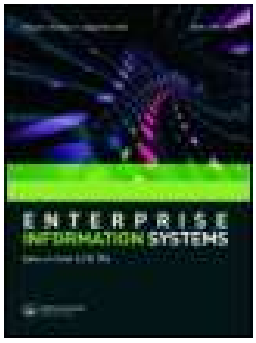

Enterprise Information Systems

ISSN: 1751-7575 (Print) 1751-7583 (Online) Journal homepage: https://www.tandfonline.com/loi/teis20

Exploring deep learning approaches for Urdu text classification in product manufacturing

Muhammad Pervez Akhter, Zheng Jiangbin, Irfan Raza Naqvi, Mohammed Abdelmajeed & Muhammad Fayyaz

To cite this article: Muhammad Pervez Akhter, Zheng Jiangbin, Irfan Raza Naqvi, Mohammed Abdelmajeed & Muhammad Fayyaz (2020): Exploring deep learning approaches for Urdu text classification in product manufacturing, Enterprise Information Systems, DOI: 10.1080/17517575.2020.1755455

To link to this article: https://doi.org/10.1080/17517575.2020.1755455

Published online: 05 May 2020.

ENTERPRISE INFORMATION SYSTEMS
https://doi.org/10.1080/17517575.2020.1755455

Taylor & Francis
Taylor & Francis Group

Exploring deep learning approaches for Urdu text classification in product manufacturing

Muhammad Pervez Akhter[a], Zheng Jiangbin[a], Irfan Raza Naqvi[a], Mohammed Abdelmajeed[b] and Muhammad Fayyaz[c]

[a]School of Software and Microelectronics, Northwestern Polytechnical University, Xian, P.R. China; [b]School of Computer Science and Technology, Northwestern Polytechnical University, Xian, P.R. China; [c]Department of Computer Science, COMSATS University Islamabad, Wah Campus, Wah Cantt, Pakistan

ABSTRACT

From last decade, machine learning (ML) techniques have been used for Urdu text processing. Due to lack of language resources, potential of deep learning (DL) models have not been exploited yet for Urdu text document classification. A text document has more noise, redundant information, and large vocabulary than short text like tweets. This study is the systematic comparison of four well-known DL models. We also compare DL models with four ML models. We also explore the various text preprocessing techniques. Experimental results show that CNN outperforms the others. Further, single-layer architecture of LSTM and BiLSTM performs better than multiple-layers architecture.

ARTICLE HISTORY
Received 8 November 2019
Accepted 5 March 2020

KEYWORDS
Text classification; deep learning; convolutional neural network; long short-term memory; text mining; machine learning

1. Introduction

The tremendous growth of Urdu text documents on the internet is creating challenges for researchers to find an automatic, reliable and fast way to organise these documents. Text document classification is a task of automatically assigning a label from a set of predefined labels to a document based on its contents. Text document classification has several applications in text mining and information retrieval like spam detection (Akhtar, Tahir, and Shakeel 2017; Jain, Sharma, and Agarwal 2018), tweet analysis (Ali et al. 2018), sentiment analysis (Mehmood, Essam, and Shafi 2019), document organisations (Tripathy, Anand, and Rath 2017; Rao et al. 2018). Urdu is a national language of Pakistan and has more than 300 million speakers all over the world (Riaz 2012) but it is a resource-poor language. The rich and complex morphological script, no capitalisation of characters, has diacritics, free word order, context-sensitive are some main characteristics of Urdu that make it more challenging for automatic text processing.

Manual classification of long text documents needs a lot of effort, time, and labour to analyse large data. As compared to a long text document, short text like tweets, reviews have only a few words, less noisy and semantically belong to a single topic. Because of a large vocabulary, more redundant features, multiple topics, and more noise, automatic text classification at the document-level is more challenging than short text classification

CONTACT Muhammad Pervez Akhter pervezbcs@gmail.com School of Software and Microelectronics, Northwestern Polytechnical University, Xian 710072, P.R. China

(Tripathy, Anand, and Rath 2017; Rao et al. 2018). A brief comparison of long text and short text processing is given in Table 1.

A raw dataset may contain hundreds or thousands of documents. A raw dataset must be preprocessed before given as input to an automated classifier because (1) it contains noise and irrelevant data, (2) classifier takes a long time to train, and (3) it degrades the performance of a classifier. Preprocessing operations like tokenised text, eliminate none-language characters, remove stopwords, and stemming are performed on raw data. For Urdu text processing, usually most frequent words (stopwords) are removed but infrequent words (rare words) are not removed from data. Different studies of text processing of other languages have shown that removing rare words in preprocessing has a confounding effect (Aggarwal 2018; Katnoria, Singh, and Kumar 2017; Song, Huang, and Ruan 2018; Ayedh et al. 2016). In this study, we investigate the performance of deep learning (DL) models after removing stopwords, and both stopwords with rare words from Urdu text.

Beyond traditional machine learning (ML) models, deep learning (DL) models are becoming popular with unique structures and the ability to learn complex features from high dimensional data. A major challenge for ML models is to reduce the dimensionality of a high dimensional feature space that may affect the performance of a model. Hand-crafted or rule-based methods of ML for dimensionality reduction are complex, time-consuming, do not perform well and, may have inconsistent results for large and complex data (Khan and Adnan 2018; Altınel and Ganiz 2018). All the DL models have various layered structures of input, hidden and output layers. Hidden layers help to extract high-level features automatically from a dataset. DL models can learn complex features automatically, process large dataset and are faster because these models use a graphics processing unit (GPU) for processing (Krig 2016).

Smart manufacturing is a networked and service-oriented paradigm that integrate many advanced technologies (e.g. IoTs, cloud computing, AI, NLP, information modelling) where all the manufacturing resources, products, services, and processes are intelligent. Therefore, a large amount of unstructured or semi-structured data are produced from heterogeneous resources (e.g. sensors, IoTs, feedback) along the whole product lifecycle. Data are the key resource for decision making by manufacturers to improve product quality and services. Along with wasting organisational resources, unstructured or poor quality data is less reliable and have little potential to make a correct decision. ML (Sharp, Ak, and Hedberg 2018) and DL models (Wang et al. 2018) are being widely used to uncover the hidden patterns, exploit the real potential of data that help in decision making during the product lifecycle. The proposed framework of (Ren et al. 2019)

Table 1. Comparison of long text vs. short text processing.

Long Text (news article, web page)	Short text (comments, tweets)
May contain hundreds or thousands of words	Only a few words
Large vocabulary size	Small vocabulary size
Many irrelevant words in the vocabulary	Few irrelevant words in the vocabulary
Multiple sentences or paragraphs within a document may semantically belong to multiple topics or classes	A single sentence that usually belongs to a single topic
More noisy text like non-language words, URLs, acronyms, redundant words	Less noisy text
Computationally inefficient. Training time and testing time are large.	Computationally efficient. Training time and testing time are usually small.
Require more hardware resources like RAM and GPU.	Require fewer hardware resources

shows that customer demand and customer feedback are two important sources of data in two of the four stages: intelligent design and intelligent maintenance and services. Sentiments, reviews, feedbacks, and public articles about the product are collected and analysed by the industry leaders and manufacturers and use it in intelligent decision making for product design, maintenance and services. Because the customers are not from the same region, therefore, their feedback and reviews can be in different languages. Text processing and text classification using ML and DL methods play an important role in decision making in product manufacturing. For example, in intelligent manufacturing, to decide about what kind of tablet attributes are desired by a specific user can be known from customer online reviews (Li, Nahar, and Fung 2015). Similarly, some manufacturers invite external stakeholders to submit ideas for innovations or to collaborate on product-service system (PSS) development. Automatic text classification methods can be used to classify and extract valuable ideas (very strong, strong, week) from a large number of submitted ideas and thereby, the open innovation of PSS development can be achieved (Ren et al. 2019).

A general process of TC using DL models is given in Figure 1. After pre-processing, documents are converted into a real-valued vector (also called word embedding). These vectors are feed into DL models. Hidden layers perform feature extraction and dimensionality reduction of feature space. One of the hidden layers (softmax layer) calculates the probability distribution over the document labels and the label with high probability is the final label of input document (Rao et al. 2018). Convolutional neural network (CNN) and recurrent neural network (RNN) are two mainstream models of deep learning. The various variants of CNN and RNN like long short-term memory (LSTM), bi-directional long short-term memory (BiLSTM) and convolutional long short-term memory (CLSTM) are widely used in text classification.

CNN extracts high-level features using convolving filters in the convolutional layer. Finding the number of filters and the appropriate size of a filter for a specific task is very difficult. Filters of large size consume more resources in training while small filters may provide inaccurate results (Amajd, Kaimuldenov, and Voronkov 2017). CNN can capture contextual information but it can't capture long-term dependencies among words. It also cannot relate current information with past information during learning. LSTM is a chain-like model that can memorise and relate current information with past information using its memory cells and it can effectively model sequential data (Jain, Sharma, and Agarwal 2018). LSTM can only relate current information in one direction (past or future). BiLSTM model can relate current information in both directions (past and

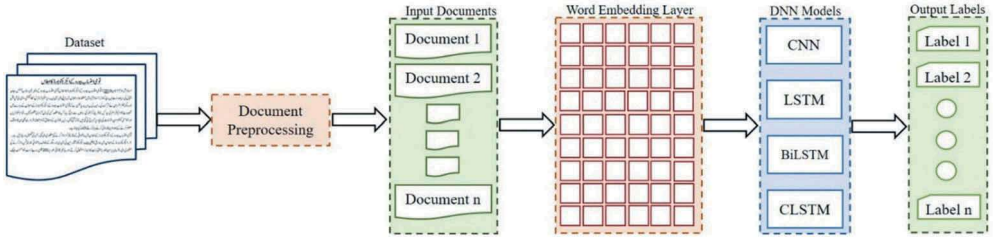

Figure 1. A general process of text classification using deep learning models.

future). It preserves both future and past context to predict the present context. This feature makes BiLSTM more effective for text processing (Liu and Guo 2019). To solve a complex problem, DL models can also be combined to design a complex model to incorporate the features of multiple models. CNN is good at extracting n-gram features and can learn various level dependencies. LSTM can capture long-term dependencies and solve the gradient vanishing problem. CLSTM is a sequential model of CNN and LSTM models. High-level features are first extracted by CNN and then are fed into LSTM to learn long-term dependencies among them (Kim and Cho 2018). Further, the single-layer architecture of LSTM and BiLSTM are popular in text processing but architectures of two-layers and three-layers have been used to capture longer-term dependencies among words (Rao et al. 2018).

From the last two decades, ML methods have been widely used for Urdu text classification but the potential of deep learning models has not been exploited yet. To the best of our knowledge, there is no systematic comparison of DL models on long text document classification of Urdu language. Because of reliable and superior performance, DL models have attracted the attention of the researchers from every field of science (Ahmad, Farman, and Jan 2019). DL models require a large training dataset for better performance but unfortunately, publically available datasets of Urdu text documents are small in size and are not suitable for classification using DL models (see Table 2). Text document classification using DL models and comparatively analyse their performance on various size datasets are the main gaps in Urdu text processing. To fill these gaps, in this study, we evaluate well-known deep learning models and compared their performance on three datasets of small, medium and large size. We also address different issues of text preprocessing methods to reduce feature set size and to improve the performance of DL models like stopwords, rare words, dataset size, and their effects on DL models.

In this study, our goal is to combine different experiments in order to determine the best model for Urdu text classification. Following are our contributions:

(1) To the best of our knowledge, it is the first study of Urdu text classification at document-level using deep learning models
(2) We perform a systematic comparison of CNN and RNN based models (LSTM and BiLSTM) in the context of long document classification. To make the comparison in a more realistic way, we also evaluate RNN models at a more deeper level (single vs. multiple LSTM layers) and CNN using multiple filters of variable-size
(3) We evaluate the performance of these models on multiple benchmark datasets of the small, medium, and large size.
(4) We apply various preprocessing methods on these datasets and explore the effects of these methods on the performance of deep learning models

Table 2. Examples of words, ligatures, and characters of Urdu language.

Urdu Words	Urdu Ligatures	Urdu Characters
پاکستان(*pakistan*, Pakistan)	پا, کستا, ن	پ, ا, ک, س, ت, ا, ن
سکول(*sakool*, School)	سکو, ل	س, ک, و, ل
کتاب(*kitab*, Book)	کتا, ب	ک, ت, ا, ب

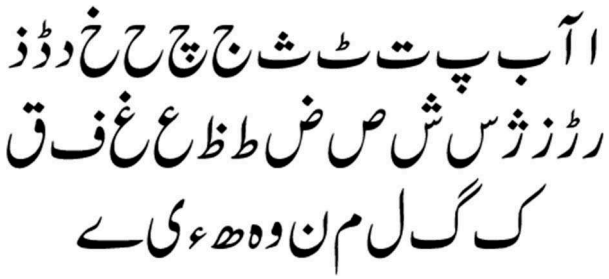

Figure 2. The basic character set of Urdu.

(5) We compared the performance of deep learning models with well-known machine learning models and found that deep learning models outperform the machine learning models

The organisation of this paper is as follows: section 2 contains a brief introduction and features of Urdu, section 3 represents a literature review. Section 4 represents the design and features of deep learning models. Datasets used for experiments are compared in section 5. Section 6 includes results and discussions. Section 7 has a conclusion and future work.

2. Urdu language and its features

Urdu is a national language of Pakistan. It is one of the major languages of South Asia and the 21st most spoken language in the world. Urdu has 38 basic characters shown in Figure 2. Urdu is written in Nastalique style that is a very complex and context-sensitive style. A word of Urdu is a combination of ligatures and a ligature is composed of a single or several characters. Words are joined from right-to-left order to form a sentence. Table 2 shows the composition of four words, their ligatures, and characters separated by a comma for understanding.

Features of Urdu language make it more complex for automated text processing as compared to other languages. Some of the important features are discussed below:

- No capitalisation: unlike English, there are no uppercase or lowercase characters in Urdu. Therefore, it is difficult to identify proper nouns and start of a sentence in Urdu using some character. For example, in 'وہ پاکستان میں رہتا ہے۔' (wo Pakistan me rehta ha, He lives in Pakistan) sentence, nouns and the start of a sentence cannot be identified.
- Right-to-left: the direction of Urdu writing is from right to left. For example, 'اردو' 'پاکستان کی قومی زبان ہے۔' is the first word and 'ہے۔' is the last word of a sentence.
- Diacritics: like Arabic, Urdu text may have few diacritics. Zer (ِ) meaning 'under', zabar (ً) meaning 'over' and pesh (ُ) meaning 'in front' are three common diacritics. Examples are: گنا (Ganna, sugarcane) and گنا (Gunna, number of times); بیکری (Bekri, daily transaction) and بکری (Bakri, goat).
- Free word order: order of words in a sentence of Urdu may be different but the meaning would be the same. For example, in 'علی نے بازار سے نئی کتاب خریدی۔'

(ali ne bazar se nai kitab kharidi) or 'علی نے نئی کتاب بازار سے خریدی' (ali ne nai kitab bazar se kharidi) both sentences have different words order but the meaning is same as 'Ali bought a new book from the market'.

- Subject-Object-Verb (SOV): unlike English, the sentence structure of Urdu is in subject-object-verb order. For example, 'جنید سکول گیا' (junaid school gia, Junaid went to school). جنید (junaid, Junaid) is a subject. سکول (sakool, school) is an object. گیا (giya, went) is a verb.
- Context Sensitivity: Urdu is a context-sensitive language. A character may have different shapes when joined with other characters to form a ligature or a word. For example, the character 'ت', when joined with 'ا', it is 'ت+ ا) 'تا). Similarly, when joined with 'ب', it is 'ت+ ب) 'تب).

Because of the lack of language resources, Urdu is known as a resource-poor language. For automatic processing of these resource-poor languages, followings are the major issues of Urdu text processing using deep learning models.

(1) Deep learning models require a large amount of training data which is not available for low-resource languages like Urdu (Kapočiūtė-Dzikienė, Damaševičius, and Woźniak. 2019)
(2) Pre-trained word embeddings like word2vec which are usually trained in an unsupervised manner on a large corpus. Until recently, these pre-trained models are available for the English language, and thus of limited usability for resource-poor languages (Balodis and Deksne 2019)
(3) The rich and complex morphological script, no capitalisation of characters, has diacritics, free word order, context-sensitive are some main characteristics of Urdu that make it more challenging for automatic text processing using deep learning methods

3. Literature review

For document-level classification of Urdu language, online news blogs are the only source of text data collection and dataset design. (Sharjeel, Nawab, and Rayson 2017) collected 1200 news articles from blogs and use that dataset for text reuse. (Basit et al. 2017) used news articles for similarity analysis of text documents. Tehseen used two datasets of news articles for the comparisons of different ML classifiers (Tehseen, Akhter, and Abbas 2015). Similarly, Malik also used news blogs for document classification of Urdu news blog articles (Usman et al. 2016). News blogs have been widely used in text processing of other languages (Wongso et al. 2017; Kılınç et al. 2015; Al-Radaideh and Al-Abrat 2019). In this study, we also use the three datasets of news articles collected from online news blogs for text classification.

In the past, most of the automated processing of Urdu text was carried out using ML approaches. (Mukhtar, Khan, and Chiragh 2017) performed sentiment classification using five-well known classifiers. (Tehseen, Akhter, and Abbas 2015) used five classifiers to comparatively analyse the performance on two datasets. (Akhtar, Tahir, and Shakeel 2017) classified emails into spam or ham using four ML classifiers. (Usman et al. 2016) used the majority voting technique to classify news articles. All the above-cited studies

used ML for classification but the potential of DL methods has not been explored at the classification of Urdu text documents.

The roman script of Urdu is different from the Nastaleeq script of Urdu. It is updated and the most popular way to express reviews, comments, feedbacks and communication on social media. It is written in English characters as English sentences. (Mehmood, Essam, and Shafi 2019) used unigram and bigram techniques and applied ML classifiers for sentiment classification. (Mehmood et al. 2019) applied discriminative feature spamming technique (DFST) on customer's reviews for sentiment classification. (Mukhtar and Khan 2017) classified roman Urdu sentiments using supervised machine learning methods. All the above studies presented used ML methods while DL methods have not been explored.

As compared to both Roman Urdu and Urdu text, very few studies have been performed for automatic processing of handwritten or printed Urdu text using DL methods. (Jain, Mathew, and Jawahar 2017) used CNN, BLSTM and CNN models for optical character recognition (OCR) on a dataset of 1500 scanned pages. (Ahmed et al. 2017) used a handwritten dataset at the page level to identify handwritten text using the BiLSTM model. (Ahmad et al. 2018) employed BLSTM and gated BLSTM for ligature recognition from sentences of Nastaleeq font of Urdu.

Yin performed a systematic comparison of CNN and RNN for question-answering, part of speech tagging and text classification tasks (Yin et al. 2017). RNN performed better than CNN on the classification task. Bassem compared three deep learning models on the author profiling task for Arabic text where LSTM performed better than CNN (Bassem and Zrigui 2020). For abusive content detection on Twitter, a comparative study of Lee showed that RNN performed better than CNN (Lee, Yoon, and Jung 2018).

In the above studies of Urdu, in pre-processing, stopwords are usually removed but the rare words are not removed from the text documents. Different studies have shown that removing both stopwords and rare words have a significant impact on the performance of a classifier like Arabic (Ayedh et al. 2016), Turkish (Çağataylı and Çelebi 2015) and English (Song, Huang, and Ruan 2018; Aggarwal 2018). In this study, we evaluate the performance of DL models on three datasets with and without removing stopwords, rare words and both form text documents.

It can be concluded from the studies mentioned above that there is a major gap in Urdu text processing using DL techniques. The major obstacle to fill this gap is the lack of language resources like datasets. The datasets used in the past studies are given in Table 3. The datasets which are publically available are not well suited for DL models because of their small size. A summary of the literature reviewed and related to this study is given in Table 4.

4. Deep learning models

Neural networks were introduced as a machine learning model in 2006 and attracted the researchers from every field because of its superior and reliable performance on large and complex datasets. Deep neural networks have shown state-of-the-art performance in image recognition, speech recognition, natural language processing, and many other fields. There are two mainstream architectures of DL models: CNN and RNN. CNN has shown superior performance at learning non-sequential data than time-series data. RNNs have a special structure that helps it to learn time series data and can retain information

Table 3. Datasets of Urdu text documents used in past few studies for classification.

Name	|classes|	|Docs|	Availability	Reference
COUNTER	05	1,200	Yes	(Sharjeel, Nawab, and Rayson 2017)
naïve	04	5,003	No	(Tehseen, Akhter, and Abbas 2015)
Ali TC dataset	06	26,067	No	(Ali and Ijaz 2009)
Malik TC dataset	07	21,769	No	(Usman et al. 2016)
EMILLE	06	46	Yes	European Language Resource Association

for a long period of time. In this section, we discuss four mostly used deep neural network-based models: CNN, LSTM, CLSTM, and BiLSMT.

4.1. Convolutional neural network (CNN)

CNN can automatically learn effective text features representation from massive text using a 1D structure (word order) in the convolutional layer. It captures local relationships among the neighbour words in terms of context windows and by using pooling layers it extracts global features. The model used in this study is shown in Figure 3. A CNN model with multiple convolutional filters of various sizes (window size) to extract variable-length features (n-grams) from the text. It performs better than a model that has a convolutional layer with multiple filters of the same size. It also helps to identify short and long-range relations in short and long documents using pooling layers (Zhou et al. 2015; Kim 2014).

Embedding layer represents the words of a document as a word vector by a lookup table. The length of this vector is taken as hyper-parameter. Let x is the input document having L words and d is the length of word vector. Therefore, $x \in R^{L\times d}$represent the input document. Let $x_i \in R^d$ be the k-dimensional word vector for the i-th word in the document. A document of length n (padded where necessary) is represented as

$$x_{1:n} = x_1 + x_2 + \ \ldots \ + x_n \tag{1}$$

where $+$ denotes concatenation operation. A filter f of length k (number of words) is chosen, where vector $n \in R^{k\times d}$represents a filter for the convolution function. For the words starting at position j till the position $(j + k - 1)$ will be processed by the filter at a time. The window w_i is denoted by

$$w_i = [x_i + x_{j+1} + \ \ldots \ + x_{j+k-1}] \tag{2}$$

Let $w \in R^{hk}$ is a convolving filter involve in convolutional operation to create a new context local feature vector from a window of h words. The number of filters can range from 1 to a few hundred depending on the text. A new feature c_i from a window of words $x_{i:i+h-1}$can be calculated as

$$c_i = f(w.x_{i:i+h-1} + \mathrm{b}) \tag{3}$$

where f is a non-linear function like sigmoid, $\cdot$ is an element-wise multiplication and $b \in R$ is a biased term. A feature map c is a set of all the extracted features of all the possible windows of words and calculated as

$$c = [c_1, c_2, \ldots, c_{n-h+1}] \tag{4}$$

Table 4. Overview of studies on text classification of Urdu and other resource-poor.

Study	Approach	Domain	Language	Dataset
(Sharjeel, Nawab, and Rayson 2017)	Naïve Bayes, N-grams	Document Text reuse	Urdu	COUNTER – 1200 news articles distributed in five classes
(Tehseen, Akhter, and Abbas 2015)	NB, KNN, SVM, DT	Document Classification	Urdu	Naïve – (5003 documents distributed into four classes
(Basit et al. 2017)	Latent Semantic Analysis	Semantic Similarity Analysis	Urdu	Overall 27 documents
(Usman et al. 2016)	SGD, NB, Linear SVC, Random Forest	Document Classification	Urdu	21,769 documents in seven classes
(Wongso et al. 2017)	Multinomial NB, SVM	Document Classification	Indonesian	5,000 documents distributed into five classes
(Kılınç et al. 2015)	NB, SVM, J48, K-NN, Random Forest	Document Classification	Turkish	3,600 documents and six categories
(Al-Radaideh and Al-Abrat 2019)	K-NN and Decision Tree	Document Classification	Arabic	Total 2,700 documents distributed into nine classes
(Mukhtar, Khan, and Chiragh 2017)	PART, Multinomial NB, SVM, DT, K-NN, IBK	Sentiment Analysis	Urdu	6,025 sentence collected from various blogs and distributed into three classes (positive, negative and neutral)
(Akhtar, Tahir, and Shakeel 2017)	SVM, RF, K-NN, NB	Detect spam emails	Urdu	–
(Mehmood et al. 2019)	LR, NB, ANN, Majority Voting	Sentiment Analysis	Roman Urdu	11,000 comments
(Khawar Mehmood, Essam, and Shafi 2019)	NB, LR, SVM, KNN, DT	Sentiment Analysis	Roman Urdu	From 779 reviews, 412 positive and 367 negative.
(Jain, Mathew, and Jawahar 2017)	CNN, BLSTM	Optical character recognition	Urdu	29,876 line images
(Ahmad et al. 2018)	Gated BLSTM	Sentence Recognition	Urdu	Total 10,000 sentences

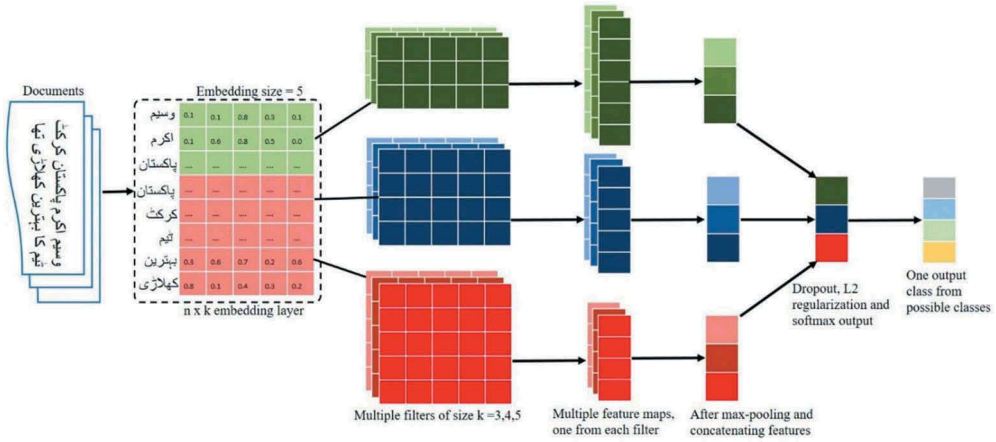

Figure 3. Single-layer multi-filter convolutional neural network.

where$c \in R^{n-h+1}$. The convolution operation is followed by a pooling operation. In this study, we used k-max pooling that select top k (most important) features from the feature map by removing low activation information for further processing and avoid overfitting due to noisy text.

$$c' = \max\{c_1, c_2, \ldots, c_{n-h+1}\} \tag{5}$$

During pooling operation, the sequence of words and the context information among words is also considered. The output layer of CNN is a fully connected softmax layer that gives a probability distribution value of the document labels.

$$y_j = w_j y_{j-1+} b_j \tag{6}$$

where y_j is an output vector of the softmax layer, y_{j-1} is an output vector of pooling layer, w_j is the transition matrix of the softmax layer, b_j is a bias factor of softmax the layer. The probability distribution over the document labels is:

$$P(i|t;\emptyset) = \frac{\exp(y_{ij})}{\sum_{k=1}^{n} \exp(y_{ij})} \tag{7}$$

The network may suffer overfitting because of a lot of hyperparameter selection. Dropout regularisation is applied to a fully connected layer to eliminate the problem of a lot of hidden units and the connection between them.

4.2. Long short-term memory (LSTM)

Traditional methods of text processing process the text independently. These methods can not relate past information with the current information during processing because of the unavailability of past information. CNN is good at capturing local context information among neighbour words but it can't capture long-term dependencies among words. Long short-term memory networks (LSTM) can capture long-term dependencies among words using its memory cells to maintain the state of the network over a long period of

time. In this way, LSTM can capture more rich and semantic information of the whole document. An LSTM model is given in Figure 4.

Input to LSTM is a word embedding layer. The input layer size is equal to the size of the word vector. The hidden layer contains LSTM units which consist of input, output and forget gates to perform input, output and forget operation respectively. These gates protect LSTM from exploding and vanishing gradient problem. Mathematical LSTM model for a word sequence is represented as below:

$$i_t = \sigma(x_t U_i + h_{t-1} W_i + b_i) \tag{8}$$

$$f_t = \sigma(x_t U_f + h_{t-1} W_f + b_f) \tag{9}$$

$$o_t = \sigma(x_t U_o + h_{t-1} W_o + b_o) \tag{10}$$

$$q_t = tanh(x_t U_q + h_{t-1} W_q + b_q) \tag{11}$$

$$p_t = f_t * p_{t-1} + i_t * q_t) \tag{12}$$

$$h_t = o_t * tanh(p_t) \tag{13}$$

i_t,o_t and f_t are input, output and forget gate respectively are generated by a sigmoid function and are an ensemble of input $x_t$and the preceding hidden stateh_{t-1}. At current step$t$, to calculate hidden step first temporary result q_t is generated by a *tanh* non-linearity over the input x_t and the preceding hidden stateh_{t-1}. Then $q_t$is with history $p_{t-1}$by input gate i_t and forgot gate $f_t$respectively to get an updated historyp_t. Finally, the output gate o_t use updated history p_t to get the final hidden state.

4.3. Bidirectional long short-term memory (BiLSTM)

Bidirectional Long Short-Term Memory (BiLSTM) is turning out the most effective RNN architecture because it uses both the future and the past information for processing current information. It is different that LSTM that only makes use of past information.

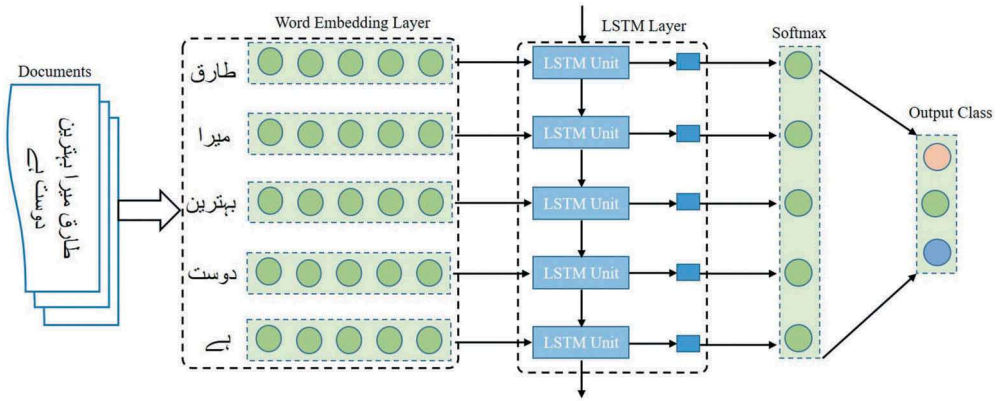

Figure 4. LSTM model.

A BiLSTM model is given in Figure 5. As compare to LSTM, BiLSTM architecture used two hidden layers one for a forward pass and the second for a backward pass. Both hidden layers are then connected to a single output layer that is connected to a fully connected layer for final output.

4.4. Convolutional long short-term memory (CLSTM)

CNN is good at extracting n-gram features from text and LSTM can learn long-term dependencies among words. CLSTM is a sequence-based model of CNN and LSTM models. CLSTM architecture is given in Figure 6. High-level features are first extracted by CNN and are feed into LSTM so that the CLSM model can learn long-range dependencies from high-level features. In this way, CLSM leaned contextual semantic features and finally classify the output label. The top-most layers, fully connected layer and the softmax layer can be used to identify the class of a document. Let $h = (h_1, h_2, \ldots, h_n)$ is the output of the LSTM units where n is the number of units. This vector is given as input to fully connected layer and its output is calculated as below:

$$d_i = \sigma(w_i * h_i + b_i) \tag{14}$$

σ is the activation function, w is the weight of the i^{th} node in the layer, b is bias then d is the output of the fully connected layer. The output of the fully connected layer is classified by the softmax layer as either the label of the document.

5. Datasets and experimental setup

5.1. Datasets for text classification

DL models are considered robust against high dimensional feature space because they implicitly select the features. To consider both high dimensional feature space and to

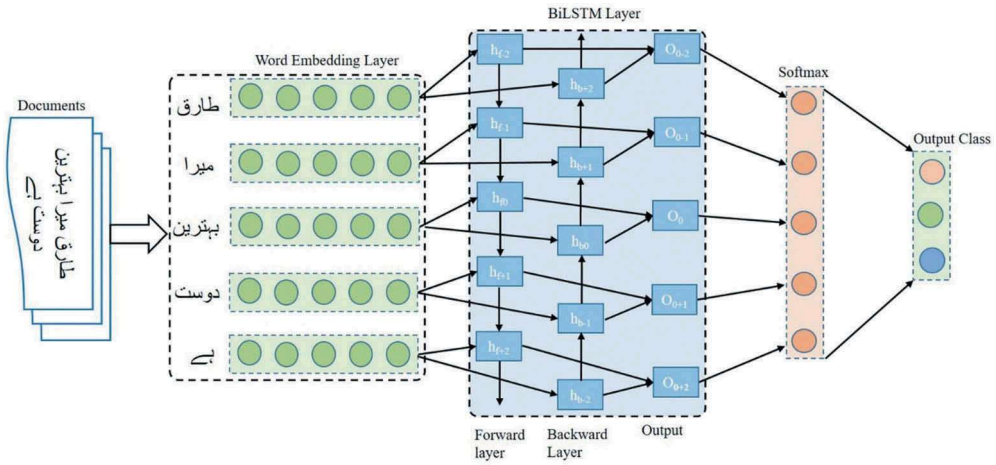

Figure 5. BiLSTM model.

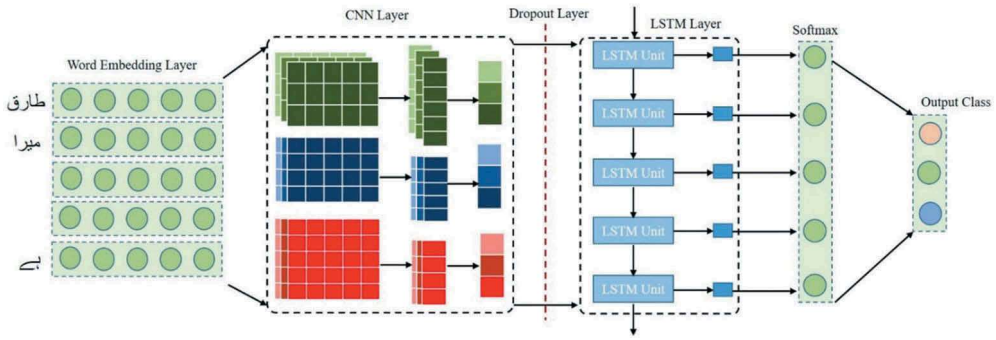

Figure 6. CLSTM model.

provide adequate data to train and evaluate the model we used three datasets of Urdu text documents of large, medium and small sizes.

- Northwestern Polytechnical University Urdu (NPUU) dataset: this dataset is a self-collected dataset. We browsed Pakistani top news websites, Express, ARY, Geo and collected news articles. We manually arranged more than ten thousand articles into six classes based on the article's content.
- Naïve dataset: this dataset contained 5003 news articles organised into four classes (Tehseen, Akhter, and Abbas 2015). It is a medium-size dataset and was designed for the classification task.
- COrpus of Urdu News Text Reuse (COUNTER) dataset: this is a small dataset designed for the study of Urdu text reuse (Sharjeel, Nawab, and Rayson 2017). It contained 1200 news documents distributed into five classes.

We cleaned the dataset and saved it in the CSV file where the label of the document was represented as integer values and the description was given as text. Sample documents of three datasets and CSV file formats are shown in Figure 7. COUNTER and naïve document samples are shown in Figure 7(a) that is in XML format. NPUU document sample is shown in Figure 7(b) that is in plain text format. CSV file format of the dataset shown in Figure 7(c) that is a common format for all datasets. Statistical comparison of the three datasets is given in Table 5.

5.2. Hyperparameter settings

We tuned hyperparameters of DL models on three datasets by applying a grid search to find the optimal value for each parameter of a model. Selecting the best parameter of a model is equals to get the best performance of the model. All the experiments were performed on Intel Core i7-7700 3.60 GHz processor, 16 GB of RAM, NVIDIA GeForce GTX 1080 graphics card, Windows 10 operating system, and Python libraries Tensorflow-GPU 1.9.0 with CUDA toolkit 9.0.

- CNN: we used embedding size 256, 128 and 32 for medium, large and small datasets. For large dataset filter size was [3, 4] while [3, 4, 5] filter size for others. 100, 64 and

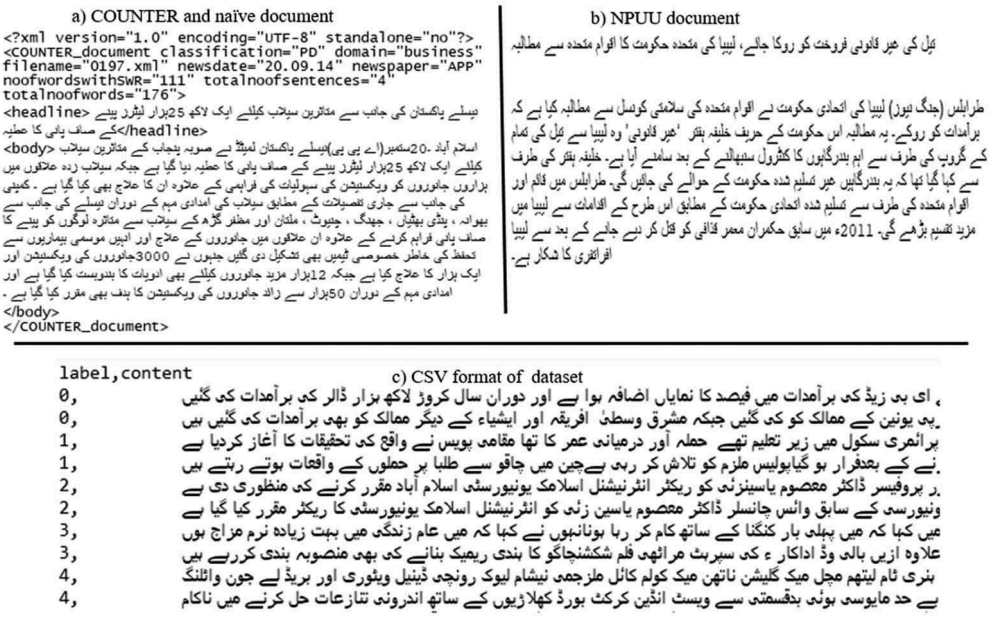

Figure 7. Samples of Urdu text documents and CSV file format with labels of each document.

Table 5. Summary of the three datasets used for classification.

Dataset Properties	COUNTER	Naïve	NPUU
Size	Small	Medium	Large
No. of Documents	1,200	5,003	10,819
Maximum length document	2,480	4,129	3,254
Minimum length document	43	47	08
Average length document	215	439	325
No. of classes	5	4	6
No. of words	288,835	2,216,845	3,611,756
Imbalanced level	High	Low	High
Split-ratio	5	5	5

128 no. of filters were taken for medium, large and small datasets respectively. For a small dataset, the dropout was 0.7 while for others 0.8 used. 120, 60 and 70 no. of epochs and Sigmoid, ReLU and Tanh activation functions were used for training medium, large and small datasets respectively.

- LSTM: 200, 100 and 160 hidden used for small, medium and large datasets. Dropout 0.5 for both small and medium datasets while 0.1 for large datasets. 20, 30, and 70 epochs used for small, medium and large datasets respectively. Embedding size is equal to a number of hidden units. For the small dataset, Tanh activation fiction while sigmoid used for medium and large datasets.
- BLSTM: Sigmoid function used for small dataset while Tanh used for small and large datasets. 160, 200, 100 hidden used for small, medium and large datasets. For the small dataset, 0.1 dropout used while 0.5 dropout for medium and large datasets. For epochs, 70, 20 and 30 epochs used for small, medium and large datasets.
- CLSTM: we used tuned parameters of LSTM and CNN for CLSTM but a number of epochs 120 used for small and medium-size datasets while 70 epochs used for large datasets.

We used Adagrad as an optimiser. Initial learning rate was taken 0.001. For all experiments, we set the batch size to 32. We divided each dataset into five training-testing ratios (50–50, 60–40, 70–30, 80–20, and 90–10) and choose the one that performed the best. The best performance was achieved on 80–20, 60–40, and 90–10 for COUNTER, naïve, and NPUU datasets respectively.

5.3. Performance measures

To measure the classification performance, we used the most common performance measures: precision, recall, F-measure and accuracy (Tripathy, Anand, and Rath 2017; Al-Radaideh and Al-Abrat 2019) and are calculated as given below.

Precision: used to measure the exactness of the classifier result and can be calculated as given below:

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

Recall: recall measures the completeness of the classifier results. It is calculated by the equation below:

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

F-measure: is the harmonic mean of precision and recall and can be calculated as:

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{17}$$

Accuracy: most common measure for classifier performance and can be calculated as given below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

Defining TP, FP, TN, FN for binary classes (i.e. positive or negative) is straightforward. For multiple classes (i.e. A, B, and C), these terms can be defined as below:

- True Positive (TP): For a class A, TP is the number of documents that actually belong to class A and are also correctly predicted as class A by a classifier
- True Negative (TN): TN is the number of documents that actually do not belong to class A and are also not predicted as class A by a classifier.
- False Positive (FP): FP is the number of documents that their actual labels do not belong to class A but are predicted as class A by a classifier.
- False Negative (FN): FN is the number of documents that their actual labels belong to class A but are predicted as documents of other classes by a classifier

6. Experimental results and discussion

COUNTER is a smaller and more imbalanced dataset than others with 1,200 documents that are distributed into five classes. It has more than two hundred and eighty thousands

of words. The experimental results of this dataset are shown in Table 6. CNN outperforms the other models with 93.3% accuracy while CLSTM outperforms with 93.6% f-measure value. These two models took advantage of their structure to extract n-gram features with parallel filters of variable size. The performance of BiLSTM is approximately the same as LSTM. Removing stopwords from the text also helps to increase the performance of all the models except CLSTM that degrades its performance. Removing both stopwords and rare words improved the performance of all the models except the LSTM model that decreased its performance after removing rare words. None of the models performed best on the presence of stopwords and rare words.

Naïve dataset is a medium-size document that contains more than five thousands of documents that are distributed into four labels. The maximum length of a document and the maximum average length of a document are also greater as compared to the other two datasets. It contained more than 2.2 million words. Naïve is a less imbalanced dataset than others. On this challenging dataset, CNN and BiLSTM models outperform the LSTM and CLSTM after removing both stopwords and rare words as given in Table 7. CNN takes advantage of contextual information extracted from neighbourhood words while BiLSTM gets the advantage to use past and present information to extract a feature. Further, both models have slightly decreased their performance when stopwords are removed from the text. BiLSTM performed better than LSTM and CLSTM models. It is concluded that all the models increased their performance after removing both stopwords and rare words from the text.

NPUU is a large-sized and more challenging dataset than naïve and COUNTER. It contained more than ten thousand documents that are distributed into six labels. It contained more than 3.6 million words and is a more imbalanced dataset than naïve. From Table 8, it can be seen that on the NPUU dataset, again CNN outperforms the other three models with 91.8% accuracy and 91.0% f-measure values after removing both stopwords and rare words from the dataset. CLSTM shows its better performance than LSTM and BiLSTM models because it combines the features extracted from CLSTM and LSTM models to predict the final label of a document. CNN and CLSTM models slightly decreased their performance while the performance of LSTM and BiLSTM models was more affected when removing only stopwords from the data.

Removing rare words up to some extent from the dataset has two main advantages: reduce vocabulary size and increase model performance. It can be seen from Figure 8 that

Table 6. Performance of deep learning models on a COUNTER dataset.

Dataset	Text	Precision	Recall	F-measure	Accuracy
CNN	Without stopwords + rare words removal	92.1	77.5	84.2	89.6
	Stopwords removal	93.0	81.7	87.0	90.6
	Stopwords + rare words removal	**95.4**	**90.2**	**92.7**	**93.3**
LSTM	Without stopwords + rare words removal	79.2	80.0	79.6	84.4
	Stopwords removal	**91.2**	**92.6**	**91.9**	**92.0**
	Stopwords + rare words removal	91.0	89.4	90.2	92.0
BiLSTM	Without stopwords + rare words removal	91.9	85.6	88.6	89.5
	Stopwords removal	91.2	90.2	90.7	90.9
	Stopwords + rare words removal	**91.5**	**91.8**	**91.7**	**92.3**
CLSTM	Without stopwords + rare words removal	85.0	83.4	84.2	86.6
	Stopwords removal	81.1	72.9	76.8	82.1
	Stopwords + rare words removal	**94.0**	**93.1**	**93.6**	**92.9**

Table 7. Performance of deep learning models on naïve dataset.

Dataset	Text	Precision	Recall	F-measure	Accuracy
CNN	Without stopwords + rare words removal	94.5	92.6	93.5	94.9
	Stopwords removal	93.8	93.0	93.4	94.7
	Stopwords + rare words removal	**94.8**	**93.6**	**94.2**	**95.4**
LSTM	Without stopwords + rare words removal	79.8	77.1	78.4	84.0
	Stopwords removal	87.8	89.4	88.6	90.6
	Stopwords + rare words removal	**90.4**	**90.8**	**90.6**	**92.5**
BiLSTM	Without stopwords + rare words removal	93.0	89.3	91.1	92.5
	Stopwords removal	91.7	89.9	90.8	92.5
	Stopwords + rare words removal	**94.8**	**93.5**	**94.1**	**95.4**
CLSTM	Without stopwords + rare words removal	89.3	90.2	89.8	92.1
	Stopwords removal	90.3	91.8	91.0	92.8
	Stopwords + rare words removal	**93.7**	**93.2**	**93.5**	**94.7**

Table 8. Performance of deep learning models on NPUU dataset.

Dataset	Text	Precision	Recall	F-measure	Accuracy
CNN	Without stopwords + rare words removal	89.9	88.0	88.9	89.7
	Stopwords removal	89.8	87.5	88.6	89.7
	Stopwords + rare words removal	**90.4**	**91.7**	**91.0**	**91.8**
LSTM	Without stopwords + rare words removal	**88.8**	87.8	**88.3**	88.7
	Stopwords removal	81.6	72.4	76.7	77.8
	Stopwords + rare words removal	86.8	**89.5**	88.1	**89.0**
BiLSTM	Without stopwords + rare words removal	86.0	83.9	84.9	86.2
	Stopwords removal	86.3	81.0	83.5	83.0
	Stopwords + rare words removal	**87.5**	**88.2**	**87.9**	**89.1**
CLSTM	Without stopwords + rare words removal	86.1	86.8	86.4	87.6
	Stopwords removal	86.2	86.4	86.3	87.3
	Stopwords + rare words removal	**87.8**	**88.9**	**88.4**	**89.2**

removing rare words caused to shrink the size of the dataset. After removing ten rare words, the size of naïve, COUNTER and NPUU was reduced up to 65%, 71%, and 69% respectively. Half of the vocabulary of each dataset was removed after removing all those words that were appeared at least five times in each dataset.

From the previous result discussion, it can be concluded that only removing stopwords from the dataset cannot improve performance. It can also be noticed that almost all the models improved its performance and showed the best performance after removing both stopwords and rare words from the three datasets. We have analysed the performance of DL models after removing those words which occurred in the dataset from one to ten times as it is shown in Figure 9. None of the models showed good performance after removing rare words one, seven, nine and ten. On small dataset, LSTM, BiLSTM, and CLSTM achieved maximum accuracy values on removing rare words up to 6, 8, and 8 respectively that is equal to removing more than half of the vocabulary from the dataset. On less balanced and less class distributed naïve dataset, the maximum performance was achieved on removing 2, 3, 4, and 5 rare words for CNN, LSTM, BiLSMT, and CLSTM models respectively that is equal to reducing dataset vocabulary from 25% to 50%. On the NPUU dataset that is imbalanced, a large size with a maximum number of classes, LSTM and BiLSTM models have shown maximum performance on 8 and 6 rare words respectively that is equal to removing more than half of the vocabulary of the dataset. While CNN and CLSTM performed well on 3 and 2 rare words that are equals to removing the vocabulary size up to 25% to 40%.

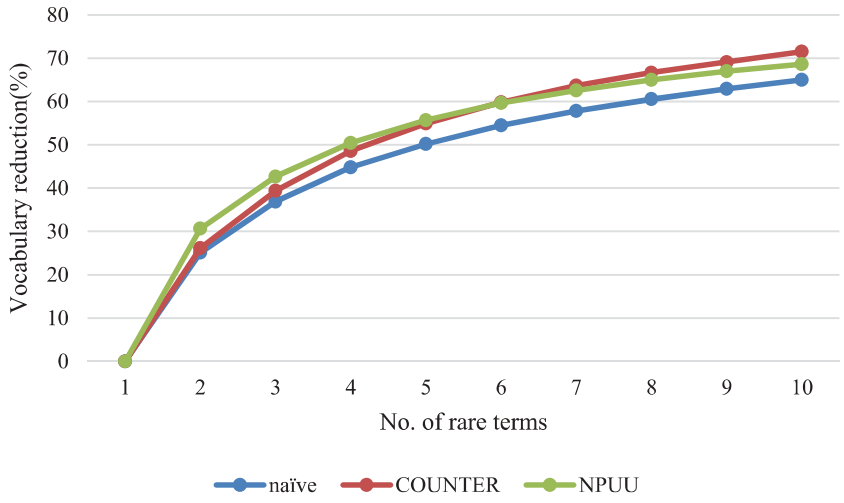

Figure 8. Effects of removing rare words on the vocabulary size of three datasets.

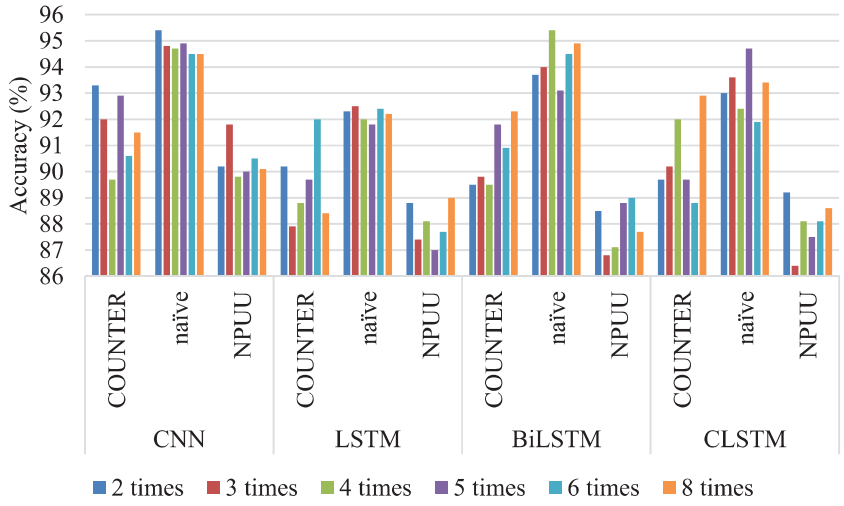

Figure 9. Removing rare words effects on the accuracy of the model.

On more imbalanced datasets having more classes like COUNTER and NPUU, LSTM and BiLSTM models can show maximum performance after removing more than half of the vocabulary. CNN's performance on all the datasets is highly affected by removing more words as it showed its maximum performance on removing 2 and 3 rare words only. On the naïve dataset that is less imbalanced and less number of classes but with maximum average length documents, all the models didn't perform well when more than five rare words were eliminated from the dataset. CNN, LSTM, BiLSMT, and CLSTM achieved maximum performance on 2, 3, 4 and 5 rare words respectively that is equal to removing less than half of the vocabulary.

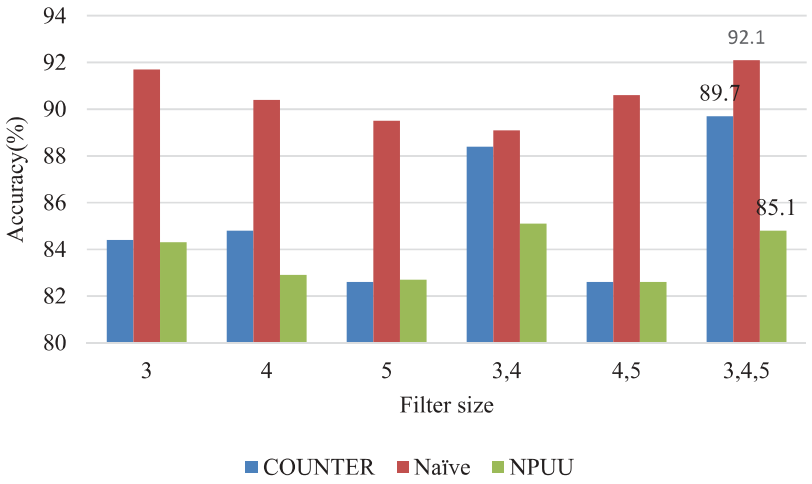

Figure 10. Performance of CNN with multiple filters of the same size vs multiple filters of variable size.

For CNN, finding an appropriate size of a filter is a time consuming and difficult task. Figure 10 shows the comparison of three different sizes of filters, single filter and well multiple filters. The comparison shows that multiple convolutional layers in parallel with different filter sizes perform better than the single convolutional layers with the same filter size. Multiple filters can exploit features of different n-grams. Our findings on Urdu text endorsed the results of (Zhou et al. 2015).

We compare LSTM and BiLSTM models at a deep level by using layers. In two-layer architecture, the hidden state of an LSTM unit is used as input to the LSTM unit in the second layer in the same time step. The aim is to capture longer-term dependencies of the input sequence. From Table 9, it can be seen that the single-layer model performs better than two-layer models on all the datasets that are contradicting to (Rao et al. 2018). BiLSTM with single-layer as well as two-layers performs better than LSTM on three datasets that endorsed the findings of (Liu and Guo 2019).

We compare the deep learning models with four well-known machine learning models. After preprocessing and feature selection using information gain, accuracy values are shown in Figure 11. It is seen that deep learning models perform better than machine learning models on three datasets. CNN outperformed the other models while NB performed better than other machine learning models. It can be noticed that there was a small merging in performance between machine learning and deep learning models on a small dataset but this margin increase significantly on large size datasets. It can be

Table 9. Single-layer vs. two-layer models of LSTM and BiLSTM (P =precision, R =recall, F =measure, A =accuracy).

	COUNTER				Naïve				NPUU			
Classifier/dataset	P	R	F	A	P	R	F	A	P	R	F	A
LSTM	91.0	89.4	90.2	92.0	90.4	90.8	90.6	92.5	86.8	**89.5**	**88.1**	89.0
2-layer LSTM	9.0	88.9	89.4	89.7	88.3	87.3	87.8	90.3	85.0	86.6	85.8	87.8
BiLSMT	**91.5**	**91.8**	**91.7**	**92.3**	**94.8**	**93.5**	**94.1**	**95.4**	**87.5**	88.2	87.9	**89.1**
2-layer BiLSTM	91.2	87.6	89.4	90.3	92.8	90.7	91.7	93.3	87.4	86.7	87.1	88.3

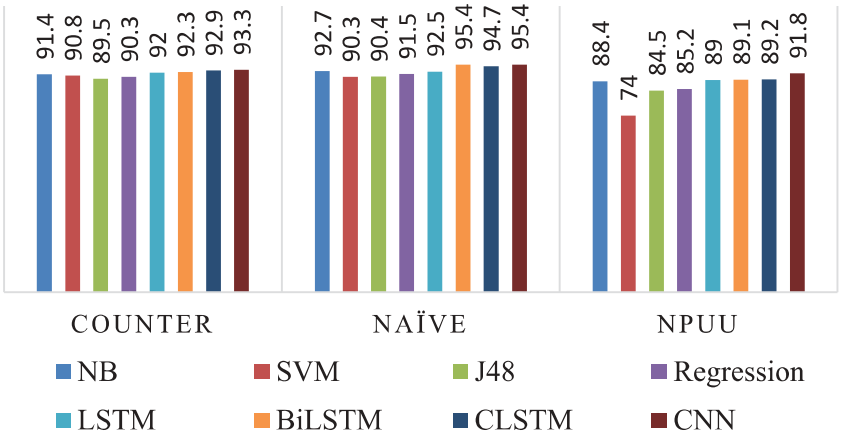

Figure 11. Comparison of machine learning and deep learning models.

concluded that deep learning models performed better than machine learning models on small as well large size dataset for long text document classification of Urdu language.

To know that is there any statistically significant difference among the classifiers, we perform statistical Friedman Test and post-hoc Nemenyi test as performed in (Demšar 2006). Friedman test used to compare multiple classifiers while Nemeni test shows the pairwise comparison among the classifiers. Friedman test ranks the classifiers for each dataset as shown in Table 10. Ranks are assigned from 1 to n based on high to low performance. Classifiers with same accuracy value assign average ranks. Although average ranks (at end of avg. column) by themselves provide a fair comparison among the classifiers.

We set a null- hypothesis that all the classifiers are equivalent if their mean ranks R_j are equal ($R_j = \frac{1}{N}\sum_i r_i^j = 2.5$). All the classifiers (except BiLSTM) are not equivalent as their average ranks are not equal with mean ranks R_j so we reject the null-hypothesis. Friedman test checks whether the measure average ranks are significantly different from the mean rank Rj =2.5.

$$X_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{19}$$

X_F^2 is undesirably conservative and derived a better statistic F_F which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degree of freedom. F_F value is 8.429 and calculated as

Table 10. Best performing algorithms and their ranks on each dataset.

	CNN		LSTM		BiLSTM		CLSTM	
Dataset	Acc.	Rank	Acc.	Rank	Acc.	Rank	Acc.	Rank
COUNTER	93.3	1	92	4	92.3	3	92.9	2
Naïve	95.4	1.5	92.5	4	95.4	1.5	94.7	3
NPUU	91.8	1	89	4	89.1	3	89.2	2
Avg. Rank		3.143		2.000		2.893		1.964

$$F_F = \frac{(N-1)X_F^2}{N(k-1)-X_F^2} \tag{20}$$

Critical value of F $(k-1, (k-1)x(N-1)) = F(3,6) = 4.76$ for $\propto = 0.05$. It shows that there is a significant difference among classifiers so we reject the null hypothesis.

After Friedman test, now we can proceed with the Nemenyi test for pairwise comparison of classifiers. A null-hypothesis can be described as the performance of two classifiers is significantly different if the corresponding average rank different by at least the critical difference (CD)

$$CD = q_{\propto}\sqrt{\frac{k(k+1)}{6N}} \tag{21}$$

For four classifier, critical value with $q_{\propto}$= 0.5 is 2.569 (see Table 5 in (Demšar 2006)) and CD value is 1.425. Pairwise comparison of all the classifiers is shown in Table 11. For example, for comparison of CNN and LSTM, $Rank_{Diff}$(CNN,LSTM) = (4–1.167) = 2.833. An average rank difference greater than CD that shows there is a significant difference between the CNN and LSTM. Comparison of these classifiers using the Nemenyi test is visually shown in Figure 12. Connected lines of the classifier show that two classifiers are not significantly different. It can be noticed that there is no statistically significant difference among the performance of CNN, CLSTM and CNN and BiLSTM. CLSTM is a hybrid classifier of CNN and LSTM models and LSTM did not perform well than CNN (see Tables 6–8) because CNN takes advantage of its structure of multiple filters of different size to extract variable-length features (see Section 4.1). Similarly, BiLSTM performs better than LSTM (see Table 9) and close to CNN because of its nature to capture features in both directions from the text.

From our experimental results on Urdu text classification on document-level, we can summarise our findings as below:

- CNN outperforms the other models on small, medium and large size datasets
- BiLSTM perform better than LSTM and CLSTM on the balanced dataset, less class distribution and long text documents

Table 11. Pairwise comparison of the classifier using average ranks and CD.

Classifiers	$Rank_{Diff}$	$Rank_{Diff}$ > CD	Significant Diff.
$Rank_{Diff}$ (CNN, LSTM)	4-1.167 = 2.833	2.833 >1.425	Yes
$Rank_{Diff}$ (CNN, BiLSTM)	2.5–1.167 = 1.33	1.33 <1.425	No
$Rank_{Diff}$ (CNN, CLSTM)	2.33–1.167 = 1.163	1.163 <1.425	No
$Rank_{Diff}$ (LSTM, BiLSTM)	4-2.5 = 1.57	1.57 >1.425	Yes
$Rank_{Diff}$ (LSTM, CLSTM)	4-2.33 = 1.73	1.73 > 1.425	Yes

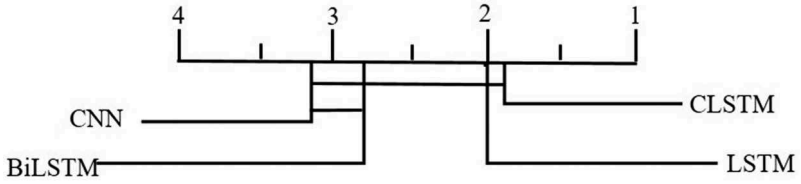

Figure 12. Comparison of all the classifiers with each other using Nemenyi test.

- CLSTM performs better than LSTM and BiLSTM on imbalanced and more class distribution dataset
- Removing stopwords cause to decrease the performance of deep learning models on a large dataset but CNN performance is more affected on all the datasets
- Models increase their performance after removing both stopwords and rare words on all the datasets except LSTM that decrease their performance on a small dataset
- Multiple filters of variable size perform better than multiple filters of the same size in CNN
- Single-layer architecture performs better than the two-layer architecture of LSTM and BiLSTM
- Deep learning models outperform the machine learning models and NB performed better than other machine learning models
- Machine learning models significantly decrease their performance on large size dataset

7. Smart manufacturing using Urdu language

In product manufacturing, product designers, maintenance providers, and decision-makers closely analyze the product features, cost, and price and customer feedback of both products (their products and competitors' products) to manufacture better quality products and provide high-quality services. Customer feedback has a significant role in product manufacturing. In the proposed framework for smart manufacturing by (Ren et al. 2019) contains four components: intelligent design, intelligent production, intelligent maintenance and service, and intelligent recovery. The proposed framework shows the importance of customer feedback in product intelligent design and intelligent maintenance and services component.

Customer feedback includes product sentiments, reviews, opinions, and public articles (either positive or negative) that are collected from various sources (i.e. internet, IoTs, EISs) from all over the world. As described in section 1, Urdu is the language of 300 million people around the world. Articles in Urdu language that includes merits and demerits of a product are published in different news blogs and website. Experts, technicians, and buyers gives their opinions and sentiments about the products. People usually prefer to give feedback in their native language so that they can express their opinion or problems more effectively. In Urdu language, various machine learning methods have been used to classify customer feedback about the products but the potential of deep learning methods have not been explored yet.

Millions of comments, emails, and articles about the product are received from customers, sellers and stored in a repository as a raw text in XML or CSV files. To convert the raw text into meaningful text, the text is preprocessed (see section 5.1). After preprocessing the text, it is given to the ML and DL models to processes and classify these millions of comments into useful labels (i.e. valuable or not valuable, relevant or irrelevant and bad, good, very good). This labelled text is then used in product lifecycle by a product designer, decision-makers and service providers in intelligent product manufacturing as it is shown in Figure 13.

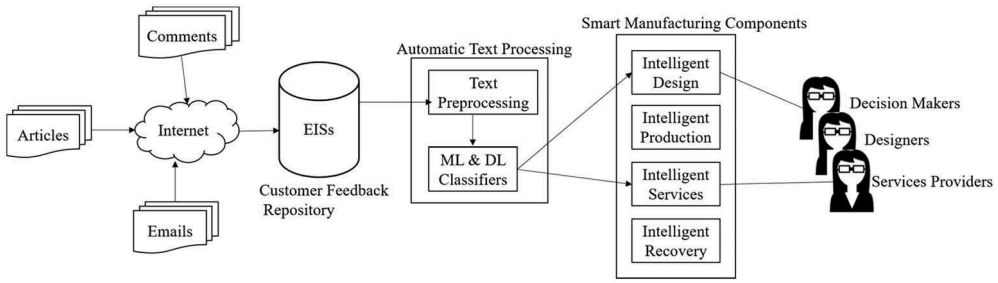

Figure 13. Role of Urdu text classification in intelligent product manufacturing.

8. Conclusion

In this study, we comparatively analysed the performance of four deep learning models to classify text documents of Urdu. On three datasets, CNN outperforms the other models while CLSTM performs better than LSTM and BiLSTM. BiLSTM performed better than LSTM on three datasets (Liu and Guo 2019). Our comparative results show that only stopwords elimination decrease the performance of these models. Removing both rare words and stopwords have a significant impact on the performance of all the used deep learning models on all the datasets except the LSTM model that decreased its performance on removing rare words form the small dataset (Aggarwal 2018; Katnoria, Singh, and Kumar 2017; Song, Huang, and Ruan 2018; Ayedh et al. 2016). Further, multiple filters of variable size perform better than multiple filters of the same size and single-layer LSTM and BiLSTM models perform better than two-layer models. Deep learning outperformed machine learning models on small as well as large size datasets.

Because Urdu is a resource-poor language there is scope for further research. Although, we achieve a significant performance but the following directions can be considered in future:

(1) We used there imbalanced datasets of various size. Performance can be evaluated using balanced datasets
(2) We are interested to analyse the performance of these models on a large dataset in term of vocabulary size and number of examples
(3) Other deep learning models like character-level CNN, predefined word embeddings like doc2vec
(4) Sentence-level representation or paragraph-level representation of the document can be applied to Urdu text documents
(5) Classification techniques other than deep learning like ensemble classification, hybrid approaches etc. can be applied to these datasets and compare the results with this study
(6) Deep learning models used in this study can also be applied to various other tasks of Urdu text processing like sentiment analysis, abusive language detection, fake news detection, etc.

Acknowledgments

The authors would like to thank all the people who worked hard and helped a lot to complete this study.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported in part by the Research and Development Plan of Shaanxi Province under Grant 2017ZDXM-GY-094 and Grant 2015KTZDGY04-01, and in part by the National Natural Science Foundation of China under Grant 61972321.

ORCID

Muhammad Pervez Akhter http://orcid.org/0000-0002-4933-8905

References

Aggarwal, C. C. 2018. "Text Sequence Modeling and Deep Learning BT." In *Machine Learning for Text*, edited by C. C. Aggarwal, 305–360. Cham: Springer International Publishing. doi:10.1007/978-3-319-73531-3_10.

Ahmad, I., X. Wang, Y. Hao Mao, G. Liu, H. Ahmad, and R. Ullah. 2018. "Ligature Based Urdu Nastaleeq Sentence Recognition Using Gated Bidirectional Long Short Term Memory." *Cluster Computing* 21 (1): 703–714. doi:10.1007/s10586-017-0990-5.

Ahmad, J., H. Farman, and Z. Jan. 2019. "Deep Learning Methods and Applications BT." In *Deep Learning: Convergence to Big Data Analytics*, edited by M. Khan, B. Jan, and H. Farman, 31–42. Singapore: Springer Singapore. doi:10.1007/978-981-13-3459-7_3.

Ahmed, S. B., S. Naz, S. Swati, and M. I. Razzak. 2017. "Handwritten Urdu Character Recognition Using 1-Dimensional {BLSTM} Classifier." *CoRR Abs/1705.0*. http://arxiv.org/abs/1705.05455.

Akhtar, A., G. R. Tahir, and K. Shakeel. 2017. "A Mechanism to Detect Urdu Spam Emails." In 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 168–172. doi:10.1109/UEMCON.2017.8249019.

Ali, A., and M. Ijaz. 2009. "Urdu Text Classification." doi:10.1145/1838002.1838025.

Ali, M., S. Khalid, M. I. Rana, and F. Azhar. 2018. "A Probabilistic Framework for Short Text Classification." In 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), 742–747. doi:10.1109/CCWC.2018.8301712.

Al-Radaideh, Q. A., and M. A. Al-Abrat. 2019. "An Arabic Text Categorization Approach Using Term Weighting and Multiple Reducts." *Soft Computing* 23 (14): 5849–5863. doi:10.1007/s00500-018-3249-z.

Altınel, B., and M. C. Ganiz. 2018. "Semantic Text Classification: A Survey of past and Recent Advances." *Information Processing & Management* 54 (6): 1129–1153. doi:10.1016/j.ipm.2018.08.001.

Amajd, M., Z. Kaimuldenov, and I. Voronkov. 2017. "Text Classification with Deep Neural Networks." In 1989 CEUR Workshop Proceedings. MIPT, RussiaNRU HSE, 362–370. Russia: CEUR-WS.

Ayedh, A., T. A. N. Guanzheng, K. Alwesabi, and H. Rajeh. 2016. "The Effect of Preprocessing on Arabic Document Categorization." *Algorithms* 9: 27. doi:10.3390/a9020027.

Balodis, K., and D. Deksne. 2019. "Fast Text-Based Intent Detection for Inflected Languages." *Information (Switzerland)* 10 (5): 1–16. doi:10.3390/info10050161.

Basit, R. H., M. Aslam, A. M. Martinez-Enriquez, and A. Z. Syed. 2017. "Semantic Similarity Analysis of Urdu Documents BT - Pattern Recognition." In *Jesús Ariel Carrasco-Ochoa, José Francisco Martínez-Trinidad, and José Arturo Olvera-López*, edited by K. L. Boyer, 234–243. Cham: Springer International Publishing.

Bassem, B., and M. Zrigui. 2020. "Gender Identification: A Comparative Study of Deep Learning Architectures." doi:10.1007/978-3-030-16660-1_77.

Çağataylı, M., and E. Çelebi. 2015. "The Effect of Stemming and Stop-Word-Removal on Automatic Text Classification in Turkish Language BT." In *Neural Information Processing*, edited by S. Arik, T. Huang, W. K. Lai, and Q. Liu, 168–176. Cham: Springer International Publishing.

Demšar, J. 2006. "Statistical Comparisons of Classifiers over Multiple Data Sets." *Journal of Machine Learning Research* 7: 1–30.

Jain, G., M. Sharma, and B. Agarwal. 2018. "Optimizing Semantic LSTM for Spam Detection." *International Journal of Information Technology*. doi:10.1007/s41870-018-0157-5.

Jain, M., M. Mathew, and C. V. Jawahar. 2017. "Unconstrained OCR for Urdu Using Deep CNN-RNN Hybrid Networks." In 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR), 747–752. doi:10.1109/ACPR.2017.5.

Kapočiūtė-Dzikienė, J., R. Damaševičius, and M. Woźniak. 2019. "Sentiment Analysis of Lithuanian Texts Using Traditional and Deep Learning Approaches." *Computers* 8: 1. doi:10.3390/computers8010004.

Katnoria, M., V. Singh, and R. Kumar. 2017. "Punjabi Document Classification Using Vector Evaluation Method." In 2017 International Conference on Computing Methodologies and Communication (ICCMC), 940–944. doi:10.1109/ICCMC.2017.8282606.

Khan, N. H., and A. Adnan. 2018. "Urdu Optical Character Recognition Systems: Present Contributions and Future Directions." *IEEE Access* 6: 46019–46046. doi:10.1109/ACCESS.2018.2865532.

Kılınç, D., A. Özçift, F. Bozyigit, P. Yıldırım, F. Yücalar, and E. Borandag. 2015. "TTC-3600: A New Benchmark Dataset for Turkish Text Categorization." *Journal of Information Science* 43 (2). SAGE Publications Ltd: 174–185. doi:10.1177/0165551515620551.

Kim, T.-Y., and S.-B. Cho. 2018. "Web Traffic Anomaly Detection Using C-LSTM Neural Networks." *Expert Systems with Applications* 106: 66–76. doi:10.1016/j.eswa.2018.04.004.

Kim, Y. 2014. "Convolutional Neural Networks for Sentence Classification." *CoRR Abs/1408.5*.

Krig, S. 2016. "Feature Learning and Deep Learning Architecture Survey BT." In *Computer Vision Metrics: Textbook Edition*, edited by S. Krig, 375–514. Cham: Springer International Publishing. doi:10.1007/978-3-319-33762-3_10.

Lee, Y., S. Yoon, and K. Jung. 2018. "Comparative Studies of Detecting Abusive Language on Twitter." *CoRR Abs/1808.1*. http://arxiv.org/abs/1808.10245

Li, S., K. Nahar, and B. C. M. Fung. 2015. "Product Customization of Tablet Computers Based on the Information of Online Reviews by Customers." *Journal of Intelligent Manufacturing* 26 (1): 97–110. doi:10.1007/s10845-013-0765-7.

Liu, G., and J. Guo. 2019. "Bidirectional LSTM with Attention Mechanism and Convolutional Layer for Text Classification." *Neurocomputing* 337: 325–338. doi:10.1016/j.neucom.2019.01.078.

Mehmood, K., D. Essam, K. Shafi, and M. K. Malik. 2019. "Discriminative Feature Spamming Technique for Roman Urdu Sentiment Analysis." *IEEE Access* 7: 47991–48002. doi:10.1109/ACCESS.2019.2908420.

Mehmood, K., D. Essam, and K. Shafi. 2019. "Sentiment Analysis System for Roman Urdu BT - Intelligent Computing." In *Advances in Intelligent Systems and Computing*, edited by K. Arai, S. Kapoor, and R. Bhatia, 29–42. Cham: Springer International Publishing.

Mukhtar, N., and M. A. Khan. 2017. "Urdu Sentiment Analysis Using Supervised Machine Learning Approach." *International Journal of Pattern Recognition and Artificial Intelligence*. doi:10.1142/S0218001418510011.

Mukhtar, N., M. A. Khan, and N. Chiragh. 2017. "Effective Use of Evaluation Measures for the Validation of Best Classifier in Urdu Sentiment Analysis." *Cognitive Computation* 9 (4): 446–456. doi:10.1007/s12559-017-9481-5.

Rao, G., W. Huang, Z. Feng, and Q. Cong. 2018. "LSTM with Sentence Representations for Document-Level Sentiment Classification." *Neurocomputing* 308: 49–57. doi:10.1016/j.neucom.2018.04.045.

Ren, S., Y. Zhang, Y. Liu, T. Sakao, D. Huisingh, and C. M. V. B. Almeida. 2019. "A Comprehensive Review of Big Data Analytics Throughout Product Lifecycle to Support Sustainable Smart Manufacturing: A Framework, Challenges and Future Research Directions." *Journal of Cleaner Production* 210. Elsevier B.V.: 1343–1365. doi:10.1016/j.jclepro.2018.11.025.

Riaz, K. 2012. "Comparison of Hindi and Urdu in Computational Context." *International Journal of Computational Linguistics and Natural Language Processing* 01 (3): 92–97.

Sharjeel, M., R. M. A. Nawab, and P. Rayson. 2017. "COUNTER: Corpus of Urdu News Text Reuse." *Language Resources and Evaluation* 51 (3): 777–803. doi:10.1007/s10579-016-9367-2.

Sharp, M., R. Ak, and T. Hedberg. 2018. "A Survey of the Advancing Use and Development of Machine Learning in Smart Manufacturing." *Journal of Manufacturing Systems* 48 (Pt C): 170–179. doi:10.1016/j.jmsy.2018.02.004.

Song, S., H. Huang, and T. Ruan. 2018. "Abstractive Text Summarization Using LSTM-CNN Based Deep Learning." *Multimedia Tools and Applications*. doi:10.1007/s11042-018-5749-3.

Tehseen, Z., M. P. Akhter, and Q. Abbas. 2015. "Comparative Study of Feature Selection Approaches for Urdu Text Categorization." *Malaysian Journal of Computer Science* 28 (2): 93–109.

Tripathy, A., A. Anand, and S. K. Rath. 2017. "Document-Level Sentiment Classification Using Hybrid Machine Learning Approach." *Knowledge and Information Systems* 53 (3): 805–831. doi:10.1007/s10115-017-1055-z.

Usman, M., Z. Shafique, S. Ayub, and K. Malik. 2016. "Urdu Text Classification Using Majority Voting." *International Journal of Advanced Computer Science and Applications* 7. doi:10.14569/IJACSA.2016.070836.

Wang, J., Y. Ma, L. Zhang, R. X. Gao, and D. Wu. 2018. "Deep Learning for Smart Manufacturing: Methods and Applications." *Journal of Manufacturing Systems* 48. The Society of Manufacturing Engineers: 144–156. doi:10.1016/j.jmsy.2018.01.003.

Wongso, R., F. A. Luwinda, B. C. Trisnajaya, and O. Rusli. 2017. "News Article Text Classification in Indonesian Language." *Procedia Computer Science* 116: 137–143. doi:10.1016/j.procs.2017.10.039.

Yin, W., K. Kann, M. Yu, and H. Schütze. 2017. "Comparative Study of {CNN} and {RNN} for Natural Language Processing." *CoRR Abs/1702.0*. http://arxiv.org/abs/1702.01923

Zhou, C., C. Sun, Z. Liu, and F. C. M. Lau. 2015. "A {C-LSTM} Neural Network for Text Classification." *CoRR Abs/1511.0*. http://arxiv.org/abs/1511.08630