Neural Computing and Applications (2024) 36:3077–3100
https://doi.org/10.1007/s00521-023-09169-6

ORIGINAL ARTICLE

Passion-Net: a robust precise and explainable predictor for hate speech detection in Roman Urdu text

Faiza Mehmood[1,2] · Hina Ghafoor[1,3,5] · Muhammad Nabeel Asim[3] · Muhammad Usman Ghani[4] · Waqar Mahmood[1] · Andreas Dengel[3,5]

Received: 15 February 2023 / Accepted: 20 October 2023 / Published online: 28 November 2023

Abstract
With an aim to eliminate or reduce the spread of hate content across social media platforms, the development of artificial intelligence supported computational predictors is an active area of research. However, diversity of languages hinders development of generic predictors that can precisely identify hate content. Several language-specific hate speech detection predictors have been developed for most common languages including English, Chinese and German. Specifically, for Urdu language a few predictors have been developed and these predictors lack in predictive performance. The paper in hand presents a precise and explainable deep learning predictor which makes use of advanced language modelling strategies for the extraction of semantic and discriminative patterns. Extracted patterns are utilized to train an attention-based novel classifier that is competent in precisely identifying hate content. Over coarse-grained benchmark dataset, the proposed predictor significantly outperforms state-of-the-art predictor by 8.7% in terms of accuracy, precision and F1-score. Similarly, over fine-grained dataset, in comparison with state-of-the-art predictor, it achieves performance gain of 10.6%, 17.6%, 18.6% and 17.6% in terms of accuracy, precision, recall and F1-score.

Keywords Roman Urdu · Hate speech detection · Language model · Interpretability · Deep learning · Attention head · Aggregation attention

1 Introduction

The last two decades have witnessed a significant increase in the development of social media platforms [1]. These platforms are facilitating in transforming world into global village in which social media users from different regions can share information [1] about everyday activities. Advancement in internet facilities enable people for quickly interacting and expressing their opinions with each other through different communication channels, such as brief text messages, tweets and sharing posts on various social media platforms including Facebook, Twitter and

✉ Muhammad Nabeel Asim
Muhammad_Nabeel.Asim@dfki.de

Faiza Mehmood
faiza.mehmood@kics.edu.pk

Hina Ghafoor
hina.ghafoor@kics.edu.pk

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

Waqar Mahmood
director@kics.edu.pk

Andreas Dengel
Andreas.dengel@dfki.de

[1] Al-Khawarizmi Institute of Computer Science (KICS), University of Engineering and Technology, Lahore, Pakistan

[2] Department of Computer Science, University of Engineering and Technology (Faisalabad Campus), Lahore, Pakistan

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

[4] Department of Computer Science, University of Engineering and Technology, Lahore, Pakistan

[5] Department of Computer Science, Rheinland Pfälzische Technische Universität, 67663 Kaiserslautern, Germany

Instagram [2, 3]. Social media platforms provide freedom for sharing any type of content; however, some users exploit the opportunity to publish fake news [4] and hate content [1, 5].

Hate speech or trolling is disparaging a person or a group of people based on a trait such as race, colour, ethnicity, gender, sexual orientation, nationality or religion [6]. Such type of controversial content propagates hostile discourse, and this enragement may manifest itself in physical violence or violent acts [7]. In a nutshell, for the development of a truly democratic society, hate speech eradication is essential to stop violence, control conflict on a larger scale and maintain law and order [8]. However, it is difficult to detect and filter hate content from social media platforms mainly due to the diversity of 7139 languages used by social media users [9].

To winnow out the spread of hate content, researchers are utilizing the power of artificial intelligence methods for the development of two different types of computational predictors, namely, language-specific [10–12] and multilingual [13–15]. Multilingual predictors lack in predictive performance because they remain fail in capturing discriminative and contextual information from different types of languages at the same time [12]. On the other hand, although language-specific predictors produce better performance, they are developed only for resource-rich languages like English [16], Spanish [17], Dutch [18, 19] and Arabic [20].

Urdu Language lacks robust and precise hate speech detectors, although it has more than 100 million speakers [21] around the world and is the national language of two different countries: Pakistan and India [21]. To communicate Urdu, speakers make use of two different writing styles: Nastaliq Urdu and Roman Urdu [22]. Nastaliq Urdu is written in Latin script and Roman Urdu is similar to English, but it follows completely free writing style [22]. Most Urdu speaking social media users use Roman Urdu language for communication. For Roman Urdu language, it is not possible to adapt existing machine and deep learning-based methodologies that have been developed for hate speech detection in resource-rich languages. The unique characters and writing style of Roman Urdu are exactly similar to the English language. However, Roman Urdu does not follow any grammatical rules and people use a free writing style while writing on social media platforms such as a word can be written in several possible ways. The English word "unconscious" is written in different ways in Roman Urdu such as "behosh", "bahosh", "bayhoosh", "byhosh" and "baihoosh". Hence, due to high variability of same words in different samples extraction of informative patterns is difficult in Roman Urdu textual data. In a nutshell, Roman Urdu requires more robust computational predictors that can deal with a large vocabulary and high variability of the same word. According to our best knowledge to date, there exist only four computational predictors for Roman Urdu hate speech detection [10, 23–25]. These predictors make use of various machine learning and deep learning-based approaches but remained fail to provide a decisive system for hate content analysis. To empower the process of hate speech detection from Roman Urdu content contributions of this paper are manifold:

1. It facilitates an optimized language model trained on large unannotated Roman Urdu data. Apart from hate speech detection, trained language model can also be used for performing multiple other tasks such as sentiment analysis, fake news detection and information retrieval.
2. It presents a novel classifier that makes use of two different types of attention mechanisms and multiple neural strategies such as learning rate decay and dropout.
3. Proposed predictor is enriched with interpretability mechanism that enables a clearer understanding of words contributions towards predictor decisions.
4. Over coarse- and fine-grained benchmark datasets, proposed predictor outperforms existing Roman Urdu hate speech predictors by a significant margin of 8.7% and 10.6% in terms of accuracy.
5. It presents a user-friendly web interface that enables more effective monitoring and intervention of hate content.

2 Literature review

Exponential growth of users over social media platforms requires advanced hate speech detectors capable of filtering unethical and hate content [6]. To promote peaceful societies, the development of computational predictors for hate speech detection is an active area of research, where aim of each newly developed predictor is to more precisely distinguish hate content from normal content [26]. Several symposiums, seminars and conferences are devoted for the exploration and smart processing of social media content related to hate speech detection [27]. Diverse platforms, such as Sentiment Analysis Symposium (SAS) [27], Workshop on Computational Approaches to Subjectivity [28], Sentiment and Social Media Analysis (WASSA) [29], Opinion Mining [30], Summarization and Diversification (WISDOM) and ACM conference for Knowledge Discovery and Data Mining (SIGKOD), provide an international forum for researchers from all over the world to share the most up-to-date studies on social data mining and its future applications in academia and industry for various

languages, such as English, Spanish, French, Dutch and Danish [31, 32]. In order to accelerate research related to natural language processing, these platforms also provide benchmark datasets for a variety of languages, such as English, Chinese, German and Arabic [13, 33, 34]. These platforms also attract researchers for the development of several applications for hate speech detection [13, 33, 34].

Although no track facilitates roman Urdu hate speech detection datasets, still various researchers have explored the potential of machine and deep learning approaches for the development of computational predictors capable of detecting offensive and hate content written in Roman Urdu language [23, 35, 36]. This section provides brief information related to existing predictors developed for Roman Urdu hate speech detection.

To fulfil the deficiency of benchmark corpora, Sajid et al. [23] developed Roman Urdu hate speech dataset that contains different comments related to YouTube videos. Authors manually tagged 16,300 comments into five distinct classes, namely, violence promotion, neutral, extremist, religious and threat. Moreover, to transform comments into statistical feature space, authors used most renowned bag of words-based feature representation approach named TF-IDF. Generated statistical feature space was passed to four different machine learning classifiers, i.e. logistic regression (LR), support vector machine (SVM), stochastic gradient descent classifier (SGD) and naive Bayes (NB). The support vector machine (SVM) classifier was found to be the highest performing classifier, with an accuracy of 77.45%.

Another Roman Urdu Hate-Speech and Offensive Language Detection (RUHSOLD)[1] dataset was developed by Rizwan et al. [24]. The dataset contains 10,012 tweets that were annotated in two different scenarios, namely, coarse-grained and fine-grained. In coarse-grained annotation, tweets were annotated against two classes: normal content and hate content. In fine-grained annotation setting, tweets were annotated against normal and four different classes of hate speech. Authors performed large-scale experimentation to transform tweets into statistical feature space using one embedding generation method FastText [24] and four different language models LASER [24], ELMo [24], BERT [24] and XLM-RoBERTa [24]. To generate more comprehensive feature space by extracting more semantic and discriminative patterns of words from tweets, they feed all the feature space generation methods with four different types of input features: uni-gram, bi-gram, tri-gram and quad-grams. To analyse the performance impact of different classifiers on Roman Urdu language, authors proposed three different deep learning classifiers, namely, CNN, BiLSTM, hybrid of Gradient Boosting Decision Tree (GBDT) and LSTM. Experimental results demonstrated that CNN classifier and BERT language model-based predictive pipeline produce highest performance for both coarse-grained and fine-grained datasets, by producing F1-scores of 0.90 and 0.75, respectively.

Akhter et al. [25] explored the potential of seven different machine learning predictors for Roman Urdu hate speech detection. Using publicly available YouTube comments dataset,[2] they transformed roman Urdu tweets into statistical feature space by taking different character and word n-gram features. Based on experimental results, authors concluded that among six different forms of character and word n-grams, character tri-grams are the most effective features for generating more comprehensive statistical feature space. The regression-based classification technique surpasses the other six machine learning classifiers. Additive logistic regression (LogitBoost) shows superior performance using character tri-gram.

Khan et al. [10] scraped several social media websites to collect around 90,000 tweets and retain only 5000 tweets related to Roman Urdu language. They annotated Roman Urdu tweets against three different categories: neutral-hostile, simple-complex and offensive-hate speech. They explored the potential of four machine learning classifiers (linear regression, SVM, Bayesian model and random forest) and one deep learning classifier (CNN) for Roman Urdu hate speech identification. Authors performed two level classification, at first level classifier discriminate between normal and hostile tweets and at second level the classifier further differentiates the hostile tweets into offensive and hate classes.

For both levels of classification, logistic regression outperforms all other machine and deep learning by producing F1-score of 0.906 for distinguishing between neutral and hostile tweets and 0.756 for discriminating between offensive and hate speech tweets.

Table 1 demonstrates the performance comparison of different predictive pipelines for Roman Urdu hate speech detection. There are several challenges associated with the Roman Urdu language that accounts for the lower predictive performance of these techniques. Unlike other resource-rich languages, Roman Urdu lacks sufficient linguistic resources to perform large-scale experimentation using deeper networks. Another challenge is the unique morphological nature of this language as a single word has numerous spelling variations which makes it difficult to generate comprehensive statistical vectors of words. Furthermore, unavailability of language processing tools such as, stemming, lemmatization and stop word lists are the major hindrances to text processing in Roman Urdu.

[1] https://github.com/haroonshakeel/roman_urdu_hate_speech.

[2] https://github.com/shaheerakr/roman-urdu-abusive-comment-detector.

Table 1 A comprehensive summary of existing Roman Urdu hate speech predictors

Predictors	Classification type	Dataset	Statistical feature representation	Classifier	Performance			
					Accuracy	Precision	Recall	F1-score
Khan et al. [10]	Binary-class	HS-RU-20	Count vectorizer	Logistic regression	0.84	0.84	0.97	0.90
					0.84	0.69	0.82	0.75
Sajid et al. [23]	Multi-Class	Roman Urdu dataset	uni + bi + tri-gram with L2 norm of TF-IDF	SVM	0.77			
Rizwan et al. [24]	Multi-Class	Coarse-grained	Language model	BERT+CNN-gram	0.90	0.90	0.90	0.90
		Fine-grained		BERT+CNN-gram	0.82	0.75	0.74	0.75
Akhter et al. [25]	Binary-Class	Roman Urdu dataset	Character-level tri-gram	LogitBoost				99.2
		Urdu offensive dataset		Simple logistic				95.9

3 Material and methods

This section describes details of proposed predictor for roman Urdu hate speech detection. Furthermore, it illustrates evaluation measures and benchmark datasets that are used to evaluate the integrity and generalizability of proposed predictor.

3.1 Proposed predictor

Recent advancement in deep learning has empowered natural language processing domain by facilitating development of useful computer-aided applications for different tasks such as sentiment analysis [37, 38], spam and non-spam email classification [39], fake news identification [40], information retrieval [41] and question answering systems [42, 43]. Initially, it was assumed that deep learning approaches produce better performance when they are trained on large datasets and these approaches remain fail to produce better performance over small datasets. However, annotation of large datasets is a costly and labour-intensive task [44, 45]. The concept of transfer learning, particularly the invention of word embedding methods, enables deep learning models to produce better performance over small datasets [46]. To date, several word embedding methods have been proposed [47–49], where motivation behind the development of each new method was to make use of large unlabelled textual data to generate statistical vectors against each word [48, 50]. Identical words must have statistical vectors in closer space while vectors of unidentical words should be distant in order to understand the semantic relation between distinct words [48].

At embedding layer, deep learning models make use of pre-trained word embeddings that facilitate them to extract more comprehensive information even from smaller datasets [51]. While training deep learning models, pre-trained word embeddings only facilitate models with pre-trained weights at only embedding layer, other layers of model are still initialized with random weights. Thus, the success of pre-trained word embeddings motivated researchers for the development of more generalized models that can provide pre-trained weights at all layers of model [51]. Following this idea, to date, several language models have been proposed [52, 53]. These models make use of large unlabelled textual data to understand the context and semantics of text [54]. Specifically, these models are trained by taking few words as input and predicting next word as output. Researchers have utilized pre-trained language models to perform different types of natural language processing tasks such as information retrieval, text classification [55], sentiment analysis [56], question answering system [54] and hate speech detection [17].

Most of the language models were originally developed for resource-rich languages and later researchers adopted them for other languages. However, these models have several hyper-parameters such as embedding vector size, number of layers, number of neurons in each layer, learning rate, weight decay, batch size and dropout [34, 57–59]. Researchers having a deeper understanding of language models can only smartly train them by selecting appropriate hyper-parameters. That is why low-resource languages still lack the availability of pre-trained language models. Such as for Roman Urdu language, researchers have trained BERT model over large unlabelled textual corpus, but still, they remained fail to produce decent

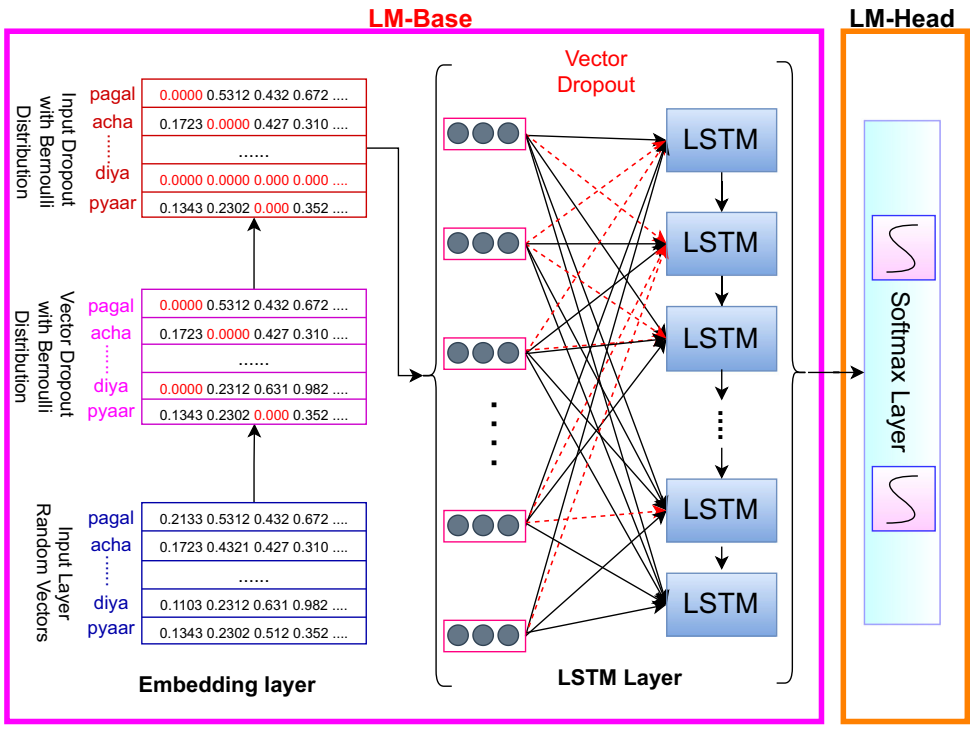

Fig. 1 Graphical illustration of proposed language model

performance for hate speech detection. On the other hand, a major drawback of deep learning models is their black-box nature which hinders explainability about contribution of different features towards prediction [60]. To open these black boxes, researchers are actively working to make decisions of deep learning predictors explainable and transparent to retrace results.

The paper in hand proposes an optimized architecture of ULMFIT language model for Roman Urdu language. Specifically, we train multiple variants of ULMFIT language model by altering its hyper-parameters such as number of layers, number of neurons, batch size and size of statistical vectors at embedding layer. Prime objective behind large-scale experimentation is to analyse the impact of different hyper-parameters for training language models over challenging language, writing of which does not follow any grammatical rules and standard dictionary. Furthermore, on top of ULMFIT model, we propose a novel classifier that makes use of two different types of attention, namely, attention head and aggregation attention. In the proposed classifier, we reap two different benefits from attention layers, firstly we utilize their potential to focus on more important words that facilitate classifier to accurately discriminate content into predefined classes. Secondly, we utilize them to make classifier decisions explainable and accountable. Following subsections briefly illustrate details of language model and proposed classifier.

3.1.1 Language model

Following working paradigm of ULMFIT language model, we designed an optimal RU-ULMFIT model that is graphically illustrated in Fig. 1. RU-ULMFIT consists of three different layers, namely, embedding layer, LSTM layer and output layer. RU-ULMFIT takes textual data in a sequential manner where its learning objective is to take a few words as input and predict next word as an output. In this way, RU-ULMFIT learns optimal weights of both embedding and LSTM layers. After training RU-ULMFIT language model in an unsupervised fashion, output layer also known as language model head is removed and language model base is connected with proposed classifier which is briefly described in Sect. 3.1.2. A comprehensive detail about all three layers of language model is provided in following subsections.

Embedding layer

The embedding layer takes a sequence of words as input and produces a 2-dimensional weight matrix where each row indicates a statistical vector of a word. The number of columns denotes the embedding dimensions and the

number of rows corresponds to unique words of vocabulary. The challenge of dense numerical representation of Roman Urdu words is addressed by empowering embedding layer with two different types of dropouts, namely, word dropout and vector dropout. Word dropout randomly makes a whole embedding vector of word zero and vector dropout randomly makes some values zero from a word embedding vector. These dropout strategies prevent model from learning specific words patterns and facilitate it for capturing semantic context of data.

LSTM layer

Statistical vectors of words generated by embedding layer are passed to LSTM layer, which is potentially more effective at identifying long-range dependencies in sequential data. The key difference between LSTM network and RNN is the use of memory cells instead of hidden layer updates. Additionally, LSTM is capable of analysing data at every time step and is far more resistant to the vanishing gradients problem. LSTM uses three different gates, namely; input gate, forget gate and output gate to regulate the flow of information. The input gate shown by i_t symbol in Eq. 1 regulates how much new information can be transmitted at the current time step. The information from the preceding time step is either lost or forwarded in the network based on the decision made by the forget gate, denoted as f_t in Eq. 2. The output gate designated as o_t in Eq. 3 uses current information to decide how much information is sent from the last time step. Mathematically, working of LSTM unit can be expressed as follow:

$$i_t = \sigma(w_i \cdot x_t + u_i \cdot h_{t-1}) \tag{1}$$

$$f_t = \sigma(w_f \cdot x_t + u_f \cdot h_{t-1}) \tag{2}$$

$$o_t = \sigma(w_\circ \cdot x_t + u_\circ \cdot h_{t-1}) \tag{3}$$

$$ci_t = tanh(w_c \cdot x_t + u_\circ \cdot h_{t-1}) \tag{4}$$

$$c_t = tanh(i_t \odot ci_t + f_t \odot ci_{t-1}) \tag{5}$$

$$h_t = (o_t \odot tanhci_t) \tag{6}$$

Here, x_t denotes a higher order residual vector fed each time step, and $[w_f, w_\circ, w_c, u_i, u_f, u_\circ]$ refer to weight matrices. A $\odot$ represents the element-wise product and c_t indicates the state of the memory cell. Model overfitting is a major issue during model training, leading to inaccurate results. This can be accomplished by limiting capacity of active neurons without modifying existing LSTM unit. This would allow LSTM to perform regularization on hidden weights using a generalized dropout mask. In contrast to traditional approaches instead of performing operations on hidden state vectors, the dropout mask acts between time step or memory cell updates before each forward or backward pass. The dropped neurons do not participate throughout a forward and backward pass, since the identical weights are maintained throughout numerous time steps.

Output layer

Softmax takes LSTM layer extracted features and predicts the next candidate word in the sentence. The predicted candidate word is compared with the actual word. Categorical cross-entropy function computes loss based on the difference between actual and predicted values. Computed loss is utilized to update weights of both LSTM and embedding layers. The mathematical expressions of softmax and categorical cross-entropy function are illustrated in Eqs. 7 and 8, respectively.:

$$f(S_i) = \frac{e_i^s}{\sum_j^C e_j^s} \tag{7}$$

$$CE = -\sum \sum_i^C t_i log(f(s_i)) \tag{8}$$

In this equation, t stands for one-hot encoding of ground truth label, s_i stands for probability score computed for each class in C, and f(s_i) stands for softmax activation used before calculating cross-entropy loss.

Algorithm 1 illustrates pseudo code of language model.

Algorithm 1 Pseudo code for Language Model Training

Input:
Roman Urdu Text **RU_Text**
Batch Size **batch-size**
Embedding Dimension **D**
Learning Rate **LR**
Dropout **Dropout**
Output: Trained Language Model (LM)
FUNCTION Data_pre-processing (RU_Text)
    Segregate RU_Text into words : *Words*
    Develop Vocabulary of unique words : *Vocab*
    Assign unique integer to each word of *Vocab*
    In text replace words with their respective integers : *Integer_Encoding*
    Prepare Input and Output
    for *i in range (0, number of samples-1)* **do**
        input.append(*Integer_Encoding*[i])
        output.append(shift one integer next in *Integer_Encoding*[i])
RETURN input, output
END FUNCTION
FUNCTION Language_Model_Initialization (D,Dropout)
    Initialize D dimensional random vector for each input word : W_vec
    Apply Vector dropout on each W_vec with Dropout : D_Vec
    Apply Word dropout on each D_Vec with Dropout : $Vectors$
    Initialize random weights of LSTM layer : $Weight_l$
    Apply Drop connect on $Weight_l$: $DCWeight_l$
    Compute Forward pass for LSTM layer $\underset{LSTM}{\rightarrow}(\underset{DCWeight_l}{\rightarrow}, Vectors)$: $LSTM\_output$
    Pass $LSTM\_output$ to Softmax Layer : $Initialized\_LM$
RETURN $Initialized\_LM$
END FUNCTION
1. inputs, outputs = Data_pre-processing (RU_text)
2. Model = Language_Model_Initialization (D, Dropout)
for *i in range(0, number of epochs - 1)* **do**
    for *i in range (0, total number of samples/batch-size)* **do**
        3. Take text samples and targets equal to batch size from inputs and outputs : $Batch\_input$, $Batch\_output$
        4. Pass $Batch\_input$ to the Model : $Batch\_Predictions$
        5. Compare $Batch\_output$ and $Batch\_Predictions$
        6. Compute cross-entropy loss : $Batch\_loss$
        7. Apply back propagation based on $Batch\_loss$ with Adam Optimizer and update according to Learning Rate LR : $Optimized\_model$
$Trained\_model = Optimized\_model$
RETURN $Trained\_model$

generalizability of vector distribution. The outputs of normalization layers are passed to two dropout layers which randomly drop values from feature vector. Processed features are further passed to fully connected layers which generate output of 50-dimensional space. These feature vectors are once more passed to the normalization and dropout layers prior to feeding these outputs to the softmax layers. Finally, predictive probabilities of softmax layers from two parallel pipelines are averaged using an ensemble layer that predicts class label. Following subsections briefly describe different modules of proposed classifier.

3.1.2 Proposed classifier

With an aim to more precisely distinguish between normal and hate content, we propose a novel classifier graphical illustration of which is shown in Fig. 2. Proposed classifier reaps the benefits of two different types of attention, namely, attention head and aggregation head, whereas at input it takes pre-trained RU-ULMFIT language model-based LSTM layer extracted features in parallel at both attention layers. Outputs of both attention layers are fed separately to normalization layers that increase

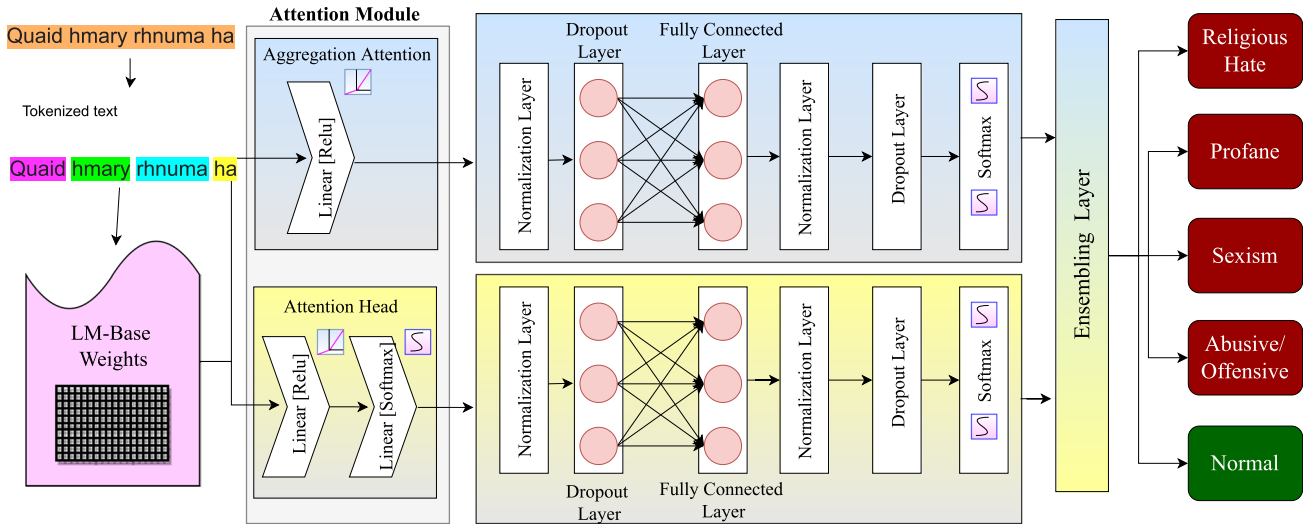

Fig. 2 Graphical illustration of proposed classifier

Attention module

Researchers have introduced diverse types of attention layers, where motivation behind each type comes from how we correlate words in a sentence or pay visual attention to different regions of an image [61]. In natural language processing, while discriminating textual samples into different classes, attention layers assign higher scores to more discriminative words which occur more frequently in one class and less frequently in other classes. In order to optimize input feature space based on their importance, proposed classifier makes use of two different types of attention mechanisms, namely, attention head and aggregated attention. Attention head aids to distinguish relevant set of words belonging to particular class, while aggregation attention learns correlation among different words of an input sentence in order to retain contextual dependency of a sentence.

Figure 3 illustrates working paradigm of attention head that assigns scores to input features based on their contribution to predict class labels. It can be seen from Fig. 3, textual words are passed to pre-trained language model, where LSTM layers extract and encode semantics and contextual information of words into d-dimensional statistical vectors by using hidden and cell states of LSTM unit. Equation 9 mathematically denotes computation of hidden state for an input word. Specifically, for current "*i*th" word hidden state "H_t" can be calculated using previous hidden state "H_{t-1}' and current input vector "X_i".

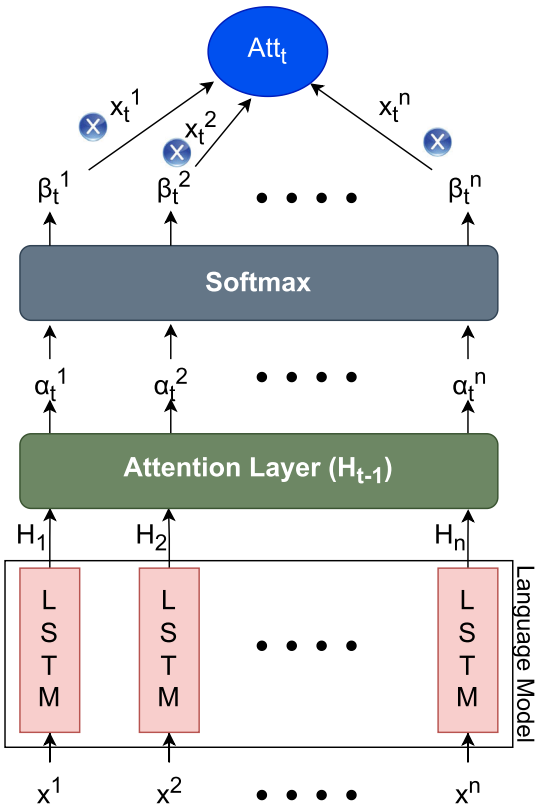

Fig. 3 Graphical illustration of attention head

$$H_t = F(H_{t-1}, X_i) \tag{9}$$

In Eq. 9, F represents nonlinear activation operation that

extracts nonlinear patterns of features. The hidden states of input sentence generated through LSTM layer are further passed to attention head layer. An attention head layer makes use of two linear layers that learn the linear relationship between different words and produces α vectors against each input word. All α vectors obtained against the corresponding input sentence are passed to softmax function that provides attention scores denoted by β. For a particular sentence $X = X_1, X_2 ....., X_n$, α_t^i value and attention wight β_t^i of "i_{th}" word at time stamp "t" are computed using Eqs. 10 and 11, respectively.

$$\alpha_t^i = \upsilon^T tanh(w_1 * [H_{t-1}, C_{t-1}] + w_2 X_i) \tag{10}$$

$$\beta_t^i = softmax(\alpha_t^i) = \frac{exp(\alpha_t^i)}{\sum_{k=1}^{n} exp(\alpha_t^i)} \tag{11}$$

In above equations, υ, w_1 and w_2 represent trainable parameters of the attention module, while H_{t-1} and C_{t-1} refer to previous hidden and cell states of an LSTM unit, respectively. Here, α^i estimates the significance of ith feature of input sentence and β^i refers to attention score of ith feature of input sentence. Finally, the weighted output Att_t of an input sentence is computed by multiplying each input vector $X_i$with its corresponding attention head score β^i, which is math mathematically expressed in Eq. 12:

$$Att_t = (\beta_t^1 X_t^1, \beta_t^2 X_t^2 ....., \beta_t^n X_t^n)^T \tag{12}$$

Hence, we obtain optimized feature vectors of input sentences by replacing d-dimensional statistical representation of input vector X_t with output Att_t of attention head. In contrast to X_t, where all input features are given equal importance, Att_t assigns higher scores to more informative features and reduces the impact of redundant and irrelevant features. Unlike attention head, the aggregation head is computationally less expensive as it comprises of only one fully connected layer that learns correlations among different words of sentences. Finally, the aggregated attention score Agg_t for an input sentence is computed by multiplying vector of each input word X_i with its corresponding output of fully connected layer denoted by C_i, which is mathematically expressed in Eq. 13. These optimized attention-based feature vectors of input sentence are then fed to classifier for predicting class label.

$$Agg_t = (C_t^1 X_t^1, C_t^2 X_t^2 ....., C_t^n X_t^n)^T \tag{13}$$

Normalization layer

The distribution of input features in all layers varies during training as the parameters of the preceding layers change, which makes it challenging to train neural networks. As a result, the training of models with saturating nonlinearity is very complex and lengthy due to the need for lower learning rates and rigorous parameter initialization. This behaviour is known as internal covariance shift. The weights obtained during prior iterations of a neural network are completely declared ineffective by internal covariance shift. As a result, generalization capacity of model is affected by internal covariance, which affects model convergence. To address the issue of internal covariance shift, batch normalization is incorporated in model architecture and carried out for each mini-batch across hidden layers of deep neural network. Batch normalization has been very effective in a variety of deep learning applications because it prevents the network input to output mapping from over-focusing any specific node of input distribution, which speeds up training, improves convergence and increases generalizability. We utilize considerably greater learning rates using batch normalization while being less concerned about initialization standards.

Dropout layer

To improve the quality of extracted hidden features by avoiding model overfitting, dropout layer incorporates a regularization factor that removes few connections among hidden layers [62]. In proposed Passion-Net predictor, we utilize Bernoulli distribution to uniformly drop connections among different neurons based on probability value. However, the neurons are randomly dropped during network training, while in testing phase, the network computes the dropping probability against each neuron.

Fully connected layer

Fully connected layers perform linear transformation for learning nonlinear relations of features. Primarily these layers facilitate predictor to assign class label based on entire context of features, because in fully connected layer, each neuron has connections with all other neurons of preceding layer.

Algorithm 2 illustrates pseudo code of proposed classifier.

Algorithm 2 Pseudo code of Classifier

```
Input:
Pre-Processed Text Processed_Text
Trained Language Model LM
Actual Labels L
Dropout D
Batch size Batch
Learning Rate LR
Output: Sample Labels (Labels)
FUNCTION Classifier (Attn_type,D)
    if Attn_type == "Attn_head" then
        Initialize Linear Layer AT_L1
        Apply RELU on AT_L1 AT_R1
        Apply Linear Layer on AT_R1 AT_L2
        Apply Softmax on AT_L2 Attn_out
    else
        Initialize Linear Layer AT_L1
        Apply RELU on AT_L1 Attn_out
    Pass Attn_out to Normalization Layer In_N1
    Pass In_N1 to Dropout Layer with D In_Dout
    Pass In_Dout to Linear Layer In_L1
    Pass In_L1 to Normalization Layer In_N2
    Pass In_N2 to Dropout Layer with D In_Dout2
    Pass In_Dout2 to Softmax Layer classifier
    RETURN classifier
END FUNCTION
FUNCTION Training (epochs, Batch, LR, attention, LM, Processed_Text, L)
    Model = Classifier (attention,D)
    for i in range(0, number of epochs - 1) do
        for i in range (0, total number of samples/batch-size) do
            Take text samples and labels equal to batch size from Processed_Text and L
              : Batch_input, Batch_output
            Pass Batch_input to the LM : Batch_Representations
            Pass Batch_Representations to Classifier : Batch_Predictions
            Compare Batch_output and Batch_Predictions
            Compute cross-entropy loss : Batch_loss
            Apply back propagation based on Batch_loss with Adam Optimizer and update
              according to Learning Rate LR : Optimized_model
RETURN Optimized_model
1. Att_Head_Model = Training (epochs, Batch, LR, Attn_head, LM, Processed_Text, L)
2. Get Att_Head_Prediction from Att_Head_Model
3. Agg_Attention_Model = Training (epochs, Batch, LR, Agg_Attention, LM,
  Processed_Text, L)
4. Get Agg_Attention_Predictions from Agg_Attention_Model
5. Labels = Average( Att_Head_Prediction, Agg_Attention_Predictions)
RETURN Labels
```

approaches have dominated machine learning-based approaches, by producing state-of-the-art performance for diverse natural language processing (NLP) [62, 68] tasks including hate speech detection [17], sentence classification [55], information retrieval [41] and machine translation [69]. However, practical applications are still dependent on machine learning-based approaches because of their understandable decisions. In contrast to this, decisions made by deep learning algorithms are not interpretable [70]. The major hindrance to use deep learning

3.1.3 Interpretability

Machine learning approaches are being used in different industrial processes ranging from automation in oil manufacturing [63] to energy management by forecasting energy generation and distribution [64, 65]. These approaches are also facilitating healthcare systems by facilitating applications competent in understanding hidden language of Genomic and Proteomic sequences [66, 67]. Moreover, in last few years, deep learning-based

models for real-world problems is their black-box nature towards features contribution and sole focus on performance.

In order to make deep learning models suitable for real-time applications, researchers are actively studying and exploring their feature extraction strategies. Understanding of these strategies also known as interpretability increases human trust in deep learning predictors by providing knowledge about features contributions for a particular decision. However, interpreting deep learning models feature extraction criterion is difficult due to their complex architectures, particularly for NLP applications that deal with discrete inputs. With an aim to make proposed predictor decisions interpretable, we utilized attention weights to highlight features contributions for a particular decision.

3.2 Benchmark datasets

This section briefly describes the details of two public benchmark datasets that are used to train and evaluate proposed Roman Urdu hate speech detector. We utilized RUHSOLD dataset, that was developed by Rizwan et al. [24]. In order to develop a comprehensive Roman Urdu hate speech dataset, authors first developed a lexicon of hateful words by searching for such keywords online and interviewing people. Developed lexicon contains abusive and derogatory words along with slurs or terms pertaining to religious hate and sexual language. Using lexicon words along with a separate collection of roman Urdu common words, authors retrieved 20,000 tweets and performed a manual preliminary analysis to find new slang, abuses and identify frequently occurring common words. Three independent annotators manually labelled Roman Urdu tweets. During the annotation process, all conflicts were resolved through majority vote among three annotators. Tweets on which a consensus cannot be reached or that could not reckon to provide sufficient information for class annotation were discarded. Furthermore, authors annotated tweets in two different settings. In first setting, tweets were annotated against two classes (Hate-Offensive, Normal) and named this annotated dataset as coarse-grained dataset. In second setting, they further annotated Hate-Offensive content against four more classes based on the level of hate. Authors referred to this type of annotated dataset as "fine-grained dataset". In terms of statistics, details of both coarse-grained and fine-grained datasets are summarized in Fig. 4.

Coarse-grained dataset contains 10013 tweets, where 4,664 tweet samples belong to "Hate-Offensive" class and 5349 tweet samples belong to "Normal" class. Authors split the dataset into train, test and validation sets with 70, 20 and 10 % ratio. This way, train, test and validation set contain 7209, 2003 and 801 tweet samples, respectively.

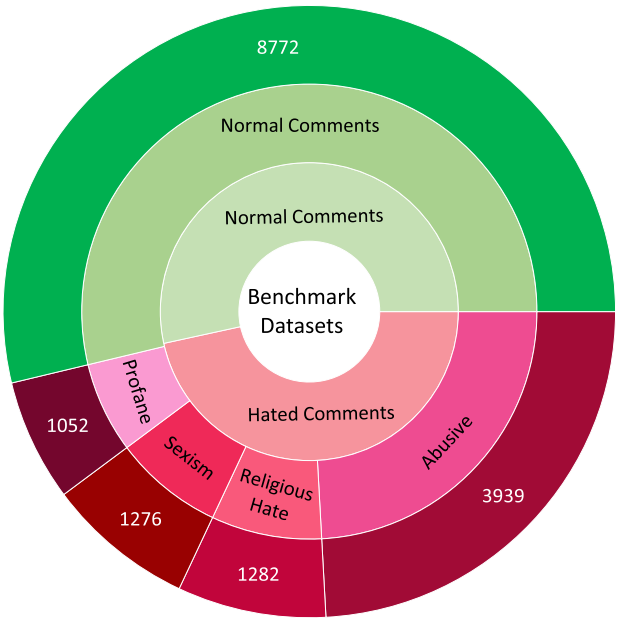

Fig. 4 Statistics of benchmark datasets

Dataset contains 20,821 unique words, where maximum number of words in a sentence are 71 and minimum number of words in a sentence is 1, while average number of words in a sentence are 16. On the other hand, there are 16,421 tweets in the fine-grained dataset, where 3939 tweets belong to "Abusive/Offensive" class, 1282 tweets belong to "Religious Hate" class, 1276 tweets belong to "Sexism" class, 1052 tweets belong to "Profane" class, and 8772 tweets belong to "Normal" class. This dataset was also split in train, test and validation sets with 44, 12, 44 split ratio. This way, train, test and validation set contain 7209, 2003 and 7209 tweet samples, respectively. There were 19,826 unique corpus words in this dataset, where maximum, minimum and average number of words in a sentence were same as for coarse-grained dataset. The label definitions of fine-grained dataset are summarized as follows:

- *Abusive/offensive* Profanity, strongly impolite, rude or vulgar language expressed with fighting or hurtful words in order to insult a targeted individual or group
- *Religious hate* Language used to express hatred towards a targeted individual or group based on their religious beliefs or lack of any religious beliefs and the use of religion to incite violence or propagate hatred against a targeted individual or group
- *Sexism* Language used to express hatred towards a targeted individual or group based on gender or sexual orientation
- *Profane* The use of vulgar, foul or obscene language without an intended target

- *Normal* Content that does not fall into any of the above categories.

3.3 Evaluation measures

In order to evaluate the performance of proposed detector, we utilize six distinct evaluation measures that have been widely used to access the performance of existing hate speech detectors [35, 36, 71–73]. We have only summarized these metrics because they are described in depth in several research studies [71–73]. The most intuitive performance metric is accuracy, which computes the ratio of correctly predicted samples to total samples. Precision indicates what percentage of all projected positive outcomes really fall into the positive class. While recall indicates what fraction of all samples in a particular class was accurately predicted in the same class by the classifier. The F1-score takes the harmonic mean of precision and recall scores. As a result, this score considers both false positives and false negatives. The mathematical expressions to compute the performance values of accuracy, precision, recall and F1-score are illustrated in Eq. 14.

$$f(x) = \begin{cases} Accuracy = \dfrac{TP + TN}{TP + TN + FP + FN} \\ Precision = \dfrac{TP}{TP + FP} \\ Recall = \dfrac{TP}{TP + FN} \\ F1 - Score = \dfrac{2 \times Precision \times Recall}{Precision + Recall} \end{cases} \tag{14}$$

In addition to these four evaluation metrics, AUC-ROC and AUPRC are used to examine the hate speech detector. The AUC-ROC curve is a performance assessment metric for classification tasks that access the performance of a model at various threshold values. AUC indicates the degree or measure of separability, while ROC is a probability curve. It indicates how well the model can discriminate between classes. The AUC indicates how well the model predicts 0 s as 0 s and 1 s as 1 s. This graph shows the relationship between Recall or True Positive Rate (TPR) and False Positive Rate (FPR). The AUC of an excellent model is close to 1, indicating that it has a high level of separability. AUC around 0 indicates a bad model, which implies it has the lowest measure of separability. In reality, it indicates that the outcome is reciprocated. It predicts 0 s to be 1 s and 1 s to be 0 s. When AUC = 0.5, the model has no ability to distinguish between classes.

AUPRC is a classification evaluation technique that visualizes performance over a range of thresholds. Apart from visual examination of a PR curve, the region under a PR curve (AUPRC) is frequently used in algorithm assessment as a generic measure of efficiency, regardless of any specific threshold or operating stage. The point (recall = 0, precision = 1) on a PR curve that corresponds to a decision threshold of 1 is the top left corner, where every case is categorized as negative since all projected probabilities are less than 1. Precision is low at the lower right corner of a PR curve, where recall = 1. Because all projected probabilities are larger than 0, this equates to a 0 decision threshold, where every sample is classified as positive. The points in between that make up the PR curve are calculated by calculating the accuracy and recall for various decision thresholds between 1 and 0.

4 Experimental setup

Proposed predictor is developed on top of seven different APIs, namely, Pytorch,[3] Pandas,[4] Fastai,[5] dash,[6] Plotly,[7] matplotlib[8] and numpy.[9] With an aim to design more robust and precise predictor, we optimized different hyper-parameters of language model and classifier. Table 2 illustrates search space of each hyper-parameter for optimization along with optimal values of all hyper-parameters. Furthermore, for a fair performance comparison with existing predictors [10, 23–25], following evaluation criteria of existing predictors [10, 23–25], we utilized standard train and validation sets to train classifier and for the optimization of hyper-parameters. Finally, we used test sets to evaluate the performance of proposed predictor.

5 Results

This section quantifies the performance of proposed classifier with random embeddings at different settings of architecture such as with different attention layers. It illustrates performance gain achieved by proposed classifier when random embedding layer is replaced with pre-trained language model. Furthermore, it summarizes the performance trends of the proposed classifier when the language model was trained with different LSTM layers. It also highlights the impact of batch size and learning rate on the predictive performance of proposed classifier. Finally,

[3] https://pytorch.org/.
[4] https://pandas.pydata.org/.
[5] https://www.fast.ai/.
[6] https://dash.plotly.com/reference.
[7] https://plotly.com/.
[8] https://matplotlib.org/.
[9] https://numpy.org/.

Table 2 A comprehensive summary of search space and optimal values of hyper-parameters

Expression	Search space	Optimal value	
		Language model	Classifier
Number of LSTM layers	1,2,3	1	–
Number of neurons	100, 150, 200, 250, 300	200	–
Weight Decay	$1e^{-1}$, $1e^{-2}$, $1e^{-3}$, $1e^{-4}$, $1e^{-5}$	$1e^{-3}$	$1e^{-5}$
Batch size	16, 32, 64, 128, 256	32	32
Dropout	$1e^{-1}$, $1e^{-2}$, $1e^{-3}$, $1e^{-4}$, $1e^{-5}$	$1e^{-3}$	$1e^{-3}$
Embedding size	32, 64, 128, 200, 256	200	200
Learning rate	$1e^{-2}$, $1e^{-3}$, $2e^{-3}$, $3e^{-3}$, $4e^{-3}$, $5e^{-3}$, $6e^{-3}$, $15e^{-4}$, $25e^{-4}$, $35e^{-4}$, $45e^{-4}$, $55e^{-4}$	$1e^{-2}$	$1e^{-3}$

it compares the performance of proposed classifier with existing Roman Urdu hate speech predictors.

5.1 In-depth performance analysis of proposed classifier

Table 3 illustrates the impact of different kinds of attention mechanisms on the predictive performance of proposed classifier. It also highlights predictive performance of proposed classifier at two different settings of pipeline by performing early and post-fusion of both attention layers. Furthermore, it quantifies the impact of randomly initialized word embeddings and pre-trained language model on the predictive performance of proposed classifier across coarse-grained and fine-grained versions of benchmark datasets in terms of accuracy, precision, recall and F1-score.

It is evident in Table 3, using random word embeddings, among all five versions of classifier, two different versions, namely, classifier without any attention layer and classifier version with attention head layer produced almost similar and highest performance figure of 87% across all four evaluation metrics over coarse-grained dataset. However, over fine-grained dataset, classifier with early fusion of attention head and aggregation attention achieve the highest performance figure of 77% across all four evaluation metrics followed by 75% performance achieved by classifier version with late fusion of attention head and aggregation.

With the use of pre-trained language model, the performance of all five different versions of classifiers gets improved. More specifically, over both coarse- and fine-grained datasets, proposed classifier version without using any attention layers manages to boost performance up to 96% and 89%, respectively, whereas the performance of other versions of classifier which reap the benefits of attention layers reach the peak of 99% and 93% in terms of precision, recall, accuracy and F1-score. Among all five versions, proposed classifier version with late fusion of attention head and aggregation achieves the best performance across both benchmark datasets.

Figure 5 sheds light on confusion matrices of all four attention-based versions of classifiers along with pre-trained language model. Over coarse-grained dataset, classifier versions using late fusion of both attentions and standalone attention head manage to produce almost similar and best true positive scores. While in terms of true negative score, late fusion-based version of the classifier beats the attention head-based classifier version with a minute margin. Hence, it can be inferred that late fusion of both attentions has superior performance than other attention-based versions of classifier. Over fine-grained dataset, once again late fusion-based classifier produces highest true positive score for the religious hate class. However, late fusion and all other versions of classifier have shown high performance fluctuation for four different classes including sexism, profane, abusive and normal. It is evident from Fig. 5 that all four versions of classifiers remain fail to accurately predict samples of three classes, namely, religious, sexism and profane, and most of the samples from these classes are wrongly categorized into abusive classes. Primarily, the context of these classes is almost similar to abusive class which hinder classifiers from accurately distinguishing between them and eventually wrongly categorized them into abusive class.

Performance comparison of all four versions of classifier based on different attention mechanisms in terms of AUROC and AUPRC is graphically illustrated in Fig. 6. It is evident from Fig. 6 that classifier versions based on attention head and late fusion of both attentions produced almost similar and highest performance across both benchmark datasets. In contrast to this, classifier versions using aggregation attention and early fusion of both attention mechanisms do not perform well over both datasets. Hence, late fusion of both attentions enables classifier to capture more discriminative features leading

Table 3 Impact of language model and attention mechanisms on the performance of proposed classifier across coarse-grained and fine-grained dataset in terms of four different evaluation metrics

Classifier	Coarse-grained dataset				Fine-grained dataset			
	Precision	Recall	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score
Simple classifier	0.877	0.877	0.877	0.877	0.729	0.737	0.737	0.731
Attention head-based simple classifier	0.87	0.87	0.87	0.87	0.76	0.759	0.759	0.755
Aggregation attention-based simple classifier	0.82	0.82	0.82	0.82	0.722	0.732	0.732	0.724
Early fusion of attention head and aggregation-based simple classifier	0.859	0.859	0.859	0.859	0.76	0.765	0.765	0.761
Post-fusion of attention head and aggregation-based simple classifier	0.855	0.855	0.855	0.855	0.747	0.753	0.753	0.748
Language models + classifier								
pre-trained language model + classifier	0.961	0.961	0.961	0.961	0.892	0.895	0.895	0.892
pre-trained language model + attention head-based classifier	0.985	0.985	0.985	0.985	0.923	0.923	0.923	0.923
pre-trained language model + aggregation attention-based classifier	0.983	0.983	0.983	0.983	0.919	0.919	0.919	0.918
pre-trained language model + early fusion of attention head and aggregation-based classifier	0.98	0.98	0.98	0.98	0.921	0.921	0.921	0.92
pre-trained language model + post-fusion of attention head and aggregation-based classifier	0.987	0.987	0.987	0.987	0.926	0.926	0.926	0.926

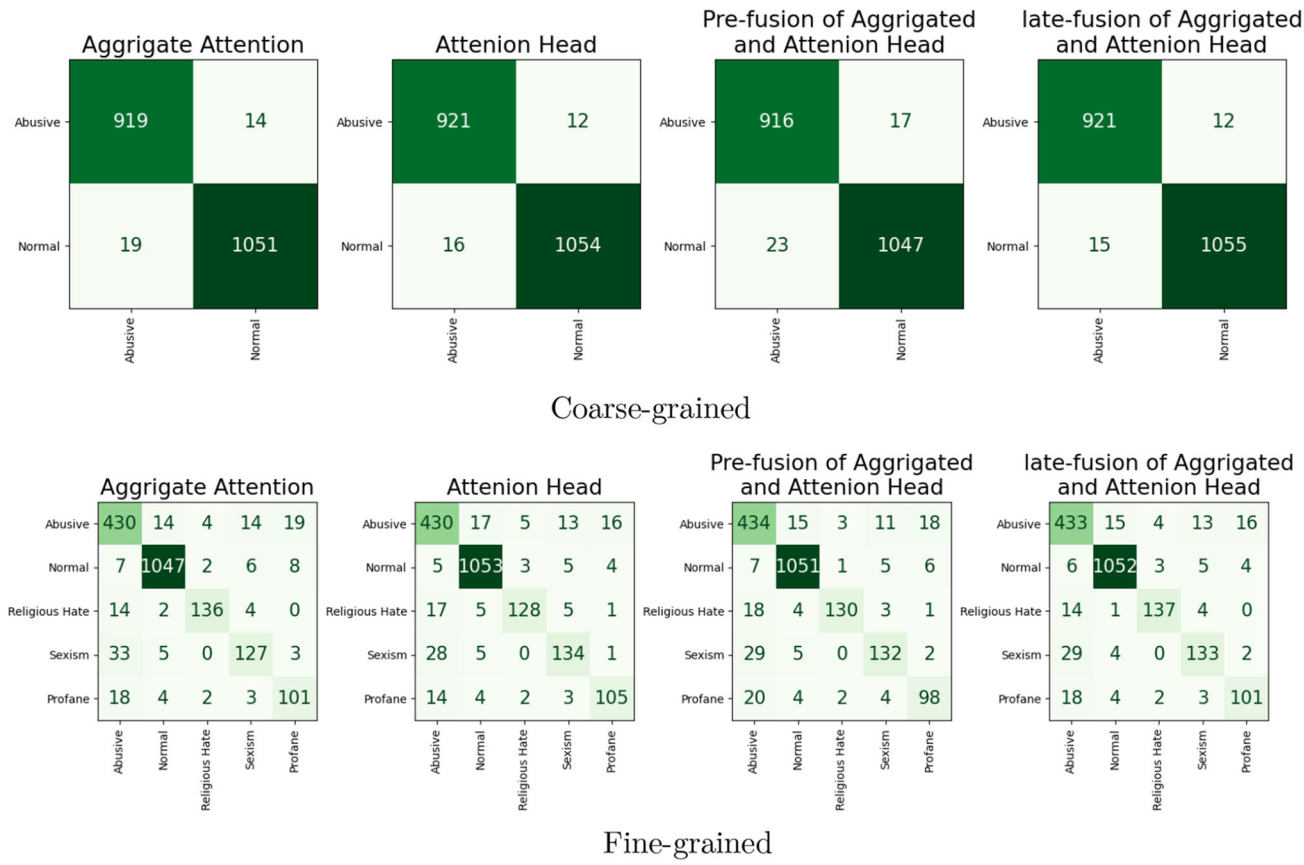

Fig. 5 Confusion matrices of coarse-grained and fine-grained datasets at different settings of proposed classifier

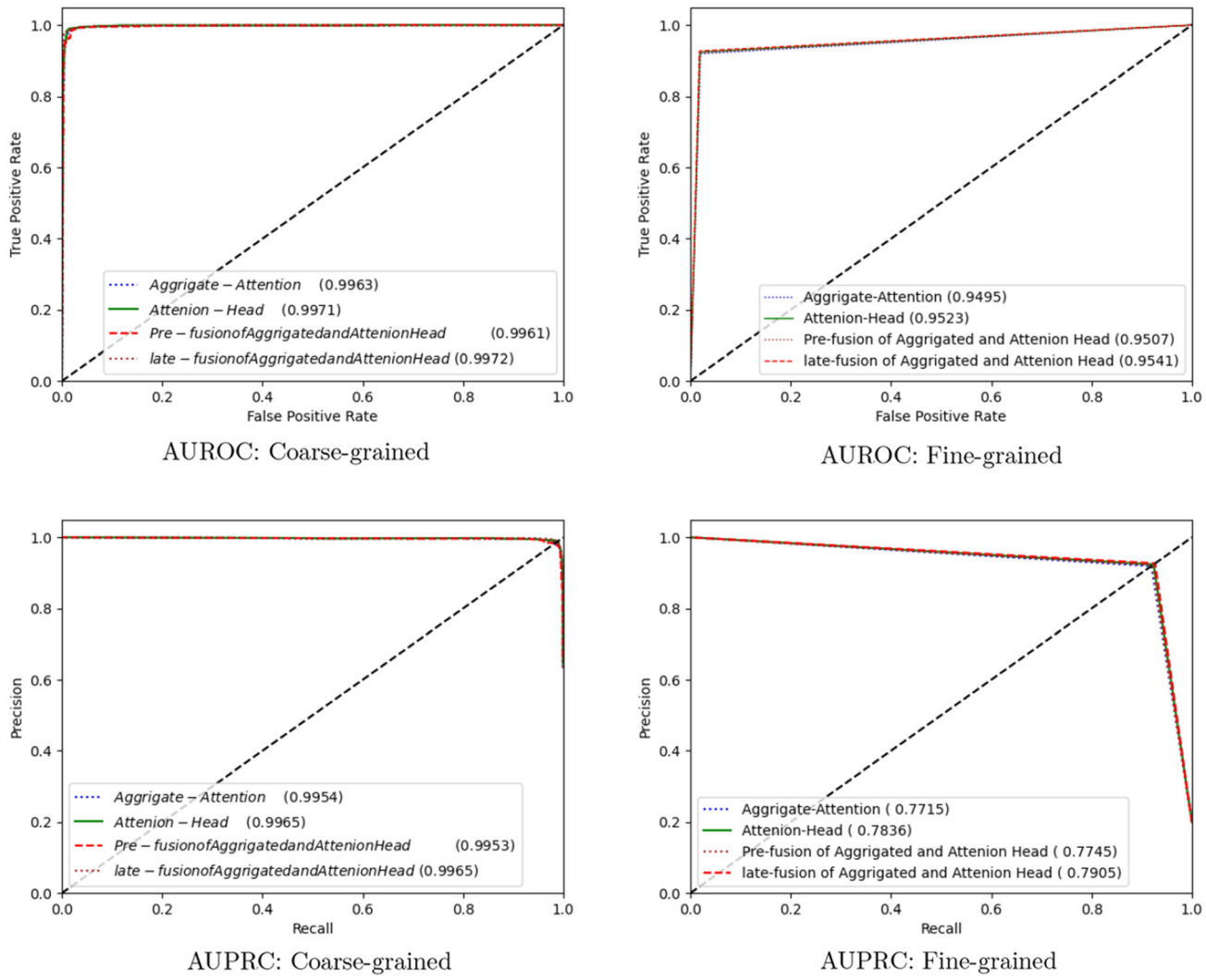

Fig. 6 AUROC and AUPRC

towards better performance in comparison with other versions of classifier across both datasets.

Figure 6 depicts that over fine-grained dataset, classifier versions utilizing aggregation attention and early fusion produce performance scores with a negligible difference and secure 3rd and last position in terms of performance. Moreover, attention head-based classifier manages to beat performances of aggregated attention and early fusion-based classifiers which secures 3rd and last position over fine-grained dataset. Hence, classifier version using late fusion of both attentions again outperforms other versions of classifiers over both benchmark datasets.

In a nutshell, across different evaluation measures, all different versions of classifiers achieve higher performance figures on coarse-grained dataset as compared to fine-grained dataset. Mainly, this is because coarse-grained dataset has only two classes, namely, hate and normal content, whereas fine-grained dataset has five different classes. It seems classifier precisely discriminates between hate and non-hate content but while discriminating hate content into further categories it is less precise as compared to discrimination between hate and non-hate. Performance of all five versions of classifier boosted when randomly initialized word embeddings are replaced with pre-trained language model. This performance booster illustrates that language model captures more comprehensive patterns among hate and non-hate content. Among different versions of proposed classifier, classifier with late fusion of both attention layers produces more prominent performance.

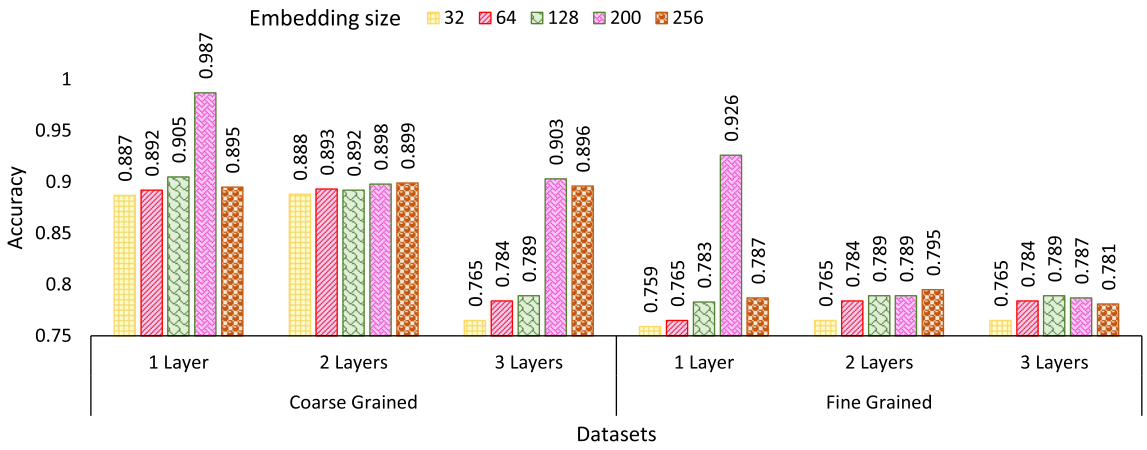

Fig. 7 Proposed classifier performance analysis using pre-trained language model with different number of LSTM layers and embedding sizes

5.2 Impact of neural architecture and model training-related hyper-parameters on the performance of proposed classifier

Figure 7 describes the impact of different numbers of LSTM layers in language model and embeddings sizes on the performance of proposed classifier that makes use of two different attention layers with late fusion strategy. In language model, a number of LSTM layers are tweaked from 1-to-3 and neural embedding sizes are changed from 32 to 256.

As shown by Fig. 7, over coarse-grained dataset, proposed classifier achieves better performance with one layer LSTM-based pre-trained language model using an embedding size of 200. With the increase of LSTM layers in pre-trained language model, the performance of proposed classifier significantly drops such as with two and three LSTM layers, accuracy declines from 99% to 90%. However, with the increase of embedding sizes up to 200, performance of proposed classifier improves regardless of number of LSTM layers in pre-trained language model.

Over fine-grained dataset, proposed classifier once again achieves better performance with single layer LSTM-based pre-trained language model and embedding size of 200. Like coarse-grained dataset, here, accuracy of proposed classifier also drops by a significant margin of 13% with the increase of LSTM layers in language model. Furthermore, change of embedding size from 32 to 256 proves effective across different numbers of LSTM layers such as one, two and three LSTM layers based pre-trained language models generate more effective embeddings using the sizes of 200, 256 and 128, respectively. However, it is important to mention that change of classifier performance with different embedding size is more prominent with only one LSTM layer based pre-trained language model and almost negligible with two and three LSTM layers based pre-trained language models.

Overall, the combination of using only one LSTM layer and higher embedding dimensions proves effective for language model training. Proposed classifier achieves the best performance across both versions of dataset using pre-trained language model with one layer of LSTM and an embedding size of 200. Also, proposed classifier achieves higher accuracy across all different settings over coarse-grained dataset as compared to fine-grained dataset. Using the optimized architecture of pre-trained language model, we optimize different hyper-parameters of proposed classifier to further enhance the predictive performance for roman Urdu hate speech detection.

Figure 8 illustrates the change in accuracy with the change of batch size from 16-to-256 over coarse-grained and fine-grained variants of benchmark dataset. Over both variants of datasets, proposed classifier achieves best

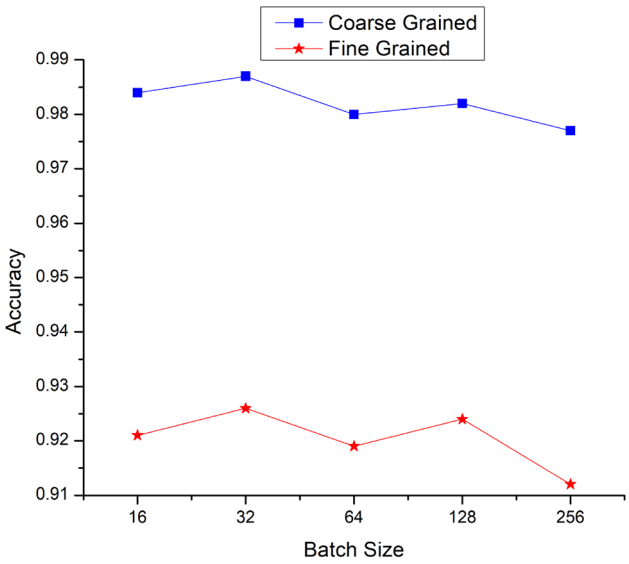

Fig. 8 Proposed classifier performance analysis using different batch sizes over coarse-grained and fine-grained version of benchmark dataset

performances around 0.987 and 0.926 with batch size of 32. Second best performance on coarse-grained around 0.984 and on fine-grained around 0.924 is achieved using batch size of 16 and 128, respectively. On both variants of dataset, proposed classifier achieves lower performance around 0.977 and 0.912 using the batch size of 256. Overall, optimal value of batch size brings a small improvement of around 1% in the performance of proposed classifier on coarse-grained dataset and a decent improvement of around 2% on fine-grained dataset.

Figure 9 demonstrates the impact of different learning rates on the performance of proposed classifier across both variants of the dataset. As is shown by Fig. 9, change in learning rate from 0.005 to 0.006 does not significantly influence the performance of proposed classifier produced on coarse-grained dataset. Proposed classifier performance of 98% remains constant at most learning rates except 0.005, 0.0045, 0.0055 and 0.006 where it drops by almost 1%. Over fine-grained dataset, proposed classifier performance only increases from 90% to 92% when learning rate is changed from 0.005 to 0.001; however, afterwards it almost keeps dropping by slight margin before ending around 89% with 0.006 learning rate.

Overall, across both variants of dataset, proposed classifier accuracy improves by almost 2% with the change in learning rate. Proposed classifier attains the peak accuracy on coarse-grained variant of dataset using learning rate of 0.001-to-0.0035, and on fine-grained variant of dataset using learning rate of 0.001. In a nutshell, proposed classifier achieves the best performance using the batch size of 32 and learning rate of 0.001 over coarse-grained and fine-grained versions of dataset.

5.3 Performance comparison of proposed predictor with existing predictors

Table 4 compares the performance of proposed and existing Roman Urdu hate speech predictors over two benchmark datasets in terms of four different evaluation measures. Over coarse-grained dataset, from existing predictors, two predictors, namely, meta predictor based on SVM, AdaBoost, random forest and deep learning predictor based on words n-gram BERT embeddings and convolutional neural network (CNN) classifier produce similar and best predictive performance of 90% in terms of accuracy, precision, recall and F1. Afterwards, four different predictors named; BERT, LAMB optimizer-based BERT, combination of LASER and BERT embeddings with GBDT classifier and conjunction of domain-specific pre-trained words n-gram embeddings and CNN classifier produce 2nd highest performance figure of 89% in terms of all evaluation measures. Four different predictors, namely, words n-gram XLM-RoBERTa embeddings-based CNN classifier, RomUrEm+CNN, RomUrEm and FastText classifier produce similar and 3rd highest performance by achieving 88% accuracy, precision, recall and F1-score. XLM-RoBERTa managed to produce 85% accuracy that is even less than the performance of pre-trained word embeddings-based CNN predictor which produce 87% accuracy. Furthermore, words n-grams based CNN predictor produces 81% performance. ELMO predictor produces 79% performance and among all existing predictors, LASER predictor produces least performance figure of 76%. On the other hand, proposed predictor, that reaps the benefits of pre-trained language model, different attention mechanisms and neural optimization strategies, manages to outperform all existing best performing predictors by significant margin of 9% in terms of accuracy, precision, recall and F1-score.

Over fine-grained datasets, from existing predictors, once again deep learning predictor based on words n-gram BERT embeddings and CNN classifier produce best performance figures of 82%, 75%, 74% and 75% in terms of accuracy, precision, recall and F1-score, respectively. Words n-gram RoBERTa embeddings-based CNN predictor produce 2nd highest performance figure of 81%, 74%, 71% and 72% in terms of accuracy, precision, recall and F1-score, respectively. Two predictors, namely, LAMB

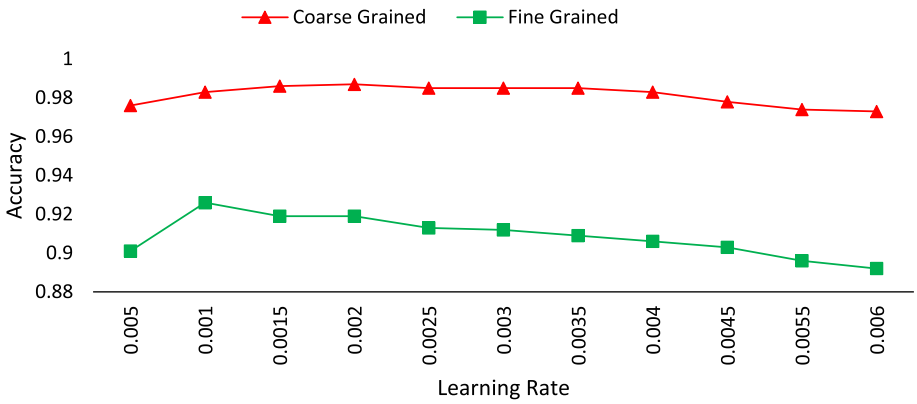

Fig. 9 Proposed classifier performance analysis using different learning rates over coarse-grained and fine-grained version of benchmark dataset

Table 4 Performance comparison of proposed and existing predictors over coarse-grained and fine-grained datasets in terms of four different evaluation metrics

Classifier	Coarse-grained dataset				Fine-grained dataset			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
LASER	0.76	0.76	0.76	0.76	0.67	0.59	0.52	0.54
ELMo	0.79	0.79	0.79	0.79	0.6	0.66	0.5	0.55
BERT	0.89	0.9	0.89	0.89	0.77	0.72	0.65	0.67
XLM-RoBERTa	0.85	0.85	0.85	0.85	0.79	0.7	0.75	0.72
FastText	0.88	0.88	0.88	0.88	0.77	0.69	0.63	0.66
RomUrEm	0.88	0.88	0.88	0.88	0.79	0.76	0.63	0.67
LSTM+GBDT	0.54	0.58	0.51	0.38	0.53	0.2	0.2	0.15
BERT+LASER+GBDT	0.89	0.89	0.89	0.89	0.8	0.73	0.7	0.71
FastText+CNN	0.87	0.87	0.87	0.87	0.78	0.7	0.67	0.68
SVM+RF+AB	0.9	0.9	0.9	0.9	0.77	0.73	0.62	0.67
BERT+LAMB	0.9	0.9	0.89	0.89	0.8	0.72	0.73	0.72
RomUrEm+CNN	0.88	0.89	0.88	0.88	0.72	0.63	0.52	0.55
BiLSTM with Attention	0.86	0.86	0.85	0.85	0.76	0.67	0.63	0.65
BERT+CNN-gram	0.9	0.9	0.9	0.9	0.82	0.75	0.74	0.75
XLM-RoBERTa+CNN-gram	0.88	0.88	0.88	0.88	0.81	0.74	0.71	0.72
FastText+CNN-gram	0.81	0.81	0.8	0.8	0.66	0.45	0.41	0.42
RomUrEm+CNN-gram	0.89	0.89	0.89	0.89	0.75	0.68	0.61	0.64
Proposed	**0.987**	**0.987**	**0.987**	**0.987**	**0.926**	**0.926**	**0.926**	**0.926**

LASER: Learning Approach for Speculative Execution and Replication, ELMo: Embeddings from Language Model, BERT: Bidirectional Encoder Representations from Transformers, XLM-RoBERTa: Cross-lingual Language Model-Robustly Optimized BERT Pre-training Approach, FastText, RomUrEm: Domain-specific Roman Urdu Embedding, LSTM+GBDT: ensemble of Long Short-Term Memory and Gradient Boosting Decision Tree, BERT+LASER+GBDT: pre-trained multilingual BERT and LASER embedding with GBDT classifier, FastText+CNN: Fasttext embedding along with CNN, SVM+RF+AB: ensemble of linear SVM, random forest and Adaboost, BERT+LAMB: pre-trained BERT embedding with LAMB optimizer, RomUrEm+CNN: Domain-specific Roman Urdu Embedding with CNN, BiLSTM with Attention, BERT+CNN-gram: pre-trained words n-gram based BERT embedding with CNN classifier, XLM-RoBERTa+CNN-gram: words n-gram based XLM-RoBERTa embeddings with CNN classifier, FastText+CNN-gram: words n-gram based Fasttext embeddings with CNN classifier, RomUrEm+CNN-gram: words n-gram based Domain-specific Roman Urdu embedding with CNN classifier

Bold values illustrate predictor's highest performance values

optimizer-based BERT and combination of LASER and BERT embeddings with GBDT classifier, manage to produce 3rd highest performance of 80% in terms of accuracy. Afterwards, XLM-RoBERTa and RomUrEm predictors produce 79% performance that is better than BERT, FastText and ML meta predictor (SVM+AB+RF) that produce 77% accuracy. CNN with pre-trained Fasttext embeddings produces performance figures 78%, 70%, 67% and 68% in terms of accuracy, precision, recall and F1-score. Proposed predictor achieves the peak performance of 92.6% in terms of all measures and this predictive performance is 11% better than existing best performing predictor.

A critical performance analysis indicates that among existing predictors best performer on both coarse- and fine-grained datasets is words n-gram BERT embeddings and CNN classifier based predictor. BERT-based predictor has produced state-of-the-art performance for several NLP classification tasks related to resource-rich languages like English but here for Roman Urdu language although it remain top performer, but it remains fail to produce decent performance due to several possible variations of same word that exist in Roman Urdu text. These advanced language models make use of masking strategy to acquire context aware representation of text; however, due to high variability of same words these models lack to explore the semantics of words. Furthermore, diversity of words-based performance degradation is also more prominent from the comparison of simple BERT predictor and n-gram based BERT predictor. Simple BERT-based predictor remains fail to capture semantics and correlations of words due to diversity of same words. Furthermore, in n-gram based settings the diversity of words is overcome at some level and it produces better performance. To overcome challenge of high variability of words, we train simple language model which is capable of learning semantics of words by

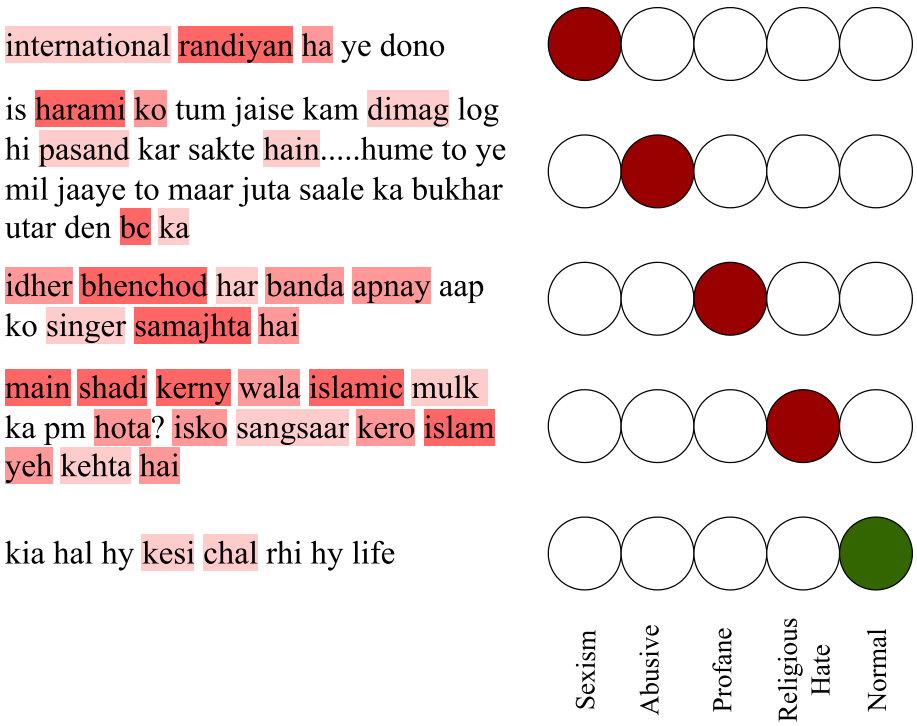

Fig. 10 A subset of test samples representing interpretability of proposed predictor

predicting every next word. Furthermore, proposed classifier also utilizes the potential of two different attention mechanisms to learn the discriminative features and a precise yet robust classifier to make accurate discrimination between hate and non-hate content. Furthermore, existing predictors are not well generalized as they produce highly variable performance on both datasets, e.g. machine learning meta predictor produces highest performance on coarse-grained dataset but over fine-grained dataset, four other predictors produce better performance as compared to meta predictor. Similarly, 2nd top performer of fine-grained dataset word n-grams based XLM-RoBERTa and CNN classifier falls on 3rd number for coarse-grained dataset.

Overall, on fine-grained dataset, all predictors achieve lower performances across all evaluation metrics as compared to their performances on coarse-grained dataset. Mainly this performance drop is due to large number of classes in fine-grained dataset. Specifically, coarse-grained dataset has only two class labels (hate, normal) and fine-grained dataset has five labels (sexism, abusive, profane, religious hate and normal). Moreover, in fine-grained dataset, context of samples is similar that confuse predictors and drop their predictive performance.

6 Proposed predictor interpretability

Proposed predictor produces decent performance over both public benchmark datasets; however, due to black-box decisions of proposed predictor, we utilize attention weights to highlight important words on the basis of which predictor decides class label. Attention layers focus on important features and assigns high score to the important features. The classification layer of predictor decides class label based on the informative features; hence, classification layer also focuses on the features which get high scores from attention layer.

In order to demonstrate the class label decision of the proposed predictor the interpretability module categorizes words into five different groups based on threshold values of attention weights ranging from 0 to 1. Each group of words is represented with a unique red colour depending on their contribution to decision-making. The opacity of red colour varies based on attention score of words, for instance, the darkest red colour indicates the group of words with highest and lightest red colour reflects the words having the least impact on decision-making.

Figure 10 illustrates interpretable decisions of proposed predictor for five different data samples. As shown in each sample, the darkest red colour indicates words that have an attention weight between 0.9 and 1.0 and are the most significant for detecting class labels. Likewise, 2nd most important group of words involved in class label decisions have attention weights between 0.7 and 0.89. Moreover,

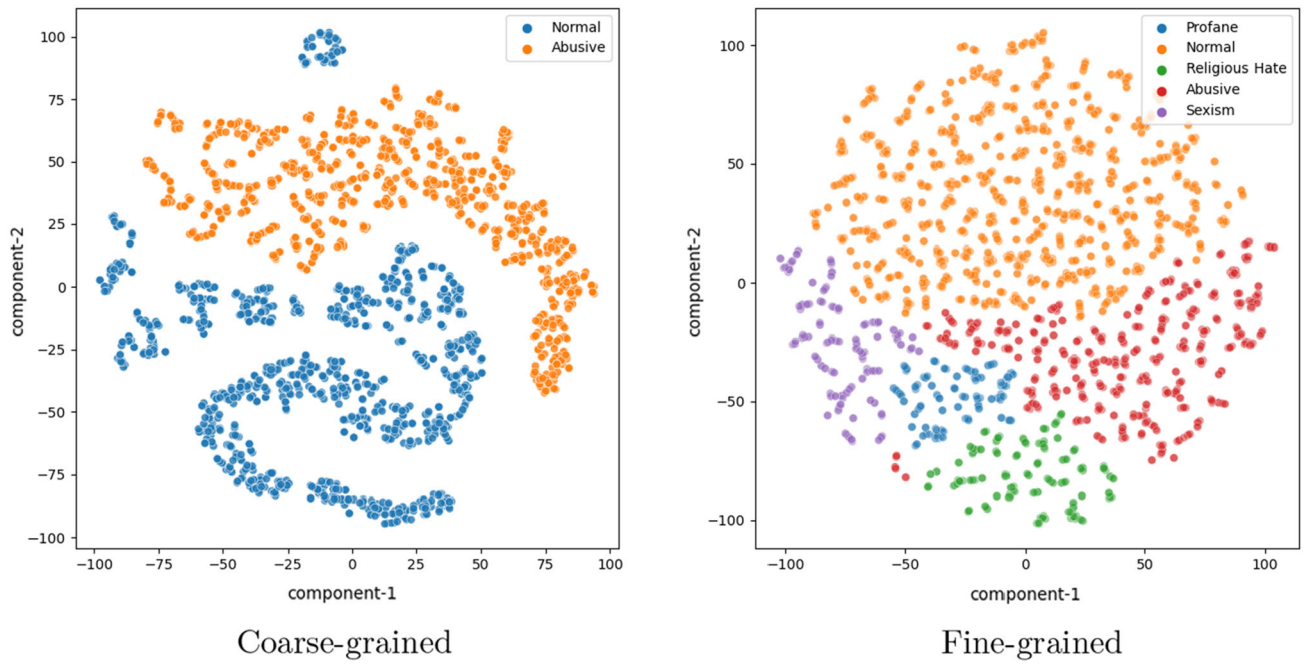

Fig. 11 Visualization of internal representation of proposed classifier

the 3rd and 4th most important group of words have their attention weights ranging from 0.5 to 0.69 and 0.20 to 0.49, respectively.

In the first sample, proposed predictor has assigned correct class label and also the interpretability module highlighted the most relevant words that put this sample into sexism class. Similarly, in 2nd, 3rd and 5th samples assigned classes and highlighted words are correct. However, in 4th sample although the predictor assigned correct class label and five words get highest and same attention scores. Among these five words, only two words belong to assigned class label but other three words do not belong to assigned class keywords, hence do not support decision of predictor. It indicates in this particular sample although proposed predictor took right decision but due to incorrect attention scores of three irrelevant words the confidence of decision is low.

7 Ablation study

To make sure proposed predictor generalizability for real-time hate speech detection, we performed two different ablation studies. In first study, we extracted and visualized internal representation of predictor. Here, objective is to make sure weather learned representation contains enough discriminative patterns which classifier can utilize for accurate prediction of relevant classes. To perform this analysis, extracted feature vectors are passed to TSNE, which reduces the dimensions and produces a two-dimensional feature space. Further reduced feature space is visualized by assigning different colours to samples of distinct classes. Figure 11 illustrates predictor internal representation for two benchmark datasets test sets. A high level analysis of Fig. 11 reveals that clusters of different classes are significantly separated from each other. This visual analysis concludes that proposed predictor is competent in extracting discriminative patterns among samples of distinct classes.

In second ablation study, objective is to make sure weather predictor makes right decisions when distribution of input vectors slightly varies. This is essential because it may be possible that predictor produces better performance on current data but it does not make right decisions in real-time settings because real-time data may slightly differ from its training corpus. To ensure predictor generalizability, we applied probability value test (p value) that illustrates predictor behaviour by slightly modifying input data. p value test first segregates data into k folds and iteratively it takes one fold as test set and other folds as training set and it computes model performance for all k folds. Furthermore, it computes model performance for number of permutations time where in each permutation it slightly varies training set distribution and computes test set performance for each fold. Specifically, for each permutation it compares predictor performance with its performance on original data. Equation 15 illustrates mathematical expression for computing p value, where permutation performance denotes predictor performance

with modified distribution and real data performance denotes to predictor performance at real data.

$$p \text{ value} = \frac{\sum (Permutation\_performance >= RealData\_performance) + 1}{Number\ of\ Permutations + 1} \quad (15)$$

Over both coarse- and fine-grained datasets, we performed experimentation under fivefold cross-validation for 100 permutations. Over coarse-grained dataset, proposed predictor produces p value of 0.01 and p value of 0.03 over fine-grained dataset. The lowest p value reveals that proposed predictor has potential to categorize Roman Urdu text into hate content or normal, even when Roman Urdu text distribution is slightly different from the distribution of text on which model is trained.

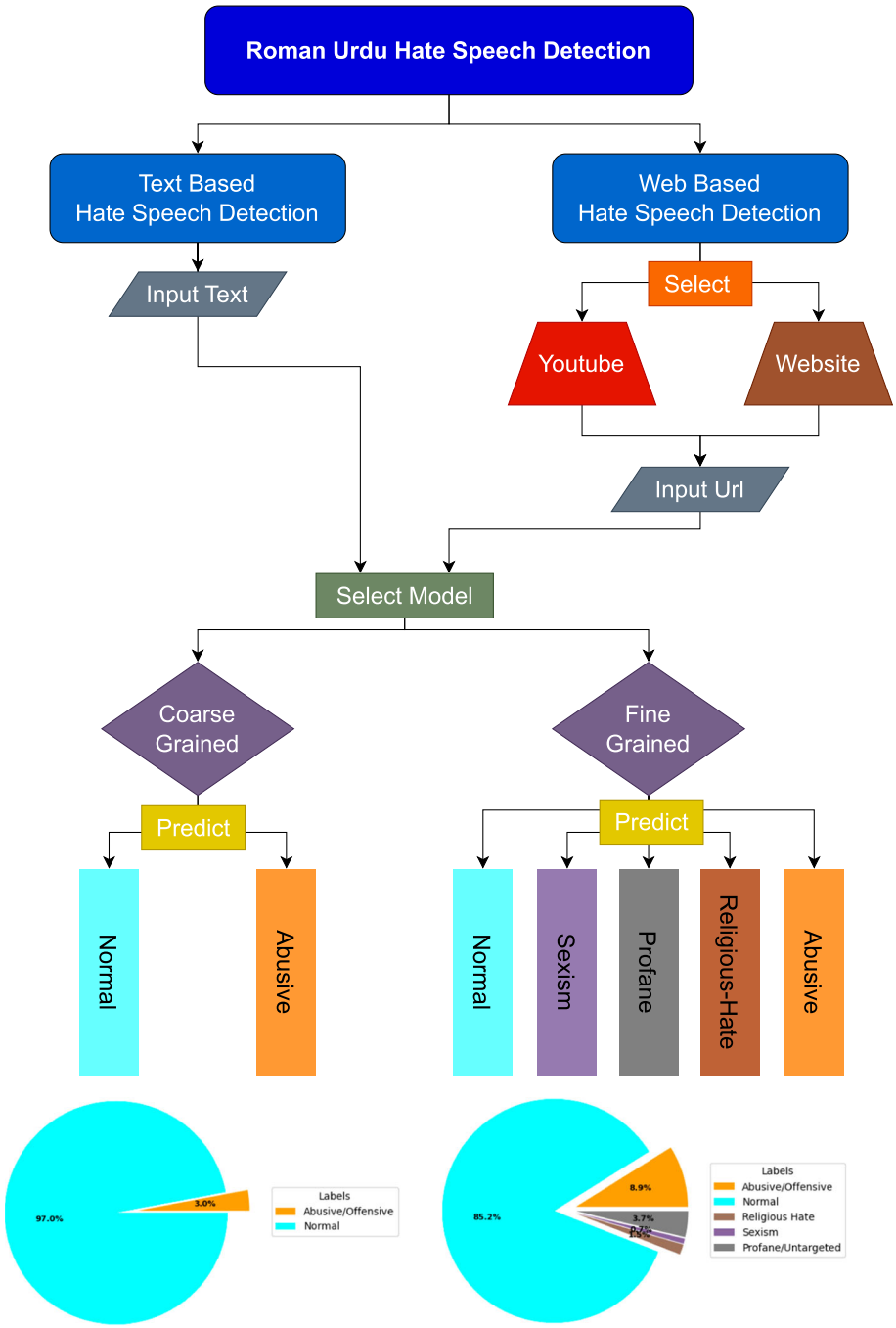

Fig. 12 Overview of web server developed for Roman Urdu Hate speech detection

8 Web interface

On top of proposed predictor, we developed web interface that will facilitate real-time hate speech detection. Figure 12 illustrates graphical workflow of web interface that provides two different options for input, namely, text-based and web-based hate speech detection. Text-based input module can be used to detect hate content from phrase pasted in text field, while web-based input module requires web or YouTube link. It will first crawl comments from the given URL, then it will separate Roman Urdu language based comments. Further, it will assign class labels to input text or comment and will also provide statistics of labels. In future, we will enrich this interface by providing predictors related to other languages.

9 Conclusion

Nowadays people are addicted to spending more time on social media platforms rather than the time they spend on other physical and social activities such as drinking, eating and sports. Proposed predictor will facilitate in developing peaceful society by eliminating and blocking spread of hate content from social media platforms. In the marathon of developing computational predictors for hate speech detection, researchers have developed two different types of predictors language-specific and multilingual models which can detect hate content from multiple languages. Among both types of predictors, multilingual predictors have less predictive performance. Language-specific powerful predictors are only developed for common languages such as English and Chinese. Specifically, for Urdu language, researchers have developed deep learning-based predictors by leveraging most recent language models; however, they remain fail to produce decent performance. Following the need of a robust and precise predictor for Urdu language, the paper in hand presents a novel predictor that makes use of language modelling strategies and different types of attention mechanisms. Experimental results reveal that over two public benchmark datasets, proposed predictor outperforms existing predictors with significant accuracy margins of 8.7%, and 10.6%. Moreover, we utilize attention scores to make proposed predictor decisions interpretable which makes it more reliable for filtering hate content from unseen data.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the Higher Education Commission Pakistan under Grant NRPU-20-16560.

Data availability Predictor source code and relevant data is available at Github repository https://github.com/Faiza-Mehmood/Passion-Net.

Declarations

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

References

1. Mathew B, Dutt R, Goyal P, Mukherjee A (2019) Spread of hate speech in online social media. In: Proceedings of the 10th ACM conference on web science, pp 173–182
2. Collins K, Shiffman D, Rock J (2016) How are scientists using social media in the workplace? PLoS ONE 11(10):0162680
3. Eriksson M, Olsson E-K (2016) Facebook and twitter in crisis communication: a comparative study of crisis communication professionals and citizens. J Conting Crisis Manag 24(4):198–208
4. Shu K, Sliva A, Wang S, Tang J, Liu H (2017) Fake news detection on social media: a data mining perspective. ACM SIGKDD Explor Newsl 19(1):22–36
5. Mondal M, Silva L.A, Benevenuto F (2017) A measurement study of hate speech in social media. In: Proceedings of the 28th ACM conference on hypertext and social media, pp 85–94
6. Djuric N, Zhou J, Morris R, Grbovic M, Radosavljevic V, Bhamidipati N (2015) Hate speech detection with comment embeddings. In: Proceedings of the 24th international conference on World Wide Web, pp. 29–30
7. Groshek J, Cutino C (2016) Meaner on mobile: incivility and impoliteness in communicating contentious politics on sociotechnical networks. Social Media+ Society 2(4):2056305116677137
8. Williams M (2019) Hatred behind the screens: a report on the rise of online hate speech
9. https://www.ethnologue.com/guides/how-many-languages
10. Khan MM, Shahzad K, Malik MK (2021) Hate speech detection in roman Urdu. ACM Trans Asian Low-Resour Lang Inf Process (TALLIP) 20(1):1–19
11. Romim N, Ahmed M, Talukder H, Islam S et al (2021) Hate speech detection in the Bengali language: a dataset and its baseline evaluation. In: Proceedings of international joint conference on advances in computational intelligence. Springer, pp 457–468
12. Mehmood F, Ghani MU, Ibrahim MA, Shahzadi R, Mahmood W, Asim MN (2020) A precisely xtreme-multi channel hybrid approach for roman Urdu sentiment analysis. IEEE Access 8:192740–192759
13. Aluru SS, Mathew B, Saha P, Mukherjee A (2020) Deep learning models for multilingual hate speech detection. arXiv preprint arXiv:2004.06465

14. Gertner A.S, Henderson J, Merkhofer E, Marsh A, Wellner B, Zarrella G (2019) Mitre at semeval-2019 task 5: Transfer learning for multilingual hate speech detection. In: Proceedings of the 13th international workshop on semantic evaluation, pp 453–459
15. Ousidhoum N, Lin Z, Zhang H, Song Y, Yeung D-Y (2019) Multilingual and multi-aspect hate speech analysis. arXiv preprint arXiv:1908.11049
16. Davidson T, Warmsley D, Macy M, Weber I (2017) Automated hate speech detection and the problem of offensive language. In: Proceedings of the international AAAI conference on web and social media, vol 11
17. Plaza-del-Arco FM, Molina-González MD, Urena-López LA, Martín-Valdivia MT (2021) Comparing pre-trained language models for Spanish hate speech detection. Expert Syst Appl 166:114120
18. Del Vigna12 F, Cimino23 A, Dell'Orletta F, Petrocchi M, Tesconi M (2017) Hate me, hate me not: hate speech detection on Facebook. In: Proceedings of the first Italian conference on cybersecurity (ITASEC17), pp 86–95
19. Struß J.M, Siegel M, Ruppenhofer J, Wiegand M, Klenner M et al (2019) Overview of germeval task 2, 2019 shared task on the identification of offensive language
20. Albadi N, Kurdi M, Mishra S (2018) Are they our brothers? Analysis and detection of religious hate speech in the Arabic twittersphere. In: 2018 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). IEEE, pp 69–76
21. Rafae A, Qayyum A, Moeenuddin M, Karim A, Sajjad H, Kamiran F (2015) An unsupervised method for discovering lexical variations in roman Urdu informal text. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 823–828
22. Shahroz M, Mushtaq MF, Mehmood A, Ullah S, Choi GS (2020) Rutut: roman Urdu to Urdu translator based on character substitution rules and unicode mapping. IEEE Access 8:189823–189841
23. Sajid T, Hassan M, Ali M, Gillani R (2020) Roman Urdu multiclass offensive text detection using hybrid features and SVM. In: 2020 IEEE 23rd international multitopic conference (INMIC). IEEE, pp 1–5
24. Rizwan H, Shakeel MH, Karim A (2020) Hate-speech and offensive language detection in roman urdu. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp 2512–2522
25. Akhter MP, Jiangbin Z, Naqvi IR, Abdelmajeed M, Sadiq MT (2020) Automatic detection of offensive language for Urdu and roman Urdu. IEEE Access 8:91213–91226
26. Pohjonen M (2019) A comparative approach to social media extreme speech: online hate speech as media commentary. Int J Commun 13:3088–3103
27. Subramanian R, Cote D, Locke J (2016) Using SAS software to enhance pedagogy for text mining and sentiment analysis using social media data
28. Barnes J, De Clercq O, Barriere V, Tafreshi S, Alqahtani S, Sedoc J, Klinger R, Balahur A (2022) Proceedings of the 12th workshop on computational approaches to subjectivity, sentiment & social media analysis. In: Proceedings of the 12th workshop on computational approaches to subjectivity, sentiment & social media analysis
29. De Clercq O, Balahur A, Sedoc J, Barriere V, Tafreshi S, Buechel S, Hoste V (2021) Proceedings of the eleventh workshop on computational approaches to subjectivity, sentiment and social media analysis. In: Proceedings of the eleventh workshop on computational approaches to subjectivity, sentiment and social media analysis
30. Sun S, Luo C, Chen J (2017) A review of natural language processing techniques for opinion mining systems. Inf Fusion 36:10–25
31. Yu C, Xia F, Qian W, Zhou A (2019) A parallel data generator for efficiently generating "realistic'' social streams. Front Comput Sci 13(5):1072–1101
32. Wright M, Filatotchev I, Hoskisson RE, Peng MW (2005) Strategy research in emerging economies: challenging the conventional wisdom. J Manag Stud 42(1):1–33
33. Qian J, Bethke A, Liu Y, Belding E, Wang W.Y (2019) A benchmark dataset for learning to intervene in online hate speech. arXiv preprint arXiv:1909.04251
34. Omar A, Mahmoud TM, Abd-El-Hafeez T (2020) Comparative performance of machine learning and deep learning algorithms for Arabic hate speech detection in osns. In: The international conference on artificial intelligence and computer vision. Springer, pp 247–257
35. Yousaf K, Nawaz T (2022) A deep learning-based approach for inappropriate content detection and classification of Youtube videos. IEEE Access 10:16283–16298
36. Nayel H.A, Shashirekha H (2019) Deep at hasoc2019: a machine learning framework for hate speech and offensive language detection. In: FIRE (working notes), pp 336–343
37. Medhat W, Hassan A, Korashy H (2014) Sentiment analysis algorithms and applications: a survey. Ain Shams Eng J 5(4):1093–1113
38. Zhang L, Wang S, Liu B (2018) Deep learning for sentiment analysis: a survey. Wiley Interdiscip Rev Data Min Knowl Discov 8(4):1253
39. Sharma AK, Sahni S (2011) A comparative study of classification algorithms for spam email data analysis. Int J Comput Sci Eng 3(5):1890–1895
40. Pérez-Rosas V, Kleinberg B, Lefevre A, Mihalcea R (2017) Automatic detection of fake news. arXiv preprint arXiv:1708.07104
41. Asim MN, Wasim M, Khan MUG, Mahmood N, Mahmood W (2019) The use of ontology in retrieval: a study on textual, multilingual, and multimedia retrieval. IEEE Access 7:21662–21686
42. Wasim M, Asim MN, Khan MUG, Mahmood W (2019) Multi-label biomedical question classification for lexical answer type prediction. J Biomed Inform 93:103143
43. Brill E, Dumais S, Banko M (2002) An analysis of the AskMSR question-answering system. In: Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002), pp 257–264
44. Li D, Bledsoe JR, Zeng Y, Liu W, Hu Y, Bi K, Liang A, Li S (2020) A deep learning diagnostic platform for diffuse large b-cell lymphoma with high accuracy across multiple hospitals. Nat Commun 11(1):1–9
45. Dabbagh SR, Rabbi F, Doğan Z, Yetisen AK, Tasoglu S (2020) Machine learning-enabled multiplexed microfluidic sensors. Biomicrofluidics 14(6):061506
46. Pasupa K, Sunhem W (2016) A comparison between shallow and deep architecture classifiers on small dataset. In: 2016 8th international conference on information technology and electrical engineering (ICITEE). IEEE, pp 1–6
47. Church KW (2017) Word2vec. Nat Lang Eng 23(1):155–162
48. Wu S, Manber U (1992) Fast text searching: allowing errors. Commun ACM 35(10):83–91
49. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543

50. Zhang W, Yoshida T, Tang X (2008) Text classification based on multi-word with support vector machine. Knowl-Based Syst 21(8):879–886
51. Qi Y, Sachan DS, Felix M, Padmanabhan SJ, Neubig G (2018) When and why are pre-trained word embeddings useful for neural machine translation? arXiv preprint arXiv:1804.06323
52. Rezaeinia SM, Rahmani R, Ghodsi A, Veisi H (2019) Sentiment analysis based on improved pre-trained word embeddings. Expert Syst Appl 117:139–147
53. Gourru A, Guille A, Velcin J, Jacques J (2020) Document network projection in pretrained word embedding space. In: European conference on information retrieval. Springer, pp 150–157
54. Su D, Xu Y, Winata GI, Xu P, Kim H, Liu Z, Fung P (2019) Generalizing question answering system with pre-trained language model fine-tuning. In: Proceedings of the 2nd workshop on machine reading for question answering, pp 203–211
55. Kant N, Puri R, Yakovenko N, Catanzaro B (2018) Practical text classification with large pre-trained language models. arXiv preprint arXiv:1812.01207
56. Araci D (2019) Finbert: financial sentiment analysis with pre-trained language models. arXiv preprint arXiv:1908.10063
57. Badjatiya P, Gupta S, Gupta M, Varma V (2017) Deep learning for hate speech detection in tweets. In: Proceedings of the 26th international conference on world wide web companion, pp 759–760
58. Abro S, Shaikh S, Khand ZH, Zafar A, Khan S, Mujtaba G (2020) Automatic hate speech detection using machine learning: a comparative study. Int J Adv Comput Sci Appl. https://doi.org/10.14569/ijacsa.2020.0110861
59. Zimmerman S, Kruschwitz U, Fox C (2018) Improving hate speech detection with deep learning ensembles. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)
60. Gaur M, Faldu K, Sheth A (2021) Semantics of the black-box: can knowledge graphs help make deep learning systems more interpretable and explainable? IEEE Internet Comput 25(1):51–59
61. Obeso AM, Benois-Pineau J, Vázquez MSG, Acosta AÁR (2022) Visual vs internal attention mechanisms in deep neural networks for image classification and object detection. Pattern Recognit 123:108411
62. Asim MN, Ghani MU, Ibrahim MA, Mahmood W, Dengel A, Ahmed S (2021) Benchmarking performance of machine and deep learning-based methodologies for Urdu text document classification. Neural Comput Appl 33(11):5437–5469
63. Sircar A, Yadav K, Rayavarapu K, Bist N, Oza H (2021) Application of machine learning and artificial intelligence in oil and gas industry. Pet Res 6:379–391
64. Mehmood F, Ghani MU, Ghafoor H, Shahzadi R, Asim MN, Mahmood W (2022) EGD-SNet: a computational search engine for predicting an end-to-end machine learning pipeline for energy generation & demand forecasting. Appl Energy 324:119754
65. Mehmood F, Ghani MU, Asim MN, Shahzadi R, Mehmood A, Mahmood W (2021) MPF-Net: a computational multi-regional solar power forecasting framework. Renew Sustain Energy Rev 151:111559
66. Asim MN, Ibrahim MA, Imran Malik M, Dengel A, Ahmed S (2021) Advances in computational methodologies for classification and sub-cellular locality prediction of non-coding RNAs. Int J Mol Sci 22(16):8719
67. Nabeel Asim M, Ali Ibrahim M, Fazeel A, Dengel A, Ahmed S (2022) DNA-MP: a generalized DNA modifications predictor for multiple species based on powerful sequence encoding method. Brief Bioinform 24:bbac546
68. Ibrahim MA, Khan MUG, Mehmood F, Asim MN, Mahmood W (2021) Ghs-net a generic hybridized shallow neural network for multi-label biomedical text classification. J Biomed Inform 116:103699
69. Singh S.P, Kumar A, Darbari H, Singh L, Rastogi A, Jain S (2017) Machine translation using deep learning: an overview. In: 2017 international conference on computer, communications and electronics (comptelix). IEEE, pp 162–167
70. El Hechi MW, Eddine SAN, Maurer LR, Kaafarani HM (2021) Leveraging interpretable machine learning algorithms to predict postoperative patient outcomes on mobile devices. Surgery 169(4):750–754
71. Hossin M, Sulaiman MN (2015) A review on evaluation metrics for data classification evaluations. Int J Data Min Knowl Manag Process 5(2):1
72. Kolo B (2011) Binary and multiclass classification. Weatherford Press, Weatherford
73. Kautz T, Eskofier BM, Pasluosta CF (2017) Generic performance measure for multiclass-classifiers. Pattern Recognit 68:111–125