$\begin{aligned}\text{Score on DSL Urdu news dataset} &= (\text{outperforming test points}/\text{total test points}) * 100\\ &= (3/16) * 100\\ &= 18.75\end{aligned}$$

Considering the trends shown in Table 18, on DSL Urdu news dataset, average performance 37.5% of NDM [47] feature selection algorithm is the highest. MMR [21] and IG [57] produce second best average performance of 18.75%. While BNS [55] marks third highest average performance of 12.75%, RDC [56] and ODDS [58] reveal

Table 18 Average performance of ten feature selection algorithms against each corpus

FR Metric	DSL Urdu News	CLE Urdu Digest 1000K	CLE Urdu Digest 1*M*	Average
RDC [56]	6.25	0	12.5	6.25
NDM [47]	**37.5**	**37.5**	12.5	**29.16**
MMR [21]	18.75	12.5	0	10.41
POISON [60]	0	6.25	6.25	4.16
GINI [59]	0	6.25	0	2.08
ACC2 [55]	0	0	18.75	6.25
ODDS [58]	6.25	6.25	**31.25**	14.58
IG [57]	18.75	12.5	0	10.41
CHISQ [48]	0	18.75	12.5	10.41
BNS [55]	12.5	0	6.25	6.25

Peak performance of every classifier is highlighted in bold for each experimental dataset

the lowest average performance of 6.25%. However, four feature selection algorithms {POISON [60], GINI [59], ACC2 [55], CHISQ [48]} do not outperform other feature selection algorithms at any test point at all. Likewise, on CLE Urdu Digest 1000K dataset, NDM [47] attains same highest average performance (37.5%). CHISQ marks second highest average performance of 18.75%, MMR [21], and IG [57] both reveal third best average performance of 12.75%. Three feature selection algorithms POISON [60], GINI [59], and ODDS [58] reveal lowest average performance of 6.25%. Among all, three algorithms {RDC [56], ACC2 [55], BNS [55]} fail to outperform other feature selection algorithms at any test point.

Furthermore, on CLE Urdu Digest 1*M* dataset, ODDS shows the best average performance of 31.25% followed by 18.75% attained by ACC2 [55]. Three feature selection algorithms {RDC [56], NDM [47], CHISQ [48]} mark the third highest average performance of 12.5%. While POISON [60] and BNS [55] manage to attain the lowest performance of 6.25%, three feature selection algorithms {MMR [21], GINI [59], IG [57]} fail to surpass the performance of other feature selection algorithms at any test point.

Taking into account the average performance of ten feature selection algorithms across all three datasets, NDM [47] performs well across all datasets; thus, it produces highest average performance of 29.16%. Contrarily, three feature selection algorithms {RDC [56], ACC2 [55], BNS [55]} show the lowest average performance of 6.25%.

Ideally, feature selection algorithm shall assign greater scores to extremely discriminative features, and lower scores to less significant or irrelevant features. Considering this, typical criteria to select highly discriminative features are as follows: features appearing very rarely in a certain class or very frequently across all classes are totally irrelevant. Thus, they shall be assigned lower scores. Contrarily, features having relative frequencies in a certain class and show in most of the corpus classes or do not show at all are greatly discriminative; hence, they must be assigned higher scores. Among all discussed feature selection algorithms, ACC2 [55] is the simplest feature selection algorithm which assigns scores by computing the absolute difference among t_{pr}, and f_{pr}; however, it assigns same scores to those features which have same frequencies in negative or positive classes. This is why it fails to perform better than other feature selection algorithms such as NDM [47], MMR [21], ODDS [58], IG [57], and CHISQ [48]. NDM [47] shows the best performance on DSL Urdu News, and CLE Urdu Digest 1000*K* datasets as it is a modified version of ACC2 [55]. NDM [47] assigns high score to those features which occur more times in one class and least occur in other classes. To achieve this, it normalizes the ACC2 [55] scores by dividing it with the minimum of t_{pr}, and f_{pr} which results in better performance. As no other feature selection algorithm considers the minimum of t_{pr}, and f_{pr} while computing score for certain feature, NDM [47] marks best performance for most datasets. However, NDM [47] score may shoot out for highly sparse and irrelevant features. For instance, consider a dataset of six classes where the t_{pr} of a feature is 0, and f_{pr} of the feature is 5, so in this case assuming that 0 is approximately equal to 0.0001 to avoid infinity, NDM [47] (0–5/0.0001= − 50,000) will assign very high score to an irrelevant feature which is not correct at all. MMR [21] which is the modified version of NDM [47] reduces such high score by multiplying NDM [47] score with the max of t_{pr},and f_{pr}. According to Rehman et al. [47], the terms located at the bottom right and top left corners of the contour plot are more important than the ones located around the diagonals. Although MMR [21] alleviates the score of highly sparse and irrelevant features but in case of small and non-skewed datasets where the term distribution is pretty balanced, MMR [21] score gets affected by the max of t_{pr},and f_{pr}. In other words, MMR [21] performs best when the dataset is large and highly skewed in nature which is not the case with experimental datasets. This is why MMR [21] fails to obliterate the performance of NDM [47]. ODDS [58] shows second best performance along with IG [57], and CHISQ [48]. ODDS [58], IG [57], and CHISQ [48] select the features in a univariate fashion; therefore, these feature selection algorithms fail to handle redundant features [152]. Also these feature selection algorithms rank the features on the basis of class relevance, but they fail to correctly discriminate those features which have large distinct values.

7.2 Results of adopted deep learning-based methodologies and the hybrid methodology

This section summarizes the performance of ten adopted deep learning methodologies on the test sets of two closed source (CLE Urdu Digest 1000K, CLE Urdu Digest 1*M*) and one publicly available presented dataset (DSL Urdu News) using F1 measure. It also reveals the performance impact created by hybrid approach which combines traditional machine learning-based feature engineering and deep learning-based automated feature engineering on the test set of all three datasets.

The performance of four CNN, two RNN, four hybrid models, and pre-trained multilingual language model BERT [42] with full features, most frequent 2*K*, 3*K*, 10*K* features, and discriminative 250, 350, and 400 features selected by filter-based feature selection algorithm (NDM) [47] is summarized in Table 19. In this table, full vocabulary is the set of unique terms found after corpus preprocessing. MF implies most frequent terms of the corpus, whereas NDM@ refers to the discriminative terms selected by NDM [47]. For discriminative terms, firstly, all unique terms of the training corpus are ranked using filter-based feature selection algorithm (NDM) [47], and then, top 250, 350, and 400 features of each class are selected in order to feed the embedding layer.

Considering the figures shown in Table 19, feeding adopted deep learning models with vocabulary of unique words, among all, Yin et al CNN-based model [125] produces the highest performance figures of 70%, 71%, and 90% on CLE Urdu Digest 1000*K*, CLE Urdu Digest 1*M*, and DSL Urdu news datasets, respectively. While Zhang et al model [126] also produces the peak performance of 90% at DSL Urdu News dataset, Yogatama et al. [127] RNN-based model replicates the peak performance of Yin et al. [125] model on CLE Urdu Digest 1*M* dataset.

Conversely, by feeding deep learning models with most frequent features instead of full features, over both DSL Urdu news, and CLE Urdu Digest 1000*K* datasets, performance of more than half of the adopted deep learning methodologies increases as compared to CLE Urdu Digest 1*M* where the performance of exactly half of the deep learning methodologies raises by a decent margin. Kalchbrenner et al. model [124] surpasses the performance of all models by marking the performance of 91% over DSL Urdu News corpus, whereas Yin et al [125] CNN-based model depicts the performance of 76% with 10K most frequent features over CLE Urdu Digest 1*M* dataset, and

Table 19 Performance of adopted deep learning methodologies, and BERT over three corpora using full, most frequent and NDM features

Model type	Models	Datasets								
		DSL Urdu News			CLE 1000*K*			CLE 1*M*		
		Full vocab	MF @3K	NDM @250	Full vocab	MF @2K	NDM @350	Full vocab	MF @10K	NDM @400
CNN	Yoon Kim et al [123]	0.88	0.90	**0.93**	0.46	0.42	0.77	0.66	0.60	0.74
	Kalchbrenner et al [124]	0.89	**0.91**	0.91	0.57	0.63	0.69	0.63	0.70	0.75
	Yin et al [125]	**0.90**	0.90	0.90	0.70	**0.67**	**0.78**	**0.71**	**0.76**	**0.80**
	Zhang et al [126]	**0.90**	0.90	0.90	0.50	0.56	0.71	0.63	0.65	0.72
RNN	Yogatama et al [127]	0.88	0.89	0.91	0.51	0.64	0.56	**0.71**	0.68	0.68
	Palangi et al [128]	0.87	0.89	0.91	0.48	0.54	0.50	0.68	0.65	0.71
HYBRID	Siwei Lai et al [142]	0.88	0.88	0.91	0.60	0.57	0.69	0.70	0.66	0.77
	Chen et al [143]	0.87	0.89	0.86	0.32	0.48	0.47	0.39	0.52	0.44
	Zhou et al [144]	0.88	0.88	0.88	0.43	0.61	0.53	0.53	0.58	0.55
	Wang et al [145]	0.88	0.90	0.90	0.66	0.62	0.50	0.58	0.57	0.56
BERT Multilingual [42]	12-layer, 768-hidden units, 12-heads	0.93	0.85	0.93	**0.77**	0.35	0.77	0.68	0.41	0.70

Peak performance of every classifier is highlighted in bold for each experimental dataset

67% over CLE Urdu Digest 1000K with 2K most frequent features.

On the account of hybrid methodology which feeds most discriminative features ranked by filter-based feature selection algorithm (NDM) [47], the performance of all models gets rocketed. The performance of Yoon Kim et al. CNN-based model [123] jumps from 88% to 93% over DSL Urdu News dataset, whereas Yin et al CNN-based model [125] performance raises from 70% to 78% over CLE Urdu Digest 1000K, and 71% to 80% over CLE Urdu Digest 1M dataset.

On the other hand, pre-trained language model BERT [42] shows decent performance with multilingual vocabulary. BERT [42] tokenizer is based on a WordPiece model which greedily builds a fixed sized vocabulary of most common words, subwords, and individual characters which best fits certain language data. BERT [42] effectively handles out of vocabulary words by generating the embeddings of subword tokens which retain most of the contextual meaning of the words, and individual characters. In this way, BERT [42] learns effective representations for the features present in experimental datasets. Through utilizing multilingual vocabulary, BERT [42] marks the promising performance of 93% over DSL Urdu news dataset, whereas for CLE Urdu Digest 1000K dataset which only consists of 270 documents, on average, BERT [42] only utilizes 20 documents of each class for training, but still it manages to mark the performance figure of 77%. As BERT [42] only supports the sequence length up to 512 due to memory overhead, this is why BERT [42] manages to achieve the limited performance of only 68% over CLE Urdu Digest 1M dataset which is the largest among all experimental datasets with an average document length of around 1100 words. BERT [42] would have achieved better results for CLE Urdu Digest 1M dataset if it had supported sequence length higher than 512. In addition, with the buffer size of 100, BERT [42] only marks the performance of 91%, 70%, and 60% as compared to 93%, 77%, and 68% achieved with the buffer size of 400 over DSL Urdu news, CLE Urdu Digest 1000K, and CLE Urdu Digest 1M datasets respectively.

Contrarily, BERT [42] does not perform well when we replace its multilingual base vocabulary with the vocabulary that only contains most discriminative features ranked by NDM [47]. In addition, its performance further gets deteriorated when a vocabulary of most frequent features is passed during fine-tuning. This is because most of the frequent features are irrelevant and also BERT [42] lacks the embeddings of most of the features. Besides, although BERT [42] with most discriminate features outshines the performance figures of 68% produced by multilingual vocabulary for CLE Urdu Digest 1M dataset; however, according to our observation, it happens only because of the fact that CLE Urdu Digest 1M has greater number of discriminative features which overlap with multilingual vocabulary as compared to CLE Urdu Digest 1M, and DSL Urdu news datasets.

To summarize, BERT [42] proves extremely effective as it almost replicates the performance of hybrid methodology. Hybrid methodology raises the performance of all adopted deep learning models up to great extent. This is because it alleviates the noise from underlay dataset and selects highly discriminative features to feed the models. Evidently, for those datasets where average document length lies within 512 tokens, BERT [42], and hybrid methodology mark similar performance figures, however for those datasets where average document length exceeds from 512 tokens, hybrid methodology outshines the performance of BERT [42].

Furthermore, in order to perform class-level comparison of hybrid methodology and state-of-the-art adopted deep learning methodologies, we present accuracy confusion matrices of both methodologies on the test set of all three datasets. To summarize the evaluation, among all 10 adopted deep learning models, three best performing deep learning models are selected including CNN given by Yin et al. [125], RNN proposed by Palangi et al. [128], and Hybrid model presented by Siwei Lai et al. [142].

Confusion matrices produced by three best performing adopted deep learning methodologies across three datasets are shown in Fig. 4. It can be noted from Fig. 4, regardless of model architecture, all three adopted methodologies are unable to produce promising performance on any dataset as automated feature engineering fails to extract discriminative features which helps the model to differentiate class boundaries. For example, in DSL Urdu news dataset, most of the testing samples of class world are wrongly classified into three classes, namely health-science, entertainment, and sports classes by the adopted models as the models extract less discriminative features, thus get confused to assign correct class labels. Likewise, in CLE Urdu Digest 1M dataset, few samples of almost every class are mistakenly classified into letter, health, culture or press classes, whereas in CLE Urdu Digest 1000K dataset, three similar classes interviews, letters, and press show the most number of samples being wrongly classified in either of classes.

In contrast, hybrid methodology performs way better as it utilizes filter-based feature selection algorithm (NDM) [47]. With the induction of feature selection algorithm (NDM) [47], a vocabulary of most discriminative features is fed to the embedding layer of adopted deep learning models which assists the model to better understand the class boundaries; thus, it raises the performance of adopted deep learning methodologies up to great extent on all three datasets. As summarized by Fig. 5, models are no more confused among highly similar classes such as letter,

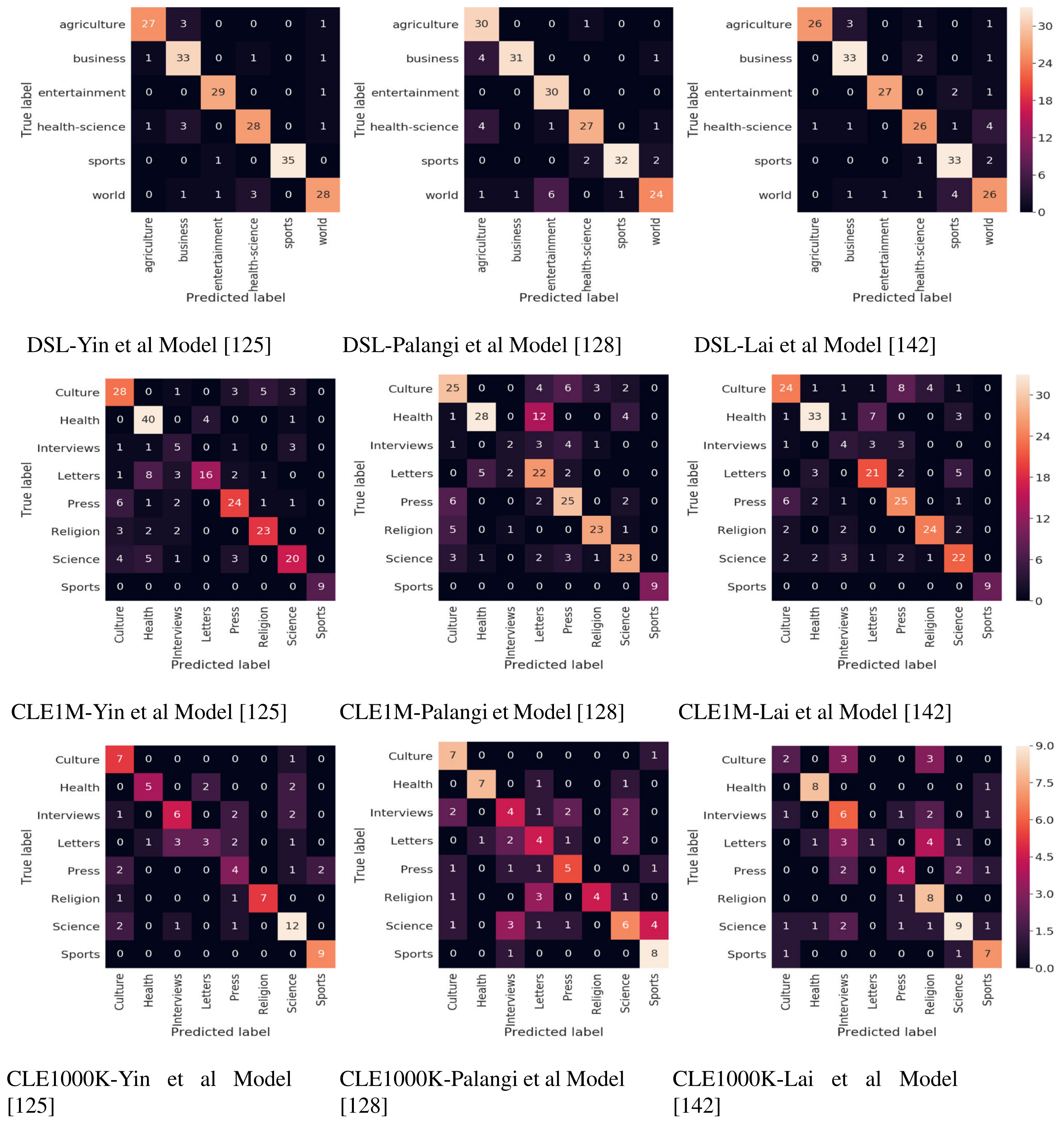
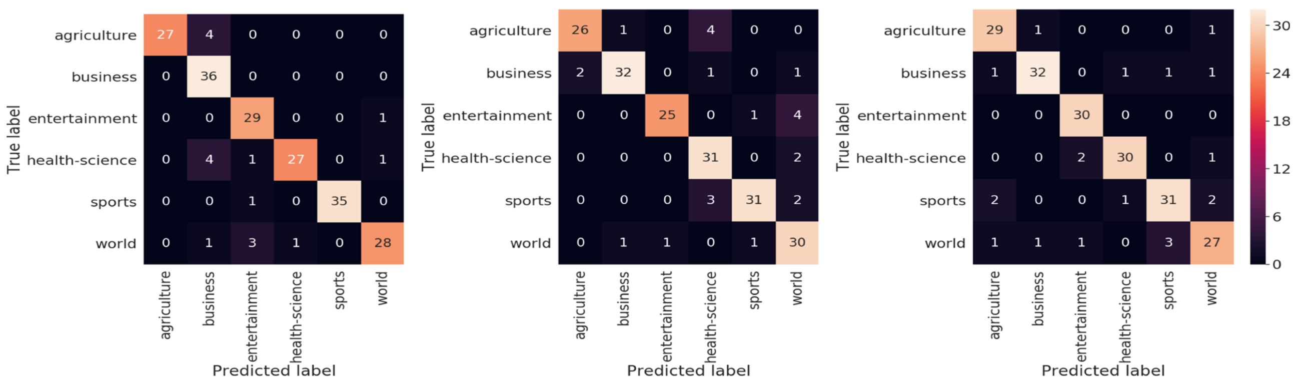


Fig. 4 Confusion matrices of adopted best performing CNN [125], RNN [128], and hybrid [142] methodologies on DSL Urdu news, CLE Urdu Digest 1*M*, and CLE Urdu Digest 1000*K* datasets

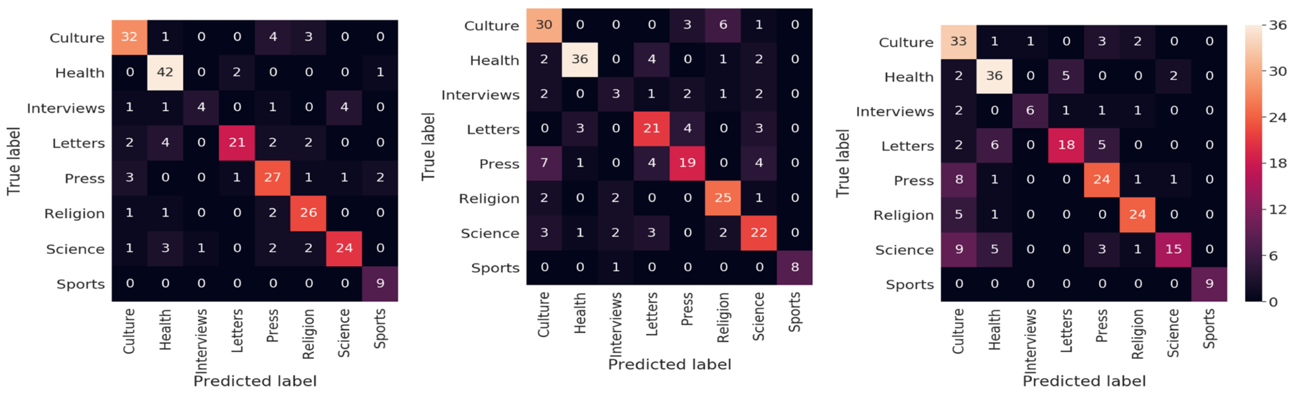
interview, and press in both CLE Urdu Digest 1*M*, 1000*K* datasets.

The general observations drawn while deploying neural network architecture for Urdu text document classification are as follows:

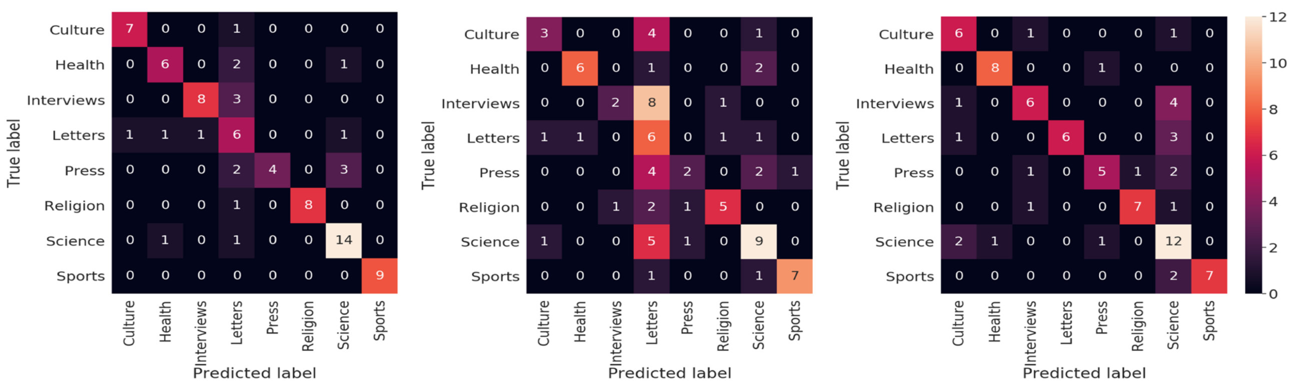
- When vocabulary of unique words is fed to the model, bidirectional LSTM outperformed all other neural architectures.
- When feeding highly discriminative features, convolution-based models are clearly the winners as they perform better than recurrent- and hybrid-based models.



DSL-Yin et al Model [125] DSL-Palangi et al Model [128] DSL-Lai et al Model [142]



CLE1M-Yin et al Model [125] CLE1M-Palangi et al Model [128] CLE1M-Lai et al Model [142]



CLE1000K-Yin et al Model [125] CLE1000K-Palangi et al Model [128] CLE1000K-Lai et al Model [142]

Fig. 5 Confusion matrices of hybrid methodology on DSL Urdu news, CLE Urdu Digest 1*M*, and CLE Urdu Digest 1000*K* datasets

However, in a few scenarios, hybrid models perform similar to CNNs but not better.

- Model with multi-layer CNN architecture with different filter sizes learns better data representation as compared to the model in which CNN layers are linearly stacked over each other.
- According to the experimentation, models give better results when embedding layer is initialized by pre-trained word vectors and get updated during training.
- Implementing wide convolutions increases the performance of models on text document classification as wide convolution equalizes the participation of all features while convolving them.

- For text document classification, the performance of deep learning model increases significantly when the model is fed with deterministic features instead of full vocabulary having all unique terms.
- Max pooling layer plays a significant role to extract discriminative features.
- Using multiple embedding layers with CNN architecture produces better results when the model is fed with deterministic features, while in all other scenarios, there is no significant difference between the performance of models that use single and multiple embedding layers.

8 Conclusion

This paper may be considered a milestone towards Urdu text document classification as it presents a new publicly available dataset (DSL Urdu News), introduces 10 filter-based feature selection algorithms in state-of-the-art machine learning-based Urdu text document classification methodologies, adopts 10 state-of-the-art deep learning methodologies, assesses the effectiveness of transfer learning using BERT, and evaluates the integrity of a hybrid methodology which harvests the benefits of both machine learning-based feature engineering, and deep learning-based automated feature engineering. Experimental results show that in machine leaning-based Urdu text document classification methodology, SVM classifier outperforms Naive Bayes as all feature selection algorithms produce better performance for two datasets (CLE Urdu Digest 1000*k*, 1*M*) with SVM classifier. NDM and CHISQ reveal the promising performance with both classifiers. Among all, GINI shows the worst performance with both classifiers. Furthermore, adopted deep learning methodologies fail to mark a promising performance with trivial automated feature engineering. Although using a vocabulary of most frequent features raises the performance of adopted deep learning methodologies, it fails to obliterate the promising performance figures of hybrid approach. The hybrid methodology has proved extremely versatile and effective with different languages. It substantially outperforms adopted deep learning-based methodologies and almost equalizes the top performance of machine learning methodologies across two datasets (DSL Urdu News, CLE Urdu Digest 1*M*). Similarly, BERT almost mimics the performance of hybrid methodology on account of those datasets where the average document length does not exceed 512 tokens. However, for datasets where average document length exceeds from 512 tokens, hybrid methodology performs better than BERT. Contrarily, for all three datasets, hybrid methodology fails to outshine the peak performance figures produced by machine learning methodology due to the small size of experimental datasets. To illustrate the point, consider the class interviews of CLE Urdu Digest 1M which has only 38 documents, so in this scenario, deep learning-based hybrid methodology only uses 22 documents for training which are not good enough at all. A compelling future line of this work would be the development of a robust neural feature selection algorithm which can assists the models to automatically select highly discriminative features from each class. In addition, comparison of diverse data augmentation approaches, investigating the impact of ensembling feature selection algorithms and assessing whether feeding the discriminative features of different filter-based feature selection algorithm at multiple channels of deep learning models can significantly raise the classification performance for Urdu language

Compliance with ethical standards

Funding Not applicable.

Conflict of interest Corresponding author on the behalf of all authors declares that no conflict of interest is present.

Availability of data and material DSL Urdu news dataset, and other pre-processing resources will be available at Github repository (https://github.com/minixain/Urdu-Text-Classification).

Code availability The source code will be available at Github repository (https://github.com/minixain/Urdu-Text-Classification).

References

1. Mulcahy M (2017) Big data statistics and facts for 2017. https://www.waterfordtechnologies.com/big-data-interesting-facts/. [Online; Accessed 1 Jan 2018]
2. Cave A (2017) What will we do when the world's data hits 163 Zettabytes in 2025. https://www.forbes.com/sites/andrewcave/2017/04/13/what-will-we-do-when-the-worlds-data-hits-163-zettabytes-in-2025/#612b04f8349a/. [Online; Accessed 1 Jan 2018]
3. Marr B (2015) Big data: 20 mind-boggling facts everyone must Read. https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/#301b174517b1/. [Online; Accessed 1 Jan 2018]
4. Idris I, Selamat A, Nguyen NT, Omatu S, Krejcar O, Kuca K, Penhaker M (2015) A combined negative selection algorithm–particle swarm optimization for an email spam detection system. Eng Appl Artif Intell 39:33–44
5. Cheng N, Chandramouli R, Subbalakshmi KP (2011) Author gender identification from text. Digit Investig 8(1):78–88
6. Bhatt A, Patel A, Chheda H, Gawande K (2015) Amazon review classification and sentiment analysis. Int J Comput Sci Inf Technol 6(6):5107–5110
7. Dilrukshi I, De Zoysa K, Caldera A (2013) Twitter news classification using svm. In: 2013 8th International conference on computer science & Education (ICCSE), pp 287–291. IEEE
8. Krishnalal G, Babu RS, Srinivasagan KG (2010) A new text mining approach based on hmm-svm for web news classification. Int J Comput Appl 1(19):98–104

9. Kroha P, Baeza-Yates R (2005) A case study: news classification based on term frequency. In: Sixteenth international workshop on database and expert systems applications, 2005. Proceedings. pp 428–432. IEEE
10. Gahirwal M, Moghe S, Kulkarni T, Khakhar D, Bhatia J (2018) Fake news detection. Int J Adv Res Ideas Innov Technol 4(1):817–819
11. Conroy Niall J, Rubin Victoria L, Chen Y (2015) Automatic deception detection: methods for finding fake news. In: Proceedings of the 78th ASIS&T annual meeting: information science with impact: research in and for the community, pp 82. American Society for Information Science
12. Shu K, Sliva A, Wang S, Tang J, Liu H (2017) Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter 19(1):22–36
13. Akram Q, Naseer A, Hussain S (2009) Assas-band, an affix-exception-list based urdu stemmer. In: Proceedings of the 7th workshop on Asian language resources, pp 40–46. Association for Computational Linguistics
14. Ali AR, Ijaz M (2009) Urdu text classification. In: Proceedings of the 7th international conference on frontiers of information technology, pp 21. ACM
15. Usman M, Shafique Z, Ayub S, Malik K (2016) Urdu text classification using majority voting. Int J Adv Comput Sci Appl 7(8):265–273
16. Ahmed K, Ali M, Khalid S, Kamran M (2016) Framework for urdu news headlines classification. J Appl Comput Sci Math 10(1):17–21
17. Sattar SA, Hina S, Khursheed N, Hamid A (2017) Urdu documents classification using naïve bayes. Indian J Sci Technol 10(29):1–4
18. Tehseen Z, Qaiser A, Muhammad Pervez A (2015) Evaluation of feature selection approaches for urdu text categorization. Int J Intell Syst Appl 7(6):33
19. Hussain S, Adeeba F, Akram Q (2016) Urdu text genre identification. In: Proceedings of conference on language and technology, 2016 (CLT 16), Lahore, Pakistan. CLE,
20. Chen G, Chen J (2015) A novel wrapper method for feature selection and its applications. Neurocomputing 159:219–226
21. Rehman A, Javed K, Babri HA, Asim N (2018) Selection of the most relevant terms based on a max–min ratio metric for text classification. Expert Syst Appl 114:78–96
22. Parlak B, Uysal AK (2016) The impact of feature selection on medical document classification. In: 2016 11th Iberian conference on information systems and technologies (CISTI), pp 1–5. IEEE
23. Prusa JD, Khoshgoftaar TM, Dittman DJ (2015) Impact of feature selection techniques for tweet sentiment classification. In: The Twenty-eighth international flairs conference
24. Alper Kursat Uysal and Serkan Gunal (2014) The impact of preprocessing on text classification. Inf Process Manag 50(1):104–112
25. Weston J, Watkins C (1998) Multi-class support vector machines. Technical report, Citeseer
26. Domingos P, Pazzani M (1997) On the optimality of the simple Bayesian classifier under zero-one loss. Mach Learn 29(2–3):103–130
27. Klenin J, Botov D (2017) Comparison of vector space representations of documents for the task of matching contents of educational course programmes. In: AIST (Supplement), pp 79–90
28. Li H, Caragea D, Li X, Caragea C (2018) Comparison of word embeddings and sentence encodings as generalized representations for crisis tweet classification tasks. en. In: New Zealand, pp 13
29. Almeida F, Xexéo G (2019) Word embeddings: a survey. arXiv preprint arXiv:1901.09069
30. Li Y, Wang X, Pengjian X (2018) Chinese text classification model based on deep learning. Future Internet 10(11):113
31. Kamath CN, Bukhari SS, Dengel A (2018) Comparative study between traditional machine learning and deep learning approaches for text classification. In: Proceedings of the ACM symposium on document engineering 2018, pp 14. ACM
32. Rubio JJ, Pan Y, Lughofer E, Chen M-Y, Qiu J (2020) Fast learning of neural networks with application to big data processes. Neurocomputing 390:294–296
33. José de Jesús Rubio (2009) Sofmls: online self-organizing fuzzy modified least-squares network. IEEE Trans Fuzzy Syst 17(6):1296–1309
34. Jesús Alberto Meda-Campaña (2018) On the estimation and control of nonlinear systems with parametric uncertainties and noisy outputs. IEEE Access 6:31968–31973
35. de José Rubio J, Enrique G, Genaro O, Israel E, David Ricardo C, Ricardo B, Jesus L, Juan Francisco N (2019) Unscented kalman filter for learning of a solar dryer and a greenhouse. J Intell Fuzzy Syst 37(5):6731–6741
36. Haider S (2018) Urdu word embeddings. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC-2018)
37. Yang W, Lu W, Zheng VW (2019) A simple regularization-based algorithm for learning cross-domain word embeddings. arXiv preprint arXiv:1902.00184
38. You S, Ding D, Canini K, Pfeifer J, Gupta M (2017) Deep lattice networks and partial monotonic functions. In: Advances in neural information processing systems, pp 2981–2989
39. Niebler T, Becker M, Pölitz C, Hotho A (2017) Learning semantic relatedness from human feedback using metric learning. arXiv preprint arXiv:1705.07425
40. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. arXiv preprint arXiv:1802.05365
41. Howard J, Ruder S (2018) Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146
42. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
43. You Y, Li J, Hseu J, Song X, Demmel J, Hsieh CJ (2019) Reducing bert pre-training time from 3 days to 76 minutes. arXiv preprint arXiv:1904.00962
44. You Y, Li J, Reddi S, Hseu J, Kumar S, Bhojanapalli S, Song X, Demmel J, Hsieh CJ (2019) Large batch optimization for deep learning: training bert in 76 minutes. arXiv preprint arXiv:1904.00962, 1(5)
45. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V(2019) Roberta: a robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692
46. Asim MN, Khan MUG, Malik MI, Dengel A, Ahmed S (2019) A robust hybrid approach for textual document classification. In: 2019 International conference on document analysis and recognition (ICDAR), pp 1390–1396. IEEE
47. Abdur R, Javid K, Babri HA (2017) Feature selection based on a normalized difference measure for text classification. Inf Process Manag 53(2):473–489
48. Resnik P (1999) Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. J Artif Intell Res 11:95–130
49. Thangaraj M, Sivakami M (2018) Text classification techniques: a literature review. Interdiscip J Inf Knowl Manag 13:117–135
50. Agarwal B, Mittal N (2014) Text classification using machine learning methods-a survey. In: Proceedings of the second

international conference on soft computing for problem solving (SocProS 2012), December 28-30, 2012, pp 701–709. Springer
51. Cai J, Luo J, Wang S, Yang S (2018) Feature selection in machine learning: a new perspective. Neurocomputing 300:70–79
52. Asim MN, Wasim M, Ali MS, Rehman A (2017) Comparison of feature selection methods in text classification on highly skewed datasets. In: First international conference on latest trends in electrical engineering and computing technologies (INTELLECT), 2017 , pp 1–8. IEEE
53. Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes MP, Shyu M-L, Chen S-C, Iyengar SS (2018) A survey on deep learning: algorithms, techniques, and applications. ACM Comput Surveys (CSUR) 51(5):1–3
54. Ruder S, Peters ME, Swayamdipta S, Wolf T (2019) Transfer learning in natural language processing. In: Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: tutorials, pp 15–18
55. Forman G (2003) An extensive empirical study of feature selection metrics for text classification. J Mach Learn Res 3:1289–1305
56. Rehman A, Javed K, Babri HA, Saeed M (2015) Relative discrimination criterion-a novel feature ranking method for text data. Expert Syst Appl 42(7):3670–3681
57. Kent JT (1983) Information gain and a general measure of correlation. Biometrika 70(1):163–173
58. Chen J, Huang H, Tian S, Youli Q (2009) Feature selection for text classification with naïve bayes. Expert Syst Appl 36(3):5432–5435
59. Park H, Kwon S, Kwon HC (2010) Complete gini-index text (git) feature-selection algorithm for text classification. In: The 2nd international conference on software engineering and data mining, pp 366–371. IEEE
60. Gao Y, Wang HL (2009) A feature selection algorithm based on poisson estimates. In: 2009 Sixth international conference on fuzzy systems and knowledge discovery, volume 1, pp 13–18. IEEE
61. Korde V, Mahender CN (2012) Text classification and classifiers: a survey. Int J Artif Intell Appl 3(2):85
62. Zhang T (2004) Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: Proceedings of the twenty-first international conference on Machine learning, p. 116. ACM
63. McCallum A, Nigam K et al (1998) A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization, volume 752, pp 41–48. Citeseer
64. Baoxun X, Guo X, Ye Y, Cheng J (2012) An improved random forest classifier for text categorization. JCP 7(12):2913–2920
65. Karegowda AG, Manjunath AS, Jayaram MA (2010) Comparative study of attribute selection using gain ratio and correlation based feature selection. Int J Inf Technol Knowl Manag 2(2):271–277
66. Yu L, Liu H (2003) Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 856–863
67. Tan S (2005) Neighbor-weighted k-nearest neighbor for unbalanced text corpus. Expert Syst Appl 28(4):667–671
68. Lopez MM, Kalita J (2017) Deep learning applied to nlp. arXiv preprint arXiv:1703.03091
69. Conneau A, Schwenk H, Barrault L, Lecun Y (2016) Very deep convolutional networks for natural language processing. arXiv preprint arXiv:1606.01781, 2
70. Camacho-Collados J, Pilehvar MT (2017) On the role of text preprocessing in neural network architectures: an evaluation study on text categorization and sentiment analysis. arXiv preprint arXiv:1707.01780
71. Ayedh A, Tan G, Alwesabi K, Rajeh H (2016) The effect of preprocessing on arabic document categorization. Algorithms 9(2):27
72. Malaviya C, Wu S, Cotterell R (2019) A simple joint model for improved contextual neural lemmatization. arXiv preprint arXiv:1904.02306
73. Yulia L (2008) Effect of preprocessing on extractive summarization with maximal frequent sequences. In: Mexican international conference on artificial intelligence, pp 123–132. Springer, 2008
74. Danisman T, Alpkocak A (2008) Feeler: emotion classification of text using vector space model. In: AISB 2008 convention communication, interaction and social intelligence, volume 1, pp 53
75. Sharma D, Cse M (2012) Stemming algorithms: a comparative study and their analysis. Int J Appl Inf Syst 4(3):7–12
76. Kanhirangat V, Gupta D (2016) A study on extrinsic text plagiarism detection techniques and tools. J Eng Sci Technol Rev 9(150–164):10
77. Latha K (2010) A dynamic feature selection method for document ranking with relevance feedback approach. ICTACT J Soft Comput 7(1):1–8
78. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1–2):273–324
79. Lal TN, Chapelle O, Weston J, Elisseeff A (2006) Embedded methods. In: Feature extraction, pp 137–165. Springer
80. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
81. Dash M, Liu H (1997) Feature selection for classification. Intell Data Anal 1(1–4):131–156
82. Ogura H, Amano H, Kondo M (2011) Comparison of metrics for feature selection in imbalanced text classification. Expert Syst Appl 38(5):4978–4989
83. Ogura H, Amano H, Kondo M (2009) Feature selection with a measure of deviations from poisson in text categorization. Expert Syst Appl 36(3):6826–6832
84. Devasena CL, Sumathi T, Gomathi VV, Hemalatha M (2011) Effectiveness evaluation of rule based classifiers for the classification of iris data set. Bonfring Int J Man Mach Interface 1:05–09
85. Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inf Process Manag 24(5):513–523
86. Ranjana A, Madhura P (2013) A novel algorithm for automatic document clustering. In: Advance computing conference (IACC), 2013 IEEE 3rd International, pp 877–882. IEEE
87. Choi D Kim P (2012) Automatic image annotation using semantic text analysis. In: International conference on availability, reliability, and security, pp 479–487. Springer
88. Huang C, Tianjun F, Chen H (2010) Text-based video content classification for online video-sharing sites. J Am Soc Inform Sci Technol 61(5):891–906
89. Tang B, He H, Baggenstoss PM, Kay S (2016) A bayesian classification approach using class-specific features for text categorization. IEEE Trans Knowl Data Eng 28(6):1602–1606
90. Rusland NF, Wahid N, Kasim S, Hafit H (2017) Analysis of naïve bayes algorithm for email spam filtering across multiple datasets. In: IOP conference series: materials science and engineering, volume 226, p. 012091. IOP Publishing
91. Watkins CJCH (1989) Learning from delayed rewards. Ph. D. thesis, King's College, Cambridge
92. Chitrakar R, Chuanhe H (2012) Anomaly detection using support vector machine classification with k-medoids clustering. In: 2012 Third Asian himalayas international conference on internet, pp 1–5. IEEE

93. Tripathy A, Agrawal A, Rath SK (2016) Classification of sentiment reviews using n-gram machine learning approach. Expert Syst Appl 57:117–126
94. Bouvrie J (2006) Notes on convolutional neural networks. http://cogprints.org/5869/
95. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press, Cambridge
96. Lee CY, Gallagher PW, Tu Z (2016) Generalizing pooling functions in convolutional neural networks: mixed, gated, and tree. In: Artificial intelligence and statistics, pp 464–472
97. Ranzato M, Huang FJ, Boureau YL, LeCun Y (2007) Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: 2007 IEEE conference on computer vision and pattern recognition, pp 1–8. IEEE
98. Scherer D, Müller A, Behnke S (2010) Evaluation of pooling operations in convolutional architectures for object recognition. In: International conference on artificial neural networks, pp 92–101. Springer
99. Wang T, Wu DJ, Coates A , Ng AY (2012) End-to-end text recognition with convolutional neural networks. In: Proceedings of the 21st international conference on pattern recognition (ICPR2012), pp 3304–3308. IEEE
100. LeCun YA, Bottou L, Orr GB, Müller KR (2012) Efficient backprop. In: Neural networks: tricks of the trade, pp 9–48. Springer
101. Xu B, Wang N, Chen T, Li M (2015) Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853
102. Ramachandran P, Zoph B, Le QV (2017) Swish: a self-gated activation function. arXiv preprint arXiv:1710.05941, 7
103. Jiuxiang G, Liu T, Wang X, Wang G, Cai J, Chen T (2018) Recent advances in convolutional neural networks. Pattern Recognit 77:354–377
104. Hochreiter S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int J Uncertain Fuzziness Knowl-Based Syst 6(02):107–116
105. Nwankpa C, Ijomah W, Gachagan A, Marshall S (2018) Activation functions: comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378
106. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167
107. Hinton GE, Nitish S, Alex K, Ilya S, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580
108. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
109. Lin M, Chen Q, Yan S (2013) Network in network. arXiv preprint arXiv:1312.4400
110. Rawat W, Wang Z (2017) Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput 29(9):2352–2449
111. Sutskever I, Martens J, Hinton GE (2011) Generating text with recurrent neural networks. In: ICML
112. Mandic D, Chambers J (2001) Recurrent neural networks for prediction: learning algorithms, architectures and stability. Wiley, Hoboken
113. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Networks 5(2):157–166
114. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: International conference on machine learning, pp 1310–1318
115. Junyoung C, Caglar G, KyungHyun C, Yoshua B (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555
116. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078
117. Conn AR, Scheinberg K , Vicente LN (2009) Introduction to derivative-free optimization. SIAM
118. Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Global Optim 13(4):455–492
119. Villemonteix J, Vazquez E, Walter E (2009) An informational approach to the global optimization of expensive-to-evaluate functions. J Global Optim 44(4):509
120. Beyer H-G (2001) The theory of evolution strategies. Springer, Berlin
121. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13(1):281–305
122. Yin W, Kann K, Yu M, Schütze H (2017) Comparative study of cnn and rnn for natural language processing. arXiv preprint arXiv:1702.01923
123. Yoon K (2014) Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882
124. Nal K, Edward G, Phil B (2014) A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188
125. Yin W, Schütze H (2016) Multichannel variable-size convolution for sentence classification. arXiv preprint arXiv:1603.04513
126. Zhang Y, Roller S, Wallace B (2016) Mgnc-cnn: a simple approach to exploiting multiple word embeddings for sentence classification. arXiv preprint arXiv:1603.00968
127. Yogatama D, Dyer C, Ling W, Blunsom P (2017) Generative and discriminative text classification with recurrent neural networks. arXiv preprint arXiv:1703.01898
128. Palangi H, Deng L, Shen Y, Gao J, He X, Chen J, Song X, Ward R (2016) Deep sentence embedding using long short-term memory networks: analysis and application to information retrieval. IEEE/ACM Trans Audio Speech Language Process 24(4):694–707
129. Vu NT, Adel H, Gupta P, Schütze H (2016) Combining recurrent and convolutional neural networks for relation classification. arXiv preprint arXiv:1605.07333
130. Wen Y, Zhang W, Luo R, Wang J (2016) Learning text representation using recurrent convolutional neural network with highway layers. arXiv preprint arXiv:1606.06905,
131. Wang J, Yu LC, Lai KR, Zhang X (2016) Dimensional sentiment analysis using a regional cnn-lstm model. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 2: Short Papers), pp 225–230
132. Chen T, Ruifeng X, He Y, Wang X (2017) Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. Expert Syst Appl 72:221–230
133. Dauphin YN, Fan A, Auli M, Grangier D (2017) Language modeling with gated convolutional networks. In: Proceedings of the 34th international conference on machine learning-volume 70, pp 933–941. JMLR. org
134. Adel H, Schütze H (2016) Exploring different dimensions of attention for uncertainty detection. arXiv preprint arXiv:1612.06549
135. Hoffmann J, Navarro O, Kastner F, Janßen B, Hubner M (2017) A survey on cnn and rnn implementations. In: PESARO 2017: the seventh international conference on performance, safety and robustness in complex systems and applications
136. Tang D, Qin B, Liu T (2015) Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 1422–1432
137. Hermanto A, Adji TB, Setiawan NA(2015) Recurrent neural network language model for english-indonesian machine

translation: Experimental study. In: 2015 International conference on science in information technology (ICSITech), pp 132–136. IEEE, 2015
138. Messina R, Louradour J (2015) Segmentation-free handwritten chinese text recognition with lstm-rnn. In: 2015 13th International conference on document analysis and recognition (icdar), pp 171–175. IEEE, 2015
139. Sundermeyer M, Ney H, Schlüter R (2015) From feedforward to recurrent lstm neural networks for language modeling. IEEE/ACM Trans Audio Speech Language Process 23(3):517–529
140. Takase S, Suzuki J, Nagata M (2019) Character n-gram embeddings to improve rnn language models. arXiv preprint arXiv:1906.05506
141. Viswanathan S, Kumar MA, Soman KP (2019) A sequence-based machine comprehension modeling using lstm and gru. In: Emerging research in electronics, computer science and technology, pp 47–55. Springer
142. Lai S, Liheng X, Liu K, Zhao J (2015) Recurrent convolutional neural networks for text classification. AAAI 333:2267–2273
143. Chen G, Ye D, Xing Z, Chen J, Cambria E (2017) Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In: 2017 International joint conference on neural networks (IJCNN), pp 2377–2383. IEEE
144. Zhou C, Sun C, Liu Z, Lau F (2015) A c-lstm neural network for text classification. arXiv preprint arXiv:1511.08630
145. Wang X, Jiang W, Luo Z (2016) Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In: Proceedings of COLING 2016, the 26th international conference on computational linguistics: technical papers, pp 2428–2437
146. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
147. Bojanowski P, Grave E, Joulin A, Mikolov T (2016) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146
148. Jeffrey P, Richard S, Christopher M (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543, Doha, Qatar, October (2014). Association for Computational Linguistics
149. Ayinde BO, Inanc T, Zurada JM (2019) On correlation of features extracted by deep neural networks. arXiv preprint arXiv:1901.10900
150. Bigi B (2003) Using kullback-leibler distance for text categorization. In: European conference on information retrieval, pp 305–319. Springer
151. Stehlík M, Ruiz MP, Stehlíková S, Lu Y (2020) On equidistant designs, symmetries and their violations in multivariate models. In: Contemporary experimental design, multivariate analysis and data mining, pp 217–225. Springer
152. Tang J, Alelyani S, Liu H (2014) Feature selection for classification: a review. Data Classification: Algorithms and Applications, pp 37