Code Mixed Entity Extraction in Indian Languages using Neural Networks

Irshad Ahmad Bhat
LTRC, IIIT-H
Hyderabad, India
irshad.bhat@research.iiit.ac.in

Manish Shrivastava
LTRC, IIIT-H
Hyderabad, India
m.shrivastava@iiit.ac.in

Riyaz Ahmad Bhat
LTRC, IIIT-H
Hyderabad, India
riyaz.bhat@research.iiit.ac.in

ABSTRACT

In this paper we present our submission for *FIRE 2016 Shared Task on Code Mixed Entity Extraction in Indian Languages*. We describe a Neural Network system for Entity Extraction in Hindi-English Code Mixed text. Our method uses distributed word representations as features for the Neural Network and therefore, can easily be replicated across languages. Our system ranked first place for Hindi-English with an F1-score of 68.24% .

1. INTRODUCTION

This paper describes our system for the *FIRE 2016 Shared Task on Code Mixed Entity Extraction in Indian Languages*. The workshop focuses on NLP approaches for identifying named entities such as Person names, Organization names, Product names, Location names etc. in code mixed text.

In this paper, we present a simple feed forward neural network for Named Entity Recognision (NER) that use distributed word representations built using *word2vec* [2] and no other language-specific resources but the unlabeled corpora.

The rest of the paper is organized as follows: In Section 2, we discuss about the data of the shared task. In Section 3, we discuss the methodology we adapted to address the problem of NER, in detail. Experiments and Results based on our methodology are discussed in Section 4. Finally we conclude in Section 5.

2. DATA

The *Entity Extraction in the Code-Mixed (CM) data in Indian Languages* shared task is meant for NER in 2 language pairs namely, *Hindi-English* (H-E) and *Telugu-English* (T-E). However, we only recived data for Hindi-English language pair.

2.1 Data Format

The training data is provided into two files; a *raw-tweets file* and an *annotation file*. (1) below, shows the format of raw tweets in train and test data, while (2) shows the format of named entity annotations for the training data.

(1) *TweetID, UserID, Tweet*

(2) *TweetID, UserID, NE-Tag, NE, startIndex, Length*

2.2 Data Statistics

Table 1 shows the train and test data stattictics after tokenization. Table 2 shows the tag statistics in the training data. Note that the *OTHER* tag in table 2 is not any NE tag. We introduced this tag for all non-NE tokens.

Data-set	**# Tweets**	**# Tokens**
Train	2,700	39,216
Test	7,429	1,50,056

Table 1: Data Statistics

Tag	**Count**	**Tag**	**Count**
ENTERTAINMENT	858	LOCOMOTIVE	11
PERSON	338	YEAR	9
LOCATION	88	MATERIALS	9
ORGANIZATION	72	TIME	8
COUNT	65	FACILITIES	8
PERIOD	59	LIVTHINGS	5
ARTIFACT	29	DISEASE	5
DATE	27	SDAY	3
MONEY	23	MONTH	3
DAY	18	*OTHER*	38366

Table 2: Tag Statistics in the training data

3. METHODOLOGY

We modelled the task into a classification problem where each token needs to be labelled with one of the 20 tags as given in table 2. For this classification task, we use a simple neural network architecture. The neural network model is the standard feed-forward neural network with a single layer of hidden units. The output layer uses softmax function for probabilistic multi-class classification. The model is trained by minimizing cross entropy loss with an $l2$-regularization over the entire training data. We use mini-batch *Adagrad* for optimization and apply *dropout*.

We explored various token level and contextual features to build an optimal Neural Network using the provided training data. These features can be broadly grouped as described below:

Contextual Word Features: They constitute the current word and 2 words to either side of the current word.

Contextual Prefix Features: They constitute the current word prefix and prefixes of 2 words to either side of the current word. All these prefixes are of length 3.

Contextual Suffix Features: They constitute the current word suffix and suffixes of 2 words to either side of the current word. All these suffixes are of length 3.

Non-lexical Features: They constitute *capitalization* feature and *length* feature. Capitalization feature represents if a word is in upper-case, lower-case or title-case. Length feature represents the token length in the form of bins: 1-5, 6-8 and rest. The non-lexical features are added for the current word only.

We include the lexical features in the input layer of the Neural Network using the distributed word representations while for

the non-lexical features we use randomly initialized 3-dimensional vectors within a range of -0.25 to $+0.25$. We use Hindi and English monolingual corpora to learn the distributed representation of the lexical units. The English monolingual data contains around 280M sentences, while the Hindi data is comparatively smaller and contains around 40M sentences. To learn the suffix and prefix embeddings, we simply crated prefix and suffix corpora from the original monolingual corpora of Hindi and English and then use *word2vec* to learn their embeddings.

Instead of a single language specific word embedding for each lexical feature, we use a concatenated vector from Hindi and English word embeddings. This approach has three main benefits. First, we do not need a language identification system to choose the embedding space of a lexical item. Second, we do not depend on a joint word embedding space which is usually trained using a costly bilingual lexicon. Third, the named entities are usually shared between the languages. This provides a two-way evidence to the training model to learn named entities. We use [1] transliteration system [1] to transliterate Roman words to Devanagari, so that we can extract their embeddings from the Hindi embeddings space. Apart from named entities Hindi words will not be present in the English embedding space and English words will not be present in the Hindi embedding space.

4. EXPERIMENTS AND RESULTS

In any non-linear neural network model, we need to tune a number of hyperparameters for an optimal performance. The hyperparameters include number of hidden units, choice of activation function, choice of optimizer, learning rate, dropout, dimensionality of input units, etc. We used 20% of training data for tuning these parameters. The optimal parameters include: *200 hidden units, adagrad optimizer, rectilinear activation function, 200 bacth size, 0.025 learning rate, 0.5 dropout and 25 training iterations*. We obtained best development set accuracy at 80 dimensional word embeddings and 20 dimensional prefix and suffix embeddings. Development set results are given in Table 3. Test set results are given in Table 4

NE-TAG	Precision	Recall	F1-score	Support
ARTIFACT	1.00	0.10	0.18	10
COUNT	0.67	0.46	0.55	13
DATE	1.00	0.43	0.60	7
DAY	1.00	1.00	1.00	4
DISEASE	0.00	0.00	0.00	1
ENTERTAINMENT	0.98	0.62	0.76	174
LOCATION	0.88	0.54	0.67	13
LOCOMOTIVE	0.00	0.00	0.00	3
MATERIALS	0.00	0.00	0.00	3
MONEY	0.00	0.00	0.00	3
MONTH	0.00	0.00	0.00	1
ORGANIZATION	1.00	0.07	0.13	14
PERIOD	0.64	0.78	0.70	9
PERSON	0.98	0.68	0.80	71
TIME	0.00	0.00	0.00	1
YEAR	1.00	1.00	1.00	2
avg / total	**0.92**	**0.57**	**0.68**	**329**

Table 3: Development set results

[1]https://www.github.com/irshadbhat/indic-trans

Precision	Recall	F1-score
80.92	**59.00**	**68.24**

Table 4: Test set results

5. CONCLUSION

In this paper, we proposed a resource light Neural Network architecture for Entity Extraction in Hindi-English Code Mixed text. The Neural Network uses distributed representation of lexical features learned from monolingual corpora. Despite the simplesity of our architecture we achieved best results.

6. REFERENCES

[1] I. A. Bhat, V. Mujadia, A. Tammewar, R. A. Bhat, and M. Shrivastava. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA, 2015. ACM.

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