Named Entity Recognition for Nepali Language

Oyesh Mann Singh, Ankur Padia and Anupam Joshi
University of Maryland, Baltimore County
Baltimore, MD, USA
{osingh1, pankur1, joshi}@umbc.edu

Abstract

Named Entity Recognition have been studied for different languages like English, German, Spanish and many others but no study have focused on Nepali language. In this paper we propose a neural based Nepali NER using latest state-of-the-art architecture based on grapheme-level which doesn't require any hand-crafted features and no data pre-processing. Our novel neural based model gained relative improvement of 33% to 50% compared to feature based SVM model and up to 10% improvement over state-of-the-art neural based models developed for languages beside Nepali.

1 Introduction

Named Entity Recognition (NER) is a foremost NLP task to label each atomic elements of a sentence into specific categories like "PERSON", "LOCATION", "ORGANIZATION" and others(Collobert et al., 2011). There has been an extensive NER research on English, German, Dutch and Spanish language (Lample et al., 2016), (Ma and Hovy, 2016), (Devlin et al., 2018), (Peters et al., 2018), (Akbik et al., 2018), and notable research on low resource South Asian languages like Hindi(Athavale et al., 2016), Indonesian(Gunawan et al., 2018) and other Indian languages (Kannada, Malayalam, Tamil and Telugu)(Gupta et al., 2018). However, there has been no study on developing neural NER for Nepali language. In this paper, we propose a neural based Nepali NER using latest state-of-the-art architecture based on grapheme-level which doesn't require any hand-crafted features and no data pre-processing.

Recent neural architecture like (Lample et al., 2016) is used to relax the need to hand-craft the features and need to use part-of-speech tag to determine the category of the entity. However, this architecture have been studied for languages like English, and German and not been applied to languages like Nepali which is a low resource language i.e limited data set to train the model. Traditional methods like Hidden Markov Model (HMM) with rule based approaches(Dey and Prukayastha, 2013),(Dey et al., 2014), and Support Vector Machine (SVM) with manual feature-engineering(Bam and Shahi, 2014) have been applied but they perform poor compared to neural. However, there has been no research in Nepali NER using neural network. Therefore, we created the named entity annotated dataset partly with the help of Dataturk[1] to train a neural model. The texts used for this dataset are collected from various daily news sources from Nepal[2] around the year 2015-2016.

Following are our contributions:

1. We present a novel Named Entity Recognizer (NER) for Nepali language. To best of our knowledge we are the first to propose neural based Nepali NER.

2. As there are not good quality dataset to train NER we release a dataset to support future research

3. We perform empirical evaluation of our model with state-of-the-art models with relative improvement of upto 10%

In this paper, we present works similar to ours in Section 2. We describe our approach and dataset statistics in Section 3 and 4, followed by our experiments, evaluation and discussion in Section 5, 6, and 7. We conclude with our observations in Section 8.

[1]https://dataturks.com/
[2]https://github.com/sndsabin/Nepali-News-Classifier

To facilitate further research our code and dataset will be made available at github.com/link-yet-to-be-updated

2 Related Work

There has been a handful of research on Nepali NER task based on approaches like Support Vector Machine and gazetteer list(Bam and Shahi, 2014) and Hidden Markov Model and gazetteer list(Dey and Prukayastha, 2013),(Dey et al., 2014).

(Bam and Shahi, 2014) uses SVM along with features like first word, word length, digit features and gazetteer (person, organization, location, middle name, verb, designation and others). It uses one vs rest classification model to classify each word into different entity classes. However, it does not the take context word into account while training the model. Similarly, (Dey and Prukayastha, 2013) and (Dey et al., 2014) uses Hidden Markov Model with n-gram technique for extracting POS-tags. POS-tags with common noun, proper noun or combination of both are combined together, then uses gazetteer list as look-up table to identify the named entities.

Researchers have shown that the neural networks like CNN(LeCun et al., 1989), RNN(Rumelhart et al., 1988), LSTM(Hochreiter and Schmidhuber, 1997), GRU(Chung et al., 2014) can capture the semantic knowledge of language better with the help of pre-trained embbeddings like word2vec(Mikolov et al., 2013), glove(Pennington et al., 2014) or fast-text(Bojanowski et al., 2016).

Similar approaches has been applied to many South Asian languages like Hindi(Athavale et al., 2016), Indonesian(Gunawan et al., 2018), Bengali(Banik and Rahman, 2018) and In this paper, we present the neural network architecture for NER task in Nepali language, which doesn't require any manual feature engineering nor any data pre-processing during training. First we are comparing BiLSTM(Hochreiter and Schmidhuber, 1997), BiLSTM+CNN(Chiu and Nichols, 2015), BiLSTM+CRF(Lample et al., 2016), BiLSTM+CNN+CRF(Ma and Hovy, 2016) models with CNN model(Collobert et al., 2011) and Stanford CRF model(Finkel et al., 2005). Secondly, we show the comparison between models trained on general word embeddings, word embedding + character-level embedding, word embedding + part-of-speech(POS) one-hot encoding and word embedding + grapheme clustered or sub-word embedding(Park and Shin, 2018). The experiments were performed on the dataset that we created and on the dataset received from ILPRL lab[3]. Our extensive study shows that augmenting word embedding with character or grapheme-level representation and POS one-hot encoding vector yields better results compared to using general word embedding alone.

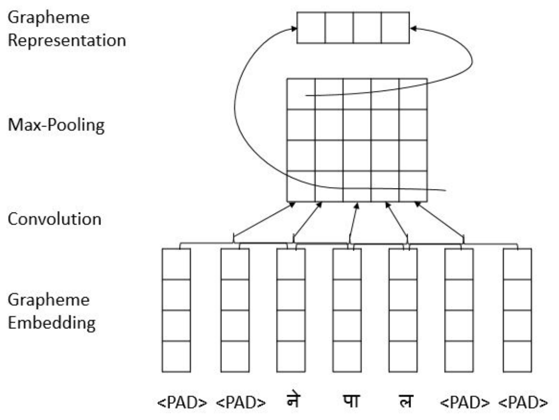


Figure 1: The grapheme level convolution neural network to extract grapheme representation. The dropout layer is applied after maxpooling layer.

3 Approach

In this section, we describe our approach in building our model. This model is partly inspired from multiple models (Chiu and Nichols, 2015),(Lample et al., 2016), and(Ma and Hovy, 2016)

3.1 Bidirectional LSTM

We used Bi-directional LSTM to capture the word representation in forward as well as reverse direction of a sentence. Generally, LSTMs take inputs from left (past) of the sentence and computes the hidden state. However, it is proven beneficial(Dyer et al., 2015) to use bi-directional LSTM, where, hidden states are computed based from right (future) of sentence and both of these hidden states are concatenated to produce the final output as h_t=[$\overrightarrow{h_t}$;$\overleftarrow{h_t}$], where $\overrightarrow{h_t}$, $\overleftarrow{h_t}$ = hidden state computed in forward and backward direction respectively.

[3]http://ilprl.ku.edu.np/

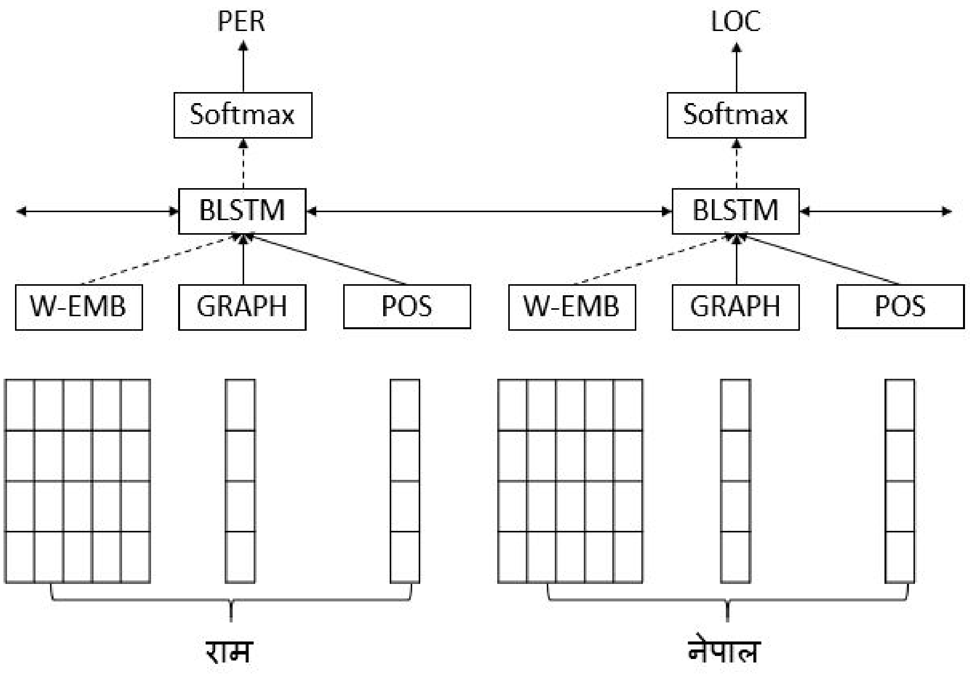


Figure 2: End-to-end model architecture of our neural network. W-EMB, GRAPH, POS represents pre-trained word embeddings, grapheme representations and POS one-hot encoding vectors. GRAPH is obtained from CNN as shown in figure 1. The dashed line implies the application of dropout.

3.2 Features

3.2.1 Word embeddings

We have used Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and FastText (Bojanowski et al., 2016) word vectors of 300 dimensions. These vectors were trained on the corpus obtained from Nepali National Corpus[4]. This pre-lemmatized corpus consists of 14 million words from books, web-texts and news papers. This corpus was mixed with the texts from the dataset before training CBOW and skip-gram version of word2vec using gensim library(Řehůřek and Sojka, 2010). This trained model consists of vectors for 72782 unique words.

Light pre-processing was performed on the corpus before training it. For example, invalid characters or characters other than Devanagari were removed but punctuation and numbers were not removed. We set the window context at 10 and the rare words whose count is below 5 are dropped. These word embeddings were not frozen during the training session because fine-tuning word embedding help achieve better performance compared to frozen one(Chiu and Nichols, 2015).

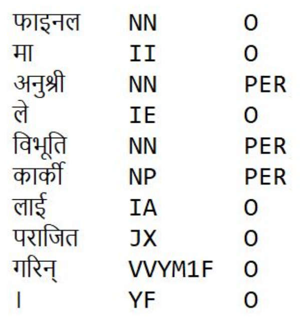
We have used fasttext embeddings in particular because of its sub-word representation ability, which is very useful in highly inflectional language as shown in Table 3. We have trained the word embedding in such a way that the sub-word size remains between 1 and 4. We particularly chose this size because in Nepali language a single letter can also be a word, for example ए, त, छ, र, ल, न, उ and a single character (grapheme) or sub-word can be formed after mixture of dependent vowel signs with consonant letters for example, छ + ौ + ं = छौं, here three different consonant letters form a single sub-word.

The two-dimensional visualization of an example word नेपाल is shown in 4. Principal Component Analysis (PCA) technique was used to generate this visualization which helps use to analyze the nearest neighbor words of a given sample word. 84 and 104 nearest neighbors were observed using word2vec and fasttext embedding respectively on the same corpus.

3.2.2 Character-level embeddings

(Chiu and Nichols, 2015) and (Ma and Hovy, 2016) successfully presented that the character-level embeddings, extracted using CNN, when combined with word embeddings enhances the NER model performance significantly, as it is able

[4]https://www.sketchengine.eu/nepali-national-corpus/



फाइनल	NN	O
मा	II	O
अनुश्री	NN	PER
ले	IE	O
विभूति	NN	PER
कार्की	NP	PER
लाई	IA	O
पराजित	JX	O
गरिन्	VVYM1F	O
।	YF	O

Figure 3: Format of a sample sentence in our dataset.

to capture morphological features of a word. Figure 1 shows the grapheme-level CNN used in our model, where inputs to CNN are graphemes. Character-level CNN is also built in similar fashion, except the inputs are characters. Grapheme or Character -level embeddings are randomly initialized from [0,1] with real values with uniform distribution of dimension 30.

3.2.3 Grapheme-level embeddings

Grapheme is atomic meaningful unit in writing system of any languages. Since, Nepali language is highly morphologically inflectional, we compared grapheme-level representation with character-level representation to evaluate its effect. For example, in character-level embedding, each character of a word नेपाल results into न + े + प + ा + ल has its own embedding. However, in grapheme level, a word नेपाल is clustered into graphemes, resulting into ने + पा + ल. Here, each grapheme has its own embedding. This grapheme-level embedding results good scores on par with character-level embedding in highly inflectional languages like Nepali, because graphemes also capture syntactic information similar to characters. We created grapheme clusters using uniseg[5] package which is helpful in unicode text segmentations.

3.2.4 Part-of-speech (POS) one hot encoding

We created one-hot encoded vector of POS tags and then concatenated with pre-trained word embeddings before passing it to BiLSTM network. A sample of data is shown in figure 3.

[5]https://uniseg-python.readthedocs.io/en/latest/index.html

4 Dataset Statistics

4.1 OurNepali dataset

Since, we there was no publicly available standard Nepali NER dataset and did not receive any dataset from the previous researchers, we had to create our own dataset. This dataset contains the sentences collected from daily newspaper of the year 2015-2016. This dataset has three major classes Person (PER), Location (LOC) and Organization (ORG). Pre-processing was performed on the text before creation of the dataset, for example all punctuations and numbers besides ',', '-', '—' and '.' were removed. Currently, the dataset is in standard CoNLL-2003 IO format(Tjong Kim Sang and De Meulder, 2003).

Since, this dataset is not lemmatized originally, we lemmatized only the post-positions like कि, कौ, ले, मा, मै, मय, जी, सँग, अघि which are just the few examples among 299 post positions in Nepali language. We obtained these post-positions from sanjaalcorps[6] and added few more to match our dataset. We will be releasing this list in our github repository. We found out that lemmatizing the post-positions boosted the F1 score by almost 10%.

In order to label our dataset with POS-tags, we first created POS annotated dataset of 6946 sentences and 16225 unique words extracted from POS-tagged Nepali National Corpus and trained a BiLSTM model with 95.14% accuracy which was used to create POS-tags for our dataset.

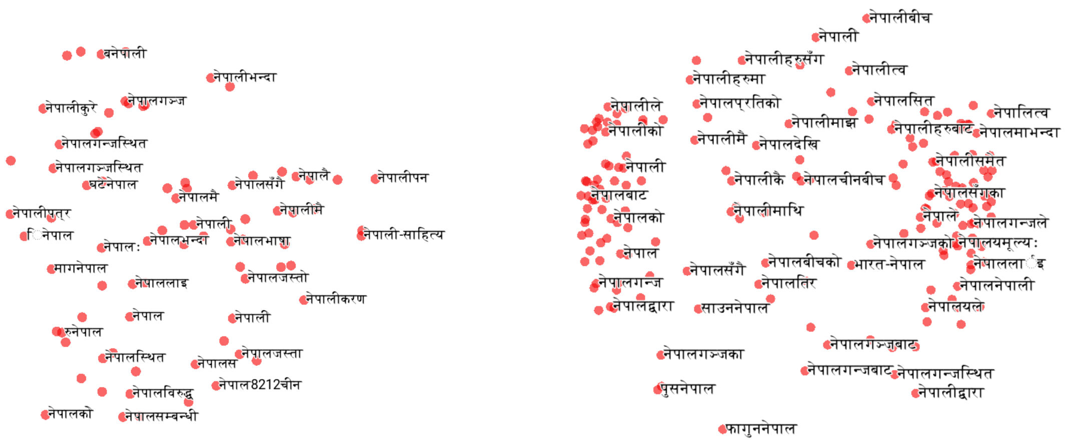
The dataset released in our github repository contains each word in newline with space separated POS-tags and Entity-tags. The sentences are separated by empty newline. A sample sentence from the dataset is presented in table 3.

4.2 ILPRL dataset

After much time, we received the dataset from Bal Krishna Bal, ILPRL, KU. This dataset follows standard CoNLL-2003 IOB format(Tjong Kim Sang and De Meulder, 2003) with POS tags. This dataset is prepared by ILPRL Lab[7], KU and KEIV Technologies. Few corrections like correcting the NER tags had to be made on the dataset. The statistics of both the dataset is presented in table 1.

Table 2 presents the total entities (PER, LOC, ORG and MISC) from both of the dataset used in

[6]https://github.com/sanjaalcorps/NepaliStemmer
[7]http://ilprl.ku.edu.np/



(a) Word2vec embedding

(b) Fasttext embedding

Figure 4: 2D Visualization of nearest neighbor word using PCA for a sample word नेपाल

Dataset	ILPRL	OurNepali
PER	262	5059
ORG	180	3811
LOC	273	2313
MISC	461	0
Total entities w/o O	1176	11183
Others - O	12683	67904
Total entities w/ O	13859	79087
Total sentences	548	3606

Table 1: Dataset statistics

Dataset	ILPRL	OurNepali
Train	754	7165
Test	188	2033
Dev	234	1985

Table 2: Dataset division statistics. The number presented are total count of entities token in each set.

our experiments. The dataset is divided into three parts with 64%, 16% and 20% of the total dataset into training set, development set and test set respectively.

5 Experiments

In this section, we present the details about training our neural network. The neural network architecture are implemented using PyTorch framework (Paszke et al., 2017). The training is performed on a single Nvidia Tesla P100 SXM2. We first run our experiment on BiLSTM, BiLSTM-CNN, BiLSTM-CRF BiLSTM-CNN-CRF using the hyper-parameters mentioned in Table 4. The training and evaluation was done on sentence-level. The RNN variants are initialized randomly from $(-\sqrt{k}, \sqrt{k})$ where $k = \frac{1}{hidden\_size}$.

First we loaded our dataset and built vocabulary using torchtext library[8]. This eased our process of data loading using its SequenceTaggingDataset class. We trained our model with shuffled training set using Adam optimizer with hyper-parameters mentioned in table 4. All our models were trained on single layer of LSTM network. We found out Adam was giving better performance and faster convergence compared to Stochastic Gradient Descent (SGD). We chose those hyper-parameters after many ablation studies. The dropout of 0.5 is applied after LSTM layer.

For CNN, we used 30 different filters of sizes 3, 4 and 5. The embeddings of each character or grapheme involved in a given word, were passed through the pipeline of Convolution, Rectified Linear Unit and Max-Pooling. The resulting vectors were concatenated and applied dropout of 0.5 before passing into linear layer to obtain the embedding size of 30 for the given word. This resulting embedding is concatenated with word embeddings, which is again concatenated with one-hot POS vector.

5.1 Tagging Scheme

Currently, for our experiments we trained our model on IO (Inside, Outside) format for both the dataset, hence the dataset does not contain any B-type annotation unlike in BIO (Beginning, Inside, Outside) scheme.

[8]https://torchtext.readthedocs.io/en/latest/

Embeddings	OurNepali					
	Raw			Lemmatized		
	Train	Test	Val	Train	Test	Val
Random	78.72	63.66	64.89	88.44	75.11	77.2
Word2Vec_CBOW	82.33	74.59	75.15	88.05	81.96	83.82
Word2Vec_Skip Gram	81.58	75.93	75.75	89.84	83.47	85.79
GloVe	87.54	76.86	76.7	92.48	82.24	84.16
fastText_Pretrained	81.57	75.06	71.96	85.76	77.6	79.78
fastText_CBOW	86.01	81.4	80.52	89.23	81.58	83.51
fastText_Skip Gram	88.31	80.6	78.85	**94.01**	**84.74**	**85.11**

Table 3: Effect of different embedding with Bi-LSTM.

5.2 Early Stopping

We used simple early stopping technique where if the validation loss does not decrease after 10 epochs, the training was stopped, else the training will run upto 100 epochs. In our experience, training usually stops around 30-50 epochs.

5.3 Hyper-parameters Tuning

We ran our experiment looking for the best hyper-parameters by changing learning rate from (0,1, 0.01, 0.001, 0.0001), weight decay from [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}], batch size from [1, 2, 4, 8, 16, 32, 64, 128], hidden size from [8, 16, 32, 64, 128, 256, 512 1024]. Table 4 shows all other hyper-parameter used in our experiment for both of the dataset.

Hyper-parameters	**Value**
LSTM - hidden size	100
CNN - Filter size	[3,4,5]
CNN - Filter number	30
Learning rate	0.001
Weight decay	1.00E-006
Batch size	1
Dropout	0.5

Table 4: Hyper-parameters of our experiments

5.4 Effect of Dropout

Figure 5 shows how we end up choosing 0.5 as dropout rate. When the dropout layer was not used, the F1 score are at the lowest. As, we slowly increase the dropout rate, the F1 score also gradually increases, however after dropout rate = 0.5, the F1 score starts falling down. Therefore, we have chosen 0.5 as dropout rate for all other experiments performed.

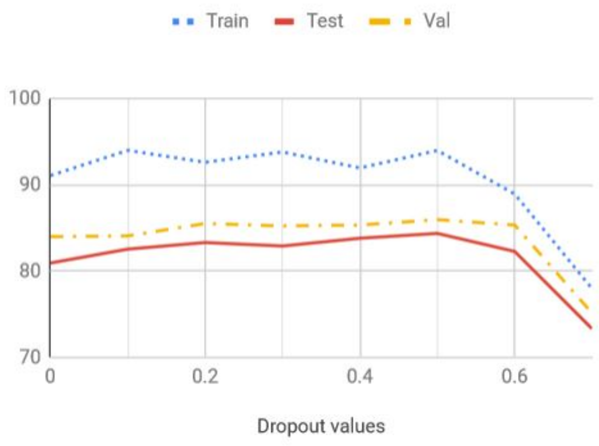


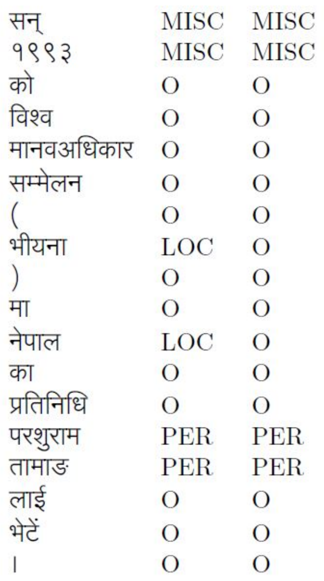
Figure 5: F1 score based on different dropout values using fastText embeddings (Skip Gram). All other hyper-parameter used for this evaluation are presented in table 4.

6 Evaluation

In this section, we present the details regarding evaluation and comparison of our models with other baselines.

Table 3 shows the study of various embeddings and comparison among each other in OurNepali dataset. Here, raw dataset represents such dataset where post-positions are not lemmatized. We can observe that pre-trained embeddings significantly improves the score compared to randomly initialized embedding. We can deduce that Skip Gram models perform better compared CBOW models for word2vec and fasttext. Here, fastText_Pretrained represents the embedding readily available in fastText website[9], while other embeddings are trained on the Nepali National Corpus as mentioned in sub-section 3.2.1. From this table 3, we can clearly observe that model using fast-

[9]https://fasttext.cc/docs/en/crawl-vectors.html



सन्	MISC	MISC
१९९३	MISC	MISC
को	O	O
विश्व	O	O
मानवअधिकार	O	O
सम्मेलन	O	O
(	O	O
भीयना	LOC	O
)	O	O
मा	O	O
नेपाल	LOC	O
का	O	O
प्रतिनिधि	O	O
परशुराम	PER	PER
तामाङ	PER	PER
लाई	O	O
भेटें	O	O
।	O	O

Figure 6: Sample output of the best model from ILPRL test dataset. First, second and third column indicates word to be predicted, ground truth and predicted truth respectively. We can see that not all the tags are classified correctly.

Text_Skip Gram embeddings outperforms all other models.

Table 5 shows the model architecture comparison between all the models experimented. The features used for Stanford CRF classifier are words, letter n-grams of upto length 6, previous word and next word. This model is trained till the current function value is less than $1e{-}2$. The hyper-parameters of neural network experiments are set as shown in table 4. Since, word embedding of character-level and grapheme-level is random, their scores are near.

All models are evaluated using CoNLL-2003 evaluation script(Tjong Kim Sang and De Meulder, 2003) to calculate entity-wise precision, recall and f1 score.

7 Discussion

In this paper we present that we can exploit the power of neural network to train the model to perform downstream NLP tasks like Named Entity Recognition even in Nepali language. We showed that the word vectors learned through fasttext skip gram model performs better than other word embedding because of its capability to represent sub-word and this is particularly important to capture morphological structure of words and sentences in highly inflectional like Nepali. This concept can come handy in other Devanagari languages as well because the written scripts have similar syntactical structure.

We also found out that stemming post-positions can help a lot in improving model performance because of inflectional characteristics of Nepali language. So when we separate out its inflections or morphemes, we can minimize the variations of same word which gives its root word a stronger word vector representations compared to its inflected versions.

We can clearly imply from tables 1, 2, and 5 that we need more data to get better results because OurNepali dataset volume is almost ten times bigger compared to ILPRL dataset in terms of entities.

8 Conclusion and Future work

In this paper, we proposed a novel NER for Nepali language and achieved relative improvement of upto 10% and studies different factors effecting the performance of the NER for Nepali language.

We also present a neural architecture BiLSTM+CNN(grapheme-level) which turns out to be performing on par with BiLSTM+CNN(character-level) under the same configuration. We believe this will not only help Nepali language but also other languages falling under the umbrellas of Devanagari languages. Our model BiLSTM+CNN(grapheme-level) and BiLSTM+CNN(G)+POS outperforms all other model experimented in OurNepali and ILPRL dataset respectively.

Since this is the first named entity recognition research in Nepal language using neural network, there are many rooms for improvement. We believe initializing the grapheme-level embedding with fasttext embeddings might help boosting the performance, rather than randomly initializing it. In future, we plan to apply other latest techniques like BERT, ELMo and FLAIR to study its effect on low-resource language like Nepali. We also plan to improve the model using cross-lingual or multi-lingual parameter sharing techniques by jointly training with other Devanagari languages like Hindi and Bengali.

Finally, we would like to contribute our dataset to Nepali NLP community to move forward the research going on in language understanding domain. We believe there should be special committee to create and maintain such dataset for Nepali NLP and organize various competitions

Dataset	OurNepali		ILPRL	
Models	**Test**	**Val**	**Test**	**Val**
Stanford CRF	75.16	79.61	56.25	72.07
BiLSTM	84.74	85.11	80.40	83.17
BiLSTM + POS	83.65	84.09	81.25	85.39
BiLSTM + CNN (C)	86.45	87.45	80.51	85.45
BiLSTM + CNN (G)	**86.71**	86.00	78.24	83.49
BiLSTM + CNN (C) + POS	85.40	86.50	81.46	86.64
BiLSTM + CNN (G) + POS	85.46	86.43	**83.08**	82.99

Table 5: Comparison of different variation of our models

	OurNepali	ILPRL
Model	**Test**	**Test**
Bam et al. SVM	66.26	46.26
Ma and Hovy w/ glove	83.63	72.1
Lample et al. w/ fastText	85.78	82.29
Lample et al. w/ word2vec	86.49	78.63
BiLSTM + CNN (G)	**86.71**	78.24
BiLSTM + CNN (G) + POS	85.46	**83.08**

Table 6: Comparison with previous models based on Test F1 score

Dataset	OurNepali			ILPRL		
Entities	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
PER	93.82	88.66	91.17	74.36	72.50	73.42
ORG	87.28	79.59	83.26	92.31	75.00	82.76
LOC	84.29	82.11	83.19	91.07	69.86	79.07
MISC	NA	NA	NA	94.94	87.21	90.91
Overall	89.45	84.14	**86.71**	89.30	77.67	**83.08**

Table 7: Entity-wise comparison using best model for respective dataset. MISC-entity is not available for OurNepali dataset.

which would elevate the NLP research in Nepal.

Some of the future works are listed below:

1. Proper initialization of grapheme level embedding from fasttext embeddings.
2. Apply robust POS-tagger for Nepali dataset
3. Lemmatize the OurNepali dataset with robust and efficient lemmatizer
4. Improve Nepali language score with cross-lingual learning techniques
5. Create more dataset using Wikipedia/Wikidata framework

9 Acknowledgments

The authors of this paper would like to express sincere thanks to Bal Krishna Bal, Kathmandu University Professor for providing us the POS-tagged Nepali NER data.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Vinayak Athavale, Shreenivas Bharadwaj, Monik Pamecha, Ameya Prabhu, and Manish Shrivastava. 2016. Towards deep learning in hindi NER: an approach to tackle the labelled data sparsity. *CoRR*, abs/1610.09756.

Surya Bahadur Bam and Tej Bahadur Shahi. 2014. Named entity recognition for nepali text using support vector machines. *Intelligent Information Management*, 6(02):21.

Nayan Banik and Md Hasan Hafizur Rahman. 2018. Gru based named entity recognition system for bangla online newspapers. In *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, pages 1–6. IEEE.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jason P. C. Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *CoRR*, abs/1511.08308.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Arindam Dey, Abhijit Paul, and Bipul Syam Purkayastha. 2014. Named entity recognition for nepali language: A semi hybrid approach. *International Journal of Engineering and Innovative Technology (IJEIT) Volume*, 3:21–25.

Arindam Dey and Bipul Syam Prukayastha. 2013. Named entity recognition using gazetteer method and n-gram technique for an inflectional language: A hybrid approach. *International Journal of Computer Applications*, 84(9).

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *CoRR*, abs/1505.08075.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

William Gunawan, Derwin Suhartono, Fredy Purnomo, and Andrew Ongko. 2018. Named-entity recognition for indonesian language using bidirectional lstm-cnns. *Procedia Computer Science*, 135:425–432.

Ayush Gupta, Meghna Ayyar, Ashutosh Kumar Singh, and Rajiv Ratn Shah. 2018. raiden11@ iecsil-fire-2018: Named entity recognition for indian languages. *FIRE Working Notes*.

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

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Suzi Park and Hyopil Shin. 2018. Grapheme-level awareness in word embeddings for morphologically rich languages. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

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*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.