Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 189 (2021) 300–311

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on AI in Computational Linguistics

Fine-grained part-of-speech tagging in Nepali text

Ingroj Shrestha[a], Shreeya Singh Dhakal[b]

[a] *The University of Iowa, Iowa City, IA, 52242, USA*
[b] *North Carolina State University, Raleigh, NC, 27606, USA*

Abstract

Part-of-speech (POS) tagging is essential in Natural Language Processing (NLP) and text analysis applications. POS-tagging is a well-researched problem, but there have been limited efforts in fine-grained Nepali POS-tagging. Highly inflectional languages like Nepali encodes information like gender, person, number, mood, and aspect within their words. Accurate disambiguation of these inflected word forms is essential in Nepali text analysis. This work shows that neural network models are great at disambiguating fine-grained morphological information encoded by words in Nepali texts. We experiment with three neural network-based architectures: BiLSTM, BiGRU, and BiLSTM-CRF. Our results show that deep-learning based models can capture fine-grained morphological information encoded by Nepali words given a large enough corpus. We trained all the models in this work using two embeddings: pre-trained multi-lingual BERT and randomly initialized Bare embeddings. We found that training a randomly initialized Bare embedding is better than the ones trained using large pre-trained multi-lingual BERT embedding for downstream tasks in Nepali like POS tagging. Among the tested models, the BiLSTM-CRF with the Bare embedding performed the best and achieved a new state-of-the-art F1 score of 98.51% for fine-grained Nepali POS tagging.

Keywords: Part-of-speech tagging; Fine-grained; Coarse-grained; Nepali; Inflections; Embeddings

1. Introduction

Part-of-Speech (POS) tagging is a fundamental task in Natural Language Processing (NLP) and text analysis applications. Given a sequence of words, a POS tagger assigns the words in the sequence with a corresponding grammatical category. However, POS tagging is not always as simple as giving a sequence of tags for a sequence of tokens as the grammatical category of words can often be ambiguous. The ambiguity

* Corresponding author.
E-mail address: ingroj-shrestha@uiowa.edu

1877-0509
10.1016/j.procs.2021.05.099

is higher for languages with complex and rich morphology, like Nepali. For this reason, most work in Nepali POS tagging is limited to coarse-grained tagsets that do not disambiguate morphological features such as gender, number, honorifics, and person encoded within a word.

Nepali is a morphologically rich language [12]. It is a complex language and has almost 75% of its general vocabulary comprising derived and inflectional words rather than headwords [2]. One of the characteristics features of the Nepali language is its rich inflectional system, especially the verbal inflection system. A verb in Nepali can easily display more than 20 inflectional forms while encoding different morphological features, including aspect, mood, tense, gender, number, honorifics, and person. The ability to disambiguate different grammatical features encoded within words is important in improving the accuracy and quality of NLP applications, especially those involving a morphologically rich language like Nepali. The existing Nepali POS taggers are limited. They are trained on a coarse-grain tagset and do not account for the fine-grained morphological information within the word categories.

Our primary focus in this research is fine-grained Nepali POS tagging. We will illustrate two different Nepali POS tagsets for texts in Nepali National Corpus, which was developed under the Bhasa Sanchar or NELRALEC (Nepali Language Resources and Localization for Education and Communication) project[1]. We will also assess different neural network architectures, including BiLSTM, BiGRU, and BiLSTM-CRF for the task. We will also experiment with two embeddings — multi-lingual BERT (mBERT) embedding and randomly initialized Bare embedding for Nepali POS tagging.

As we will show below, with a large enough corpus, deep learning-based approaches can capture fine-grained morphological categories as good as coarse-grained categories. We report a new state-of-the-art performance for fine-grained Nepali POS tagging, which is an improvement over the existing one [5] by about 6%. We will also show that models using randomly initialized Bare embeddings perform better than those using mBERT embedding for Nepali POS tagging. To our best knowledge, our work is the first work that uses mBERT for a downstream NLP task in Nepali. We also establish a need for further analysis of mBERT for morphologically challenging languages like Nepali with this work.

2. Related Work

Several efforts have been made towards the development of a POS tagger for Nepali. The very first effort in POS tagging in Nepali is the NELRALEC (Nepali Language Resources and Localization for Education and Communication) or the Bhasa Sanchar (भाषा संचार) project, led by Madan Puraskar Pustakalaya and funded by EU Asia ITC committee. Nepali National Corpus (NNC) with over 17 million words and NELRALEC tagset with 112 fine-grained POS Tags were developed under this project. The NELRALEC project also developed a Nepali tagger using the same framework as Unitag [5] and primarily using linguistic knowledge, disambiguation rules, and a Markov model-based probabilistic model. The Nepali Unitag tagger had an accuracy of 93%.

Shahi [13] proposed a Support Vector Machine (SVM) based part of speech tagger, which uses a dictionary of 11,147 unique words collected from FinalNepaliCorpus. The SVM tagger used a coarse tagset with 43 tags and a feature set, including word-level features, POS features, n-grams (up to trigrams), and ambiguity classes. The tagger relies on the SVMlight tool, a binary classifier, and uses the one v/s rest strategy for classification. The tagger shows an overall accuracy of 93.27% and 90.06% for unknown words.

A Hidden Markov Model (HMM) based tagger for Nepali was proposed in [10]. The dataset used in this work was annotated with 42 POS tags using SANCHAY[2], an annotated tool developed by IIT, Hyderabad. The proposed tagset follows the guidelines of Indian Languages Corpora Initiative[3] and Bureau of Indian Standard[4]. The work achieves an accuracy of 96% for known words. They report that the HMM tagger does not perform well for unknown words.

[1] https://www.lancaster.ac.uk/staff/hardiea/nepali/postag.php
[2] http://ltrc.iiit.ac.in/anil/sanchay/
[3] http://sanskrit.jnu.ac.in/ilci/index.jsp
[4] https://bis.gov.in/

Sinha et al. [15] combined a Hidden Markov Model (HMM) with a lexicon and linguistic rules for Nepali POS tagging. This work also used a coarse POS tagset with 43 tags and was evaluated on a corpus containing 15, 430 words. The hybrid method first runs each word through a word segmenter to get the root and searches for the root in the lexicon. If the word is found in the lexicon and contains a single tag, the corresponding tag is assigned to the word. Otherwise, the word is passed to an HMM-based model that returns a tagged output sequence. The words that are not recognized in either of the previous two steps are disambiguated with a rule-based module. The hybrid tagger achieves an accuracy of 93.15%.

Yajnik [18] applied an HMM and Viterbi algorithm to Nepali POS tagging. The work used the coarse NELRALEC tagset containing 41 tags and a corpus of 45,000 Nepali words. They found the Viterbi algorithm to be more accurate than HMM with the accuracy of 95.43% and 76.97%, respectively. The work also extracted a tag-wise error rate for the Viterbi algorithm, which showed Tag NN(Common noun) to be the most misclassified tag.

More recent works in Nepali POS Tagging use neural network-based methods. Yajnik [19] proposed three neural network-based architectures for Nepali POS tagging: Radial Basis Function (RBF) network, General Regression Neural Networks (GRNN), and Feed forward Neural network. GRNN was found to perform the best among the three with a reported accuracy of 98.32%. [11] explored different deep learning algorithms, including Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), Gated Recurrent Network (GRU), and their bidirectional variants for Nepali POS tagging. A Nepali corpus generated by translating 4325 English sentences from the PENN Treebank corpus tagged with 43 POS tags was used to evaluate the algorithms. Bidirectional variants of the models, RNN, LSTM, and GRU, performed the best. Hence, we use the Bidirectional LSTMs and GRUs based models for our experiments. The accuracy of the models increased with the increase in the word embedding size, with an embedding size of 300 giving the optimal performance. Following this, we also used an embedding size of 300 for our experiments that use the randomly initialized Bare embeddings.

Limited work has been done in terms of fine-grained POS Tagging for Nepali. Since Nepali is a morphologically rich language, fine-grained distinctions of grammatical categories of words are useful in improving the accuracy and quality of NLP applications for Nepali. This work explores how well the deep learning models capture the grammatical categories encoded by Nepali words using a fine-grained POS tagset containing 112 tags.

3. Nepali Morphology and NELRALEC tagset

Nepali morphology is highly inflectional and agglutinative [6]. Inflections in Nepali encode grammatical categories of words such as gender, number, tense, person, and level of honorifics. In Nepali, inflection occurs only by suffixation. Inflectional suffixes are regular and transparent in their function, meaning they do not change the lexical category and meaning of the root word [14]. Hence, inflectional variants of a root usually have the same lexical category and meaning as the root. Words in Nepali are also formed by compounding and agglutination. Compounding is a highly productive process in Nepali, especially for the Nepali verbs. Hence, Nepali verbs exhibit a complex structure.

The NELRALEC tagset and tagging scheme considers the postpositions or case markers as separate units. The postpositions or inflectional markers identified in NELRALEC are usually compounded with nouns and sometimes with pronouns and adjectives in Nepali. POS tags in NNC are assigned to tokens extracted by splitting the postpositions from the base words. For example रामले(raam-le) would be split into राम(ram) and ले(le) and tagged as राम*/NP* and ले*/IE*. The splitting is done because space between a base word and case markers is not consistent in Nepali. The tags are fully hierarchical, for example in the tag *DDM*, the first *D* indicates *Determiner*, *DD* indicates *Demonstrative Determiner*, finally, *DDM* denotes *Masculine Demonstrative Determiner*.

3.1. Nominal Inflections in Nepali: Nouns and Inflectional Markers

Nepali nouns inflect for seven different cases and two dimensions of number. Postpositions mark the inflections in Nepali nouns. Since postpositions are considered separate tokens, the NELRALEC tagset identifies only two categories of noun *Common Noun* and *Proper Noun*. Nouns in Nepali also express forms for gender, but these are lexical and not inflectional, so these distinctions are not identified in the NELRALEC tagset. The tagset contains ergative-instrumental (le(ले)), accusative-dative (laaii(लाई)), genetive (ko/kaa/kii(को/का/की)) postpositions and gender distinctions for these postpositions.

3.2. Nominal Inflections in Nepali: Adjective

Adjectives in Nepali inflect for two dimensions of number and gender are marked by the different vowel endings. NELRALEC identifies gender agreement in adjectives. Masculine and feminine adjectives are characterized by "o" and "i" endings, respectively. The tagset also recognizes adjectives borrowed from *Sanskrit* language.

3.3. Nominal Inflections in Nepali: Pronouns

The NELRALEC tagset makes distinctions for pronouns based on types: personal, possessive, reflexive, demonstrative, interrogative, and relative and the inflectional forms for each type. There are three personal pronouns in Nepali: first, second, and third. Like nouns and adjectives, personal pronouns inflect for two dimensions of number. Inflections due to number are marked by the plural marker (haru(हरु)) or by suppletive forms for some of the pronouns, like haamii(हामी) for we. Second and third-person pronouns shown five levels of honorifics. Besides, Nepali pronouns also exhibit direct and oblique forms. The tagset also identifies distinctions based on the gender agreement expressed by the pronouns in Nepali.

3.4. Verbal Inflections in Nepali

Verbs in Nepali are strongly inflected and exhibit highly productive compounding. Nepali verbs inflect to encode aspect, mood, tense, person, honorifics, number, and gender. Verbs in Nepali are very complex; a single verb token can encode different grammatical categories. The NELRALEC tagset contains 29 tags for different grammatical categories of verbs.

One thing to note about the NELRALEC tagset is that finite verbs are not marked for tense, aspect, and modality to simplify the complex verbal inflection system of Nepali verbs. This simplification also keeps POS tags for verbs consistent with the rest of the tagging scheme. Also, the tagging is done for compound verbs based on the last identifiable unit in the compounded verb.

4. Methods

We perform the POS tagging using Bidirectional LSTM (BiLSTM), Bidirectional GRU (BiGRU), and BiLSTM-CRF. We use randomly initialized Bare and Google's multi-lingual BERT embeddings in our experiments.

4.1. Embedding Layer

We represent a given text *(T)* as a word embedding *(e)* using Bare (randomly initialized embedding) and multi-lingual BERT embedding. The embedding layer takes n word embeddings of length d and these embeddings are concatenated in a sequential order as they occur in the text i.e., $T = [e_1 \| e_2 \| ... \| e_n]$, where e_i represents the embedding of i-th word and $e_i \in \mathbb{R}^d$. The input text longer than n is truncated and the text shorter than n is padded with zeros at the end.

4.2. Bidirectional LSTM (BiLSTM)

LSTM[7] is built on the top of traditional RNNs by adding four gates — input gate, output gate, forget gate, and modulation gate to overcome the short-term memory issue of RNNs. These gates control the flow of input (discriminate important input information) and help better capture the context of the given text. These gates are explained with related equations in [16]. BiLSTM is the concatenation of forward LSTM ($\overrightarrow{LSTM}(T)$) and backward LSTM ($\overleftarrow{LSTM}(T)$), where backward LSTMs adds an advantage of capturing future input features along with past information to better capture context.

We fetch the embedding representation of the given text to BiLSTM to get the hidden representation (H). H is represented as $[\overrightarrow{LSTM}(T)||\overleftarrow{LSTM}(T)]$, where $H \in \mathbb{R}^{2h}$, and h is the hidden size of LSTM. The output H is then fed to *Output Layer* to obtain the tag for each word in the given text.

4.3. Bidirectional GRU (BiGRU)

GRU[3] is a variant of LSTM, which also overcomes the short-term memory issue of vanilla RNNs. GRU uses two gates — reset gate and update gate instead of four gates as in LSTM, to capture important information. GRU is simpler in architecture and performs fewer calculations compared to LSTM. However, GRU performs as well as LSTM[3]. BiGRU is concatenation of forward GRU ($\overrightarrow{GRU}(T)$) and backward GRU ($\overleftarrow{GRU}(T)$). The output of BiGRU is then fed to *Output Layer*.

4.4. BiLSTM Conditional Random Field (BiLSTM-CRF)

BiLSTM, followed by a softmax classifier, assigns a tag to each word in a text with conditional independence. However, while assigning tags to a given input sequence, it is important to consider tag syntax. For example, in Nepali text, *Third person non-honorific singular verb (VVYN1)* tag is never followed by *Third person non-honorific singular verb (IA)* e.g, garyo(गर्यो) */IA* is not followed by laaii(लाई)*/VVYN1*. CRF will capture this kind of unlikeness in the tag distribution in a text via a graphical model representation of BiLSTM predictions. In other words, CRF captures dependencies across the output tags of an input text, i.e., CRF uses past and future predicted tags to predict the current tag like BiLSTM using past and future input information to capture context better. We also used BiLSTM-CRF[8] for POS tag labeling, where the CRF model is added on the top of the BiLSTM model.

In BiLSTM-CRF, the output (probability distribution of all unique tags) of BiLSTM followed by the *Output Layer* is fed to the CRF layer. CRF layer will assign the tag sequence based on the highest probability by constructing the transitional probability of output tags.

4.5. Output layer

In the case of BiLSTM and BiGRU model, the output obtained from previous layers is flattened and fed to an output layer of size t with *softmax* activation t represents the number of unique tags in the training set.

5. Experiments

5.1. Data and Tagset Description

We used the Nepali National Corpus (NNC)[17] corpus for all of our experiments. NNC is the largest Nepali POS-tagged corpus with over 17 million manually and semi-manually words tagged with 112 POS-tags, listed in Tables 3. It is a rich corpus with texts collected from various sources including, books, newspapers, and the web.

We combined the core sample (CS contains a Nepali match for the FLOB and Frown Corpora) and general corpus (GC contains texts obtained from books, newspapers, journals and web) of NNC. We found

no instance of tokens with *DJF*, *IKX*, and *MLM* tags in the CS and GS corpus. Hence, there are 109 distinct tags in our dataset.

We split the dataset into train (75%), valid (10%) and test (15%) set. The overall dataset information is provided in Table 1. Tables 3, 4 provides fine-grained and coarse-grained tagset distribution. As specified in Table 3 (using *), three tags specified earlier are missing in training set. In addition to these tags, *VOTN1* and *VOTX2* are missing in validation and testing set, respectively.

Table 1: Data distribution (tokens include words and delimiters)

	train	**valid**	**test**	**overall**
# of sentences	723,106	96,415	144,622	964, 143
# of tokens	13,442,811	1,793,767	2,679,503	17,916,081
# of unique tokens	287,500	99,374	12,4621	511,495
# of unique tags	109	108	108	109
average # of tags/sentence	19	19	19	19

Coarse-grained and fine-grained tagset: To assess the ability of our models to learn fine-grained POS tags and compare it to the models' performances on coarse-grained POS tags, we mapped the 112 fine-grained NELRALEC tags to coarse-grained tags similar to the one in [1]. We grouped the tags and eliminated the distinctions in terms of grades of honorifics and gender for pronouns. Similarly, the distinctions between types and gender forms are eliminated for Pronoun Determiners. Different verb forms for gender, number, honorifics, and gender are removed by mapping the distinct verb forms into a more coarse-grained label. Different categories of foreign words are combined into one *FW*. The complete mapping of the fine-grained tagset to the coarse-grained tagset is given in Table 2.

Table 2: Coarse-grained and fine-grained tagset

Coarse grained	**Fine grained**
NN	NN
NP	NP
IE	IE
IA	IA
CC	CC
MM	MM
JJX	JX
JJD	JT
RR	RR
II	II
IH	IH
MOX	MOX
MLX	MLX
VBI	VI
VBN	VN
PRF	PRF
UU	UU
QQ	QQ
TT	TT
YF	YF
YM	YM
YQ	YQ
YB	YB
FU	FU

Coarse grained	**Fine grained**
FB	FB
CS	CSA, CSB
MOM	MOM, MOF, MOO
FW	FF, FS, FO, FZ
MLM	MLM, MLF, MLO
RRO	RD, RK, RJ
JJM	JM, JF, JO
IKO	IKM, IKO, IKF, IKX
DTX	DDX, DKX, DJX, DGX
VBKO	VDM, VDF, VDO, VDX
PP	PMX, PTN, PTM, PTH, PXH, PXR
VBO	VE, VQ, VCN, VCM, VCH, VS, VR
DTM	DDM, DDF, DKM, DKF, DJM, DJF, DGM, DGF, DDO, DKO, DJO, DGO
VF	VVMX1, VVMX2, VVTN1, VVTX2, VVYN1, VVYX2, VVTN1F, VVTM1F, VVYN1F, VVYM1F, VOMX1, VOMX2, VOTN1, VOTX2, VOYN1, VOYX2
PPP	PMXKM, PMXKF, PMXKO, PTNKM, PTNKF, PTNKO, PTMKM, PTMKF, PTMKO, PRFKM, PRFKF, PRFKO, PMXKX, PTNKX, PTMKX, PRFKX

Table 3: Fine grained tag distribution (* no token(s) with the given tag)

Tag	train (%)	valid (%)	test (%)
NN	28.96	28.99	28.92
II	6.50	6.48	6.49
IKM	6.03	6.01	6.03
JX	5.87	5.87	5.86
YF	4.90	4.89	4.91
VVYN1	4.13	4.11	4.15
VE	3.39	3.39	3.38
YM	3.12	3.16	3.12
MM	2.75	2.76	2.73
IKO	2.66	2.68	2.65
CC	2.48	2.47	2.47
IE	2.41	2.41	2.42
TT	2.37	2.36	2.38
DDX	2.29	2.27	2.30
NP	2.27	2.29	2.26
VI	1.80	1.79	1.80
IH	1.67	1.67	1.68
VN	1.61	1.60	1.59
IA	1.42	1.41	1.43
VVYX2	1.39	1.40	1.38
VQ	1.31	1.29	1.31
RR	1.26	1.26	1.26
YQ	0.98	0.97	0.98
YB	0.66	0.68	0.66
JM	0.60	0.60	0.60
FS	0.52	0.53	0.53
PMX	0.50	0.50	0.50
RD	0.48	0.48	0.49
VDX	0.43	0.43	0.43
DKX	0.42	0.42	0.41
CSA	0.39	0.39	0.39
FB	0.32	0.32	0.33
VVMX1	0.31	0.30	0.31
VDO	0.30	0.30	0.31
MLO	0.23	0.24	0.24
DJX	0.19	0.19	0.19
PRFKM	0.17	0.17	0.18
FO	0.17	0.17	0.17
RK	0.17	0.16	0.16
PMXKM	0.16	0.17	0.17
IKF	0.16	0.16	0.16
PRF	0.15	0.15	0.15
DGX	0.13	0.13	0.14
VOMX2	0.12	0.12	0.12
VVMX2	0.11	0.11	0.11
VVYM1F	0.11	0.11	0.11
MLX	0.11	0.11	0.10
DDM	0.10	0.11	0.11
JO	0.09	0.09	0.09
MOM	0.09	0.09	0.09
CSB	0.08	0.08	0.08
PXH	0.08	0.08	0.08
DGM	0.07	0.07	0.07
FF	0.07	0.07	0.06
VCM	0.07	0.07	0.06
PTH	0.06	0.06	0.06
PTM	0.06	0.06	0.06
DDO	0.0544	0.0531	0.0553
FZ	0.0501	0.0494	0.0496
VOYN1	0.0493	0.0472	0.0478
VS	0.0486	0.0489	0.0497
RJ	0.0472	0.0511	0.0502
VDM	0.0464	0.0459	0.0468
PRFKO	0.0461	0.0455	0.0447
PMXKO	0.0417	0.0404	0.0405
PRFKX	0.0382	0.0379	0.0392
VVYN1F	0.0337	0.0322	0.0340
UU	0.0334	0.0340	0.0338
VVTX2	0.0331	0.0379	0.0357
DKM	0.0271	0.0261	0.0275
VCH	0.0220	0.0224	0.0223
JF	0.0203	0.0209	0.0202
DGO	0.0199	0.0202	0.0210
MOX	0.0189	0.0175	0.0188
PTMKM	0.0176	0.0159	0.0176
MLF	0.0128	0.0140	0.0128
VVTN1	0.0125	0.0138	0.0125
VBMX1	0.0119	0.0119	0.0125
VCN	0.0118	0.0117	0.0114
VR	0.0089	0.0093	0.0079
PMXKF	0.0076	0.0070	0.0077
PTN	0.0065	0.0058	0.0077
JT	0.0061	0.0060	0.0053
PMXKX	0.006	0.0079	0.0056
PXR	0.0057	0.0052	0.0050
DJM	0.0053	0.0055	0.0054
VOYX2	0.0051	0.0041	0.0048
PTNKM	0.0045	0.0049	0.0053
PRFKF	0.0045	0.0041	0.0053
DKO	0.004	0.0043	0.0049
QQ	0.0036	0.0043	0.0040
FU	0.0033	0.0035	0.0038
PTMKO	0.003	0.0028	0.0026
DGF	0.0016	0.0009	0.0015
VDF	0.0015	0.0015	0.0014
PTMKF	0.0012	0.0013	0.0009
VVTN1F	0.0012	0.0014	0.0011
PTMKX	0.0012	0.0013	0.0011
DDF	0.001	0.0014	0.0012
VVTM1F	0.0008	0.0011	0.0008
DKF	0.0007	0.0008	0.0007
PTNKF	0.0007	0.0010	0.0010
PTNKO	0.0006	0.0008	0.0005
DJO	0.0005	0.0004	0.0006
MOO	0.0004	0.0004	0.0004
PTNKX	0.0003	0.0003	0.0004
MOF	0.0002	0.0002	0.0
VOTN1	0.0001	0.0*	0.0001
VOTX2	0.0001	0.0001	0.0*
MLM	0.0*	0.0*	0.0*
DJF	0.0*	0.0*	0.0*
IKX	0.0*	0.0*	0.0*

Table 4: Coarse grained tag distribution

Tag	train (%)	valid (%)	test (%)
NN	28.960	28.994	28.915
IKO	8.843	8.844	8.838
II	6.496	6.481	6.486
VF	6.312	6.296	6.333
JJX	5.869	5.874	5.861
YF	4.898	4.887	4.912
VBO	4.852	4.845	4.849
YM	3.119	3.165	3.121
DTX	3.028	3.011	3.041
MM	2.746	2.759	2.732
CC	2.483	2.468	2.467
IE	2.413	2.406	2.421
TT	2.369	2.355	2.380
NP	2.270	2.292	2.258
VBI	1.798	1.785	1.800
IH	1.670	1.674	1.679
VBN	1.609	1.603	1.591
IA	1.423	1.407	1.428
RR	1.258	1.262	1.259
YQ	0.983	0.967	0.977
FW	0.807	0.814	0.810
VBKO	0.783	0.780	0.786
PP	0.712	0.708	0.712
JJM	0.702	0.712	0.711
RRO	0.692	0.688	0.703
YB	0.657	0.680	0.662
PPP	0.511	0.507	0.520
CS	0.471	0.470	0.465
FB	0.324	0.318	0.330
DTM	0.286	0.290	0.293
MLM	0.245	0.250	0.249
PRF	0.150	0.147	0.151
MLX	0.105	0.107	0.104
MOM	0.087	0.087	0.091
UU	0.033	0.034	0.034
MOX	0.019	0.018	0.019
JJD	0.006	0.006	0.005
QQ	0.004	0.004	0.004
FU	0.003	0.004	0.004

5.2. Experimental Settings

We use two embeddings[5] – BareEmbedding (randomly initialized embedding of dimension d=300) and pre-trained multilingual BERT-base cased (mBERT) embedding ($d = 768$[6]). To use mBERT vocabulary for embedding, we first tokenized the text using mBERT tokenizer. The sequence length n is set to the maximum length of sequences in every batch. The batch size is fixed to 64. We use categorical cross-entropy as a loss function and adopt Adam optimizer [9] with a learning rate of 0.01. For LSTM and GRU, we use a hidden size (h) of 128. We also use a dropout of 0.4 between BiLSTM and successive layers. We evaluated the models using a micro F1-score, and early stopping was applied to the models based on the performance on the validation set. All the models are adapted from Kashgari[7], which are implemented in Keras[4].

6. Results and Discussions

We evaluated the models based on their performance on unknown, known, and all tokens. For the models using Bare embedding, the tokens in the test set that were not present in the training set were considered unknown (UNK), while the rest were known (Known) tokens. There were 44,025 UNK tokens for models using Bare embedding. Since we used the mBERT tokenizer for the models using mBERT embedding, the UNK tokens were chosen based on the mBERT vocabulary. For models using mBERT embedding, there were 25,209 UNK tokens.

The results for each experiment are shown in Table 5. The BiLSTM-CRF model with Bare embedding performed the best for both fine-grained and coarse-grained tagging. Similarly, the performances on the known tokens were best for BiLSTM-CRF with Bare embedding. However, the BiLSTM model with Bare embedding learned the tags for the unknown tokens better. On average, we saw better performances for models using Bare embedding than their mBERT counterparts, i.e., 7.19% -7.80% (on coarse-grained) and 7.31%-7.54% (on fine-grained).

Table 5: Overall performance

Model	Embed	μ F1 (coarse-grained)	μ F1 (fine-grained)
BiGRU	Bare (All)	0.9803	0.9803
	Bare (UNK)	0.7958	0.7932
	Bare (Known)	0.9834	0.9834
	BERT (All)	0.9095	0.9116
	BERT (UNK)	0.7335	0.6993
	BERT (Known)	0.9102	0.9125
BiLSTM	Bare	0.9836	0.9823
	Bare (UNK)	0.8148	0.8002
	Bare (Known)	0.9864	0.9853
	BERT (All)	0.9176	0.9154
	BERT (UNK)	0.7841	0.7847
	BERT (Known)	0.9182	0.9159
BiLSTM-CRF	Bare	0.9848	0.9851
	Bare (UNK)	0.8068	0.7987
	Bare (Known)	0.9878	0.9883
	BERT (All)	0.9147	0.9164
	BERT (UNK)	0.7817	0.7799
	BERT (Known)	0.9152	0.9170

[5] https://kashgari.readthedocs.io/en/v2.0.1/embeddings/
[6] https://github.com/google-research/bert
[7] https://github.com/BrikerMan/Kashgari

Performances on fine-grained vs coarse-grained tagset: The performance of our best model (BiLSTM-CRF model with Bare embedding) on fine-grained tagset is slightly better than that on coarse-grained tagset by 0.03%. Comparable F1 scores were observed for BiGRU and BiLSTM for coarse-grained and fine-grained tagging tasks. The BiLSTM model gave slightly better F1 scores for coarse-grained tagging except when tagging unknown tokens using mBERT embeddings. Overall, we observed that the models gave similar F1 scores for both fine-grained and coarse-grained tagging. Hence, with a large enough corpus, deep-learning best models capture fine-grained distinctions of word forms as good as the coarse-grained.

Effect of varying embeddings: The models performed better for the randomly initialized and trained Bare embedding than for the pre-trained mBERT embedding. While mBERT embedding has been proven to be great for cross-lingual generalization, our results show for a more downstream task like POS tagging for Nepali, training a randomly initialized Bare embedding gives better results. This indeed give a strong motivation to further explore the mBERT vocabulary and embedding specific to Nepali language.

For a model trained using mBERT embedding, word pieces of the words are marked as unknown (UNK) if the word pieces are not in the subwords represented by the model. So, the overall performance is similar to the one where we considered known tokens only. However, the performance on the unknown tokens is 14.32% - 23.36% (across coarse-grained and fine-grained) less than the performance on known tokens.

6.1. Tag wise performance

This section will discuss the tag-wise performance for our best performing model, BiLSTM-CRF (with Bare embedding). We will see how the performance of this model on the coarse-grained task compares to that on fine-grained tagging.

Table 6: Coarse-grained tag-wise performance for BiLSTM-CRF with Bare Embedding

Tag	**P**	**R**	**F1**	**Tag**	**P**	**R**	**F1**	**Tag**	**P**	**R**	**F1**
YB	1.0	1.0	1.0	MM	0.9964	0.9883	0.9923	TT	0.9795	0.9741	0.9768
IA	0.9999	0.9999	0.9999	VBN	0.9927	0.9913	0.9920	JJX	0.979	0.9655	0.9722
IE	0.9999	0.9999	0.9999	II	0.9924	0.9909	0.9916	RRO	0.9611	0.9797	0.9703
YQ	1.0	0.9998	0.9999	DTX	0.9922	0.9879	0.9901	JJD	0.9853	0.9504	0.9675
PPP	0.9999	0.9998	0.9998	FB	0.9893	0.9872	0.9883	NP	0.9664	0.9482	0.9572
IH	0.9987	1.0	0.9994	PP	0.9889	0.9864	0.9877	VBKO	0.9893	0.9132	0.9498
YF	0.9984	0.9998	0.9991	MOM	0.9801	0.9926	0.9863	CS	0.9539	0.9412	0.9475
PRF	0.9988	0.9993	0.9990	VBO	0.9910	0.9750	0.9830	RR	0.9588	0.9345	0.9465
YM	0.9998	0.9974	0.9986	MOX	0.9939	0.9722	0.9829	JJM	0.9870	0.9077	0.9457
IKO	0.9976	0.9991	0.9984	DTM	0.9750	0.9910	0.9829	UU	0.9336	0.9449	0.9392
MLM	0.9961	0.9997	0.9979	NN	0.9738	0.9901	0.9819	FW	0.9830	0.8860	0.9320
MLX	0.9996	0.9896	0.9946	VF	0.9760	0.9811	0.9785	FU	1.0	0.8333	0.9091
CC	0.9917	0.9954	0.9935	VBI	0.9937	0.9607	0.9769	QQ	0.8317	0.7778	0.8038

Table 7: Fine-grained tag-wise performance for BiLSTM-CRF with Bare Embedding

Tag	P	R	F1	Tag	P	R	F1	Tag	P	R	F1
MOF	1.0	1.0	1.0	DDX	0.9964	0.9979	0.9971	CSB	0.9863	0.9380	0.9615
PTNKF	1.0	1.0	1.0	DGX	0.9945	0.9997	0.9971	CSA	0.9463	0.9768	0.9613
PMXKO	1.0	1.0	1.0	DGO	0.9947	0.9964	0.9956	JF	0.9698	0.9500	0.9598
PTMKM	1.0	1.0	1.0	PMX	0.9919	0.9988	0.9954	NP	0.9592	0.9543	0.9568
PMXKX	1.0	1.0	1.0	MLX	1.0	0.9907	0.9953	RK	0.9639	0.9431	0.9534
PRFKF	1.0	1.0	1.0	PTH	0.9929	0.9974	0.9951	RR	0.9542	0.9514	0.9528
PTM	1.0	1.0	1.0	CC	0.9947	0.9938	0.9943	VDO	0.9877	0.9197	0.9525
MLF	1.0	1.0	1.0	DJX	0.9953	0.9910	0.9931	VOYN1	0.9769	0.9251	0.9503
DDF	1.0	1.0	1.0	II	0.9924	0.9937	0.9931	VCN	0.9964	0.9052	0.9486
PTMKX	1.0	1.0	1.0	PMXKF	0.9904	0.9952	0.9928	VDX	0.9957	0.9034	0.9473
DGF	1.0	1.0	1.0	MM	0.9976	0.9879	0.9927	JM	0.9890	0.8997	0.9422
PTMKF	1.0	1.0	1.0	DKO	1.0	0.9846	0.9922	UU	0.9175	0.9680	0.9421
PTMKO	1.0	1.0	1.0	VN	0.9921	0.9921	0.9921	VBMX1	0.9868	0.8896	0.9356
YQ	1.0	1.0	1.0	VE	0.9932	0.9858	0.9895	VVYM1F	0.9916	0.8815	0.9333
PTNKX	1.0	1.0	1.0	IKF	0.9878	0.9908	0.9893	VVMX1	0.9874	0.8847	0.9332
DKF	1.0	1.0	1.0	FB	0.9898	0.9885	0.9891	VVYN1F	0.9543	0.8946	0.9235
PTNKM	1.0	1.0	1.0	MOM	0.9781	0.9983	0.9881	FZ	0.9764	0.8728	0.9217
PTNKO	1.0	1.0	1.0	RJ	0.9970	0.9770	0.9869	VVMX2	0.9907	0.859	0.9202
IKO	1.0	1.0	1.0	VDF	1.0	0.9737	0.9867	VDM	0.9615	0.8571	0.9063
YB	0.9999	1.0	1.0	FS	0.9963	0.9770	0.9865	FU	0.9659	0.8333	0.8947
IE	1.0	0.9999	0.9999	MOX	0.9919	0.9781	0.985	FF	0.9138	0.8763	0.8947
IA	0.9999	0.9999	0.9999	TT	0.9850	0.9824	0.9837	VCM	0.9601	0.8257	0.8878
PMXKM	1.0	0.9998	0.9999	RD	0.9806	0.9840	0.9823	QQ	0.9767	0.7778	0.8660
IKM	0.9995	0.9998	0.9997	NN	0.9730	0.9912	0.9820	VS	0.8995	0.786	0.8389
PRFKO	1.0	0.9992	0.9996	VQ	0.9893	0.9669	0.9780	VCH	0.9393	0.7508	0.8346
PRFKM	0.9992	1.0	0.9996	VVYN1	0.9728	0.9822	0.9775	VVTX2	0.9515	0.7395	0.8323
IH	0.9990	1.0	0.9995	VI	0.9927	0.9607	0.9764	VVTN1	0.9912	0.6766	0.8043
YF	0.9994	0.9996	0.9995	VVYX2	0.9876	0.9655	0.9764	VVTM1F	0.7895	0.7143	0.7500
DGM	1.0	0.9989	0.9995	VOYX2	1.0	0.9535	0.9762	DJM	0.8762	0.6389	0.7390
DKM	1.0	0.9986	0.9993	JX	0.9817	0.9655	0.9735	FO	0.9391	0.5999	0.7321
DDM	1.0	0.9986	0.9993	DKX	0.9762	0.9691	0.9726	VVTN1F	1.0	0.5333	0.6957
YM	0.9993	0.9990	0.9992	JO	0.9851	0.9581	0.9714	PTN	0.7727	0.5805	0.6630
PRF	0.9990	0.9993	0.9991	VOMX2	0.9895	0.9509	0.9698	VR	0.9626	0.4882	0.6478
DDO	0.9987	0.9987	0.9987	PXR	1.0	0.9398	0.9690	VOTN1	0.0	0.0	0.0
MLO	0.9964	0.9998	0.9981	JT	0.9853	0.9504	0.9675	DJO	0.0	0.0	0.0
PRFKX	1.0	0.9952	0.9976	PXH	0.9737	0.9537	0.9636	VOTX2	0.0	0.0	0.0
								MOO	0.0	0.0	0.0

Performance of tags with respect to tag distribution: The BiLSTM-CRF with Bare embedding captured fine-grained POS information in text well. Table 7 shows the tag-wise performance of the BiLSTM-CRF with Bare embedding. As shown in the table, the models have high F1 scores across many of the tags. 42/109 tags have an F1 score of 1.0. Similarly, 14/109, 35/109, 8/109 have a F1 score of 0.99, [0.90-0.99), and [0.80-0.90), respectively. Out of the remaining ten tags, 6 of them are above 0.65, and the rest have 0.0 F1. The four tags that had the F1 score of 0.0 were *VOTN1*, *DJO*, *VOTX2* and *MOO*. They were also the tags with low distribution in the training set (0.0001% - 0.0004%). Note that we exclude the tag not present in the training and testing set in the evaluation.

The tag *NN* with the highest distribution (28.96%) in training set (Table 3) is ranked 41 (F1 of 0.982)[8] in tag wise performance (Table 7). Likewise, tag *II* with second highest (6.5%) is ranked 26 (F1 of 0.993).

[8] Note that we provide rank 1 to the tag with highest F1 and higher the rank value indicates low performance of the tag.

Interestingly, except for the four tags with a 0.0 F1 score discussed earlier, several tags with low occurrences in the dataset were predicted correctly and ranked higher (closer to rank 1). Some of such tags are MOF (Rank:1, F1:1.0, train: 0.0002%), PTNKX (Rank:1, F1:1.0, train: 0.0003%), and PRFKO (Rank:6, F1:0.9996, train:0.0461%). These accurate predictions of low occurring tags indicate that tag-wise performance does not correlate with tag distribution in the training set.

Tag-wise performance on fine-grained tags vs their coarse-grained counterpart: As mentioned in the previous section, the BiLSTM-CRF model with Bare embedding performs the best for coarse and fine-grained tagging. Also, the overall performance for fine-grained tagging is better than that for coarse-grained tagging. We observed a similar performance for the tags in the fine-grained set and their corresponding coarse-grained tags (as shown in Table 2. The F1 scores for tags that were same across both the tagsets, like *NN*, *NP*, *IE*, and so on, were similar. One exception, however, was the QQ tag for which its tag-wise F1 score for fine-grained was higher than that for the coarse-grained experiment by about 6%.

As for the fine-grained tags that were grouped the F1 scores for many of them were within the range of 2% to their corresponding coarse-grained tags. For example, 9 out of 12 fine-grained tags corresponding to the *DTM* tag from coarse-grained tagset has similar performance (*DTM* in coarse: 0.98 F1, corresponding fine-grained tags: 0.99 - 1.0 F1) with two tags *DJM* and *DJO* underperforming and one of them (*DJF*) absent in the testing set. As an another example, 5 out of 6 fine-grained tags corresponding to the tag *PP* from coarse-grained have similar performances (fine-grained: *PTM*, *PMX*, *PTH* (0.99-1.0 F1), *PXR*, *PXH* (0.96-0.97 F1), coarse: *PP* (0.988 F1)) with one exception *PTN* (0.66 F1). Similar results were observed for the other tags of coarse-grained tags and their corresponding fine-grained tags.

7. Conclusion and Future Works

In this work, we applied three deep learning models for fine-grain POS tagging for the Nepali language. We showed that deep learning models could capture fine-grained morphological features like gender, person, number, and honorifics that are encoded within words in highly inflectional languages like Nepali with a large enough dataset. Our work reports a new state of the art for fine-grained Nepali POS tagging. While our models achieved great results on fine-grained tags, there are a few tags that the models fail to learn accurately. In future work in this line, we would like to study these errors and employ techniques to mitigate them.

We also experimented with randomly initialized Bare embeddings and compared its results to the new mBERT embeddings for Nepali POS tagging. We showed that training a much smaller randomly initialized embedding can be more useful for fine and coarse-grained tagging in Nepali. We want to explore the mBERT vocabulary and the embedding space for the Nepali language in future work.

References

[1] Bal, B.K., 2009. Towards building advanced natural language applications-an overview of the existing primary resources and applications in Nepali, in: Proceedings of the 7th Workshop on Asian Language Resources (ALR7), pp. 165–170.
[2] Bal, B.K., Shrestha, P., Pustakalaya, M.P., 2004. Nepali Spellchecker. PAN Localization Working Papers 2007, 316–318.
[3] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 .
[4] Chollet, F., et al., 2015. Keras. URL: https://github.com/fchollet/keras.
[5] Hardie, A., 2005. Automated part-of-speech analysis of Urdu: conceptual and technical issues. .
[6] Hardie, A., Lohani, R., Yadava, Y., 2011. Extending corpus annotation of nepali: advances in tokenisation and lemmatisation. Himalayan Linguistics 10, 151–165.
[7] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.
[8] Huang, Z., Xu, W., Yu, K., 2015. Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991 .
[9] Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization. ICLR .
[10] Paul, A., Purkayastha, B.S., Sarkar, S., 2015. Hidden Markov model based part of speech tagging for Nepali language, in: 2015 International Symposium on Advanced Computing and Communication (ISACC), IEEE. pp. 149–156.

[11] Prabha, G., Jyothsna, P., Shahina, K., Premjith, B., Soman, K., 2018. A deep learning approach for part-of-speech tagging in Nepali language, in: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE. pp. 1132–1136.
[12] Prasain, B., 2011. Computational analysis of nepali morphology: A model for natural language processing. Ph.D. thesis. Faculty of Humanities and Social Sciences of Tribhuvan University.
[13] Shahi, T., 2012. Support Vector Machine Based POS Tagging For Nepali Text. Ph.D. thesis. Masters Dissertation, Central Department of Computer Science and IT 2012 ….
[14] Shrestha, I., Dhakal, S.S., 2016. A new stemmer for Nepali language, in: 2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Fall), IEEE. pp. 1–5.
[15] Sinha, P., Veyie, N.M., Purkayastha, B.S., 2015. Enhancing the performance of part of speech tagging of nepali language through hybrid approach. International Journal of Emerging Technology and Advanced Engineering 5, 354–359.
[16] Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., Saenko, K., 2014. Translating videos to natural language using deep recurrent neural networks. arXiv preprint arXiv:1412.4729 .
[17] **[dataset]** Yadava, Y.P., Hardie, A., Lohani, R.R., Regmi, B.N., Gurung, S., Gurung, A., McEnery, T., Allwood, J., Hall, P., 2008. Construction and annotation of a corpus of contemporary Nepali. Corpora 3, 213–225.
[18] Yajnik, A., 2017. Part of speech tagging using statistical approach for Nepali text. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering 11, 76–79.
[19] Yajnik, A., 2018. ANN Based POS Tagging For Nepali Text. International Journal on Natural Language Computing 7, 13–18.