Multimedia Systems (2022) 28:2027–2041
https://doi.org/10.1007/s00530-020-00672-7

SPECIAL ISSUE PAPER

Multi-input integrative learning using deep neural networks and transfer learning for cyberbullying detection in real-time code-mix data

Akshi Kumar[1] · **Nitin Sachdeva**[1]

Published online: 13 July 2020

Abstract
Automatic detection of cyberbullying in social media content is a natural language understanding and generic text classification task. The cultural diversities, country-specific trending topics hash-tags on social media, the unconventional use of typographical resources such as capitals, punctuation, emojis and easy availability of native language keyboards add to the variety and volume of user-generated content compounding the linguistic challenges. This research focuses on cyberbullying detection in the code-mix data, specifically the Hinglish, which refers to the juxtaposition of words from the Hindi and English languages. We explore the problem of cyberbullying prediction and propose MIIL-DNN, a multi-input integrative learning model based on deep neural networks. MIIL-DNN combines information from three sub-networks to detect and classify bully content in real-time code-mix data. It takes three inputs, namely English language features, Hindi language features (transliterated Hindi converted to the Hindi language) and typographic features, which are learned separately using sub-networks (capsule network for English, bi-LSTM for Hindi and MLP for typographic). These are then combined into one unified representation to be used as the input for a final regression output with linear activation. The advantage of using this model-level multi-lingual fusion is that it operates with the unique distribution of each input type without increasing the dimensionality of the input space. The robustness of the technique is validated on two datasets created by scraping data from the popular social networking sites, namely Twitter and Facebook. Experimental evaluation reveals that MIIL-DNN achieves superlative performance in terms of AUC-ROC curve on both the datasets.

Keywords Social media · Cyberbullying · Deep learning · Multi-lingual · Code-mix

1 Introduction

Social media has reshaped communication by facilitating healthy discussions and candid conversations in which people engage on the community-centric platform by sharing ideas, thoughts and information. As one of the popular and modern means of communication, social networking sites provide a constructive platform for market research, decision-making process and government intelligence [1, 2]. Undoubtedly, its mass adoption, effortless availability and popularity can get users united in a very short time and allow gathering opinion from different people on an issue in just a click. But this virtual social world can also fuel and witness different anti-social activities such as scams, fake news, rumors and cyberbullying. Social media may seem positive and safe, but it affects our daily lives more than we can think of. According to a study by Harvard University, "self-disclosure on social networking sites lights up the same part of the brain that also ignites when taking an addictive substance. The reward area in the brain and its chemical messenger pathways affect decisions and sensations" [3]. The overuse of social media can disrupt psychologically leading to social withdrawal, depression, anxiety and insomnia. Further, social media hacks and oversharing makes one's identity extremely vulnerable.

A large portion of the youngsters has progressed toward becoming casualties of cyberbullying which is one the most severe and unethical way to harass or humiliate others. It

✉ Akshi Kumar
akshikumar@dce.ac.in

Nitin Sachdeva
nits.usit@gmail.com

[1] Department of Computer Science and Engineering, Delhi Technological University, Delhi, India

refers to bullying a person or a group by sending inappropriate messages by means of electronic communication [4]. The target of cyberbullying is to harass, humiliate or to harm the reputation of cyber victim and to create infinite social dilemmas. Cyberbullying is not confined to any particular country or religion and has spread its roots all over the world. Recent statistics[1] are frightening and show steady growth in cyberbullying. A 2007 Pew Research study[2] found "32% of teens have been victims of some type of cyberbullying". Nearly a decade later, 2016 and 2019 studies by the Cyberbullying Research Center[3] found those numbers were almost unchanged. In 2019, a new poll released by UNICEF and the UN Special Representative[4] of the Secretary-General (SRSG) on Violence against Children revealed that "one in three young people in 30 countries said they have been a victim of online bullying, with one in five reporting having skipped school due to cyberbullying and violence". Young children, women, and people with a non-traditional sexual orientation are the most common targets for online bullies. That is, by 2019, nearly 43% of teens reported they were victims of cyberbullying. In the Indian context, a recent study that was conducted amongst 630 adolescents in Delhi-NCR reported that one in every tenth respondent, that is, around 9.2% had experienced cyberbullying. The survey also discussed the increased vulnerability with the rise of the internet and that only 50% of the victims report the harassment [5].

Social media has made cyberbullying a lot easier than it used to be due to it being much reckless in reach and virality that too with anonymity and without any restrictions. Social media cyberbullying is most prevalent on Instagram (42%), followed by Facebook (37%) and Snapchat (31%).[5] Typically, online bullying involves sending or posting harmful content or negative comments about a person. It intends to embarrass or humiliate a person to ruin his/her dignity, confidence and self-esteem [6]. The results of cyberbullying are dangerous and may affect the victim socially, mentally or psychologically. Hence, it is important to promptly detect cyberbullying to prevent it from becoming a global epidemic. Automatic detection of cyberbullying in social media content is a natural language understanding and generic text classification task. The cultural diversities, country-specific trending topics—hash-tags on social media and easy availability of native language keyboards add to the variety and volume of user-generated content compounding the linguistic challenges. The users are usually more comfortable conversing in their native language. Recent statistics[6] show that Hindi is the fourth most widely spoken language around the world with about 310 million native speakers, coming in only after Chinese, Spanish and English. An upsurge in the use of hybrid of Hindi and English languages has been observed [7]. The availability of keyboards with 'Devanagari' scripts on mobile phones has made it a popular language choice. A research study from Parshad et al. [8] shows that people are more fluent in Hinglish than in English alone. The anglicization of language, that is, to make or become English in sound, appearance, or character, is a semeiotical practice by the Millennials and generation-Z on social media. The mixture of languages can be observed in the text as follows:

- Transliterated Code-mix: I hate this girl!!! Ek dum ghatiya aur cheap hai 👹🐽
- Literal Code-switch: I hate this girl!!! एक दम घटिया और cheap है 👹🐽

The first case is an example of transliterated bilingual code-mixing where one language/script word (Hindi) is transcribed into a source language (English) such that the source phonetics is preserved. This is also known as phonetic typing. The second example describes a literal bilingual code-switch, where the actual words of one language (Hindi) are mashed up with the other language (English) demonstrating language alternation [9]. The research presented in this paper focuses on cyberbullying detection in the code-mix data, specifically the Hinglish, which refers to the juxtaposition of words from Hindi and English language. Further people express themselves in a certain style without adhering to the formal mechanisms for grammar and punctuation. The unconventional use of typographical resources such as capitals, punctuation and emojis is another commonly observed phenomenon in real-time social media text. Thus, it is necessary to comprehend the multiple inputs as features to train a learning model for classifying the incoming post as bullying or non-bullying. The hierarchical learning capabilities and generalization offered by deep learning architectures have made them a popular choice within the natural language text processing and specifically within the research area of cyberbullying detection. We explore the problem of cyberbullying prediction and propose MIIL-DNN, a multi-input integrative learning model based on deep neural networks. MIIL-DNN combines information from three sub-networks to detect and

[1] https://www.firstsiteguide.com/cyberbullying-stats/.

[2] https://www.pewresearch.org/internet/wp-content/uploads/sites/9/media/Files/Reports/2007/PIP-Cyberbullying-Memo.pdf.pdf.

[3] https://www.cyberbullying.org/.

[4] https://www.unicef.org/press-releases/unicef-poll-more-third-young-people-30-countries-report-being-victim-online-bullying.

[5] https://www.ditchthelabel.org/wp-content/uploads/2017/07/The-Annual-Bullying-Survey-2017-1.pdf.

[6] https://www.en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers.

classify bully content in real-time code-mix data. It takes three inputs, namely English language features, Hindi language features (transliterated Hindi converted to the Hindi language) and typographic features, which are learned separately using sub-networks and are then combined into one unified 'concat feature' representation to be used as the input for a final regression output. The three sub-networks are:

- English sub-network: Capsule Network with dynamic routing is used to GENERATE semantic context vectors using pre-trained GloVe embeddings.
- Hindi sub-network: We use the Google Transliteration—Translator toolkit[7] to implement a language transformation module where the transliterated Hindi text is converted to Hindi language text. Bi-directional LSTM is then used to generate the feature vector using pre-trained word embedding for the Hindi language provided by fastText.
- Typographic sub-network: MLP is used to operate over typographic feature vector.

These sub-networks are then concatenated together to form the final multi-input integrative learning model which generates a regression output with linear activation. Characteristically, MIIL-DNN is the foremost model-level feature fusion deep neural architecture for code-mix data which also uses transfer learning to increase the size of the training dataset. The robustness of the technique is validated on two datasets created by scraping data from the popular social networking sites namely, Twitter and Facebook. Experimental evaluation of MIIL-DNN reveals that it achieves superlative performance in terms of AUC-ROC curve on both the datasets.

The paper is organized as follows: in the next section, a brief discussion on related studies is given. Section 3 explicates the details about the proposed model followed by the results and discussion in Sect. 4. The last section, Sect. 5 presents the conclusion of the work undertaken.

2 Related studies

Cyberbullying has been majorly studied as an application area of hate-speech detection and toxic comment classification. Few systematic reviews on cyberbullying are available in the literature [6, 10, 11]. Most of the primary studies done on automatic cyberbullying detection are based on using machine learning and deep learning techniques on monolingual content, i.e. based on the use of one language only, primarily English [12–25].

Studies have also been reported on cyberbullying detection in languages other than English. Ptaszynski et al. [26] reported a study to detect cyberbullying in Polish. Gordeev [27] studied verbal aggression detection in Russian and English language image boards. Ibrohim et al. [28] and Pratiwi et al. [29] studied hate speech and abusive language identification in Indonesian tweets. Haider et al. [30] proposed a multilingual cyberbullying detection system using machine learning and natural language processing techniques and validated their model on content written in the Arabic language from Facebook and Twitter data. In another study, Haider et al. [31] extended their previous work and provided a solution for detecting cyberbullying in Arabic content and stopping cyberbullying. Pawar et al. [32] proposed a multilingual cyberbullying detection system for the detection of cyberbullying in two Indian languages namely: Hindi and Marathi. Arreerard et al. [33] proposed a model for classification of defamatory Facebook comments in the Thai language using machine learning classifiers. Tarwani et al. [34] developed a system to detect cyberbullying in Hindi–English code-mixed Instagram and YouTube comments using eight machine learning techniques. Bohra et al. [35] reported the use of supervised machine learning technique on a dataset of code-mixed Hindi–English tweets from Twitter for hate speech detection. In the same year, Singh et al. [36], used multimodal naïve Bayes, decision tree, support vector machine, multilayer perceptron, long-short term memory and convolutional neural networks for aggression detection in code-mix corpus created from Facebook. Results showed that the best accuracy of around 73% was obtained using convolutional neural networks. Santosh et al. [37] detected hate speech from Hindi–English code-mixed social media text using two LSTM models namely the sub-word level and hierarchical model with attention. Gupta [38] utilized bi-directional sequence models to tackle a classification problem in categorizing social content written in Hindi–English into abusive, hate-inducing and not offensive categories. Various secondary studies on cyberbullying detection on multilingual content have also been reported [39, 40].

3 The proposed MIIL-DNN model

As a typical natural language text classification task, automatic detection of bully content depends on feature engineering and learning model. Social media has created a new 'text-speak' genre of language which is more direct or casual or polemical. Shorthand English has become a social norm and is full of abbreviations, hashtags emojis and new-fangled uses of punctuation. It consists of some novel words (such as *selfie*), wordplay (*greaaatttttt* for *great*), neologisms (*l8r* for *later*), and Internet slangs (*TTLY* for *talk to you later*). While English dominates this shortened

[7] Google Input Tools: https://www.google.co.in/inputtools/try/.

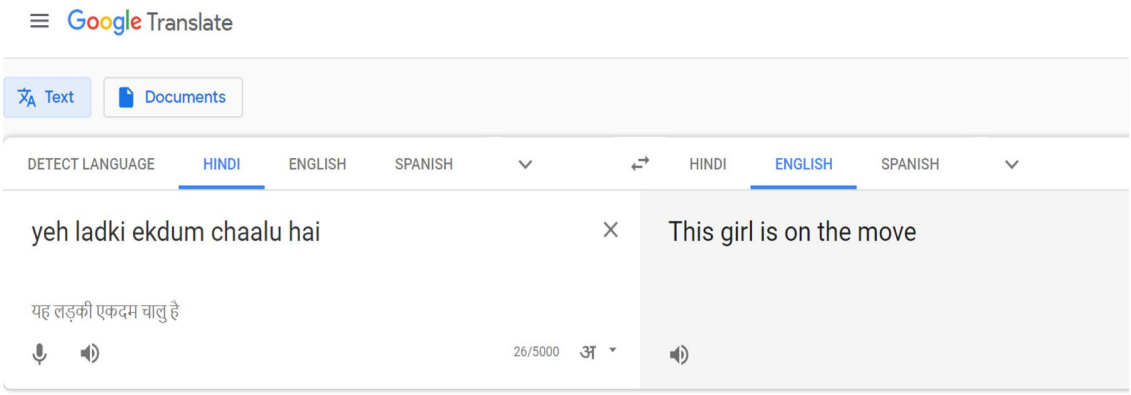

Fig. 1 Example of translation ambiguity

text-speak, a vast amount of static and dynamic web content is continuously generated by non-native writers. Multilinguality is a commonly observed phenomenon. All these multiple inputs make the task of automated text analytics computational intensive. A critical challenge is to find techniques for multi-lingual input-type fusion which can either be done at an early or a later stage (early fusion, late fusion) or at a model level. While early fusion takes a combined representation to train the network, in late fusion, the features of each language are examined and classified independently and the results are fused as a decision vector to obtain the final decision. Early fusion suffers because it increases the dimensionality of the input data without considering the unique distribution of each input type and further demands normalization to avoid giving added weight to the input type with more dimensions. Though late or decision level fusion is easy as compared to early feature fusion and facilitates the use of best suitable classifier or model to learn its features, it significantly isolates interactions among different features. Moreover, as different classifiers are used for the analysis task, the learning process of all these classifiers at the decision-level fusion stage becomes tedious and time-consuming. We propose a medial fusion strategy, that is, the model-level fusion which resolves the cons of both early and late fusion. It exploits correlation in data as different sub-networks are used to operate over features which are learned separately for each input type and then combined into one unified representation.

Text classification in a multilingual code-mix input can either be done by translating the input into a mono-lingual dataset (English only) or by using a language-dependent method (English and Hinglish) without translation. The translation method has a serious shortcoming as it may cause an ambiguity or failure of the translation resulting in wrong semantics and feature vector generation used to train the model. For example, the English translation of the Hindi transliterated text "yeh ladki ekdum chaalu hai" is wrongly translated to "This girl is on the move" (Fig. 1).

On the other hand, the language-dependent model requires a large labeled training dataset for every new language, which is a computationally expensive job. Therefore, to bridge the limitation between translated method and language-dependent method, we use the Google Transliteration—Translator toolkit, such that word-level transliteration is done to convert Hinglish text to Hindi. This also enables to capture the right textual interpretation, for example, the correct transliteration to Hindi for "yeh ladki ekdum chaalu hai" is "यह लड़की एकदम चालु है".

The hierarchical learning capabilities and generalization offered by deep learning architectures have made them a popular choice within natural language text processing [41]. The most sophisticated bullying classification methods are trained on general corpora with vast amounts of labeled data which are not suitable to a code-mix data (English and a low-resource language like Hindi). Transfer learning methods look like a promising solution to this challenge of the scarcity of labeled data. In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. The core idea behind these models is that by training language models on very large corpora and then initializing down-stream models with the weights learned from the language modeling task, a much better performance can be achieved The initialized layers can range from the single word embedding layer to the whole model [42].

Thus, the proposed integrative learning network, MIIL-DNN combines information from three sub-networks trained using three inputs, namely English, Hindi and typographic, respectively. We use transfer learning by fine-tuning the pre-trained word embeddings (GloVe for English and fastText for Hindi) for the domain-specific words to increase the size of the training dataset. These three sub-networks

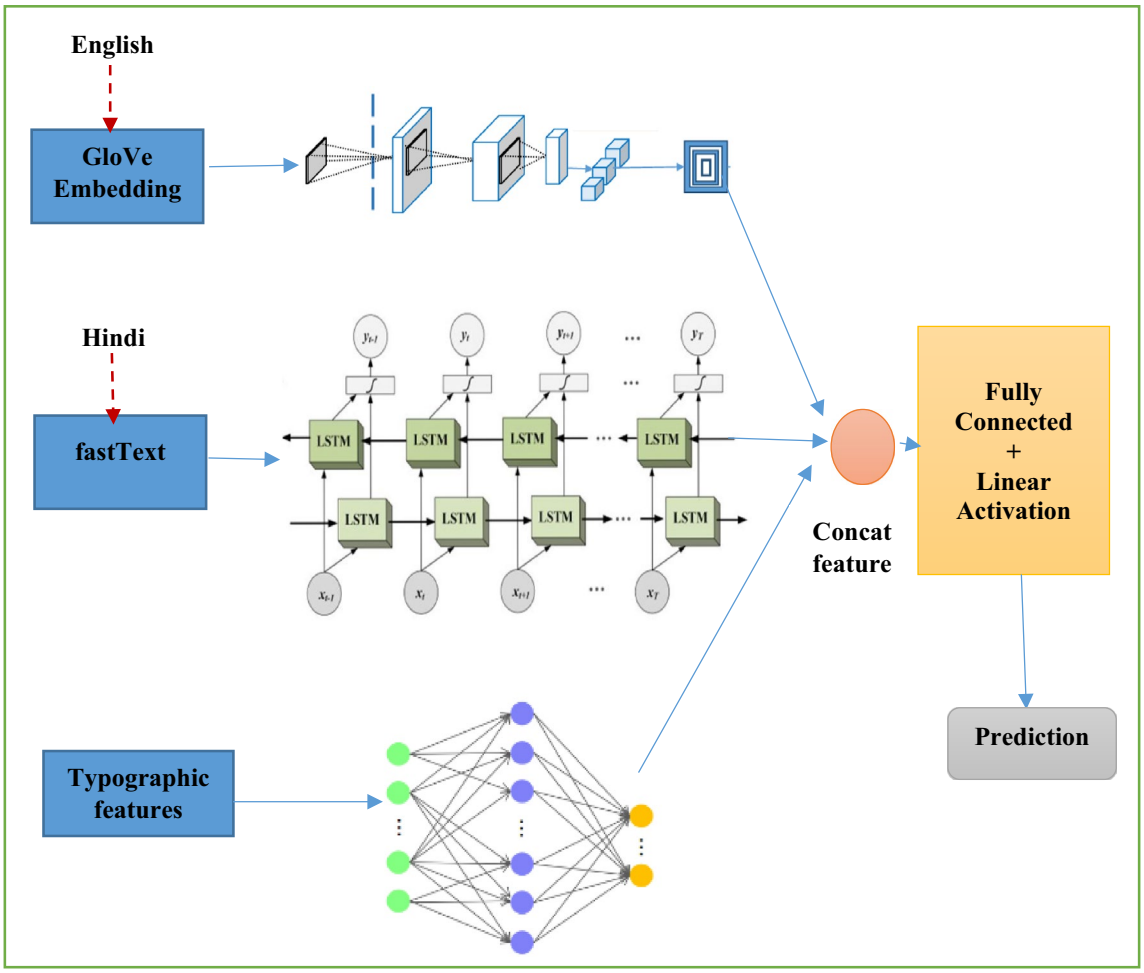

Fig. 2 The proposed MIIL-DNN architecture

include capsule network with dynamic routing [43] sub-network to generate English semantic context vectors using pre-trained GloVe embeddings. Hindi bi-directional LSTM sub-network used to generate the feature vector using pre-trained word embedding for the Hindi language provided by fastText and Typographic feature sub-network where MLP is used to operate over typographic input data. Subsequently, a model-level feature fusion of sub-network outputs is done to generate the output class. That is, these sub-networks are then concatenated together to form the final multi-input integrative learning model which generates a regression output with linear activation. The following sub-sections explicate the details of MIIL-DNN. Figure 2 depicts the architecture of the proposed MIIL-DNN network.

3.1 Dataset creation

Two datasets were created by scraping data from the popular social networking sites namely, Twitter and Facebook. Data was based on the selection of certain hashtags and keywords from the domain of politics, public figures, entertainment, etc. and was restricted to code-mixed 'Hinglish (Hindi + English)' language. The posts fetched from Facebook were "profile-based". The most popularly searched profiles of Sh. Narendra Modi ji (Prime Minister of India), Mr. Shahrukh Khan (actor), public profiles of NDTV (news channel) and Jawaharlal Nehru University (university in India) were observed for the data analysis. GraphAPI was used for the extraction of Facebook comments. For Twitter, "topic-based" tweets were scraped that belonged to the most trending topics such as "#Ind VS Pak, #Beaf Ban, #movies". Tweepy tool was used for the extraction of tweets from Twitter. Also, the posts that were solely written in English or Hindi or code-mix of English and Hindi were removed using manual filtering. Finally, two datasets with 6500 (English–Hindi) code-mixed posts each, for both Facebook, Dataset 1 (DS-I) and Twitter, Dataset 2 (DS-2) were created.

The datasets were annotated for two categories, namely cyberbullying and non-bullying. The details for the tag-categorization are given in Table 1.

Table 1 Tags and their counts for both the datasets

	Dataset 1 (DS-I) [Facebook]	Dataset 2 (DS-II) [Twitter]
Bullying (B)	3275	3350
Non-bullying (NB)	3225	3150
Total	6500	6500

Table 2 Average post-length in different class text for both the datasets

	Dataset 1 (DS-I) [Facebook]	Dataset 1 (DS-II) [Twitter]
B	27.75	27.035
NB	27.63	26.75

Table 3 Average word length in different class text for both the datasets

	Dataset 1 (DS-I) [Facebook]	Dataset 2 (DS-II) [Twitter]
B	4.505	4.76
NB	4.24	4.10

Tables 2 and 3 give the details about the average post and word length in different class text, respectively.

3.2 Data pre-processing

The primary intent of pre-processing was to transform the data for the extraction of features [44]. The process included

- Removing tags, numbers, URLs, mentions, stopwords and punctuations.
- Spell check, lemmatization and stemming.
- The tokens are converted to lower case.
- Substituting slangs and emojis using the *SMS Dictionary*[8] and *emojipedia*[9] respectively.
- Punctuations are usually discarded during data pre-processing phase but in casual or informal writing such as text message or online posts, these are used as a technique to add emphasis to written text. Therefore, the count of each punctuation mark (!, ?,., capitalization, 'x', "x") is extracted as typographic feature set to train the model [9].

Tokenization [45] of Facebook posts and tweets was then done using the TreebankWordTokenizer of Python Natural Language Toolkit (NLTK).[10] Subsequently, language transformation is done to decode the Hinglish language tokens using transliteration into Hindi. Transliteration is the conversion of text written in one script (language) into text written in another script (language), while maintaining the pronunciation to the greatest possible extent [46]. There is no change in grammar or meaning. Unlike translation which tells the meaning in the target language, transliteration is based on the pronunciation in the target language, and not on the meaning. For example, for the Hindi phrase 'मुझे उसका तरीका बलिकुल अच्छा नहीं लगता', its translation in English would be 'I don't like her way' and 'Mujhe uska tareeka bilkul achha nahi lagta' is the transliterated Hindi. We use the Google Transliteration—Translator toolkit[11] to implement this language transformation module where the transliterated Hindi text is converted to the Hindi language.

3.3 Feature extraction

Manual feature extraction is computationally expensive [47] whereas feature learning techniques such as word embeddings enable vector representations of a word in a vector space where words sharing certain semantic or syntactic relationships exist in close vicinity of each other. Such knowledge allows us to do away with manual feature engineering required to gain semantic and local contextual insight. Subsequently in transfer learning, the embedding layer is initialized using third-party embeddings such as GloVe, Word2Vec or fastText and the semantic information between words that was learned during the embedding process is transferred. In this work, we use the GloVe pre-trained embedding and the fastText pre-trained embedding to initialize the English and Hindi sub-networks respectively.

The count-based GloVe embedding is used to seed the sub-network for the English language feature vector generation [48]. This feature vector is given as input to the CapsNet. A capsule is trained to specify the features of the object and its likelihood. Thus the objective of the capsule is not just feature detection but also to train the model to capture the context features. Similarly, we use a pre-trained word embedding for the Hindi language provided by fastText to train a bidirectional LSTM sub-network such that word features $H=(h_1, h_2, \ldots, h_n)$ are concatenated from both directions.

Additionally, punctuations such as exclamation mark, quotation marks, capitalization add emphasis in written informal text and are significant signs which assist to comprehend the context inconsistency or intensity within the

[8] SMS Dictionary. *Vodacom Messaging*. Retrieved 16 March 2012.

[9] https://www.emojipedia.org/.

[10] https://www.nltk.org/.

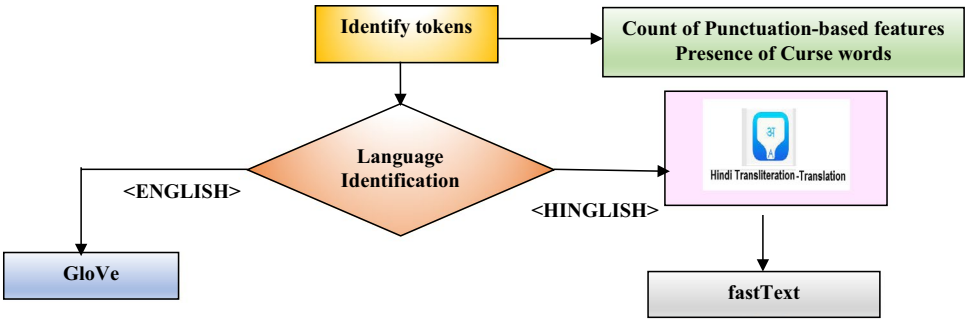

Fig. 3 Feature extraction in MIIL-DNN

text [9]. Similarly, target curse words[11] also act as textual indicators and therefore the presence of offensive/profane words must be included as an important typographic feature. Thus, the typographic feature vector t with six tuples is $<r, e, p, u, q, c>$, where r is the frequency of recurring alphabetic character (that is, if recurrence >2 set $r=1$, else 0) and e, p, u and q defines the count of exclamation marks, periods, uppercase letters, single quotes (‘’) or double quotes (“”), respectively, and c defines the presence of curse word within the text. The conceptual flow of feature extraction is shown in Fig. 3.

Different deep learning models are then applied, as sub-networks for this multi-input data that are English and Hindi language input mapped to real-valued vectors using pre-trained word embeddings GloVe and fastText, respectively, and numeric/categorical pragmatic data. These inputs are fed into the respective sub-networks namely, CapsNet for English, bi-LSTM for Hindi and multi-layer perceptron (MLP) for pragmatic to model an integrative learning network which combines information from the sub-networks. Each sub-network of this deep learning network is explained in the following sections.

3.4 Capsule network (CapsNet) with dynamic routing sub-network for English input

A capsule network is composed of many capsules [49]. A capsule can be a neuron or set of neurons which output a vector rather than a single value scalar. This vector usually carries additional information that would otherwise be lost by the summation process (max-polling). The key concepts of capsule nets include substituting the scalar-output feature detectors of CNNs with vector-output capsules and replacing the max-pooling with “routing-by-agreement.” The purpose of the capsule is not only to detect a feature but, also to train the model to learn the variant. Various routing algorithms such as static, dynamic, clustering and attention based have been proposed in the relevant literature on text classification. Most of the work relies on the customary dynamic routing algorithm where basically the capsules ‘vote’ which capsule to output to [43]. In contrast to CNNs which require training on large datasets, the generalization capabilities of CapsNets on smaller datasets make them competent and conducive for use in various real-life applications. The following subsections explicate the details:

3.4.1 Embedding layer

The embedding layer of a neural network converts an input from a sparse representation into a distributed or dense representation. Word embedding facilitates natural language understanding by means of semantic parsing such that the meaning from text is extracted preserving the contextual similarity of words. In this research, we pre-train the model on a general dataset using GloVe word embeddings and use transfer learning to train it on the domain-specific problem. The GloVe embedding is pre-trained on Twitter (2B tweets, 27B tokens, 1.2 M vocab, uncased, 200d vectors) data.

3.4.2 Encoding layer: capsule network

The matrix of word vectors produced by GloVe is converted into a feature vector of one dimension by the encoding layer. The encoding layer is implemented as a capsule network. The network consists of the convolution layer, the primary caps layer, and the class caps layer such that the outputs from one capsule (child) are routed to capsules in the next layer (parent). The detailed functionality of each layer is explained in the following subsections.

- Convolution layer

Extraction of features from the given input is done by the convolution layer. This layer performs the convolution operation and generates a feature f_i for the given filter as given in the following equation:

[11] https://www.cs.cmu.edu/~biglou/resources/bad-words.txt.

$$f_i = \varphi\left(\sum_{h=1}^{h}\sum_{d=1}^{d} K_{h,d}X_{i+h,d} + b_i\right), \tag{1}$$

where f_i $(i = 1, 2, 3, \ldots, n)$ represents the feature produced, $\boldsymbol{\varphi}$ is ReLU function used or activation, $K_{h,d} \in \mathbb{R}^{h\times d}$ is the filter, X_i represents the input word vector, and b_i is a bias term.

- Primary caps layer

This low-level layer takes the previous convolution layer scalar output to generate vector outputs called capsules. Capsule networks can be visualized as tree-like representations that learn transformations to associate the parts of an object to the whole. Capsules provide a way to detect parts of objects identifying the child and parent capsules such that the output of the capsule gets sent to an appropriate parent in the layer above. The key question that needs to be answered is which parts belong to which parents. A powerful non-linear dynamic routing captures the part-whole relationship dynamics of the capsules and ensures the output of a capsule is sent to a suitable parent. This ensures that if after applying a transformation to the part, we have the same or a similar feature vector to that of the parent, then we update a parameter for the likelihood that the two capsules are linked as parent/child. Another key issue is that of whether a part actually exists or not. This is determined by the length of the feature vector of a capsule.

The output of a single capsule u_i is multiplied by a translation matrix W_{ij} to produce a vector $u_{j|i}$.

$$u_{j|i} = W_{ij}u_i, \tag{2}$$

where capsule '*i*' is in the current lower level primary caps layer whereas capsule '*j*' is in the next level layer. Using the iterative routing-by-agreement mechanism, lower level capsule sends its output to higher level capsules whose activity vectors have a big scalar product with the prediction coming from the lower level capsule (Eq. 3):

$$s_j = \sum_i c_{ij}u_{j|i}, \tag{3}$$

where c_{ij} is the coupling coefficient that is calculated using a softmax function during the dynamic routing process as given by

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})}, \tag{4}$$

where $i \in [1, a]$ and $j \in [1, k]$, and k represents the number of classes $b_{ij} = b_{ij} + u_{j|i}.v_j.b_{ij}$ is the initial logits (prior probabilities that capsule i should be coupled to capsule j).

That is, till now we have multiplied the output of the previous capsule by weight matrices to encode the spatial relationships, then multiplied them with coupling coefficients to just receive the relevant information from the previous capsules. A non-linear "squashing" function is used to normalize the length of the output vector of each capsule to [0, 1]. Thus, on applying the squashing function s_j, the output vector v_j is given as

$$v_j = \frac{\left|\left|s_j\right|\right|^2}{1 + \left|\left|s_j\right|\right|^2}\frac{s_j}{\left|\left|s_j\right|\right|}. \tag{5}$$

The dynamic routing algorithm is summarized in Algorithm 1.

Algorithm 1 Dynamic routing algorithm [42]

Initialize logit parameters b_{ij}= 0 for all capsule i in layer l and capsule j in layer $(l + 1)$.

1: **for** 1:*MaxIter* **do**

2: $c_{ij} = \frac{\mathbf{exp}(\boldsymbol{b_{ij}})}{\sum \mathbf{exp}(\boldsymbol{b_{ijk}})}$ for all capsule i in layer l.

3: $s_j = \sum_{\boldsymbol{i}} \boldsymbol{c_{ij}}\,\widehat{\boldsymbol{u}}_{\boldsymbol{j|i}}$ and $v_j = \frac{\|\boldsymbol{s_j}\|^2}{\mathbf{1}+\|\boldsymbol{s_j}\|^2}\frac{\boldsymbol{s_j}}{\|\boldsymbol{s_j}\|}$ for all capsule j in layer $(l + 1)$.

4: $b_{ij} = b_{ij} + <\widehat{\boldsymbol{u}}_{\boldsymbol{j|i}}, \boldsymbol{v_j}>$ for all capsule i in layer l and capsule j in layer $(l + 1)$.

5: **end for**

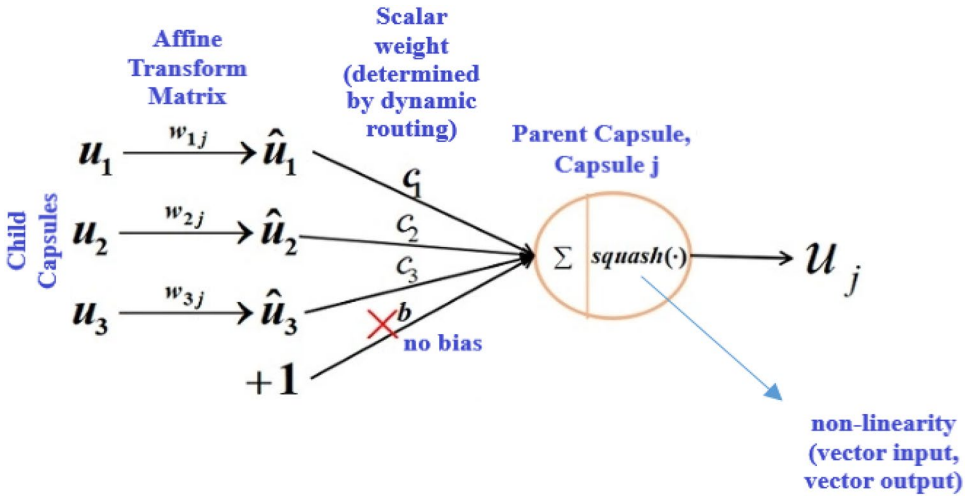

Fig. 4 Operations within a capsule

- Class caps layer

The output of the lower level capsule which is a linear combination of different predictions is sent to an appropriate parent in the layer above. Moreover, according to the degree of agreement, as measured by the dot product, between the prediction and the final output of the higher level capsule, i.e. after the squashing, the routing-by-agreement algorithm increases or decreases the coupling to adjust different contributions of different capsules. After the prediction vectors are calculated, they are linearly summed as in Eq. 3 to get the total input of the capsule, which is then squashed as Eq. 5 to calculate the output of this capsule. The prediction made by the network after convergence is of course the class with the largest output vector norm. The final class caps layer outputs a vector to represent the existence of the entity. That is, the length of the activation vector characterizes the probability of the existence of the entity. We refer to the normalized outputs from the class caps layer and use them as features for our bully detection classifier. Figure 4 summarizes the operations within a capsule.

3.5 Bi-directional long short-term memory (bi-LSTM) sub-network for Hindi input

Training a model on a huge dataset and then re-using the pre-trained model for a target task (transfer learning), can be valuable to low-resource languages such as Hindi, where the amount of labeled data is limited. Here, we use a pre-trained word embedding for Hindi language provided by fastText to train a bidirectional LSTM sub-network such that word features $H = (h_1, h_2, \ldots, h_n)$ are concatenated from both directions. This model is trained using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives.

A long short-term memory (LSTM) is a type of recurrent neural network that addresses the vanishing/exploding gradient problem with RNNs. LSTMs introduce the concept of cell states, which provide "highways" for the gradient to flow backward through time freely, thereby making it more resistant to the vanishing gradient problem. The cell state can be thought of almost like data stored in a computer's memory. LSTMs can "remember" or "forget" information in the cell state by using specialized neurons called "gates". This way, LSTMs can retain long-term dependencies and connect information from the past to the present. There are three major gates, namely the forget gate, input gate and the output gate (Eqs. 6–11):

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right), \tag{6}$$

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right), \tag{7}$$

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right), \tag{8}$$

$$s_t = tanh\left(W_s \bullet \left[h_{t-1}, x_t\right] + b_s\right), \tag{9}$$

$$c_t = f_t * c_{t-1} + i_t * s_t, \tag{10}$$

$$h_t = tanh\left(c_t\right) * o_t, \tag{11}$$

where

- x_t is the tth word vector that is it denotes the word representation of w_t,
- $W_i W_f W_o W_s$ are model parameters,
- $b_i b_f b_o b_s$ represents the bias vectors,
- σ is the sigmoid function used as the gate activation function,

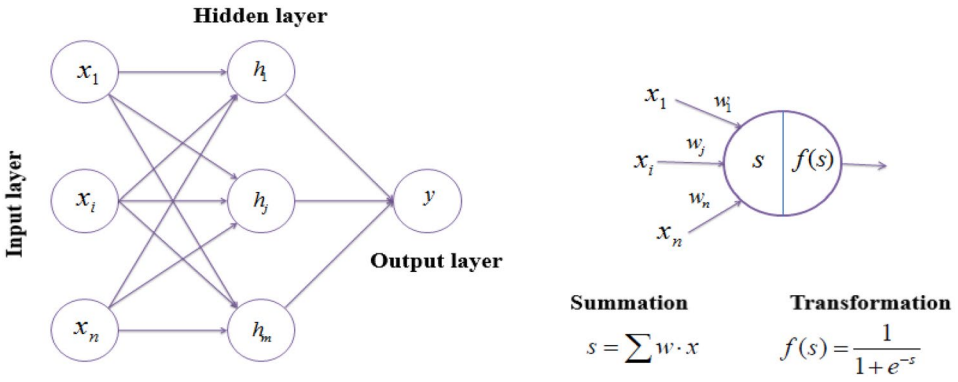

Fig. 5 MLP architecture

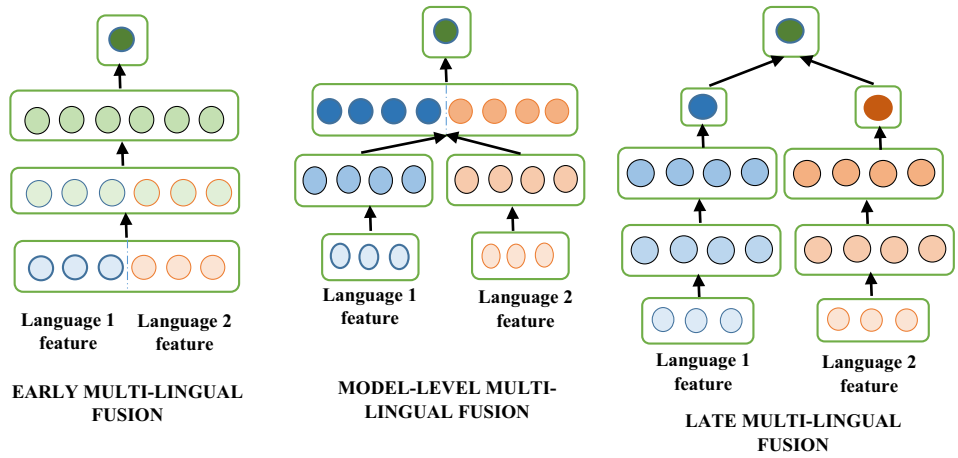

Fig. 6 Multi-lingual fusion strategies

- $*$ represents the product (element-wise),
- tanh is the hyperbolic tangent function.

Bidirectional learning in LSTMs trains two LSTMs to allow the propagation of input in both backward (previous time steps) as well as forward (later time steps) direction in time to make predictions about the current state. This adds past and future context as a bonus to the network and improves the results. We use bidirectional LSTM (bi-LSTM) [50] to obtain word features $H = (h_1, h_2, \ldots, h_n)$ concatenated from both directions. A forward LSTM processes the sentence (tweet/post) from x_1 to x_n, while a backward LSTM processes from x_n to x_1. For word x_t, a forward LSTM obtains a word feature as $\overrightarrow{h_t}$ and a backward LSTM obtains the feature as $\overleftarrow{h_t}$ [9]. Then h_i is calculated using (12):

$$h_i = \left(\overrightarrow{h_i} \odot \overleftarrow{h_i}\right), \tag{12}$$

where h_i is the output of the ith word, $\odot$ function is a concatenation function. Generally, different merge modes can be used to combine the outcomes of the bi-LSTM layers. These are concatenation (default), multiplication, average, and sum.

$\overrightarrow{h}$ is the forward hidden sequence and $\overleftarrow{h}$ is the backward hidden sequence calculated iteratively for time step from $t = T$ to 1 for the backward layer and $t = 1$ to T for the forward layer.

3.6 Multi-layer perceptron (MLP) sub-network for pragmatic features

The pragmatic feature vector is trained using a multilayer feed-forward neural network which is a special type of fully-connected network with multiple single neurons. MLP can be viewed as a logistic regression classifier where the input is first transformed using a learnt non-linear transformation Φ. This transformation projects the input data into a space where it becomes linearly separable. This intermediate layer is referred to as a hidden layer. A single hidden layer is sufficient to make MLPs a universal approximator. The types of layers in a typical MLP are as follows as shown in Fig. 5:

- Input layer: input variables, sometimes called the visible layer.
- Hidden layers: layers of nodes between the input and output layers. There may be one or more of these layers.

Table 4 Hyper-parameters for the model

	Hyper-parameter	Value
CapsNet	No. of convolution layers	1
	Batch size	64
	No. of filters	32
	Activation	ReLU
	Learning rate	0.001
	Kernel_size	3
	No. of capsules in PrimaryCaps layer	8
	No. of nodes in PrimaryCaps layer	64
	Routing time	2
	Dimension of each capsule	8
	Length of PrimaryCaps	2
	Length of Digi Caps	2
	Optimizer	Adam
Bi-LSTM	Number of layers	1
	No. of nodes in each layer	64
	Batch size	32
	Optimizer	Adam
MLP	Number of hidden layer	1
	No. of nodes in hidden layer	32
	Batch size	32
	Activation function of hidden layer	ReLU
	Optimizer	Adam
Vector dimensions of the word embeddings	fastText; GloVe	300; 300

- Output layer: a layer of nodes that produce the output variables.

In this work, the MLP consists of a single hidden layer which is fully connected to the input layer as well as the output layer. The standard logistic sigmoid is used as the activation function in the MLP.

3.7 Concatenation and output

Typically, multi-lingual fusion strategies can be categorized into early, model-level and late fusion. The early multi-lingual fusion strategy involves concatenation of features from different languages, the model-level multi-lingual fusion involves concatenation of high-level feature representations from different languages and the late multi-lingual fusion involves the fusion of predictions from different languages as shown in Fig. 6.

In this work, the output features from the sub-networks are concatenated to generate the final concat feature using model-level multi-lingual fusion strategy. This concat feature is the shared representation which combines the high-level representation features of each input type. This fusion strategy helps to complete the essence of multi-input integrative learning proposed in this work. Unlike early fusion, this strategy helps to circumvent the curse of dimensionality and synchronization between different features and at the same time does not isolate interactions among different languages as in late fusion. Finally, the shared representation is given to the fully-connected layer which generates a regression output with linear activation to detect cyberbullying for code-mixed social media textual content.

4 Results and discussion

We experimented with the two datasets of 6500 tweets and posts each. The Facebook dataset, DS-I consisted of 3275 posts as bullying and 3225 as non-bullying posts and the Twitter dataset, DS-II consisted of 3350 tweets as bullying and 3150 tweets as non-bullying. We performed tenfold cross-validation and calculated the AUC curve. We used the Scikit-learn library and Keras deep learning library with Theano backend. Since three different models were used

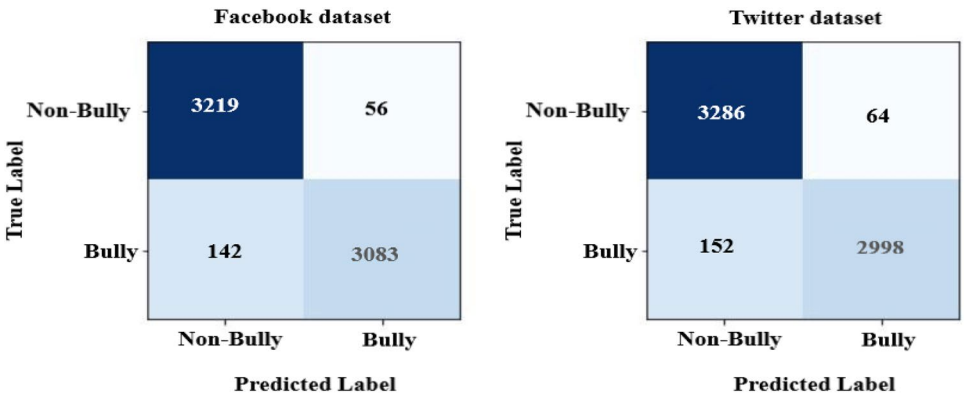

Fig. 7 Confusion matrix for DS-I and DS-II

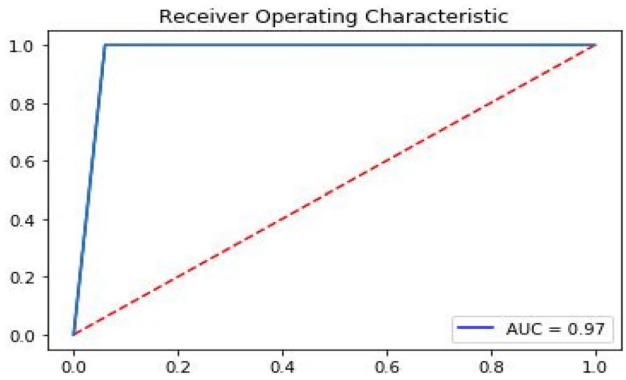

Fig. 8 Performance of MIIL-DNN on DS-I and DS-II

based on the input-type, the tuning of each model's respective hyper-parameters was performed. The choice of model parameters was as given in Table 4.

Figure 7 illustrates a summary of prediction results using confusion matrix. It helps to visualize the proportions between the predicted and true label for both the datasets.

The proposed model reports a performance of approximately 0.97 for both the datasets as shown in Fig. 8. This is primarily because it combines an automatic feature extraction mechanism with the robustness, dynamism and flexibility of the deeper neural architectures such as CapsNet and bi-LSTM.

As we proposed training a CapsNet model for English tweets/posts, it was imperative to evaluate the robustness of this sub-network. We compared its performance with the existing state-of-the-art Toxic Comment Classification Challenge dataset[12] from a Kaggle competition. The dataset contains 159,571 Wikipedia manually labeled comments categorized as toxic; severe toxic; obscene; threat; insult and identity hate. All these categories account for cyberbullying whereas any comment with value = 0 in all fields will indicate non-cyberbullying, i.e. non-toxic comments. As per https://www.kaggle.com, the first place solution reported a performance of 0.9885 using a bi-GRU with the pseudo-labeling technique. The performance of the best single model of the competition was around 0.9869 and a single layer RNN-capsule network with GRU cell performed at 0.9857. Srivastava et al. [51] used capsule network with focal loss and achieved a ROC-AUC of 98.46 on the Kaggle toxic comment dataset. The performance of the proposed CapsNet was thus comparable at 0.9841. Figure 9 shows the ROC curves for all the toxic comment categories.

[12] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge.

The Hindi bi-LSTM model was compared with other deep neural architectures, namely convolution neural network (CNN) and LSTM. The accuracy results are shown in Table 5.

The variations for typographic content using four different classifiers, namely Naïve Bayes (NB), decision tree (DT), support vector machine (SVM) and MLP was also evaluated. Figure 10 depicts the accuracy results. Comparable results were observed for NB, SVM and MLP. The choice of MLP was made to give coherence to the model architecture using only neural networks.

The results were also compared with the existing models of Hindi–English code-mix cyberbullying detection. Table 6 depicts the analysis.

5 Conclusion

Intelligent adaptive models are required to deal with the information overload on the chaotic and complex social media portals due to its enormous growing usage. This has resulted in high incidences of cyberbullying on the web as compared to customary bullying practices. One of the challenges of cyberbullying these days is dealing with the use of code-mixed languages for bullying others. This work proffered a model-level feature fusion model using deep neural networks to classify the incoming real-time post into bullying or non-bullying categories. The contribution of the research is twofold: first, the problem of cyberbullying is taken as a generic case such that classification is done into two broad categories that is bullying and non-bullying as compared to earlier works on either toxic comment classification or hate-speech detection. This comprehensibility and generalization of the proposed model makes it easily scalable and applicable to high dimensional cross-platform, cross-lingual real-time datasets as well. Secondly, the proposed integrative learning network, MIIL-DNN uses a model-level multi-lingual fusion to combine information from three sub-networks to generate the final output. We achieve an appreciable ROC–AUC of 0.97 on the datasets.

The primary limitations of working with a low-resource language like Hindi and its combination with English suffers due to lack of tools, benchmark datasets and learning techniques. This research warrants a new line of inquiry to address these challenges and apply deep neural architectures to achieve a superlative performance. As a future direction, we would like to examine the decision-level (late) multi-lingual fusion and study its effect on the performance. We would also like to explore pre-trained language models like Embeddings from Language Models (ELMo) and bidirectional encoder representations from transformers (BERT) instead of the pre-trained

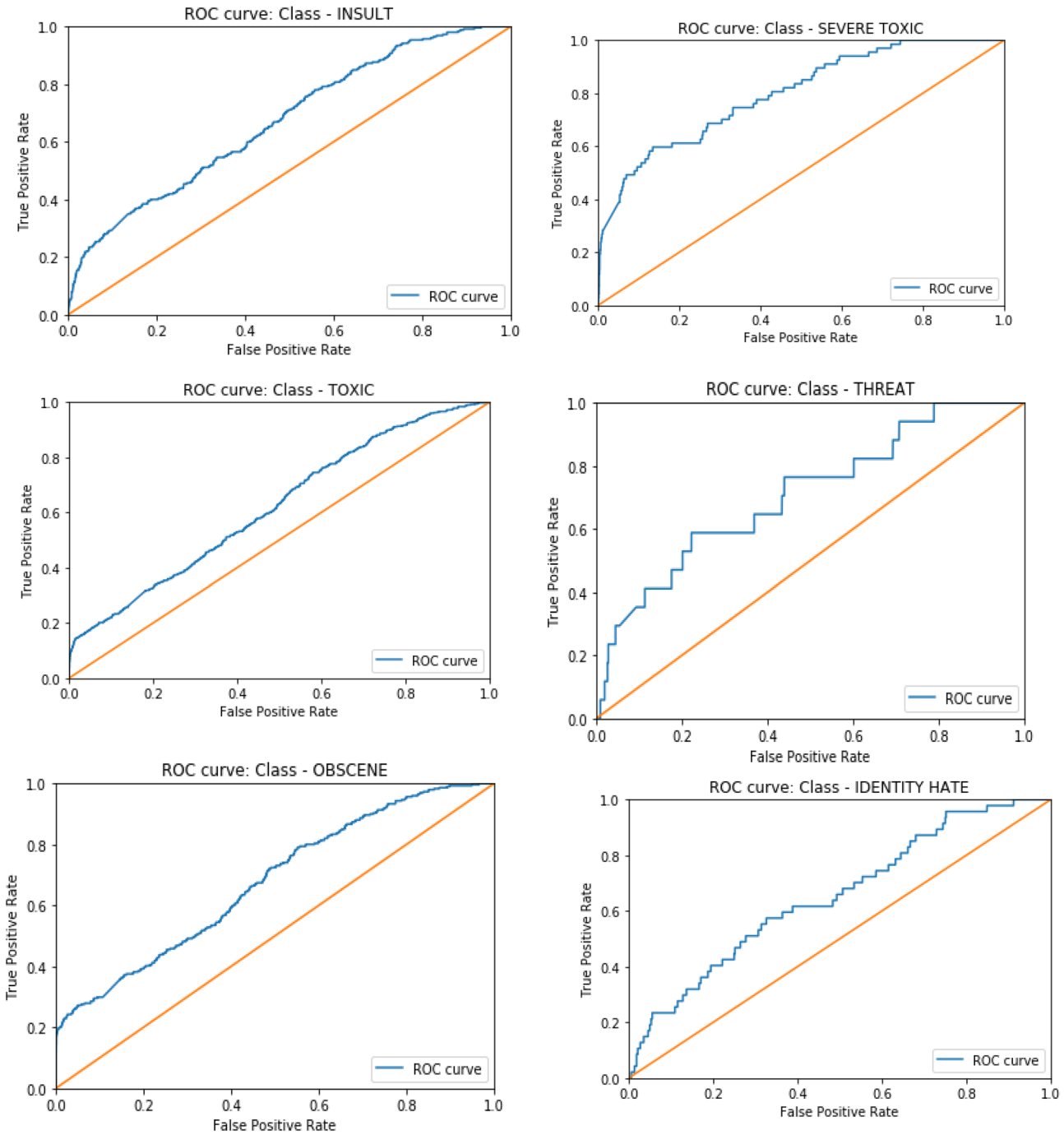

Fig. 9 Performance results of toxic comment categories

Table 5 Variations with Hindi sub-network

Model	Accuracy
CapsNet + CNN + MLP	0.9635
CapsNet + LSTM + MLP	0.9603
CapsNet + Bi-LSTM + MLP	0.9706

embeddings (GloVe and fastText) used. These dynamic, context-dependent, and instance-specific embeddings can improve the analytics on a bigger code-mixed dataset and a Hindi only dataset too.

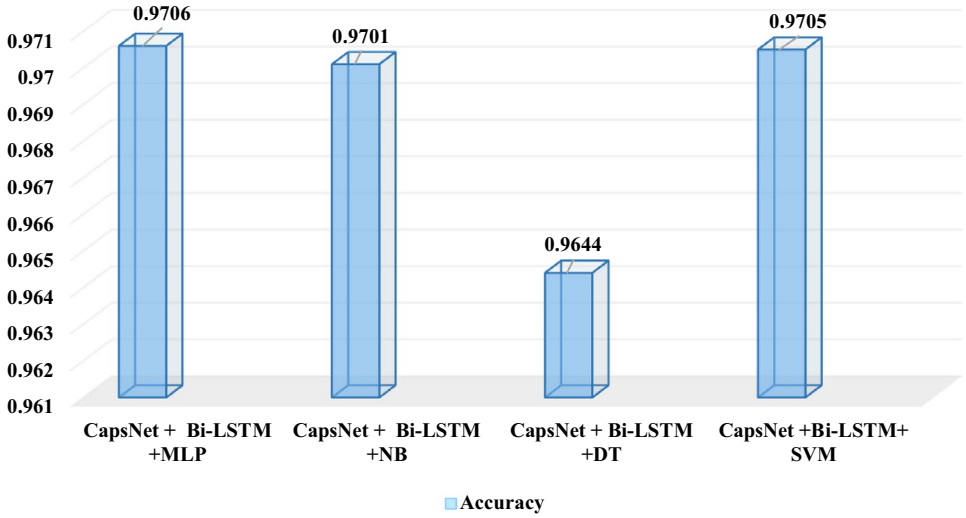

Fig. 10 Accuracy results for typographic classifier variations

Table 6 Comparison with existing models

References	Dataset	Objective	Model	Performance
Tarwani et al. [34]	Instagram and YouTube comments	Cyberbullying detection	Eight machine learning techniques	80.26% accuracy
Bohra et al. [35]	Twitter tweets	Hate-speech detection	Support vector machines with radial basis function kernel	71.7% accuracy
Singh et al. [36]	Facebook	Aggression detection	Convolutional neural networks	73% accuracy
Santosh et al. [37]	Tweets	Hate-speech detection	Sub-word level LSTM model Hierarchical LSTM model with attention based on phonemic sub-words	69.8% accuracy 66.6% accuracy
Gupta [38]	Tweets	Abusive, hate-inducing and not offensive categories	Bi-LSTM	77% recall
Proposed MIIL-DNN	Facebook posts and Twitter tweets	Cyberbullying detection	Capsule Net + bi-LSTM + MLP	97% accuracy

References

1. Kumar, A., Jaiswal, A.: Systematic literature review of sentiment analysis on Twitter using soft computing techniques. Concurr. Comput. Pract. Exp. **32**(1), e5107 (2020)
2. Kumar, A., Sharma, A.: Systematic literature review on opinion mining of big data for government intelligence. Webology **14**(2), 6–47 (2017)
3. Brown L (2012) New Harvard study shows why social media is so addictive for many. [online] WTWH Marketing Lab. https://www.marketing.wtwhmedia.com/new-harvard-study-shows-why-social-media-is-so-addictive-for-many/. Accessed 27 Jan 2020
4. Campbell, M.A.: Cyber bullying: an old problem in a new guise? J. Psychol. Couns. Sch. **15**(1), 68–76 (2005)
5. Child Rights and You (CRY): Online Safety and Internet Addiction (A Study Conducted Amongst Adolescents in Delhi-NCR). Child Rights and You, New Delhi (2020)
6. Kumar, A., Sachdeva, N.: Cyberbullying detection on social multimedia using soft computing techniques: a meta-analysis. Multimed. Tools Appl. **78**(17), 23973–24010 (2019)
7. Patra, B.G., Das, D., Das, A.: Sentiment analysis of code-mixed Indian languages: an overview of SAIL_Code-Mixed Shared Task@ ICON-2017. arXiv preprint. arXiv:1803.06745 (2018)
8. Parshad, R.D., Bhowmick, S., Chand, V., Kumari, N., Sinha, N.: What is India speaking? Exploring the "Hinglish" invasion. Phys. A **449**, 375–389 (2016)
9. Jain, D., Kumar, A., Garg, G.: Sarcasm detection in mash-up language using soft-attention based bi-directional LSTM and feature-rich CNN. Appl. Soft Comput. **91**, 106198 (2020). https://doi.org/10.1016/j.asoc.2020.106198
10. Rosa, H., Pereira, N., Ribeiro, R., Ferreira, P.C., Carvalho, J.P., Oliveira, S., Trancoso, I.: Automatic cyberbullying detection: a systematic review. Comput. Hum. Behav. **93**, 333–345 (2019)
11. Salawu, S., He, Y., Lumsden, J.: Approaches to automated detection of cyberbullying: a survey. IEEE Trans. Affect. Comput. **1**, 1–20 (2017)
12. Reynolds, K., Kontostathis. A., Edwards, L.: Using machine learning to detect cyberbullying. In: Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference, vol. 2, pp. 241–244. IEEE (2011)
13. Dinakar, K., Reichart, R., Lieberman, H.: Modeling the detection of textual cyberbullying. In: International AAAI Conference on Web and Social Media, North America, July 2011 (2016)
14. Dadvar, M., Trieschnigg, D., Ordelman, R., de Jong, F.: Improving cyberbullying detection with user context. In: European Conference on Information Retrieval, pp. 693–696. Springer, Berlin, Heidelberg (2013)
15. Dadvar, M., Trieschnigg, D., de Jong, F.: Experts and machines against bullies: a hybrid approach to detect cyberbullies. In: Canadian Conference on Artificial Intelligence, pp. 275–281. Springer, Cham (2014)
16. Kontostathis, A., Reynolds, K., Garron, A., Edwards, L.: Detecting cyberbullying: query terms and techniques. In: Proceedings

of the 5th Annual ACM web Science Conference, pp. 195–204 (2013)
17. Potha, N., Maragoudakis, M., Lyras, D.: A biology-inspired, data mining framework for extracting patterns in sexual cyberbullying data. Knowl. Based Syst. **96**, 134–155 (2016)
18. Hosseinmardi, H., Rafiq, R.I., Han, R., Lv, Q., Mishra, S.: Prediction of cyberbullying incidents in a media based social network. In: Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 186–192 (2016)
19. Hammer, H.L.: Automatic detection of hateful comments in online discussion. In: International Conference on Industrial Networks and Intelligent Systems, pp 164–173. Springer, Cham (2016)
20. Sarna, G., Bhatia, M.P.: Content based approach to find the credibility of user in social networks: an application of cyberbullying. Int. J. Mach. Learn. Cybern. **8**(2), 677–689 (2017)
21. Zhang, X., Tong, J., Vishwamitra, N., Whittaker, E., Mazer, J.P., Kowalski, R., Hu, H., Luo, F., Macbeth, J., Dillon, E.: Cyberbullying detection with a pronunciation based convolutional neural network. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 740–745 (2016)
22. Zhao, R., Mao, K.: Cyberbullying detection based on semantic-enhanced marginalized denoising autoencoder. IEEE Trans. Affect. Comput. **8**(3), 328–339 (2017)
23. Zhao, R., Zhou, A., Mao, K.: Automatic detection of cyberbullying on social networks based on bullying features. In: Proceedings of the 17th International Conference on Distributed Computing and Networking, pp. 43–48 (2016)
24. Raisi, E., Huang, B.: Cyberbullying detection with weakly supervised machine learning. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 409–416. ACM (2017)
25. Rakib, T.B., Soon, L.K.: Using the Reddit Corpus for cyberbully detection. In: Asian Conference on Intelligent Information and Database Systems, p. 180. Springer, Cham (2018)
26. Ptaszynski, M., Pieciukiewicz, A., Dybała, P.: Results of the PolEval 2019 shared task 6: first dataset and open shared task for automatic cyberbullying detection in Polish Twitter. In: Proceedings of the PolEval2019 Workshop, p. 89 (2019)
27. Gordeev, D.: Automatic detection of verbal aggression for Russian and American image boards. Procedia Soc. Behav. Sci. **236**, 71–75 (2016)
28. Ibrohim, M.O., Budi, I.: Multi-label hate speech and abusive language detection in Indonesian Twitter. In: Proceedings of the Third Workshop on Abusive Language Online, pp. 46–57 (2019)
29. Pratiwi, N.I., Budi, I., Jiwanggi, M.A.: Hate Speech Identification using the Hate Codes for Indonesian Tweets. In: Proceedings of the 2019 2nd International Conference on Data Science and Information Technology, pp. 128–133 (2019)
30. Haidar, B., Chamoun, M., Serhrouchni, A.: Multilingual cyberbullying detection system: detecting cyberbullying in Arabic content. In: 2017 1st Cyber Security in Networking Conference (CSNet), pp. 1–8. IEEE (2017)
31. Haidar, B., Chamoun, M., Serhrouchni, A.: A multilingual system for cyberbullying detection: Arabic content detection using machine learning. Adv. Sci. Technol. Eng. Syst J. **2**(6), 275–284 (2017)
32. Pawar, R., Raje, R.R.: Multilingual cyberbullying detection system. In: 2019 IEEE International Conference on Electro Information Technology (EIT), pp. 040–044. IEEE (2019)
33. Arreerard, R., Senivongse, T.: Thai defamatory text classification on social media. In: 2018 IEEE International Conference on Big Data, Cloud Computing, Data Science and Engineering (BCD), pp. 73–78. IEEE (2018)
34. Tarwani, S., Jethanandani, M., Kant, V.: Cyberbullying detection in Hindi–English code-mixed language using sentiment classification. In: International Conference on Advances in Computing and Data Sciences, pp. 543–551. Springer, Singapore (2019)
35. Bohra, A., Vijay, D., Singh, V., Akhtar, S.S., Shrivastava, M.: A dataset of Hindi–English code-mixed social media text for hate speech detection. In: Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media, pp. 36–41 (2018)
36. Singh, V., Varshney, A., Akhtar, S. S., Vijay, D., Shrivastava, M.: Aggression detection on social media text using deep neural networks. In: Proceedings of the 2nd Workshop on Abusive Language Online (ALW2) ,pp. 43–50 (2018)
37. Santosh, T.Y.S.S., Aravind, K.V.S.: Hate speech detection in Hindi–English code-mixed social media text. In: Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, pp. 310–313 (2019)
38. Gupta, V.K.: "Hinglish" language-modeling a messy code-mixed language. arXiv preprint. arXiv:1912.13109 (2019)
39. Haidar, B., Chamoun, M., Yamout, F.: Cyberbullying detection: a survey on multilingual techniques. In: 2016 European Modelling Symposium (EMS), pp. 165–171. IEEE (2016)
40. Al-Hassan, A., Al-Dossari, H.: Detection of hate speech in social networks: a survey on multilingual corpus. In: 6th International Conference on Computer Science and Information Technology (2019)
41. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. IEEE Comput. Intell. Mag. **13**(3), 55–75 (2018)
42. Araci, D.: FinBERT: financial sentiment analysis with pre-trained language models. arXiv preprint. arXiv:1908.10063 (2019)
43. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems, pp. 3856–3866 (2017)
44. Kumar, A., Srinivasan, K., Cheng, W.H., Zomaya, A.Y.: Hybrid context enriched deep learning model for fine-grained sentiment analysis in textual and visual semiotic modality social data. Inf. Process. Manag. **57**(1), 102141 (2020)
45. Loper, E., Bird, S.: NLTK: The natural language toolkit. In: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, vol. 1, pp. 63–70. Association for Computational Linguistics (2002)
46. Knight, K., Graehl, J.: Machine transliteration. Comput. Linguist. **24**(4), 599–612 (1998)
47. Kumar, A., Jaiswal, A.: Swarm intelligence based optimal feature selection for enhanced predictive sentiment accuracy on Twitter. Multimed. Tools Appl. **78**(20), 29529–29553 (2019)
48. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
49. Zhao, W., Ye, J., Yang, M., Lei, Z., Zhang, S., Zhao, Z.: Investigating capsule networks with dynamic routing for text classification. arXiv preprint. arXiv:1804.00538 (2018)
50. Graves, A., Jaitly, N., Mohamed, A.R.: Hybrid speech recognition with deep bidirectional LSTM. In: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 273–278. IEEE (2013)
51. Srivastava, S., Khurana, P., Tewari, V.: Identifying aggression and toxicity in comments using capsule network. In: Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), pp. 98–105 (2018)