Named Entity Recognition for Hindi-English Code-Mixed Social Media Text

by

Vinay Singh, Deepanshu Vijay, Syed S. Akhtar, Manish Shrivastava

in

56th Annual Meeting of the Association for Computational Linguistics
(*NEWS 2018 (ACL-2018)*)

Melbourne, Australia

Report No: IIIT/TR/2018/-1

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2018

Named Entity Recognition for Hindi-English Code-Mixed Social Media Text

Vinay Singh, Deepanshu Vijay, Syed S. Akhtar, Manish Shrivastava
Language Technologies Research Centre (LTRC)
International Institute of Information Technology Hyderabad, Telangana, India
{vinay.singh, deepanshu.vijay, syed.akhtar}@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

Named Entity Recognition (NER) is a major task in the field of Natural Language Processing (NLP), and also is a subtask of Information Extraction. The challenge of NER for tweets lies in the insufficient information available in a tweet. There has been a significant amount of work done related to entity extraction, but only for resource-rich languages and domains such as the newswire. Entity extraction is, in general, a challenging task for such an informal text, and code-mixed text further complicates the process with it's unstructured and incomplete information. We propose experiments with different machine learning classification algorithms with word, character and lexical features. The algorithms we experimented with are Decision tree, Long Short-Term Memory (LSTM), and Conditional Random Field (CRF). In this paper, we present a corpus for NER in Hindi-English Code-Mixed along with extensive experiments on our machine learning models which achieved the best f1-score of 0.95 with both CRF and LSTM.

1 Introduction

Multilingual speakers often switch back and forth between languages when speaking or writing, mostly in informal settings. This language interchange involves complex grammar, and the terms "code-switching" and "code-mixing" are used to describe it Lipski. Code-mixing refers to the use of linguistic units from different languages in a single utterance or sentence, whereas code-switching refers to the co-occurrence of speech extracts belonging to two different grammatical systems Gumperz. As both phenomena are frequently observed on social media platforms in similar contexts, we use only the code-mixing scenario in this work.

Following are some instances from a Twitter corpus of Hindi-English code-mixed texts also transliterated in English.

T1 : *"Finally India away series jeetne mein successful ho hi gayi :D"*

Translation: *"Finally India got success in winning the away series :D"*

T2 : *"This is a big surprise that Rahul Gandhi congress ke naye president hain."*

Translation: *"This is a big surprise that Rahul Gandhi is the new president of Congress."*

However, before delving further into code-mixed data, it is important to first address the complications in social media data itself. First, the shortness of micro-blogs makes them hard to interpret. Consequently, ambiguity is a major problem since semantic annotation methods cannot easily make use of co-reference information. Second, micro-texts exhibit much more language variation, tend to be less grammatical than longer posts, contain unorthodox capitalization, and make frequent use of emoticons, abbreviations and hashtags, which can form an important part of the meaning. Most of the research has, however been focused on resource rich languages, such as English Sarkar, GermanTjong Kim Sang and De Meulder, French Azpeitia et al. and Spanish Zea et al.. However entity extraction and recognition from social media text for Indian languages Saha et al.; Ekbal and Bandyopadhyay; Malarkodi et al. and Code-Mixed text Gupta et al. have been

Proceedings of the Seventh Named Entities Workshop, pages 27–35

introduced a bit late. Chieu and Ng A shared task in FIRE-15 workshop[1] and explicitly NER task on Code-Mixed in FIRE 2016[2].

The structure of the paper is as follows. In Section 2, we review related research in the area of Named Entity Extraction on code-mixed social media texts. In Section 3, we describe the corpus creation and annotation scheme. In Section 4, we discuss the data statistics. In Section 5, we summarize our classification systems which includes the pre-processing steps and construction of feature vector. In Section 6, we present the results of experiments conducted using various character, word level and lexical features using different machine learning models. In the last section, we conclude our paper, followed by future work and the references.

2 Background and Related work

Bali et al. performed analysis of data from Facebook posts generated by English-Hindi bilingual users. Analysis depicted that significant amount of code-mixing was present in the posts. Vyas et al. formalized the problem, created a POS tag annotated Hindi-English code-mixed corpus and reported the challenges and problems in the Hindi-English code-mixed text. They also performed experiments on language identification, transliteration, normalization and POS tagging of the Dataset. Sharma et al. addressed the problem of shallow parsing of Hindi-English code-mixed social media text and developed a system for Hindi-English code-mixed text that can identify the language of the words, normalize them to their standard forms, assign them their POS tag and segment into chunks. Barman et al. addressed the problem of language identification on Bengali-Hindi-English Facebook comments.

In Named Entity Recognition there has been significant research done so far in English and other resource rich languages Morwal et al.; Srihari et al., but same cannot be said for code-mixed text due to lack of structured resources in this domain. Bhargava et al. proposed a hybrid model for NER on Hindi-English and Tamil-English code-mixed Dataset. Bhat et al. proposed a neural network architecture for NER on Hindi-English code-mixed Dataset. Code-mixing got attention in FIRE-2016 with the introduction of tasks on Code-Mixed resources. Now code-mixing has found its application in different areas such as Query Labeling Bhargava et al., Sentiment Analysis Bhargava et al., Question Classification etc.

3 Corpus and Annotation

The corpus that we created for Hindi-English code-mixed tweets contains tweets from last 8 years on topics like politics, social events, sports, etc. from the Indian subcontinent perspective. The tweets were scrapped from Twitter using the Twitter Python API[3] which uses the advanced search option of twitter. The mining of the tweets are done using some specific hash-tags and are mined in a json format which consist all the information regarding the tweets like time-stamps, URL, text, user, replies, etc. Extensive pre-processing (Section 5.4) was carried out to remove the noisy and non-useful tweets. Noisy tweets are the ones which comprise only of hashtags or urls. Also, tweets in which languages other than Hindi or English are used were also considered as noisy and hence removed from the corpus . Furthermore, all the tweets which were either in only English or used Devanagari script text are removed too, keeping only the code-mixed tweets. Further cleaning of data is done in the annotation phase.

3.1 Annotation: Named Entity Tagging

We label the tags with the present three Named Entity tags 'Person', 'Organization', 'Location', which using the BIO standard become six NE tags (B-Tag referring to beginning of a named entity and I-Tag refers to the intermediate of the entity) along with the 'Other' tag to all those which don't lie in any of the six NE tags.

'Per' tag refers to the 'Person' entity which is the name of a Person, twitter handles and common nick names of people. The 'B-Per' states the beginning and 'I-Per' for the name of the Person, if the Person name or reference is split into multiple continuous. In the example **T3** we show the instance of 'Per' tag in a tweet chosen from our corpus.

T3: *"modi/B-Per ji/I-Per na/Other kya/Other de/Other rakha/Other hai/Other media/B-Org ko/Other ?/Other"*

Translation: "What has modi ji given to media?"

[1] http://fire.irsi.res.in/fire/2015/home

[2] http://www.au-kbc.org/nlp/CMEE-FIRE2016/

[3] https://pypi.python.org/pypi/twitterscraper/0.2.7

Tag	Count of Tokens
B-Loc	762
B-Org	1,432
B-Per	2,138
I-Loc	31
I-Org	90
I-Per	554
Total NE tokens	5,007

Table 1: Tags and their Count in Corpus

'Loc' tag refers to the location named entity which is assigned to the names of places for eg. 'Kashmir', '#Delhi', 'Hindustan', etc. The 'B-Loc' states the beginning and 'I-Loc' intermediate of name of the location, if the location name is split into multiple tokens. Example **T4** shows the instance of 'Loc' tag.

T4 : *"jis/Other ki/Other asar/Other saudi/B-Loc arab/I-Loc mein/Other bhi/Other dikhai/Other de/Other raha/Other hai/Other corruption/Other ke/Other khilaf/Other"*
Translation: "The effect of which is visible in saudi arab against corruption"

'Org' tag refers to social, political groups like Dalit, Bhartiya, Bhartiya Jnata Party (BJP), Hindus, Muslims, social media organizations like facebook, twitter, whatsapp, etc. and also govt. institutions like Reserve bank of India (RBI), banks, Swiss banks, etc. 'B-Org' states the beginning and 'I-Org' intermediate of name of the organization, if the organizations' name is split into multiple tokens. Example **T5** shows instance of 'Org' tag in the tweet.

T5: *"saare/Other black/Other money/Other to/Other swiss/B-Org bank/I-Org mein/Other the/Other"*
Translation: "all of the black money was in the swiss bank"

With these six NE tags and the seventh 7th tag as "Other" we annotated 3,638 tweets which meant tagging 68,506 tokens. The annotated Dataset with the classification system is made available online.[4] The distribution of the tags in the Dataset is shown in Table 1.

[4]https://github.com/SilentFlame/Named-Entity-Recognition

	Cohen Kappa
B-Loc	0.98
B-Org	0.96
B-Per	0.94
I-Loc	0.98
I-Org	0.91
I-Per	0.93

Table 2: Inter Annotator Agreement.

3.2 Inter Annotator Agreement

Annotation of the Dataset for NE tags in the tweets was carried out by two human annotators having linguistic background and proficiency in both Hindi and English. In order to validate the quality of annotation, we calculated the inter annotator agreement (IAA) between the two annotation sets of 3,638 code-mixed tweets having 68,506 tokens using Cohen's Kappa coefficient Hallgren. Table 2 shows the results of agreement analysis. We find that the agreement is significantly high. Furthermore, the agreement of 'I-Per' and 'I-Org' annotation is relatively lower than that of 'I-Loc', this is because, the presence of uncommon/confusing names of Organization as well as Person with unclear context.

4 Data statistics

Using the twitter API we retrieved 1,10,231 tweets. After manually filtering as described in Section 3, we are left with 3,638 code-mixed tweets. This number is close to the size of Dataset provided by FIRE 2016 which introduced the NER task for code-mixed text in 2015 with it's one shared task on Entity recognition on code-mixed data. Table 1 shows the distribution of different tags in the corpus. We use the standard CONLL tags (Loc, Org, Per, Other) for tagging in the annotation stage. The Named Entity (NE) Tag Person ("Per"), Organization ("Org") and Location ("Loc") are the ones we used to tag our corpus tokens. The 'Person' tag comprises names of famous people, politicians, actresses, sports personalities, news reporters and social media celebrities and their twitter handles and nick names if used frequently as known to the annotator (like "Pappu for Mr. Rahul Gandhi"). 'Organizations' comprises names of social or political organizations as well as major groups present in India, eg. Bhartiya Janta Party (BJP), Hindu, Muslim, twit-

Tag	Precision	Recall	F1-score
B-Loc	0.47	0.50	0.49
B-Org	0.54	0.59	0.56
B-Per	0.65	0.60	0.63
Other	0.97	0.97	0.97
I-Loc	0.27	0.30	0.29
I-Org	0.23	0.22	0.22
I-Per	0.43	0.38	0.40
avg / total	0.94	0.94	0.94

Table 3: Decision Tree Model with 'max-depth=32'

ter, etc. The Tag 'Location' comprises names of cities, towns, states and countries of the world. Major of the location entities are present for Indian subcontinent places in the corpus. The ones which does not lie in any of the mentioned tags are assigned 'Other' tag.

5 System Architecture

In this section we'll explain working of different machine learning algorithms we used for experiments on our annotated dataset.

5.1 Decision Tree

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too. Szarvas et al. takes a multilingual named entity recognition system using boosting and C4.5 decision tree learning algorithm. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label. In decision trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node. The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Handling this is know the attributes selection. We have different attributes selection measure to identify the attribute which can be considered as the root note at each level. The popular attribute selection measures:

- Information gain
- Gini index

Information gain: Using information gain as a criterion, we try to estimate the information contained by each attribute. By calculating entropy measure of each attribute we can calculate their information gain. Information Gain calculates the expected reduction in entropy due to sorting on the attribute. Information gain can be calculated as:

$$H(X) = E_X[I(X)] = -\sum_{x \in X} p(x) log(p(x))$$

Where $p(x)$ is the probability of a class for the feature we are calculating information gain. The node/feature with lowest entropy is chosen as root and process is repeated for other level feature selection.

Gini index: It refers to a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower gini index should be preferred. It is calculated as:

$$Gini - index = 1 - \sum_j p_j^2$$

Where p_j is the probability of a class for a given feature we are calculating gini index for.

5.2 Conditional Random Field (CRF)

For sequence labeling (or general structured prediction) tasks, it is beneficial to consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence. For example, in POS tagging an adjective is more likely to be followed by a noun than a verb, and in NER with standard BIO2 annotation (Tjong Kim Sang and Veenstra, 1999) I-ORG cannot follow I-PER. Therefore, we model label sequence jointly using a conditional random field (CRF) (Lafferty et al., 2001), instead of decoding each label independently. Since here we are focusing on sentence level and not individual positions hence it is generally known that CRF can produce higher tagging accuracy.

Say we are given a sequence of inputs we denote by X where $X = (x_1, x_2, x_3, \ldots, x_m)$ which are nothing but the words of the sentence and $S = (s_1, s_2, s_3, \ldots, s_m)$ as the sequence of output states, i.e the named entity tags. In conditional random field we model the conditional probability as

$$p(s_1, s_2, \ldots, s_m | x_1, x_2, \ldots, x_m)$$

Tag	Precision	Recall	F1-score
B-Loc	0.76	0.57	0.65
B-Org	0.67	0.33	0.44
B-Per	0.82	0.56	0.67
I-Loc	0.70	0.23	0.34
I-Org	0.68	0.27	0.39
I-Per	0.75	0.43	0.55
Other	0.96	0.99	0.98
avg / total	0.95	0.95	0.95

Table 4: CRF Model with 'c1=0.1' and 'c2=0.1' and 'L-BFGS' algorithm

We do this by defining a feature map

$$\Phi(x_1, x_2, \ldots, x_m, s_1, s_2, \ldots, s_m) \in \Re^d$$

that maps the entire input sequence X paired with an entire state sequence S to some d-dimensional feature vector. Then we can model the probability as a log-linear model with parameter vector $w \in \Re^d$

$$p(s|x; w) = \frac{exp(w.\Phi(x, s))}{\sum_{s'} exp(w.\Phi(x, s'))}$$

where s' ranges over all possible input sequences.

For the estimation of w, we assume that we have a set of n labelled examples $(x^i, s^i)_{i=1}^n$. Now we define regularized log likelihood function L as

$$L(w) = \sum_{i=1}^{n} log(p(s^i|x^i; w)) - \frac{\lambda_2}{2}||w||_2^2 - \lambda_1||w||_1$$

The terms $\frac{\lambda_2}{2}||w||_2^2$ and $\lambda_1||w||_1$ force the parameter vector to be small in the respective norm. This penalizes the model complexity and is known as regularization. The parameters λ_2 and λ_1 allows to enforce more or less regularization. The parameter vector w^* is then estimated as

$$w^* = argmax_{w \in \Re^d} L(w)$$

If we estimated the vector w^*, we can find the most likely tag for a sentence s^* for a given sentence sequence x by

$$s^* = argmax_s p(s|x; w^*)$$

5.3 LSTMs

Recurrent neural networks (RNN) are a family of neural networks that operate on sequential data. They take an input sequence of vectors $(x_1, x_2, \ldots, x_n)$ ad return another sequence $(h_1, h_2, \ldots, h_n)$ that represents some information about the sequence at every step of the input. In theory RNNs can learn long dependencies but in practice they fail to do so and tend to be biased towards the most recent input in the sequence.Bengio et al. Long Short Term Memory networks usually just called "LSTMs" are a special kind of RNN, capable of learning long-term dependencies. Here with our data where tweets are not very long in the size LSTMs can provide us a better result as keeping previous contexts is one of the specialty of LSTM networks. LSTM networks were first introduced by Hochreiter and Schmidhuber and then were refined and popularized by many other authors. They work well with large variety of problems specially the one consisting of sequence and are now widely used. They do so using several gates that control the proportion of the input to give to the memory cell, and the proportion from the previous state to forget.

5.4 Pre-processing

This step is done to make the data uniform which will be beneficial for our system. The preprocessing step consist of

- Removing noisy tweets
- Removing links from tweets
- Tokenization
- Separating words which appear continuous (i.e Modi.ji.Ke.Liye as 'Modi ji Ke Liye')
- Converting to lowercase
- Token encoding (mapping of tokens to their Tags)

5.5 Features

The feature set consists of word, character and lexical level information like char N-Grams of Gram size 2 and 3 for suffixes, patterns for punctuation, emoticons, numbers, numbers inside strings, social media specific characters like '#', '@' and also previous tag information, and the same all

Tag	Precision	Recall	F1-score
B-Loc	0.71	0.59	0.64
B-Org	0.62	0.37	0.47
B-Per	0.78	0.57	0.66
I-Loc	0.57	0.26	0.36
I-Org	0.60	0.26	0.36
I-Per	0.70	0.42	0.52
Other	0.97	0.99	0.98
avg / total	0.95	0.95	0.95

Table 5: CRF Model with 'Avg. Perceptron' Algorithm

features of the previous and next tokens are used as context features.

in this paper we have used the following feature vectors for training of our supervised model.

1. **Character N-Grams:** Character N-Grams are language independent Majumder et al. and have proven to be very efficient for classifying text. These are also useful in situations when the text suffers from errors such as misspellings Cavnar et al.; Huffman; Lodhi et al.. Group of characters can help in capturing semantic meaning, especially in the code-mixed language where there is an informal use of words, which vary significantly from the standard Hindi and English words. We use character N-Grams as one of the features, where n is 2 and 3.

2. **Word N-Grams:** Bag of word features have been widely used for NER tasks in languages other than English Jahangir et al.. Thus we use word N-Grams, where we used the previous and the next word as a feature vector to train our model. These are also called contextual features.

3. **Capitalization:** It is a very general trend of writing any language in Roman script that people write the names of person, place or a things starting with capital letter von Däniken and Cieliebak or for aggression on someone/something use the capitalization of the entire entity name. This will make for two binary feature vectors one for starting with capital and other for the entire word capitalized.

4. **Mentions and Hashtags:** It is observed that in twitter users generally tend to address other people or organization with their user names which starts with '@' and to emphasize on something or to make something notable they use '#' before that word. Hence presence of these two gives a good probability for the word being a named entity.

5. **Numbers in String:** In social media content, users often express legitimate vocabulary words in alphanumeric form for saving typing effort, to shorten message length, or to express their style. Examples include abbreviated words like gr8' ('great'), 'b4' ('before'), etc. We observed by analyzing the corpus that alphanumeric words generally are not NEs. Therefore, this feature serves as a good indicator to recognize negative examples.

6. **Previous Word Tag:** As mentioned in word N-Gram feature the context helps in deciding the tag for the current word, hence the previous tag will help in learning the tag of current word and all the I-Tags always come after the B-Tags.

7. **Common Symbols:** It is observed that currency symbols as well as brackets like '(', '[', etc. symbols in general are followed by numbers or some mention not of importance. Hence are a good indicator for the words following or before to not being an NE.

6 Experiments

This section present the experiments we performed with different combinations of features and systems.

6.1 Feature and parameter experiments

In order to determine the effect of each feature and parameter of different models we performed several experiments with some set of feature vectors at a time and all at a time simultaneously changing the values of the parameters of our models like criterion ('Information gain', 'gini'), maximum depth of the tree for Decision tree model, optimization algorithms, loss functions in LSTM, regularization parameters and algorithms of optimization for CRF like 'L-BFGS' [5], 'L2 regularization' [6], 'Avg. Perceptron', etc. In all the models

[5]https://en.wikipedia.org/wiki/Limited-memory_BFGS

[6]https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

we mentioned above we validated our classification models with 5-fold cross-validation.

Tables 3 shows the experiment result on Decision tree model with maximum depth = 32 which we arrived at after fine empirical tuning. Tables 4 and 5 provides the experiments on CRF model. The c1 and c2 parameters for CRF model refers to L1 regularization and L2 regularization. These two regularization parameters are used to restrict our estimation of w^* as mentioned in Section 5.1. When experimented with algorithm of optimization as 'L2 regularization' or 'Average Perceptron' there is not any significant change in the results of our observation both in the per class statistics as well as the overall. We arrived at these values of $c1$ and $c2$ after fine empirical tuning. Table 4 and 5 refers to this observation.

Next we move to our experiments with **LSTM** model. Here we experimented with the optimizer, activation function along with the number of units as well as number of epochs. The best result that we came through was with using 'softmax' as activation function, 'adam' as optimizer and 'sparse categorical cross-entropy' for our loss function. Table 7 shows the statistics of running LSTM on our Dataset with 5-fold cross-validation having validation-split of 0.2 with our char, word and lexical feature set of our tokens. Table 6 shows one prediction instance of our **LSTM** model.

6.2 Results and Discussion

From the above results we can say that our system learns from the structure of the text the specific NE types like from Table 6 we can see that our system is understanding well as it tagged most tokens correctly.

We also observe that our system is getting confused in the 'Org' names that resemble to name of locations like 'America' is tagged as 'B-Org' this is because our system has seen many 'American' tokens tagged as 'B-Org' hence this confusion.

From the example in the Table 11 we can see that our system learns to tag tokens that starts with '#' as beginning of a NE but majority of the time tags it as 'B-Per' which is a problem. Our model needs to learn more generic details about these specific characters.

For 'Loc' and 'Other' tags our system works good, giving accurate predictions. The presence of confusing names of 'Location', 'Organization' and that of 'Person' in our corpus makes it difficult for our machine learning models to learn the proper tags of these names. For eg. 'Hindustan' is labeled as 'B-Loc' in our annotation and 'Hindustani' is as 'B-Org' as the former is one of the 5 names of the country India and the later represent the citizens which makes it a group representation which we used for Organization during our annotation. Hence lexically similar words with different tags makes the learning phase of our model difficult and hence some incorrect tagging of the tokens as we can see in Table 6.

Word	Truth	Predicted
#ModiMeetTrump	Other	Other
kya	Other	Other
#Modi	B-Per	B-Per
gi	I-Per	Other
#America	B-Loc	B-Per
main	Other	Other
#Trump	B-Per	B-Per
ke	Other	Other
shaath	Other	Other
mil	Other	Other
kar	Other	Other
#Pakistan	B-Loc	B-Org
ka	Other	Other
koi	Other	Other
rasta	Other	Other
nikalenge	Other	Other
kya	Other	Other
hoga	Other	Other
#EidMubarak	Other	Other
#India	B-Loc	B-Per
#India	B-Loc	B-Org

Table 6: An Example Prediction of our LSTM Model

Tag	Precision	Recall	F1-score
B-Loc	0.91	0.89	0.86
B-Org	0.80	0.57	0.63
B-Per	0.88	0.82	0.87
I-Loc	0.93	0.47	0.60
I-Org	0.91	0.89	0.89
I-Per	0.82	0.76	0.78
Other	0.87	0.83	0.84
avg / total	0.96	0.93	0.95

Table 7: LSTM model with "optimizer='adam'"

7 Conclusion and future work

In this paper, we present a freely available corpus of Hindi-English code-mixed text, consisting of tweet ids and the corresponding annotations. We also present NER systems on this Dataset with experimental analysis and results. This paper first explains about the reason of selection of some features specific to this task at the same time experimenting our results on different machine learning classification models. Decsion Tree, CRF and LSTM models worked with a best individual f1-score of 0.94, 0.95 and 0.95 which is good looking at the fact that there haven't been much research in this domain.

To make the predictions and models' result more significant the size of the corpus needed to be expanded. Our corpus has just 3,638 tweets, but due to unavailability of Hindi-English Code-Mixed Dataset it is difficult to get large corpus for our system.

Our contribution in this paper includes the following points:

1. Annotated corpus for Hindi-English Code-Mixed, kind of which are not available anywhere on the internet.

2. Introduction an addressing of Hindi-English Code-Mixed data as a research problem.

3. Proposal of suitable features targeted towards this task.

4. Different models which deals with sequential tagging and multi-class classification.

5. Developing machine learning models on our annotated corpus for the NE task.

As a part of future work, the corpus can be annotated with part-of-speech tags at word level which may yield better results. Moreover, the Dataset contains very limited tweets having NE tokens. Thus it can be extended to include more tweets more of these specific NE tokens as well as introducing a more number of tags on the existing corpus. The annotations and experiments described in this paper can also be carried out for code-mixed texts containing more than two languages from multilingual societies in future.

References

Andoni Azpeitia, Montse Cuadros, Seán Gaines, and German Rigau. 2014. Nerc-fr: supervised named entity recognition for french. In *International Conference on Text, Speech, and Dialogue*, pages 158–165. Springer.

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. " i am borrowing ya mixing?" an analysis of english-hindi code mixing in facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the first workshop on computational approaches to code switching*, pages 13–23.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Rupal Bhargava, Yashvardhan Sharma, and Shubham Sharma. 2016a. Sentiment analysis for mixed script indic sentences. In *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*, pages 524–529. IEEE.

Rupal Bhargava, Yashvardhan Sharma, Shubham Sharma, and Abhinav Baid. 2015. Query labelling for indic languages using a hybrid approach. In *FIRE Workshops*, pages 40–42.

Rupal Bhargava, Bapiraju Vamsi, and Yashvardhan Sharma. 2016b. Named entity recognition for code mixing in indian languages using hybrid approach. *Facilities*, 23:10.

Irshad Ahmad Bhat, Manish Shrivastava, and Riyaz Ahmad Bhat. 2016. Code mixed entity extraction in indian languages using neural networks. In *FIRE (Working Notes)*, pages 296–297.

William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. *Ann arbor mi*, 48113(2):161–175.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Pius von Däniken and Mark Cieliebak. 2017. Transfer learning and sentence level features for named entity recognition on tweets. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 166–171.

Asif Ekbal and Sivaji Bandyopadhyay. 2008. Bengali named entity recognition using support vector machine. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*.

John J Gumperz. 1982. *Discourse strategies*, volume 1. Cambridge University Press.

Deepak Gupta, Shubham Tripathi, Asif Ekbal, and Pushpak Bhattacharyya. 2016. A hybrid approach for entity extraction in code-mixed social media data. *MONEY*, 25:66.

Kevin A Hallgren. 2012. Computing inter-rater reliability for observational data: an overview and tutorial. *Tutorials in quantitative methods for psychology*, 8(1):23.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Stephen Huffman. 1995. Acquaintance: Language-independent document categorization by n-grams. Technical report, DEPARTMENT OF DEFENSE FORT GEORGE G MEADE MD.

Faryal Jahangir, Waqas Anwar, Usama Ijaz Bajwa, and Xuan Wang. 2012. N-gram and gazetteer list based named entity recognition for urdu: A scarce resourced language. In *24th International Conference on Computational Linguistics*, page 95.

John Lipski. 1978. Code-switching and the problem of bilingual competence. *Aspects of bilingualism*, 250:264.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

P Majumder, M Mitra, and BB Chaudhuri. 2002. N-gram: a language independent approach to ir and nlp. In *International conference on universal knowledge and language*.

CS Malarkodi, RK Pattabhi, and Lalitha Devi Sobha. 2012. Tamil ner–coping with real time challenges. In *24th International Conference on Computational Linguistics*, page 23.

Sudha Morwal, Nusrat Jahan, and Deepti Chopra. 2012. Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing (IJNLC)*, 1(4):15–23.

Sujan Kumar Saha, Sanjay Chatterji, Sandipan Dandapat, Sudeshna Sarkar, and Pabitra Mitra. 2008. A hybrid approach for named entity recognition in indian languages. In *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian languages*, pages 17–24.

Kamal Sarkar. 2015. A hidden markov model based system for entity extraction from social media english text at fire 2015. *arXiv preprint arXiv:1512.03950*.

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Srivastava, Radhika Mamidi, and Dipti M Sharma. 2016. Shallow parsing pipeline for hindi-english code-mixed social media text. *arXiv preprint arXiv:1604.03136*.

Rohini Srihari, Cheng Niu, and Wei Li. 2000. A hybrid approach for named entity and sub-type tagging. In *Proceedings of the sixth conference on Applied natural language processing*, pages 247–254. Association for Computational Linguistics.

György Szarvas, Richárd Farkas, and András Kocsor. 2006. A multilingual named entity recognition system using boosting and c4. 5 decision tree learning algorithms. In *International Conference on Discovery Science*, pages 267–278. Springer.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979.

Jenny Linet Copara Zea, Jose Eduardo Ochoa Luna, Camilo Thorne, and Goran Glavaš. 2016. Spanish ner with word representations and conditional random fields. In *Proceedings of the Sixth Named Entity Workshop*, pages 34–40.