LIMSI_UPV at SemEval-2020 Task 9: Recurrent Convolutional Neural Network for Code-mixed Sentiment Analysis

Somnath Banerjee[1], **Sahar Ghannay**[1], **Sophie Rosset**[1], **Anne Vilnat**[1], **Paolo Rosso**[2]
[1] LIMSI, CNRS, Université Paris-Saclay, Orsay, France
[2] Universitat Politècnica de València, Spain
[1]firstname.lastname@limsi.fr
[2]prosso@dsic.upv.es

Abstract

This paper describes the participation of LIMSI_UPV team in SemEval-2020 Task 9: Sentiment Analysis for Code-Mixed Social Media Text. The proposed approach competed in SentiMix Hindi-English subtask, that addresses the problem of predicting the sentiment of a given Hindi-English code-mixed tweet. We propose Recurrent Convolutional Neural Network that combines both the recurrent neural network and the convolutional network to better capture the semantics of the text, for code-mixed sentiment analysis. The proposed system obtained 0.69 (best run) in terms of F1 score on the given test data and achieved the 9th place in the SentiMix Hindi-English subtask.

1 Introduction

In this digital era, users express their personal thoughts and opinions regarding a wide range of topics on social media platforms such as blogs, micro-blogs (e.g., Twitter), and chats (e.g., WhatsApp and Facebook messages). Multilingual societies like India with a decent amount of internet penetration widely adopted such social media platforms. However, the regional language influences the proliferation of the Hindi-English Code-Mixed (CM) data. Sentiment analysis of these end-user data from social media is a crucial resource for commerce and governance. However, in contrast to the classical sentiment analysis methods, which were originally designed for dealing with well-written product reviews, CM texts from social media often contain misspellings (often intentional), badly cased words, letter substitutions, ambiguities, non standard abbreviations, improper use of grammar, etc.

CM poses several unseen difficulties to natural language processing (NLP) tasks such as word-level language identification, part-of-speech tagging, dependency parsing, machine translation and semantic processing. In the last few years, a number of workshops such as *Linguistic Code-Switching Workshops*[1] and shared tasks such as *Mixed Script Information Retrieval* (Banerjee et al., 2020) have been organized due to the emerging popularity of code-mixing. To promote research in this area, Task 9 of SemEval-2020 was devoted to CM sentiment analysis in Twitter. The goal of the task was to automatically classify the polarity of a given CM Twitter post into one of the three predefined categories: *positive*, *negative* and *neutral*. The CM languages are English-Hindi and English-Spanish; for a more detailed description of the task see (Patwa et al., 2020).

In this paper, we present a deep learning approach, using a Recurrent Convolutional Neural Network for the task of automatic CM sentiment classification of tweets. The rest of the paper is structured as follows. Section 2 provides background in brief. Section 3 provides the system overview and Section 4 describes our approach in detail. In Section 5, we discuss the analysis and evaluation results for our system. We conclude our work in Section 6.

2 Background

Sentiment classification is the task of detecting whether a textual item (e.g., a product review, a blog post, an editorial, etc.) expresses a POSITIVE or a NEGATIVE opinion in general or about a given entity, e.g., a product, a person, a political party, or a policy (Nakov et al., 2016). Classifying tweets according to

[1]https://code-switching.github.io/2020/

sentiment has many applications in political science, social sciences, market research, and many others (Martínez-Cámara et al., 2014; Mejova et al., 2015). Although initially sentiment identification was focused on newswire text (Baccianella et al., 2010), later research turned towards social media (Rosenthal et al., 2015). Since 2013, a sentiment classification task on Twitter data have been organized in SemEval campaigns.

Most of the earlier approaches to this problem were based on hand crafted features and sentiment lexicons (Pak and Paroubek, 2010; Mohammad et al., 2013). These features were then used as input to classifiers (e.g., Support Vector Machines). However, such approaches required extensive domain knowledge, were laborious to define, and can lead to incomplete or over-specific features.

Recently, researchers pay their attention to sentiment polarity detection on CM data. However, a few research work have been carried out in particular Hindi-English CM data with different approaches: lexicon lookup (Sharma et al., 2015), sub-word with CNN-LSTM (Joshi et al., 2016), Siamese networks (Choudhary et al., 2018), dual Encoder Network with features (Lal et al., 2019). Lai et al. (2015) proposed Recurrent Convolutional Network for text classification which is a foundational task in many NLP applications. We followed this model in our task.

3 System overview

We are inspired by the model proposed in (Lai et al., 2015) particularly proposed for the text classification task. The proposed model takes sequence of CM words as input and provides sentiment polarity class as output. The recurrent structure of the proposed model captures the contextual information during the learning of the word representation, and the max-pooling layer identifies the key CM words. If T, s and θ denote a CM tweet made up of sequence of CM words (T = w_1, w_2, w_3, ..., w_n), the sentiment polarity class and parameters of the neural network respectively, $p(s|T, \theta))$ denotes the probability of the tweet T having sentiment polarity s, where s could be any one of sentiment polarity classes, i.e., $s \in$*{positive, negative, neutral}*.

In this model, a CM word is represented by combining the word and its both side contexts. If $c_l(w_i)$ and $c_r(w_i)$ denote the left-side and right-side contexts of a CM word w_i, a word is represented as a concatenation of the left-side context vector $c_l(w_i)$, the word embedding $e(w_i)$ and the right-side context vector $c_r(w_i)$.

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)]$$

where, x_i is the representation of the i^{th} word (i.e., w_i) in T.

Both the contexts (i.e., $c_l(w_i)$ and $c_r(w_i)$) are calculated as follows:

$$c_l(w_i) = f(W^{(l)}c_l(w_{i-1}) + W^{(s_l)}e(w_{i-1}))$$
$$c_r(w_i) = f(W^{(r)}c_r(w_{i+1}) + W^{(s_r)}e(w_{i+1}))$$

Where,

f: is a non-linear activation function;

$e(w_{i-1})$: the word embedding of word w_{i-1};

$W^{(l)}$: a matrix that transforms the hidden layer (context) into the next hidden layer;

$W^{(sl)}$: a matrix that is used to combine the semantic of the current word with the next word's left context.

In the forward scan, the recurrent structure of the model obtains all the c_l of the CM tweet, whereas, it obtains all the c_r in a backward scan of the CM tweet. After obtaining the x_i for the word w_i, a linear transformation together with the tanh activation function is applied to x_i and the result is sent to the next layer:

$$y_i^{(2)} = \tanh(W^{(2)}x_i + b^{(2)})$$

After calculating all of the representations of words, a max-pooling layer is applied. Hence, the pooling layer utilizes the output of the recurrent structure as the input. This layer attempts to find the most important latent semantic factors in the CM tweet:

$$y^{(3)} = \max_{i=1}^{n} y_i^{(2)}$$

Finally, in the output layer, a softmax function is applied that provides the sentiment polarity probabilities:

$$y^{(4)} = W^{(4)} y^{(3)} + b^{(4)}$$

$$p_i = \frac{\exp(y_k^{(4)})}{\sum_{k=1}^{n} \exp(y_k^{(4)})}$$

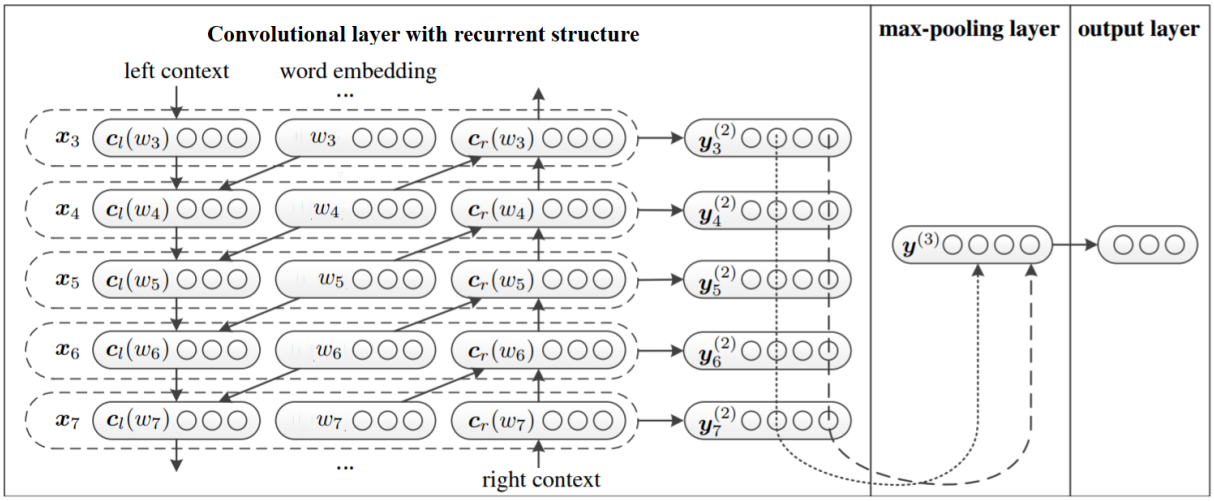

Figure 1: Recurrent Convolutional Neural Network (Lai et al., 2015) model

The Figure 1 depicts the architecture of the proposed model. From Figure 1, we could see that $c_l(w_n)$ captures the left-side context, whereas, $c_l(w_1)$ captures the right-side context of a word w_i.

4 Experimental setup

Data: For sentiment analysis on Hindi-English CM tweets, we used the dataset provided by the organizers of Task 9 at SemEval-2020. The training dataset consists of 15 thousand tweets. Whereas, the validation dataset as well as test dataset contain 3 thousand tweets each. The details of the dataset are given in (Patwa et al., 2020). For this task, we did not use any external dataset.

Preprocessing: The CM Twitter data is not formal in nature. Also, often casing is not followed properly. As a result, it misleads the language processing task. Therefore, we converted the entire tweets into lower case. After observing the CM dataset, we found that the urls are not contributing any information. While writing, the writer of a tweet addresses a person by mentioning the person's name followed by @ or by twitter id which starts with @. Similarly, to specify a particular topic, the writer uses the topic name that starts with #. To capture the semantics and syntax of a tweet, we replaced the person and a topic being addressed with a MENTION and a TOPIC token respectively. The following steps were applied on each CM tweet:

- Tokens were converted to lowercase.
- If a token is http or https, the following tokens are merged to identify as links and were deleted.
- Garbage tokens (e.g., â€¦, à ¥¥, etc.) were deleted.
- If a token is @, the following token is merged with the first token (i.e., @) and the new token is replaced with a MENTION token. For example, the tokens @ and *bomanirani* are merged and replaced with a MENTION token.
- If a token is #, the following token is merged with the first token (i.e., #) and the new token is replaced with a TOPIC token. For example, the tokens # and *LoveIsLove* are merged and replaced with a TOPIC token.

- Emoji's with text were divided into two tokens. For example, *he*☺ becomes *he* and ☺.

- If a token contains more than one emoji, each emoji was considered as a token. For example, ☺☹ becomes ☺ and ☹.

Embeddings: Following Collobert et al. (2011), a lot of authors argued that word embedding plays a vital role to improve natural language task performance. Hence, we experimented the use of word embeddings to improve the performance of our proposed models. Using the fastText (Bojanowski et al., 2017), we prepared two embedding models: Skip-gram and Cbow. After empirically evaluating the performance on validation set, the embeddings' dimensionality was set to 300 for all the embeddings. The embeddings are trained on training data using the parameters: lr=0.05, context window=5, epochs=5, minimal number of word occurences=5, dimensionality=300.

Experiment: We carried out two experiments with similar settings except different word embedding approaches: Skip-gram for SkipGRun, and Cbow for CbowRun.

Hyper-parameters: After evaluating the model performance on the validation data, the optimal values of the hyper-parameters were set. We used the following list of hyper-parameters: learning rate = 0.6, word embedding vector size = 300, hidden layers= 2, hidden layer size = 64, context vector size = 5, dropout rate = 0.1, optimizer = stochastic gradient descent, loss function= negative log likelihood, and batch size = 64.

5 Results and Analysis

As mentioned in the previous section, we carried out experiments with our proposed model using two settings. All the presented experiments are evaluated on the test data for the given task. The performance of the systems were evaluated using F1 averaged across the positive, negative and the neutral.

	Precision	Recall	F1 score
Baseline	-	-	0.654
SkipGRun	0.6952	0.6893	**0.6913**
CbowRun	0.6566	0.6556	0.6560

Table 1: Evaluation results on test data

The organizer baseline F1 scores for the validation and test data are 0.58 and 0.654 respectively. The details of the baseline are given in (Patwa et al., 2020). The obtained results with our submitted runs are given in Table 1. For SkipGRun, we achieved 0.6913 of F1 score with 0.6952 and 0.6893 of precision and recall respectively. The SkipGRun outperformed the CbowRun by around 0.40 in terms of F1 score. CbowRun outperformed the organizers' baseline by a slight margin, however, SkipGRun outperformed the baseline by around 0.4 in terms of averaged F score.

For SkipGRun, the confusion matrix and the performance of class-wise sentiment polarities are presented in Table 2 and Table 3, respectively. The SkipGRun is the best among the two submitted runs. From the confusion matrix (cf. Table 2), we can observe that the system is more successful in identifying the *positive* polarity class (F1 score 0.76) than the polarity classes: *negative* (F1 score 0.71) and *neutral* (F1 score 0.62). This run performed well on distinguishing a *positive* tweet from *negative* tweet and vice versa. Therefore, only 35 *positive* tweets were misclassified as *negative* and 24 *negative* tweets were misclassified as *positive*. However, it has a problem to separate *positive* and *negative* tweets from *neutral*. Hence, a lot of *positive* and *negative* tweets were misclassified as *neutral*. From Table 2, we could say that around 24% of *positive* tweets and 28% of *negative* tweets were identified as *neutral*. Similarly, a number of *neutral* tweets were misclassified as *positive* and *negative*. From Table 3, it is evident that the positive tweets were the easiest class to predict. The F1 scores were 0.76 (positive class), 0.71 (negative class) and 0.62 (neutral class).

Table 4 and Table 5 present the confusion matrix and the performance of class-wise sentiment polarities for CbowRun. Like SkipGRun, the CbowRun is also able to successfully identify the *positive* polarity class (F1 score 0.72) in comparison to others (cf. Table 4). For CbowRun, we observed similar characteristics of results like SkipGRun. Although the results obtained for *negative* tweets are almost similar with

	Positive	Negative	Neutral
Positive	729	35	236
Negative	24	624	252
Neutral	175	210	715

Table 2: SkipGRun: Confusion matrix

	Precision	Recall	F1 score
Positive	0.79	0.73	0.76
Negative	0.72	0.69	0.71
Neutral	0.59	0.65	0.62

Table 3: SkipGRun: Polarity class-wise performance

SkipGRun (F1 score 0.71) and CbowRun (F1 score 0.70), the difference in performance is notable for *positive* and *neutral* tweets. From Table 5, we can see that for CbowRun the F1 scores were 0.73 (positive class), 0.70 (negative class) and 0.57 (neutral class).

	Positive	Negative	Neutral
Positive	709	41	250
Negative	33	630	237
Neutral	232	240	628

Table 4: CbowRun: Confusion matrix

	Precision	Recall	F1 score
Positive	0.73	0.71	0.72
Negative	0.69	0.70	0.70
Neutral	0.56	0.57	0.57

Table 5: CbowRun: Polarity class-wise performance

To get a deeper analysis of the results, we also performed the error analysis. We observed that the system could not identify the sentiment when Hindi lyrics are used in tweets. For example: *RT MENTION Aankho ki hai ye khawaise ki chehre se teri na hate ...* (id:36925, gold:*positive*, predicted:*negative*); *RT MENTION Saare jahan se achha # Hindustan hamara Ham bulbulain hai iski yeh gulsitan hamara...* (id:32707, gold:*positive*,predicted:*neutral*).

The system often failed to identify the sentiment, when a long tweet consists of a number of complete or incomplete sentence with mixed sentiments. For example: *RT MENTION I miss childhood days ... No problems ... No hates ... No shames ... No stress ... No heartbreaks ... Go school ... Life was easy* (id:25011, gold:*neutral* predicted:*positive*); *RT MENTION PM Modi won 356 seats & thanked all 130 crore people for faith in democracy . Sonia won 52 seats & thanked only the 12 cro* (id:30674, gold:*positive*, predicted:*neutral*)

The proposed system often failed to identify the tweets that have any punctuation or delimiter to separate the sentence clauses. For example: *Jb Koi aapke liye kam krta h aapke liye 5sal din rat ek krta h aap pe vishvas krta h aapka vishvas jit ta h jis* (id:31020, gold:*neutral*, predicted:*positive*)

We observed that mostly the system misclassified the *positive* and *negative* tweets as *neutral* and vice versa. However, we observed that there are some tweets that may arise some arguments, such as *MENTION Beautiful words and its one of my fav song* (id:31662), *MENTION Sir me from Bihar Love you You have Great job for Bjp continue We are with you* (id:39755), etc. tagged as *neutral* and our system identified as *positive*. Identifying the polarity of an entity or entity's aspect in the tweets along with more training data could help to resolve these issues.

6 Conclusion

This paper describes the approach we proposed for SemEval-2020 Task 9: Sentiment Analysis for CM Social Media Text (SentiMix Hindi-English). In our approach, we pre-processed the CM tweets and proposed a Recurrent Convolutional Neural Network for the sentiment analysis of CM tweets. We submitted two runs and obtaining promising results: our best run obtained 0.691 of F1 averaged across the positives, negatives and the neutral. We observed that the proposed architecture occasionally strives to separate the *positive* and *negative* polarities from the *neutral* and vice versa.

For future work, we will explore the performance of our model with larger corpora against the testing set. Also, we would like to investigate other embedding choices such as BERT (Devlin et al., 2019). Moreover, due to the impact that irony and sarcasm have on sentiment analysis (Hernández Farıas and Rosso, 2016) it would be interesting to apply deep learning techniques to detect irony (Zhang et al., 2019) but in a code-mixed scenario.

Acknowledgements

The research work of the first four authors was supported by ERA-Net CHIST-ERA LIHLITH Project funded by ANR (France) project ANR-17-CHR2-0001-03. The research work of the last author was partially funded by the Spanish MICINN under the project MISMIS-FAKEnHATE on Misinformation and Miscommunication in social media: FAKE news and HATE speech (PGC2018-096212-B-C31).

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Somnath Banerjee, Monojit Choudhury, Kunal Chakma, Sudip Kumar Naskar, Amitava Das, Sivaji Bandyopadhyay, and Paolo Rosso. 2020. MSIR@FIRE: a comprehensive report from 2013 to 2016. *SN Computer Science*, 1(1):1–15.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Nurendra Choudhary, Rajat Singh, Ishita Bindlish, and Manish Shrivastava. 2018. Sentiment analysis of code-mixed languages leveraging resource rich languages. *arXiv preprint arXiv:1804.00806*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

DI Hernández Farıas and Paolo Rosso. 2016. Irony, sarcasm, and sentiment analysis. chapter 7. *Sentiment Analysis in Social Networks. Morgan Kaufmann*, pages 113–127.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI Conference on Artificial Intelligence*.

Yash Kumar Lal, Vaibhav Kumar, Mrinal Dhar, Manish Shrivastava, and Philipp Koehn. 2019. De-mixing sentiment from code-mixed text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 371–377.

Eugenio Martínez-Cámara, M Teresa Martín-Valdivia, L Alfonso Urena-López, and A Rturo Montejo-Ráez. 2014. Sentiment analysis in twitter. *Natural Language Engineering*, 20(1):1–28.

Yelena Mejova, Ingmar Weber, and Michael W Macy. 2015. *Twitter: a digital socioscope*. Cambridge University Press.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: building the state-of-the-art in sentiment analysis of tweets. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 321–327.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California, June. Association for Computational Linguistics.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Dan Garrette, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. SemEval-2020 Sentimix Task 9: overview of sentiment analysis of code-mixed tweets. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain, Sep. Association for Computational Linguistics.

Sara Rosenthal, Saif M Mohammad, Preslav Nakov, Alan Ritter, Svetlana Kiritchenko, and Veselin Stoyanov. 2015. SemEval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015)*, page 451–463. Association for Computational Linguistics.

Shashank Sharma, PYKL Srinivas, and Rakesh Chandra Balabantaray. 2015. Text normalization of code mix and sentiment analysis. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1468–1473. IEEE.

Shiwei Zhang, Xiuzhen Zhang, Jeffrey Chan, and Paolo Rosso. 2019. Irony detection via sentiment-based transfer learning. *Information Processing & Management*, 56(5):1633–1644.