A Deep Neural Network based Approach for Entity Extraction in Code-Mixed Indian Social Media Text

Deepak Gupta, Asif Ekbal, Pushpak Bhattacharyya
Indian Institute of Technology Patna, India
{deepak.pcs16, asif, pb}@iitp.ac.in

Abstract

The rise in accessibility of web to the mass has led to a spurt in the use of social media making it convenient and powerful way to express and exchange information in their own language(s). India, being enormously diversified country have more than 168 millions users on social media. This diversity is also reflected in their scripts where a majority of users often switch between their native languages to be more expressive. These linguistic variations make automatic entity extraction both a necessary and a challenging problem. In this paper, we report our work for entity extraction in a code-mixed environment. Our proposed approach is based on the popular deep neural network based Gated Recurrent Unit (GRU) archirecture that automatically discovers the higher level features from the text. We do not make use of any handcrafted features or rules, and therefore our proposed model is quite generic in nature. Our experiments on two benchmark datasets of English-Hindi and English-Tamil language pairs show the F-scores of 66.04% and 53.85%, respectively.

Keywords: Code-Mixing, Gated Recurrent Unit (GRU), Entity Extraction, Indian Language

1. Introduction

The phenomenal growth in user-generated contents on popular social media platforms like Twitter and Facebook has established a new perspective in Natural Language Processing (NLP). It incurs several challenges ranging from the incorrect spellings to the highly creative words ("f9" for "fine"), word lengthening ("gooooood" for "good"), phonetic typing, abbreviation ("IDK" for "I don't know") and so on. In recent times there has been a rising interest in building several applications on social media contents such as mood identification, opinion mining, etc. However, a majority of the existing research is restricted towards resource-rich languages such as English, some of the European languages and few Asian languages. The growing contents of underprivileged languages in the Web has necessitated investigating automated techniques that could build solutions involving these. Code-mixing refers to the mixing of more than one language that often makes the task more complex for building automated techniques. Non-native English speakers do not always use unicode encoding scheme to write in social media. Instead, they often use the most frequently used transliterated forms with the English words or phrases. These multilingual speakers often mix multiple languages in addition to anglicisms to be more expressive. There are 22 official languages mentioned in the Indian constitution. The report also reveals that 30 languages are spoken by more than one million native speakers, and 122 are spoken by more than $10,000$ people. Language diversity and dialect changes instigate frequent code-mixing in Indian languages. Hence, Indians are multilingual by adaptation and necessity, and frequently change and mix languages while writing in social media platforms. These pose additional difficulties in building automated tools for social media analytics. Named Entity Recognition (NER) is a primary task in information extraction. It aims at identifying the names of entities and classifying them into some predefined categories such as people, location, organization and product. This task can also be thought of as a two-step process, *viz.* entity detection and entity classification (Grishman and Sundheim, 1996). There are a significantly large body of works existing in Indian languages, but these are mostly related to the domains such as newswire. Nowadays, information extraction in micro-blogs has become an active research topic (Cano Basave et al., 2013), following the early experiments which showed this genre to be extremely challenging for the state-of-the-art algorithms (Derczynski et al., 2015; Bontcheva et al., 2014). The shortness of micro-blogs makes them hard to interpret. The social media text normally carries less discourse information per document, and threaded structure is fragmented across multiple documents. Apart from this, short text (tweets) also exhibits more language variations, tend to be less grammatical than the longer posts, contains unorthodox capitalization, and makes use of frequent abbreviations, hashtags and emoticons. These information also participates to decide the intention or meaning of a short text. To combat these problems, researcher has focused on microblog-specific information extraction algorithms, e.g. NER on Twitter data using Conditional Random Field (CRF) (Ritter et al., 2011) or hybrid methods (Van Erp et al., 2013). Particular attention is given to micro-text normalization (Han and Baldwin, 2011), as a way of removing some of the linguistic noise prior to Part-of-Speech (PoS) tagging and NER. Several Machine Learning (ML) techniques have already been applied for the NER tasks such as Hidden Markov Model(HMM) (Bikel et al., 1997), Maximum Entropy (Borthwick, 1999; Kumar and Bhattacharyya, 2006), Support vector Machine (SVM) (Isozaki and Kazawa, 2002), Conditional Random Field (CRF) (Li and McCallum, 2003) etc. Few systems for entity extraction in social media texts involving Indian language have been reported in FIRE-2015 workshop (Rao et al.,). In recent times, a benchmark setup for entity extraction involving Indian languages was introduced in FIRE-2016 shared

task (Rao and Devi, 2016). Some of the challenges for entity extraction in a code-mixed environment are as follows:

- The dataset contains tweets utterance in code mixed as well as in mono-lingual text.
- Introduction of a diverse set of entities, not limited to the only traditional set of entity types such as person, location, organization etc. There are 22 different types of entities that need to be extracted from text.
- Various resources and/or tools such as sentence splitter, tokenizer, PoS tagger, chunker etc. are not readily available in the required measure. In our current work, we develop a system for entity extraction using a deep Gated Recurrent Unit (GRU) architecture. We evaluate the proposed system for two language pairs, namely English-Hindi (EN-HI) and English-Tamil (EN-TA).

2. Methodology

In this section at first we define the code mixed entity extraction problem, and then present our proposed deep neural network based model.

2.1. Problem Definition

The problem of code-mixed entity extraction comprises of two sub-problems, *viz.* entity extraction and entity classification. Mathematically, the problem of code-mixed entity extraction can be described as follows: Let be S a code-mixed sentence having n tokens $t_1, t_2 \dots t_n$. A set of pre-defined entity category $C = \{C_1, C_2, \dots C_k\}$. The goal is to extract set of code mixed entities CME=$\{x \in S \mid x$ is a token or pharse$\}$. Thereafter, each of the code mixed entities $ce \in$ CME has to be classified into one of the pre-defined categories denoted by C. More formally the input of the system is a tweet associated with a user id and tweet id. The output of the system is all the entities with their corresponding categories present in the tweet. Some of the tweets from the dataset are shown in Table 1.

Langauage Pair	Sample Tweet
English-Hindi	awesome track ☺ listening to Toota Jo Kabhi Tara - A Flying Jatt - Atif Aslam; Sumedha Karmahe
	RiftWood Productions presents to you the season finale of Le' Bill & Giddy,La Muje'r
English-Tamil	@adhu_idu111 @7hillsm @beingsalmankhan @memephobhia @rajinifc salman sultan did 580cr. and for salman 200 crs cakewlk;
	IruMugan will be a Class + Mass movie like Thani Oruvan. The biggest plus is the screenplay - Thambi Ramaiyah..

Table 1: Some of the tweets from the dataset. The colored words represent the entities.

2.2. Approach for Entity Extraction

In order to recognize the entity from a code-mixed sentence, it is necessary to have a model which can process forward and backward tokens together in order to decide the type of the current token. Inspired by these works (Dernoncourt et al., 2017; Yao et al., 2013), we adapted a bi-directional GRU-based deep learning model for code mixed entity extraction which can use their memory to process arbitrary sequences of inputs in both the directions. Our proposed model is a layered architecture that follows the following steps:

1. Character and word level embedding layer
2. Input sequence processing layer
3. Output sequence optimization layer

We will begin by the first describing the bi-directional gated recurrent unit (GRU). Thereafter, each of the component layers is described.

2.2.1. Bi-directional gated recurrent units

Gated recurrent unit (GRU) was proposed by (Cho et al., 2014) to make each recurrent unit to adaptively capture dependencies at different time scales. GRU is very similar to the Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997). Similar to LSTM unit, GRU has gating units that modulate the flow of information inside the unit, however, without having separate memory cells. GRU has two neural gates, *update* and *reset*, that control the flow of information, as in LSTM. The update gate controls how much of past state should matter now. Similarly, reset gate allows the model to drop information that is irrelevant in future. Specifically, a GRU network successively reads the input token t_i, as well as the previous hidden state h_{i-1}, and generate the new memory content c_i and hidden unit h_i.

$$\begin{aligned} \mathbf{z}_i &= \sigma(\mathbf{W}_z t_i + \mathbf{V}_z \mathbf{h}_{i-1} + \mathbf{b}_z) \\ \mathbf{r}_i &= \sigma(\mathbf{W}_r t_i + \mathbf{V}_r \mathbf{h}_{i-1} + \mathbf{b}_r) \\ \mathbf{c}_i &= tanh(\mathbf{W} t_i + \mathbf{V}(\mathbf{r}_i \odot \mathbf{h}_{i-1}) + \mathbf{b}) \\ \mathbf{h_i} &= z_i \odot \mathbf{h}_{i-1} + (1 - \mathbf{z}_i) \odot \mathbf{c}_i \end{aligned} \tag{1}$$

where $\mathbf{z}$ and $\mathbf{r}$ are the input and reset gates, respectively. Here σ, $\odot$ represents the sigmoid function and element-wise multiplication, respectively. The $\{W_z, W_r, W, V_z, V_r, V\}$ and $\{b_z, b_r, b\}$ are the weight-matrices and bias vectors, respectively. Eq 1 can compute the forward an backward hidden state at time i by moving the forward or backward along the input sequence. We compute the forward hidden state $\overrightarrow{h_t}$ and backward hidden state $\overleftarrow{h_t}$. The final hidden state z_i computed by augmenting both the hidden states, i.e. $z_i = [\overrightarrow{h_i} \oplus \overleftarrow{h_i}]$ where $\oplus$ is the concatenation operator.

2.2.2. Character and word level embedding layer

The input to this layer is the word t at time i from the code mixed sentence S. It produces output as the word vector. The representation of the word in higher dimensional space is known as word embedding which can be obtained from the pre-trained embedding model trained on the huge unlabeled corpus using several methods, such as Glove (Pennington et al., 2014), word2vec (Mikolov et al., 2013) or fastText (Bojanowski et al., 2016). The word embedding is

Entity Types	No. of Entities (EN-HI / EN-TA)	Entity Type	No. of Entities (EN-HI / EN-TA)
COUNT	132 / 94	FACILITIES	10 / 23
PLANTS	1 / 3	PERSON	712 / 661
PERIOD	44 / 53	MATERIALS	24 / 28
LOCOMOTIVE	13 / 5	LOCATION	194 / 188
ENTERTAINMENT	810 / 260	YEAR	143 / 54
MONEY	25 / 66	DATE	33 / 14
TIME	22 / 18	ORGANIZATION	109 / 68
LIVTHINGS	7 / 16	QUANTITY	2 / 0
DISEASE	7 / 5	DAY	67 / 15
ARTIFACT	25 / 18	SDAY	23 / 6
MONTH	10 / 25	DISTANCE	0 / 4
Total no. of entity (EN-HI /EN-TA)		**2413 / 1624**	

Table 2: Statistics of the dataset for English-Hindi and English-Tamil language pairs

obtained through the Glove pre-trained word embedding[1]. It has the desirable properties in capturing the syntactic and semantic representation of words. This property helps the model to identify the similar entities. However, it still suffers from several issues such as, they cannot handle out-of-vocabulary words, misspelled words and variations in noun or verb phrase. When comes to the code mixing text this issue even becomes more crucial to resolve. One way to resolve this issue is by utilizing character based word embedding which incorporates each individual character of a token to generate its vector representation. This approach, therefore, allows the model to learn lexical patterns (e.g. suffix or prefix) which eventually can help in capturing out-of-vocabulary words and some other information which are difficult to capture through word embedding.

Now we define the way to compute the character embedding. Let $c_1^i, \dots c_l^i$ be the character sequence of word t_i having length l. We initialize the character representation $V(c_k^i)$ randomly of each character k in token t_i. By this way, we generate the sequence of character representations $c_{1:k}$. The generated sequence is provided as an input to the Bi-GRU model, which outputs the character word embedding $C(i)$. Finally, the output of word embedding for the i_{th} word is the concatenation of word embedding $W(i)$ and the character word embedding $C(i)$. In short, the layer takes the sequence of words $t_{1:n}$ as input and produces a sequence of word embeddings $e_{1:n}$ as output.

2.2.3. Input sequence processing layer

This layer takes the previous layer's output i.e. the sequence of word embedding $e_{1:n}$ as the input and passes this sequence to a Bi-GRU unit. Bi-GRU unit as discussed in Section-2.2.1. generates the output states h_i for the i^{th} element of the input sequence $e_{1:n}$. Thereafter, each h_i of the Bi-GRU unit is given to a feed-forward neural network. This network computes the vectors of probability score p_i. The length of the probability vector $len(p_i)$ = $(2 \times$ number of possible named entity category + $1)$. This length exhibits the BIO (B-beginning, I-intermediate and O-outside) encoding of the entity classes.

2.2.4. Output sequence optimization layer

This layer takes the output (sequence of probability vector) of the previous layer as input and produces the label sequence as the output. However, the label sequence $C_{1:n}$ can be obtained for C_i as follows: $C_i = argmax_k(p_i[k])$. This greedy approach costs the performance of a model where the final output is the sequence of labels. This strategy does not account the dependencies between the subsequent labels. However, these dependencies capture at certain level in the input sequencing layer. But, we can still allow the model to directly learn these dependencies in the last layer of the model. To achieve this transition probability two subsequent labels are utilized. The final score of a label sequence can be defined as follows:

$$Score(C_{1:n}) = \sum_{k=2}^{n} M[C_{k-1}][C_k] + \sum_{k=1}^{n} p_i[C_k] \quad (2)$$

Where M is 2-D matrix that contains the transition probabilities between the two subsequent labels. While training, model maximizes the log-likelihood probability of the gold label sequence. In the testing phase, a sequence of predicted labels which maximize the score is chosen as the final label sequence.

3. Dataset and Experimental Setup

In this section, we report the datasets used in the experiments and experimental setup.

3.1. Dataset

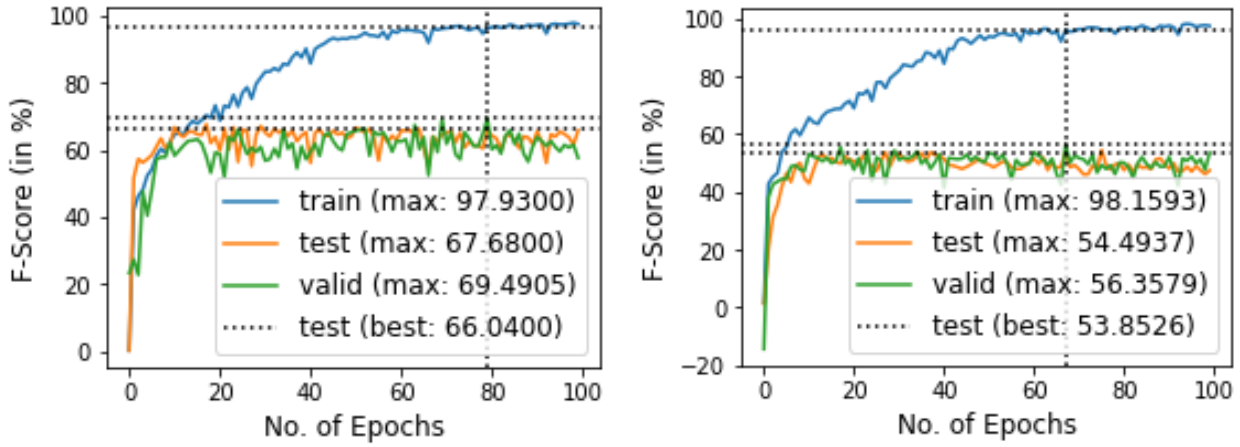
In order to evaluate the system performance, we use two language pairs: English-Hindi (EN-HI) and English-Tamil (EN-TA), which have been made available through the CMEE shared task (Rao and Devi, 2016). Datasets are crawled from the tweeter. The tweets are mixed in nature containing English-Hindi and English-Tamil scripts. The dataset contains a total of 22 different type of entities. Majority of entities are from 'Entertainment', 'Person' 'Location' and 'Organization' categories. A brief statistics of the dataset is provided in Table-2. English-Hindi tweet dataset contains a total of $2,700$ tweets from $2,699$ tweeter users. Similarly, English-Tamil tweet dataset contains a total of $2,183$ tweets from $1,866$ tweeter users. As some of the tweets from EN-TA language pair dataset are in Tamil script, in order to keep everything in English we transliterate[2] those tweets into English. We evaluate the system performance using the standard metrics, precision, recall and F-score.

[1] http://nlp.stanford.edu/data/glove.840B.300d.zip

[2] https://github.com/deepak1357/indic-trans

Models	English-Hindi			English-Tamil		
	Precision	Recall	F-Score	Precision	Recall	F-Score
SVM	72.25	50.12	59.18	69.38	36.12	47.50
MEMM	73.89	51.07	60.39	68.75	37.61	48.62
CRF	75.14	52.29	61.66	69.27	38.02	49.09
Our model	74.29	59.44	66.04	67.93	44.15	53.85

Table 3: Performance comparison of our proposed model with ML based baseline classifier.



(a) English-Hindi

(b) English-Tamil

Figure 1: Validation curve for both pair of dataset

3.2. Experimental Setup

We perform 5-fold cross validation on the both language pair datasets as described in Section 3.1.

We optimize the network by setting the following hyper-parameters through 5-fold cross-validation experiments: word embedding dimension=300, character embedding dimension=50, character-based word embedding GRU dimension=50, input sequence processing GRU dimension: 100, dropout probability: 0.5, dropout probability=0.3, batch size=50, # epochs=100.The network is trained using Adam optimization algorithm (Kingma and Ba, 2014) by updating all the parameters. This reflects the generalization of our optimum hyper-parameter selection over two completely different language pair datasets.

4. Result and Analysis

For comparison, we design three strong baselines based on traditional supervised techniques. Our baseline systems are based on Support vector machine (SVM) [3], Maximum entropy markov model [4] and Conditional random field [5] classifiers. In order to train the classifier the feature set described in (Gupta et al., 2016) is extracted. We evaluate the performance of all three baselines using 5-fold cross validation to be consistent with our proposed neural network.

Table-3 reports the performance of our proposed approach with all the three baselines for both the language pair datasets. We also plot the validation curve for both the language pair datasets. Fig 1 shows the validation curve between F-score and no. of epochs. The model achieves the best F-score value of 66.04% in 79 epochs for EN-HI (i.e. English-Hindi). Similarly, we obtain the best F-score value of 53.85% for EN-TA language pair dataset with 67 epochs. There could be two possible reasons behind the low F-score values on EN-TA dataset :

1. less instances of entities in the dataset as compared to EN-HI.
2. lower hit ratio (availability of word) of EN-TA word (37.46%) as compared to EN-HI word (55.70%) in word embedding matrix.

We further improve the system performance by using the word embedding trained on an appropriate code-mixing dataset. Our proposed approach outperforms all the baselines for both the language pairs. We observe that we achieve significant gains in terms of recall as compared to the baselines. This may be attributed to the fact our proposed network learns the hidden and useful features from a text, which handcrafted features can not infer always.

The best-performing system (Bhat et al., 2016) in the shared task reported the F-score value of 68.24% in English-Hindi language pair. They have used several hand-crafted features to train the feed-forward neural network. For English-Tamil language pair the best- performing system (Gupta et al., 2016) reported the F-score value of 44.12 using the handcrafted features with CRF as the underlying classifier. The evaluation shows that our

[3] http://chasen.org/ taku/software/yamcha/

[4] https://nlp.stanford.edu/software/classifier.html

[5] https://taku910.github.io/crfpp/

proposed system achieves higher performance compared to these systems. However, it is to be noted that our reported results are on cross-validation and we have not been able to perform experiments on the test data as this is not publicly available. Hence, it will not be fair to compare these existing systems with our proposed system.
Our obtained results are reliable as we perform experiments on cross-validation Significance test shows that the improvement over all the baselines are statistically significant as (p-*value* < 0.044).

4.1. Error Analysis

In order to get an idea about the errors, we perform in-depth analysis of the outputs of the systems. We have segregated our prediction inaccuracies based on its genre and have mentioned some of the observations below:
(1) Incorrect Entity: This error is caused when the predicted entity is misclassified into the incorrect entity type. This appears mainly due to the irregular casing of multiple words in a tweet. One possible reason for incorrect classification of tags also includes words that convey more than one semantic information.
(2) Missed Entity: This error is generated when the system fails to predict the entity tag and mis-classified it into other-than-NE. The possible causes for this error include:

- **Contextual information:** Words appearing at the start and end of tweets are easily recognized as NE, but the system fails to predict many entities present in the middle of a tweet.
- **Presence of abbreviated words:** Due to the presence of a large number of abbreviations, many words were left un-categorized by our system. Words like *SRK* and *FB* were often confused with *"Shah Rukh Khan"* and *Facebook*, leading to incorrect predictions by our system.

5. Conclusion and Future Works

In this paper, we have described an approach for entity extraction from the code-mixed tweets. We have proposed a deep learning model utilizing bi-directional GRU architecture. The proposed deep learning architectures discover the hidden features from the tweets automatically to categorize them into one of the pre-defined classes. The system has been evaluated for two language pairs, namely English-Hindi and English-Tamil. Experimental results show that our system achieves encouraging performance for both the language pairs. Empirical evaluation shows that our proposed system performs better as compared to the feature-based supervised model. In future, we will investigate the usages of shared representation of code mixed words in different languages.

Acknowledgements

Asif Ekbal greatfully acknowledges the Young Faculty Research Fellowship (YFRF) Award, supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

6. Bibliographical References

Bhat, I. A., Shrivastava, M., and Bhat, R. A. (2016). Code mixed entity extraction in indian languages using neural networks. In *FIRE (Working Notes)*, pages 296–297.

Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. (1997). Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, pages 194–201. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Bontcheva, K., Derczynski, L., and Roberts, I. (2014). Crowdsourcing named entity recognition and entity linking corpora. *The Handbook of Linguistic Annotation (to appear)*.

Borthwick, A. (1999). *A maximum entropy approach to named entity recognition*. Ph.D. thesis, Citeseer.

Cano Basave, A. E., Varga, A., Rowe, M., Stankovic, M., and Dadzie, A.-S. (2013). Making sense of microposts (# msm2013) concept extraction challenge.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Derczynski, L., Maynard, D., Rizzo, G., van Erp, M., Gorrell, G., Troncy, R., Petrak, J., and Bontcheva, K. (2015). Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2):32–49.

Dernoncourt, F., Lee, J. Y., Uzuner, O., and Szolovits, P. (2017). De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*, 24(3):596–606.

Grishman, R. and Sundheim, B. (1996). Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gupta, D., Tripathi, S., Ekbal, A., and Bhattacharyya, P. (2016). A hybrid approach for entity extraction in code-mixed social media data. *MONEY*, 25:66.

Han, B. and Baldwin, T. (2011). Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics.

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

Isozaki, H. and Kazawa, H. (2002). Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kumar, N. and Bhattacharyya, P. (2006). Named entity recognition in hindi using memm. *Techbical Report, IIT Mumbai*.

Li, W. and McCallum, A. (2003). Rapid development of hindi named entity recognition using conditional random fields and feature induction. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3):290–294.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Rao, P. R. and Devi, S. L. (2016). Cmee-il: Code mix entity extraction in indian languages from social media text@ fire 2016-an overview. In *FIRE (Working Notes)*, pages 289–295.

Rao, P. R., Malarkodi, C., and Devi, S. L.). Esm-il: Entity extraction from social media text for indian languages@ fire 2015–an overview.

Ritter, A., Clark, S., Etzioni, O., et al. (2011). Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Van Erp, M., Rizzo, G., and Troncy, R. (2013). Learning with the web: Spotting named entities on the intersection of nerd and machine learning. In *# MSM*, pages 27–30.

Yao, K., Zweig, G., Hwang, M.-Y., Shi, Y., and Yu, D. (2013). Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528.

7. Language Resource References

Rao, P. R. and Devi, S. L. (2016). Cmee-il: Code mix entity extraction in indian languages from social media text@ fire 2016-an overview. In *FIRE (Working Notes)*, pages 289–295.