ISSN 2007-9737

Character Embedding for Language Identification in Hindi-English Code-mixed Social Media Text

P. V. Veena, M. Anand Kumar, K. P. Soman

Amrita University, Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India

m_anandkumar@cb.amrita.edu, veenakrt27@gmail.com, kp_soman@amrita.edu

Abstract. Social media platforms are now widely used by the people to express their opinion or interest. The language used by the users in social media earlier was purely English. Code-mixed text, i.e., mixing of two or more languages, is commonly seen now. In code-mixed data, one language will be written using another language script. So to process such code-mixed text, identification of language used in each word is important for language processing. The main objective of the work is to propose a technique for identifying the language of Hindi-English code-mixed data used in three social media platforms namely, Facebook, Twitter, and WhatsApp. The classification of Hindi-English code-mixed data into Hindi, English, Named Entity, Acronym, Universal, Mixed (Hindi along with English) and Undefined tags were performed. Popular word embedding features were used for the representation of each word. Two kinds of embedding features were considered - word-based embedding features and character-based context features. The proposed method was done with the addition of context information along with the embedding features. A well-known machine learning classifier, Support Vector Machine was used to train and test the system. The work on Language Identification in code-mixed text using character-based embedding is a novel approach and shows promising results.

Keywords. Language identification, code-mixed, character embedding, word embedding, support vector machine, 3-gram embedding, context appending.

1 Introduction

Humans use natural language as their medium of communication. Natural Language Processing (NLP), is an area of Artificial Intelligence where we train the machine to understand and process the text to make human-computer interactions more efficient. Applications of NLP lies under several fields like machine translation, text processing, entity extraction and so on [15]. A large amount of data is now available on the Web as text. With the emergence of several social media platforms and availability of a large amount of text data in them, NLP plays a great role in understanding and generating data today.

The social media platforms are used widely today by people to discuss the interests, hobbies, reviews on products, movies and so on. In earlier days, the language used in such platforms was purely English. Today mixing multiple languages together is a popular trend. These kinds of languages are called code-mixed language.

An example of Hindi-English code-mixed text is shown below:

Array yaar, ek super idea **thi mere paas**...
hi hi hi en en hi hi hi

We can observe from the example that the Hindi words, tagged as *hi*, were written in Roman Script instead of Unicode characters.

The code-mixed text is more often ambiguous. In the above example, when a non-Hindi user reads the word '*Array*' he can get confused whether the word is English or Hindi. So to remove such ambiguity, identification of language is important. For processing monolingual text, the primary step would be Part-Of-Speech (POS), tagging of the text. But in case of social media text, the primary

feature to be considered is identification of the language particularly for code-mixed text [4].

The language identification for code-mixed text proposed in this paper is implemented using word embedding models. The term word embedding refers to the vector representation of the given data capturing the semantic relation between the words in the data. The work is a generalized approach because this system can be extended for other NLP applications since only word embedding features are considered. The work involves features obtained from two embedding models, word-based embedding and character-based embedding. A comparison on the performance of the two models with the addition of contextual information is performed in this paper. The training and testing of the system is done using machine learning based classifier, Support Vector Machine [10].

The remainder of the paper is organized as below: Section 2, gives a brief overview of the related works. A discussion on the methodology proposed in this paper is given in Section 3. The word-based embedding is explained in Section 3.1 and Section 3.2, describes character-based embedding method. The dataset description is stated in Section 4. Experimental results obtained is explained in Section 5. Section 6, analyses the inferences obtained from the work done. The conclusion of the paper and the future work is given in Section 7.

2 Related Work

Language Identification (LID), is a primary task in many text processing applications and hence several research is going on this area especially with the code-mixed data. Earlier work includes a shared task on Language Identification for twitter data performed at Spanish Society for Natural Language Processing (SEPLN) 2014 contest. The conference focused on code-mixed Catalan-English, Spanish-English, Portuguese-English and Basque-English text [23]. Extended Markov Models were used to perform word-level language identification on Twitter code-mixed English-Nepali, English-Mandarin, English-Spanish, and Arabic-Arabic dialects [11]. Another task was performed for LID on code-switched task with English-Spanish and Modern Standard Arabic - Dialectal Arabic dataset in [13].

A shared task on Mixed Script Information Retrieval (MSIR) 2015 was conducted in which a subtask includes language identification of 8 code-mixed Indian Languages, Telugu, Tamil, Marathi, Bangla, Gujarati, Hindi, Kannada, and Malayalam, each mixed with English [19]. The MSIR language identification task was implemented by using machine learning based SVM classifier and obtained an accuracy of 76% [16]. Word level language identification was performed for English-Hindi using supervised methods in [9]. Naive Bayes classifier was used to identify the language of Hindi-English data and an accuracy of 77% was obtained [7].

Language Identification is also performed as a primary step to several other applications. [6], implemented a sentiment analysis system which utilized MSIR 2015 English-Tamil, English-Telugu, English-Hindi, and English-Bengali code-mixed dataset. Another emotion detection system was developed for Hindi-English data with machine learning based and Teaching Learning Based Optimization (TLBO), techniques [20]. Part-of-Speech tagging was done for English-Bengali-Hindi corpus including the language identification step in [5].

Since code-mixed script is the common trend in the social media text today, many researches are going on for the information extraction from such text. An analysis of behavior of code-mixed Hindi-English Facebook dataset was done in [2]. POS Tagging technique was performed on code-mixed social media text in Indian languages [22]. A shared task was organized for entity extraction on code-mixed Hindi-English and Tamil-English social media text [18]. Entity extraction for code-mixed Hindi-English and Tamil-English dataset was performed with embedding models [17]. Many NLP applications consider word embedding as an efficient feature. [17], implemented entity extraction system using word-embedding features. It was also considered to be efficient in implementing user profiling system for social networks [1].

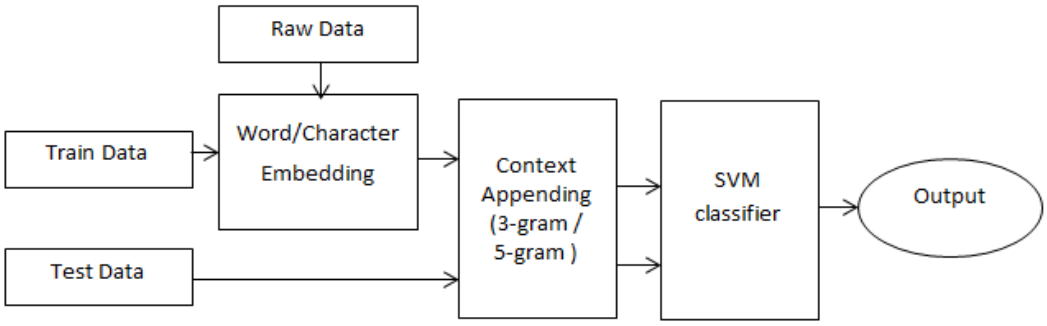


Fig. 1. Methodology of the proposed system

3 Methodology

In this work two systems were developed based on word-based embedding features and character-based context features. For the character-based system, same procedure as that of word-based is done except that the vectors are character vectors. The methodology of the proposed system is illustrated in Fig. 1.

For the embedding to capture the word representation more effectively, additional data apart from train and test data must be provided to the embedding model. The additional data used here is also a code-mixed Hindi-English social media data collected from other shared tasks. The input for the word embedding will be the train data and the additionally collected dataset. The embedding model generates the vector of each vocabulary (unique), words present in the data. Along with extracting the feature vectors of the train data, its context information is also extracted. The incorporation of the immediate left and right context features with the features of the current word is called 3-gram context appending. 5-gram features were also extracted, which is the extraction of features from two neighboring words before and after the current word.

So if the vocabulary size of the training data is $|V|$, and the embedding feature size generated is 100 for each word, then after context appending with 3-gram features, a matrix of size $|V|$ x 300 is obtained. 5-gram appending will result in a matrix of size $|V|$ x 500.

The test data was also given to the embedding models. The data were then appended with the 3-gram and 5-gram context information. These are then fed to a machine learning based classifier, SVM-Light [10], to train and test the system.

3.1 Word-Based Embedding Model

The word-based embedding model is used to find the feature vectors that are useful in predicting the neighboring tokens in a context. The feature vector for this model is generated using Skip-gram architecture of popular Word2vec package proposed by Mikolov et al. Apart from the skip-gram model, another architecture Continuous Bag of Words (CBOW), is also present [12].

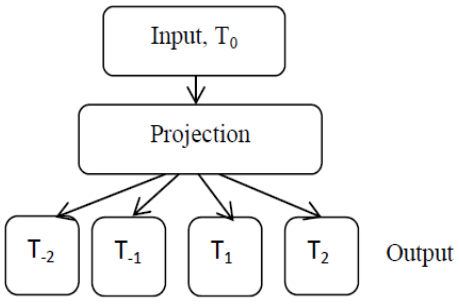


Fig. 2. Skip-gram Model

The CBOW constructs the target word from their context information while skip-gram model functions in reverse. The illustration of Skip-gram model is shown in Fig. 5.

Here the input token is T_0 which is fed to a log-linear classifier to predict the neighboring words. T_{-2}, T_{-1}, T_1 and T_2 are the words that are before and after the current word.

When the data is given to the Skip-gram model, it maximizes the average log probability, given by L, which is formulated as in Eq. 1:

$$L = \frac{1}{\mathrm{N}} \sum_{\mathrm{n}=1}^{\mathrm{N}} \sum_{-\mathrm{x} \leq \mathrm{i} \leq \mathrm{x}} \log \mathrm{p}(\mathrm{T_{n+i}} | \mathrm{T_n}). \tag{1}$$

In the equation, *N* is the total number of words in the train data and *x* is the context size. *p* is the softmax probability which is given using the Eq. 2:

$$p(T_j|T_k) = \frac{\exp(V'_{T_j}(V_{T_k}))}{\sum\limits_{w=1}^{W} \exp(V'_T(V_{T_k}))}, \tag{2}$$

where *W* is the vocabulary size, $p(T_j|T_k)$, is the probability of occurrence of the next word given the current word representation. *V'* is the output vector representation.

The dataset along with the additional dataset collected were given to the skip-gram model. The vector size to be generated were fixed as 100. The skip-gram model generates a vector of size 1×100 for each vocabulary word in the dataset. From this, the vectors for the training data were extracted. The context appending features were then extracted from this file. The final training file for the classifier will consist of the tokens in the train data, their language tag and the 3-gram and 5-gram context feature vectors extracted. Thus three training files are generated with $|V|$ x 101, $|V|$ x 301 and $|V|$ x 501 dimension.

The test data with its corresponding context appended vectors are fed to the classifier for testing the system.

3.2 Character-Based Embedding Model

The procedure for character embedding is the same as that of skip-gram based word embedding. Each token in the train data is split into characters and these are fed to the system. This will generate a vector for each character with the vector size fixed as 100. The vectors generated for each character is used to create vectors for each token as in Eq. 3 [14]:

$$y = x + Sh(W, c_{t-k}, ..., c_{t+k}; C). \tag{3}$$

where *x* and *S* are the softmax parameters and *h* is the combination of character and word embedding features. *C* stands for character vectors and *W* stands for word vectors. c_{t-k},... c_{t+k}, are the characters in the train data.

The procedure to generate a particular token's feature from the character vectors is shown in Fig. 3:

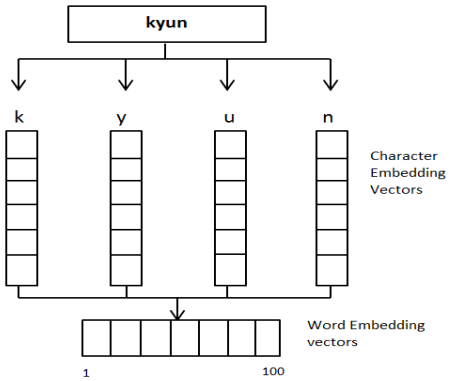


Fig. 3. Character embedding technique

The word *kyun* is split into characters and given to the system to produce embedding feature vector. The vectors are generated for each character in the word. These are then transformed to produce the character-based embedding vector of the word *kyun* using Eq. 3.

The vectors for each token is then used to extract the context feature vectors. The feature vector with context features is appended along with the language tag and is fed to the classifier for training the system. Similar procedure is done for the test file. The vectors generated from character embedding model, is then transformed as a context matrix for the test data. This context matrix with the test words are fed to the classifier for testing the system.

ISSN 2007-9737

Table 1. Dataset Size of ICON 2016 POS Tagging Task

Data	Number of Sentences		Number of Tokens		Average tokens per Sentence	
	Train Data	Test Data	Train Data	Test Data	Train Data	Test Data
Facebook	772	111	20615	2167	26.70	19.52
Twitter	1096	110	17311	2163	15.79	19.66
Whatsapp	763	219	3218	802	4.22	3.66

4 Dataset Description

The dataset used for this work is obtained from POS Tagging task for Hindi-English code-mixed social media text conducted by ICON 2016 [8]. The dataset contains text of three social media platforms namely Facebook, Twitter and Whatsapp. The train data provided contains the tokens of the dataset with its corresponding language tag and POS tag. The POS tags are omitted since they are not used for this work. The tags present in the dataset are as follows:

1. acro
   - representing acronym
   - eg: *tv* for television
2. en
   - indicating english words
   - eg: *and, there*
3. hi
   - indicating hindi words
   - eg: *aisa, mera*
4. ne
   - indicating named entities like Person, Location and Organization
   - eg: *India, Facebook*
5. mixed
   - indicating words of Hindi-English combination
   - eg: *H3, Indiawaala*
6. univ
   - indicating tokens containing special characters and numbers
   - eg: *'@','$' , 0-9*
7. undef
   - indicating unidentified words
   - eg: *t.M, T*

All the seven tags are present in the Facebook dataset, where *'en', 'hi', 'ne', 'univ'* are the tags present in Twitter and Whatsapp data.

The size of the train and test data used is tabulated in Table 1. From the table, it can be observed that the average tokens per comment of Whatsapp train and test data is very less than Facebook and Twitter data. This may be due to the fact that Facebook and Twitter data mostly contains news articles and comments which makes the average tokens per comment count to be more while Whatsapp contains conversational short messages.

For generating the embedding vectors, more dataset has to be provided to efficiently obtain the distributional similarity of the data. The additional dataset collected along with the training data will be given to the embedding model. The Hindi-English additional code-mixed data were collected from Shared task on Mixed Script Information Retrieval (MSIR), conducted in year 2016 [3] & 2015 [19] and shared task on Code-Mix Entity Extraction task conducted by Forum for Information Retrieval and Evaluation (FIRE), 2016 [21]. Most of the data collected for embedding is Hindi-English code-mixed Twitter data. The size of the dataset used for embedding is given in Table 3.

That is, for context size 3, the features of the immediate neighbor is extracted and for context size 5, features of two words before and after the current word is extracted. Context appending was

Table 2. Cross Validation Results obtained for Character Embedding

		Known	Known Ambiguous	Unknown	Overall
Facebook	1-gram	98.17	87.91	84.86	95.71
	3-gram	98.09	87.38	83.67	95.46
	5-gram	98.02	86.56	84.53	95.50
Twitter	1-gram	94.18	71.22	76.91	89.56
	3-gram	94.28	71.93	77.09	89.74
	5-gram	93.92	70.98	77.08	89.48
Whatsapp	1-gram	96.45	59.46	73.79	89.66
	3-gram	96.55	56.71	72.42	89.59
	5-gram	96.35	60.18	74.72	89.92

Table 3. Number of sentences in the dataset used for embedding

Dataset		Count
ICON 2016 Train Data	Facebook	772
	Twitter	1096
	Whatsapp	763
MSIR 2016	Train	6139
	Test	4886
MSIR 2015	Train	389
CMEE-IL 2016	Train	2700
	Test	7429
Total		24228

done for each Facebook, Twitter and Whatsapp train as well as test data. These were given to the SVM-based classifier for training and testing. A 10-fold cross-validation was performed while training the classifier.

The cross-validation accuracies obtained for Facebook, Twitter, and Whatsapp with 1-gram, 3-gram and 5-gram features for character based embedding model and word based embedding model is given in Table 2 and Table 4 respectively. When comparing the overall accuracy obtained for Facebook, Twitter, and Whatsapp, we can see that the accuracy obtained with character-based embedding model is more than that with the word-based model.

It can also be observed that in the word-based embedding model, 3-gram-based features gives more accuracy than 1-gram and 5-gram context feature model while in character-based model 5-gram gives more accuracy than 1-gram and 3-gram context features except in the case of Twitter. For character-based embedding approach the unknown accuracies are more while using 5-gram meanwhile the unknown accuracy decreases for Facebook and Twitter data with 5-gram based word embedding technique.

The test data along with 1-gram, 3-gram and 5-gram feature vector was given to the classifier for testing. The predicted tags were compared with the gold standard test to evaluate the system. The system was evaluated based on Precision, Recall and F-measure performance metrics. Table 5, gives the results obtained for each tag in Facebook dataset with both character-based and word-based 1-gram, 3-gram and 5-gram models. In addition to these tags, the Facebook data also holds *'mixed'* and *'undef'* tags. They are not shown in the table since they resulted in 0 scores.

The Twitter and Whatsapp data contains 4 tags which are *'en', 'hi', 'ne'* and *'univ'*. The Precision, Recall and F-measure scores obtained for each tag are tabulated in Table 6 and Table 7 respectively.

5 Discussions

When observing Table 5 which shows the performance of Facebook Hindi-English code-mixed data, we can see that the F-score for *en*, *hi* and *univ* is better using character embedding than word embedding. It can also be seen that the 3-gram embedding gives better result than 1-gram and 5-gram for character based embedding

Table 4. Cross validation results obtained for word embedding

		Known	Known Ambiguous	Unknown	Overall
Facebook	1-gram	97.41	81.62	82.59	94.68
	3-gram	97.29	80.46	82.22	94.51
	5-gram	97.09	78.95	81.52	94.22
Twitter	1-gram	91.11	50.06	75.27	86.94
	3-gram	92.39	58.99	74.77	87.76
	5-gram	92.22	56.97	74.16	87.43
Whatsapp	1-gram	94.77	38.58	73.81	88.68
	3-gram	95.02	48.18	73.19	88.93
	5-gram	95.52	45.76	74.47	89.12

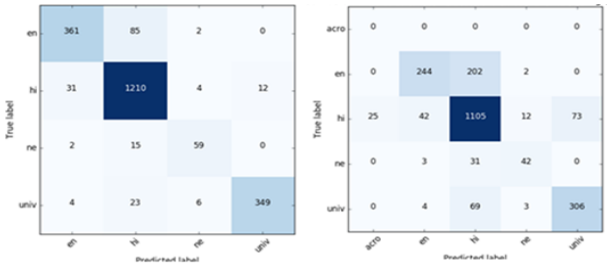
Table 5. Precision, recall and F-measure obtained for Facebook data

Type of embedding		acro			en			hi			ne			univ		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Character embedding	1-gram	95.24	76.92	85.11	85.83	85.48	85.65	91.76	94.12	92.92	98.48	61.76	64.95	99.13	97.85	98.48
	3-gram	95.00	73.08	**82.61**	86.45	87.34	**86.89**	92.42	94.39	**93.41**	70.65	63.73	**67.01**	99.34	97.85	98.59
	5-gram	95.00	73.08	82.61	86.19	86.72	86.45	92.23	94.12	93.36	65.35	64.71	65.02	99.35	98.06	**98.70**
Word embedding	1-gram	90.48	73.08	**80.85**	85.86	84.44	85.15	91.16	92.92	92.03	61.90	63.73	62.80	98.91	97.20	98.05
	3-gram	78.26	69.23	73.47	86.48	83.61	85.02	91.31	93.66	92.47	64.42	65.69	**65.05**	99.34	97.63	**98.48**
	5-gram	78.26	69.23	73.47	87.42	83.61	**85.47**	91.47	93.66	**92.55**	58.56	63.73	61.03	99.13	97.63	98.37

which is highlighted in bold characters. However for word-based embedding model, the 5-gram embedding gives better F-score for *en* and *hi* tags while 3-gram for *univ* and *ne* tags.

From the performance of Twitter data tabulated in Table 6, it is clearly seen that the word embedding 3-gram based model gives less score than other models. An increase of 20% can be observed for *en* and *ne* tags when using character-based embedding and almost 10% for *hi* tag. When analyzed, it was found that *hi* and *acro* tags were confused by the system when using word embedding based 3-gram model. This was solved when the size of the n-gram was increased to 5-gram. It can also be seen that the scores are slightly increase when the context information is included.

Table 7, holds tagwise accuracy for Whatsapp data. It can be observed from the table that same scores are obtained for *ne* and *univ* tags in the case of character embedding while the score decreases when using word embedding. For all the tags, the scores are less when compared to the Facebook and Twitter data. This is due to the fact that the average number of tokens per comment



(a) (b)

Fig. 4. Confusion matrix for Twitter with 3-gram embedding (a) Character-based and (b) Word-based

is very less than Facebook and Twitter (see Table 1). So the system needs more context information to identify the language. That is why the 5-gram embedding gives a better result than 1-gram and 3-gram for both character and word embedding techniques.

Fig. 4, shows a confusion matrix generated for Whatsapp dataset with 5-gram based embedding features. From the figure, it can be seen that for 'en' tag, the word-based method is better than the character-based embedding method while

Table 6. Precision, recall and F-measure obtained for Twitter data

Type of embedding		en			hi			ne			univ		
		P	R	F	P	R	F	P	R	F	P	R	F
Character embedding	1-gram	90.51	80.04	84.95	90.59	96.18	93.31	80.56	76.32	78.38	96.11	90.58	93.26
	3-gram	90.70	80.58	**85.34**	90.77	96.26	**93.44**	83.10	77.63	**80.27**	96.68	91.36	**93.94**
	5-gram	89.68	81.47	85.38	91.16	95.94	93.49	76.62	77.63	77.12	96.91	90.31	93.50
Word embedding	1-gram	87.94	83.04	85.42	91.62	94.75	93.16	86.76	77.63	**81.94**	94.09	91.86	92.96
	3-gram	83.28	54.46	65.86	78.54	87.91	82.96	71.19	55.26	62.22	80.74	80.10	80.42
	5-gram	89.71	81.70	**85.51**	91.23	96.02	**93.57**	78.67	77.63	78.15	97.20	90.84	**93.91**

Table 7. Precision, Recall and F-measure obtained for Whatsapp data

Type of embedding		en			hi			ne			univ		
		P	R	F	P	R	F	P	R	F	P	R	F
Character embedding	1-gram	75.41	40.17	52.42	70.22	93.36	80.15	100.00	25.00	**40.00**	94.74	65.06	77.14
	3-gram	79.13	39.74	52.91	70.49	95.13	**80.98**	100.00	25.00	**40.00**	95.95	65.14	**77.60**
	5-gram	77.42	41.92	**54.39**	70.88	94.25	80.91	100.00	25.00	**40.00**	95.95	65.14	**77.60**
Word embedding	1-gram	72.06	42.79	53.70	71.18	91.81	80.19	75.00	25.00	**37.50**	91.14	66.06	76.60
	3-gram	71.77	38.86	50.42	69.90	92.48	79.62	42.86	25.00	31.58	95.89	64.22	**76.92**
	5-gram	71.60	52.84	**60.80**	73.48	88.27	**80.20**	33.33	25.00	28.57	87.65	65.14	74.74

for 'hi' tags more words were identified with character-based embedding method.

The character-based technique with 5-gram embedding features delivered better accuracy than the other models except for Twitter dataset. This may be due to the fact that most of the data used for embedding is Twitter thereby obtaining precise word representation when using 5-gram word-based embedding methodology. So, if sufficient amount of data is available for any social media platform, then the word-based embedding technique would be the effective methodology.

A graphical representation of accuracies obtained for Facebook, Twitter and Whatsapp data with character-based embedding model is shown in Fig. 5. From the graph it can be seen that, the 'univ' tag holds maximum accuracy for all 3 social media text while the 'ne' tag holds minimum accuracy. When comparing 'hi' and 'en', the former shows better accuracy than the latter.

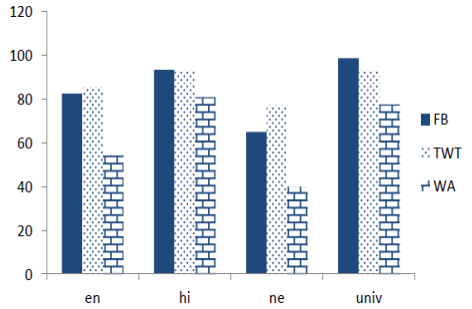


Fig. 5. Analysis of tag-wise F-scores obtained for Facebook (FB), Twitter (TWT) and Whatsapp (WA)

6 Conclusion and Future Work

The main objective of the system is to identify the language of the given data. Since code-mixed data is the popular trend among the users of social media platforms like Facebook, and Twitter, identifying language from them becomes the primary task of many NLP applications. For this work, a Hindi-English code-mixed dataset of Facebook, Twitter, and Whatsapp was used for developing the language identification system. Word-based and character-based 3-gram and 5-gram embedding vectors are used as features and a machine learning based SVM classifier is used for training and testing. The predicted labels obtained from the four models for each of the datasets were compared with the gold-standard dataset. The system was evaluated by performance metrics, Precision, Recall and F-measure. From the work, it was inferred that character-based 5-gram embedding system produced better results for

Facebook and Whatsapp code-mixed data while for Twitter dataset, the word-based approach was effective. It was analyzed that, word-based embedding system will be sufficient if the sufficient embedding data is available from that particular social media platform.

The work on Language Identification for Hindi-English code-mixed data can be extended by using more unsupervised data. In future, the work can be done for code-mixed datasets of other languages like Malayalam-English and Tamil-English with varying the n-gram size. Deep-Learning techniques like Recurrent Neural Networks (RNN) based approaches can be used along with the character embedding features.

References

1. **Alekseev, A. & Nikolenko, S. (2017).** Word embeddings for user profiling in online social networks. *Computación y Sistemas*, Vol. 21, No. 2.
2. **Bali, K., Jatin, S., Choudhury, M., & Vyas, Y. (2014).** "i am borrowing ya mixing?" An Analysis of English-Hindi Code Mixing in Facebook. *EMNLP 2014*, pp. 116.
3. **Banerjee, S., Chakma, K., Naskar, S., Das, A., Rosso, P., Bandyopadhyay, S., & Choudhury, M. (2016).** Overview of the Mixed Script Information Retrieval (MSIR) at FIRE-2016. *CEUR Workshop Proceedings*, Vol. 1737, pp. 94–99.
4. **Barman, U., Das, A., Wagner, J., & Foster, J. (2014).** Code mixing: A challenge for Language Identification in the Language of Social Media. *EMNLP 2014*, Vol. 13.
5. **Barman, U., Wagner, J., & Foster, J. (2016).** Part-of-speech tagging of code-mixed social media content: Pipeline, stacking and joint modelling. *EMNLP 2016*, pp. 30.
6. **Bhargava, R., Sharma, Y., & Sharma, S. (2016).** Sentiment Analysis for Mixed Script Indic Sentences. *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016*, pp. 524–529.
7. **Ethiraj, R., Shanmugam, S., Srinivasa, G., & Sinha, N. (2015).** NELIS - Named Entity and Language Identification System: Shared Task System Description. volume 1587, pp. 43–46.
8. **Jamatia, A. & Das, A. (2016).** Task Report: Tool Contest on POS Tagging for Code-mixed Indian Social Media (Facebook, Twitter, and Whatsapp) Text@ ICON 2016.". *Proceeding of ICON 2016*.
9. **Jhamtani, H., Bhogi, S. K., & Raychoudhury, V. (2014).** Word-level language identification in bi-lingual code-switched texts. pp. 348–357.
10. **Joachims & Thorsten (1999).** Svmlight: Support vector machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, Vol. 19, No. 4.
11. **King, L., Baucom, E., Gilmanov, T., Kübler, S., Whyatt, D., Maier, W., & Rodrigues, P. (2014).** The IUCL+ System: Word-Level Language Identification via Extended Markov Models. *EMNLP 2014*, pp. 102.
12. **Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013).** Distributed Representations of Words and Phrases and their Compositionality. pp. 3111–3119.
13. **Molina, G., Rey-Villamizar, N., Solorio, T., AlGhamdi, F., Ghoneim, M., Hawwari, A., & Diab, M. (2016).** Overview for the Second Shared Task on Language Identification in Code-Switched Data. *EMNLP 2016*, pp. 40.
14. **Quoc, L. & Tomas, M. (2014).** Distributed Representations of Sentences and Documents. *31st International Conference on Machine Learning, ICML 2014*, Vol. 4, pp. 2931–2939.
15. **R., W., Carbonell, J., Grosz, B., Lehnert, W., Marcus, M., R., P., & R, W. (1989).** White Paper on Natural Language Processing. Association for Computational Linguistics, pp. 481–493.
16. **Rahul Venkatesh Kumar, R. M., Anand Kumar, M., & Soman, K. P. (2015).** AmritaCEN-NLP @ FIRE 2015 Language Identification for Indian Languages in Social Media Text. volume 1587, pp. 26–28.
17. **Remmiya Devi, G., Veena, P. V., Anand Kumar, M., & Soman, K. P. (2016).** AMRITA-CEN@FIRE 2016: Code-mix Entity Extraction for Hindi-English and Tamil-English tweets. *CEUR Workshop Proceedings*, Vol. 1737, pp. 304–308.
18. **R.K., P. R. & Devi, S. (2016).** CMEE-IL: Code Mix Entity Extraction in Indian Languages from Social Media Text@FIRE 2016 - An Overview. volume 1737, pp. 289–295.
19. **Sequiera, R., Choudhury, Monojit, Gupta, P., Rosso, P., Kumar, S., Banerjee, S., Naskar, S.,**

Bandyopadhyay, S., Chittaranjan, G., Das, A., & Chakma, K. (2015). Overview of FIRE-2015 Shared Task on Mixed Script Information Retrieval. volume 1587, pp. 19–25.

20. **Sharma, S., Srinivas, P., & Balabantaray, R. (2016).** Emotion Detection using Online Machine Learning Method and TLBO on Mixed Script. volume 10, pp. 5.
21. **Srinidhi Skanda, V., Singh, S. ., Remmiya Devi, G., Veena, P. V., Anand Kumar, M., & Soman, K. P. (2016).** CEN@ Amrita FIRE 2016: Context based Character Embeddings for Entity Extraction in Code-Mixed Text. *CEUR Workshop Proceedings*, Vol. 1737, pp. 321–324.
22. **Vyas, Y., Gella, S., Sharma, J., Bali, K., & Choudhury, M. (2014).** POS tagging of English-Hindi Code-Mixed Social Media Content. pp. 974–979.
23. **Zubiaga, A., San Vicente, I., Gamallo, P., Campos, J. R. P., Loinaz, I. A., Aranberri, N., Ezeiza, A., & Fresno-Fernández, V. (2014).** Overview of tweetlid: Tweet Language Identification at SEPLN 2014. pp. 1–11.

Article received on 03/08/2016; accepted on 09/10/2016.
Corresponding author is P. V. Veena.