Development of a Dataset and Deep Learning Models for Named Entity Recognition and Joint Word Segmentation in Sanskrit

Thesis to be submitted in fulfillment of the requirements for the degree

of

Master of Technology

by

Razorshi Prozzwal Talukder
21CS60A03

Under the guidance of

Prof. Pawan Goyal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Student Department
Indian Institute of Technology,
Kharagpur
India - 721302

CERTIFICATE

This is to certify that we have examined the thesis entitled **Neural Model for Joint Learning of Named Entity Recognition and Word Segmentation in Sanskrit**, submitted by **Razorshi Prozzwal Talukder** (Roll Number: *21CS60A03*) a postgraduate student of **Department of Computer Science and Engineering** in fulfillment for the award of the degree of M.Tech. We hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the Post Graduate Degree for which it has been submitted. The thesis has fulfilled all the requirements as per the regulations of the Institute and has reached the standard needed for submission.

Pawan Goyal

Prof. Pawan Goyal

Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology,
Kharagpur

Place: Kharagpur
Date:

ACKNOWLEDGEMENTS

I express my sincere gratitude to Sir **Pawan Goyal**, Associate Professor, Department of Computer Science Engineering, Indian Institute of Technology, Kharagpur for his supervision and guidance during the project. His valuable advice, encouragement, suggestions, and friendly attitude toward me, during this work, helped a long way in bringing this project report to this stage. I am also thankful to him for extending all kinds of help and for providing the necessary facilities required during this work.

I would like to express my special thanks to my mentor **Sujoy Sarkar** for continuous and extensive support.

Razorshi Prozzwal Talukder
IIT Kharagpur
Date:

Contents

List of Figures

Chapter 1

Abstract

Named Entity Recognition (NER) is a fundamental task in Natural Language Processing (NLP) that involves identifying and categorizing named entities in text into pre-defined categories such as person names, locations, and organizations. While this task has been studied extensively for many languages, there are still challenges when it comes to low-resource languages such as Sanskrit. Sanskrit, an ancient Indian language, poses unique challenges in NER due to its complex morphology, lack of standardization, and word segmentation issues. In this paper, we create a comprehensive annotated dataset for Named Entity Recognition (NER) and Word Segmentation tasks in Sanskrit and evaluate the performance of existing NER models on this dataset. The main contribution of this work is the classification of 10K names from Mahabharata into three categories, namely PER, LOC, and MISC. Various Deep Learning models based on Bidirectional LSTM, CNN, and CRF with different embeddings at the token and character level, are applied to the dataset. The results show that the best F score of 85.5 is achieved by using the fastText embedding model in character embedding mode in CNN with random data split. As our dataset is based on a single book there are many repeated mentions. So we have also partitioned the data such that there is less overlap between the train and test sets on which the Best F Score is 55.71 is achieved. A joint training model is also proposed and developed. The NER and Segmentation combined corpus contain a total of 6.3L tokens, out of which 1.4L tokens are in Sandhied form. The percentage of tokens that contain entities is 10.13% of the total tokens and 25.52% of the total entities are Sandhied in other words. This highlights the importance of developing joint models for both NER and segmentation tasks.

Chapter 2

Introduction

The challenge of locating and classifying named things in a text is crucial to natural language processing. Named entities are words or phrases that stand in for the names of individuals, groups, locations, dates, or numbers. The process of recognizing proper nouns in the text and categorizing them into predetermined groups is known as named entity recognition. Sanskrit, an Indian language with a rich morphology compared to other languages, has received less attention than English and other European languages from researchers working on NER. Low-resource languages like Sanskrit, however, lack sufficient annotated data, and human annotation is time- and money-consuming. It would be appealing, especially for languages with limited resources, to have a sequence labeling tool that enables human annotation while also providing automatic suggestions to speed up annotation and enhance productivity. Sanskrit, for example, has a rich morphological heritage, yet the majority of tools with these capabilities have been created for Western European languages, which have more available resources [1]. The goal of this effort is to produce a thoroughly annotated dataset for Sanskrit word segmentation and named entity recognition (NER) tasks, as well as to test the effectiveness of existing NER models on the Mahabharata dataset. Here we can see an example from Volume 1 (Adi Parva), Chapter 1 (Adivansavatarana Parva), Verse 48.

kvasthāḥ vīrāḥ pāṇḍavāḥ te babhūvuḥ kutas ca etat śrutavantaḥ priyam te janamejayaḥ uvāca kvasthāḥ vīrāḥ pāṇḍavāḥ te babhūvuḥ

Here, *pāṇḍavāḥ* and *janamejayaḥ* both are Name Entity. The impact of joint learning of Word Segmentation and NER tasks is studied. The Classification of the

10K names from the Mahabharata into the three categories of Person, Location, and Miscellaneous is the main contribution of this work. Using two distinct Mahabharata editions that differ by chapters, verses, and even words, annotated data for NER and Word Segmentation tasks are prepared programmatically. By presenting the assignment as the shortest path to finding the issue in a graph, the goal is to identify a mapping between the two corpora. Segmentation, POS, and NER data are compiled from the data in both corpora to build a single corpus. Different Deep Learning models, such as Bi-LSTM, CNN, and CRF, were employed with various embeddings at the token and character level to train a dataset. Among these models, the one with fastText embedding in character embedding mode in CNN with random data split. However, as the dataset was based on a single book, there were many repeated mentions, so the data was partitioned to reduce overlap between the train and test sets. Additionally, a joint training model was proposed and developed that used token representation for the NER task and character representation for the segmentation task, with CRF used at the last layer in both cases. The combined corpus for NER and Segmentation contained 6.3L tokens, of which 1.4L tokens were in Sandhied form. Of all the tokens, only 10.13% contained entities, and among these entities, 25.52% were Sandhied in other words. These findings indicate the importance of creating joint models for NER and Segmentation tasks. The study contributes to NER for low-resourced languages, particularly Sanskrit, and highlights the effectiveness of joint learning for Word Segmentation and NER tasks. The results offer insights into the performance of various models and embeddings, and the suggested joint training model demonstrates promising results.

Chapter 3

Literature Survey

- **Bidirectional LSTM-CRF Models for Sequence Tagging [2]**
  The paper "Bidirectional LSTM-CRF Models for Sequence Tagging" proposes a novel approach to sequence tagging, which combines bidirectional Long Short-Term Memory (LSTM) neural networks with Conditional Random Fields (CRF) in a unified framework. The authors claim that their model achieves state-of-the-art performance on several benchmark datasets. The authors start by describing the problem of sequence tagging, which involves assigning a label to each element in a sequence of data (e.g., part-of-speech tagging or named entity recognition). They then explain the limitations of previous approaches, such as hidden Markov models (HMMs) and maximum entropy Markov models (MEMMs), which rely on hand-crafted features and struggle with long-term dependencies.

  The authors introduce their proposed model, which consists of a bidirectional LSTM layer followed by a CRF layer. The bidirectional LSTM layer captures both forward and backward context, allowing the model to make better use of long-term dependencies. The CRF layer models the dependencies between adjacent labels, further improving performance.

  The authors then describe their experimental setup, where they evaluate their model on three benchmark datasets: CoNLL-2000 chunking, CoNLL-2003 named entity recognition, and the BioCreative IV chemical-disease relation (CDR) task. They compare their model to several baselines, including HMMs and MEMMs, as well as other neural network models such as the basic LSTM and the LSTM-CRF. The results show that their model outperforms all baselines on all three datasets, achieving state-of-the-art performance.

The authors also conduct an ablation study to investigate the contributions of the bidirectional LSTM and the CRF layers. They find that both components are important for achieving high performance, with the bidirectional LSTM layer contributing more to performance gains on the CoNLL datasets and the CRF layer contributing more to gains on the CDR dataset.

In conclusion, the paper "Bidirectional LSTM-CRF Models for Sequence Tagging" proposes a novel approach to sequence tagging that combines bidirectional LSTMs and CRFs in a unified framework. The authors demonstrate that their model achieves state-of-the-art performance on several benchmark datasets and shows the importance of both the bidirectional LSTM and CRF layers.

- **Sanskrit Word Segmentation Using Character-level Recurrent and Convolutional Neural Networks [3]**
  The paper "Sanskrit Word Segmentation Using Character-level Recurrent and Convolutional Neural Networks" proposes a deep learning approach for Sanskrit word segmentation. The authors argue that due to the complex morphology and the absence of word boundaries in Sanskrit, word segmentation is a challenging task that requires advanced machine-learning techniques.

  On paper, most previous approaches for Sanskrit word segmentation relied on rule-based methods or on the use of external resources such as dictionaries or morphological analyzers. However, they argue that such approaches have limitations, such as being unable to handle unseen words or being computationally expensive.

  The authors then introduce their proposed method, which is based on recurrent and convolutional neural networks (RNNs and CNNs, respectively). Their approach consists of two phases: a training phase, in which the model is trained on a large corpus of Sanskrit texts, and an inference phase, in which the model is used to segment new texts.

  The authors evaluate their method on a benchmark dataset for Sanskrit word segmentation and compare it to several baseline methods, including rule-based methods and machine learning methods based on n-gram models or conditional random fields (CRFs). Their results show that their proposed method outperforms the baselines in terms of precision, recall, and F1 score.

The authors conclude that their approach represents a significant improvement over previous methods for Sanskrit word segmentation and that it has the potential to be applied to other tasks in Sanskrit natural language processing (NLP). They also discuss some limitations of their approach, such as the fact that it relies on a large training corpus and that it may not generalize well to other Sanskrit texts.

Overall, the paper provides a detailed and well-structured literature survey on Sanskrit word segmentation and related work, as well as a clear and rigorous presentation of the proposed method and its evaluation. The paper makes a valuable contribution to the field of Sanskrit NLP and demonstrates the potential of deep learning techniques for handling complex morphological structures in languages with rich morphology.

- **Neural Chinese Named Entity Recognition via CNN-LSTM-CRF and Joint Training with Word Segmentation** [4]
  The paper "Neural Chinese Named Entity Recognition via CNN-LSTM-CRF and Joint Training with Word Segmentation" proposes a novel approach for Chinese named entity recognition (NER) that combines convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and conditional random fields (CRFs) with joint training of word segmentation.

  The authors begin by introducing the importance of NER and the challenges specific to the Chinese language, such as the lack of clear word boundaries and the high degree of ambiguity in Chinese characters. They then review previous work in Chinese NER, highlighting the limitations of rule-based and feature-based methods, and the success of neural network-based approaches.

  The proposed model consists of three main components: a CNN layer for feature extraction, an LSTM layer for context modeling, and a CRF layer for decoding the output sequence. The authors also incorporate joint training of word segmentation, which they argue can improve NER performance by better capturing the underlying structure of the Chinese language.

To evaluate their model, the authors conduct experiments on two publicly available Chinese NER datasets and compare their results to several baseline models. They find that their model achieves state-of-the-art performance on both datasets, demonstrating the effectiveness of their approach.

The authors conclude by discussing the implications of their findings, such as the potential for their model to be applied to other languages with similar characteristics to Chinese, and the importance of joint training with word segmentation for improving NER performance.

Overall, this paper provides a comprehensive literature survey of Chinese NER and proposes a novel approach that achieves state-of-the-art performance through the combination of CNNs, LSTMs, and CRFs with joint training of word segmentation.

- **Enhancing Entity Boundary Detection for Better Chinese Named Entity Recognition [5]**
  The paper "Enhancing Entity Boundary Detection for Better Chinese Named Entity Recognition" aims to address the challenges in Chinese named entity recognition (NER), particularly in entity boundary detection. The authors highlight that the identification of entity boundaries is crucial in achieving high accuracy in Chinese NER, as Chinese is a character-based language, making it difficult to distinguish between named entities and non-named entities.

  The paper provides a comprehensive review of related works in Chinese NER, with a focus on entity boundary detection. The authors discuss the different approaches used in previous studies, including rule-based, dictionary-based, and machine learning-based methods. They note that rule-based and dictionary-based methods are limited in their ability to handle complex named entities and variations in their representations. On the other hand, machine learning-based methods have shown promising results, particularly those using deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

  The authors introduce their proposed model, which utilizes a combination of CNN and RNN to enhance entity boundary detection in Chinese NER. They also introduce a novel character-level encoding method that captures the semantic and syntactic features of Chinese characters, allowing the model to effectively distinguish between named and non-named entities. The paper provides a detailed description of the model architecture and the experimental setup used to evaluate its performance.

The authors evaluate their model on two widely used Chinese NER datasets and compare it with state-of-the-art models. The experimental results show that their model outperforms existing models in terms of entity boundary detection accuracy, achieving an F1-score of 94.57% and 93.31% on the two datasets, respectively.

Overall, the paper provides a valuable contribution to the field of Chinese NER, particularly in entity boundary detection. The proposed model and character-level encoding method demonstrate significant improvements in accuracy, which can be beneficial in various applications such as information extraction and natural language processing.

- **OPDAI at SemEval-2022 Task 11: A hybrid approach for Chinese NER using outside Wikipedia knowledge** [6]

  The paper titled "OPDAI at SemEval-2022 Task 11: A hybrid approach for Chinese NER using outside Wikipedia knowledge" presents a novel hybrid approach for Chinese Named Entity Recognition (NER) using outside Wikipedia knowledge. The authors aim to improve the performance of existing NER models by leveraging external knowledge sources.

  The paper starts by providing an overview of the existing methods for Chinese NER, which are mostly based on deep learning techniques such as BiLSTM-CRF and BERT. However, these models often suffer from data sparsity and lack of contextual information, which results in poor performance.

  To address these issues, the authors propose a hybrid approach that combines the strengths of existing models with external knowledge sources from Wikipedia. Specifically, they use the Wikipedia API to extract information related to entities mentioned in the text, such as their categories, related entities, and descriptions. This information is then incorporated into the model as additional features, which improves the contextual understanding of the model.

  The authors also conducted experiments on the Chinese NER dataset provided by SemEval-2022 Task 11, and the results demonstrate that their approach outperforms existing state-of-the-art models in terms of the F1-score. The authors also performed ablation studies to investigate the contribution of external knowledge sources to the performance of the model.

Overall, the paper provides a comprehensive overview of the existing methods for Chinese NER and proposes a novel hybrid approach that leverages external knowledge sources to improve the performance of existing models. The experimental results demonstrate the effectiveness of the proposed approach and highlight the importance of incorporating external knowledge sources for NER tasks.

Chapter 4

Motivation

A great deal of accurate work has been done in English and other foreign languages like Spanish, Chinese, etc., study in Indian languages is still in its early stages. For European languages, particularly English and East Asian languages, accurate NER systems are already accessible. The NER issue is still very much a problem for South and South East Asian languages. The nature of the problem differs for Indian languages due to a number of factors. For instance, in the European language, when a substantial percentage of first names are not used as common words, there are many regularly used words (common nouns) that can also be used as names (proper nouns) [7].

Named Entity Recognition (NER) is an important task in Natural Language Processing (NLP), which involves identifying and categorizing named entities in text into predefined categories such as person names, organization names, location names, etc. In the case of the Sanskrit language, NER is important because Sanskrit has a rich literary heritage and is considered to be the language of the ancient Indian scriptures such as the Vedas, Puranas, and Upanishads. These texts contain a wealth of information about ancient Indian society, culture, and religion, and named entities play an important role in identifying and understanding this information. In the context of ancient Indian history, NER can be used to identify the names of historical figures, kingdoms, and dynasties mentioned in the texts. Similarly, in the context of religion, NER can be used to identify the names of gods, goddesses, and other mythical figures mentioned in the texts. NER can help in unlocking valuable information from Sanskrit texts and making it accessible to scholars and researchers, thus facilitating a deeper understanding of ancient Indian society, culture, and religion.

Generally in the English language, there are different words in sentences. In the Sanskrit language, sometimes create a new word from two or three words combined together. Which meaning is also different. In that case, identifying NER is very difficult. For solving that issue we will approach three different methods Phonemic changes (Sandhi) obscure the word boundaries in a sentence by combining multiple words, resulting in long and complex word forms. A named entity may also get combined with other words. For example, 'śrībrahmovāca' translates to 'Śrī Brahmā said'. Here Brahmā is a named entity that is easily identifiable in English as it is present as a separate word. Also, the capitalization of the NE is a major feature used by NER systems for European languages. Even multiple NEs may be present in a single word. For example, the word 'sakarṇaduḥśāsanasaubalānāṁ' contains three NEs: 'karṇa, duḥśāsana, saubalānām'. So to prepare a dataset one needs to segment the sandhi words in a corpus. However, word segmentation in Sanskrit is not a trivial task as the number of possible segments for a given sentence can be very large. Segmentation tools are available in popular computational Sanskrit platforms like Sanskrit Heritage Reader (SHR), but human intervention is required to decide the final solution (Goyal et al., 2012) [1].

Joint Model is needed for NER detection because Sanskrit is a highly inflected language with complex grammatical structures. This makes it difficult to accurately identify named entities based on surface-level features alone. In our work, Joint Model is very needed for Segment word and NER detection in the Sanskrit language.

Chapter 5

Objective

The objective of this project is to create a comprehensive dataset with annotations for Word Segmentation and Named Entity Recognition (NER) tasks in Sanskrit. With the use of this dataset, the effectiveness of current NER models is assessed. Two separate Mahabharata editions with various chapters, verses, and even words are used to create the NER and Word Segmentation annotations programmatically. The issue is tackled by using the shortest path problem in a graph to identify a mapping between the two corpora. Segmentation, POS, and NER data are compiled from the data in both corpora to build a single corpus. We obtained information for this research from two websites, Itihasa and the Digital Corpus of Sanskrit (DCS), which had a total of 18 volumes and 109,712 mentions, and 73,590 poems in 73,590 verses. We used a segmenter model that combines RNN and CNN to segment the data from DCS and Itihasa. By concurrently dividing compounds and resolving phonetic merges (Sandhi), this technique tokenizes Sanskrit. Following segmentation, the data was manually divided into three classes: Person, Location, and Miscellaneous. We filtered the data and changed the Sanskrit script from IAST format to SLP format in order to obtain accurate matching. The similarity between the DCS and Itihasa data was then determined. Next, we identified Named Entities (NER) from the dataset using our BiLSTM and CRF model. This model relies on character-based word representations learned from the supervised corpus and unsupervised word representations obtained from unannotated corpora as two sources of knowledge about words. Three different word embedding models (GloVe, Albert, and fastText) were used to train the dataset. Finally, we suggested a new CNN-LSTM-CRF-based Join model for the Sanskrit language. The main goal was to separate words and find NER in the input.

Chapter 6

Scope and Challenges

There are so many scopes and challenges. For lack of dataset, we are working on only Mahabharat text and get only verses from data. There are no modern Sanskrit data. Sanskrit has complex grammar and morphology with a large number of inflections [8], which makes it difficult to identify named entities. The meaning of a word in Sanskrit can change depending on its context, making it harder to accurately identify named entities. Sanskrit has a large vocabulary, and many proper names are not commonly used in everyday language. This means that training data for NER may be limited [9]. There is a lack of standardized annotation guidelines for NER in Sanskrit, which makes it harder to develop and evaluate NER models. Sanskrit texts often contain compound words, which can be challenging to parse and analyze for NER.

Although the study is to provide a toolset that would make it easier to conduct NLP research on South Asian languages with limited resources. Another goal is to offer a complete toolset for low-resource South Asian language NLP tasks such as tokenization, part-of-speech tagging, named entity recognition, dependency parsing, and machine translation [10]. Four languages—Nepali, Maithili, Bhojpuri, and Awadhi—are among the least resourced in South Asia. So, there are so many similar problems in other languages [11]. MT for English, Hindi, and Sanskrit languages, including differences in grammar, syntax, and vocabulary between languages. The authors note that the performance of MT systems can be affected by the complexity of the target language's morphology and syntax. The lack of parallel corpora and language resources for low-resource languages like Sanskrit is also a significant challenge in MT research [12]. Indian epics are subjected to NER and entity classification. It can be challenging to create a uniform strategy for NER and entity categorization due to the variance in language usage across different geographic areas and historical periods, which can cause inconsistencies in the text. Multiple names and titles for the same thing, as well as the usage of honorifics and epithets, can reduce accuracy by adding noise and ambiguity to the dataset. It can be challenging to create models with high accuracy because Indian epics are fictional narratives with numerous characters and events that may have several uncommon and unique entities that are absent from other books. The lack of annotated data for Indian epics poses a challenge in developing and evaluating ML models for NER and entity classification tasks [13].

Overall, NER in Sanskrit is a challenging task that requires a deep understanding of the language's grammar, morphology, and vocabulary. While there are many potential applications for NER in Sanskrit, such as analyzing ancient texts or developing natural language processing tools for modern Sanskrit, more research and development are needed to overcome these challenges[14].

Chapter 7

Dataset

7.1 Existing Dataset

For detecting NER from the Sanskrit Language we have used the Itihasa dataset. Entities are not classified and entity boundaries are not identified.

Total no of verses	73590
Total number of mentions	109712
Unique entities	12636
Total verses without entities	25198

Table 7.1: Itihasa Dataset Details

7.2 Manual Classification of Mahabharata Names

The existing dataset conatins around 10K unique entity names from Mahabharata. These names descriptions are obtained from a dictionary and based on the description manually we have classified them.

Pearson	**Location**	**Miscellaneous**
9534	599	532

Table 7.2: Number of Manual Annotated Data

Person	**Location**	**Miscellaneous**
100946	4479	4287

Table 7.3: Data from Mentions

7.3 NER Dataset by Applying Neural Word Segmentor

We applied the Sanskrit word segmenter for segmenting our Itihasa dataset. It is end-to-end neural network models that tokenize Sanskrit by jointly splitting compounds and resolving phonetic merges (Sandhi) [15].

Total no of tokens (before segmentation)	987868
Total no of tokens (after segmentation)	1363095
Token contains a single entity name and is not segmented	63049

Table 7.4: Statistics of NER Dataset by applying Word Segmentor

The Segmentation is also not 100% correct. We may get error propagation. Here we have present the data that we faced wrong segmentation.

Token contains multiple names and is not segmented	56
Token contains a single entity name and is segmented	7008
Token contains entity name Sandhied with other words and the name is not segmented	33484
Token contains entity name Sandhied with other words and the name is segmented	6565

Table 7.5: Data of wrong Segmentation after applying Word Segmentor

7.4 NER Dataset with Gold Segmentation

7.4.1 Methodology to find Mapping between Two Corpora

To identify the similarities between DCS and Itihasa Corpora, we started by selecting a particular volume and chapter from each corpus. We compared the two by analyzing metrical line matching and word matching. As an example, a table demonstrating these comparisons is provided below.

	Total Lines	Set (Line)	Common Lines (DCS vs Itihasa)	Total Words in File	Set (Words)	Common Words (DCS vs Itihasa)	Difference (Itihasa-DCS)
DCS	962	932	-	6277	2525	-	-
Itihasa	789	781	443	5171	2287	2011	276

Table 7.6: Matching between Itihasa and DCS Corpus (Volume: 1, Chapter:2)

To accomplish this we programmatically analyzed the text of each subchapter in both the Itihasa and DCS corpora. After finding the matching subchapters programmatically, we manually inspected the output to verify the correctness of the identified subchapter matches.
We then used these subchapter matches to identify corresponding metrical lines in both corpora, which were further analyzed for similarities and differences.

Manual Matching of Chapters:

Total Number of Subchapters in Itihasa: 2108
Total Number of Subchapters in DCS: 1995
Total number of chapters selected: 1646

To solve the problem of finding the optimal matching between lines from Itihasa and DCS, we represented the problem as a graph and used the shortest path algorithm to find the optimal path. We created a directed graph where the start node was connected to all nodes in the first column, and all nodes in the last column were connected to the end node. We then added edges between adjacent nodes and diagonal nodes to create a fully connected graph. The cost of each edge was determined by the distance between the lines, calculated using the partial ratio matching approach. Finally, we used the shortest path algorithm to find the optimal path from the start node to the end node. We iterated over the nodes in the path and extracted the corresponding lines from Itihasa and DCS, along with the similarity score. This approach allowed us to create a mapping between the Itihasa and DCS verses, which we used to create a dataset containing segmentation and named entity recognition information.

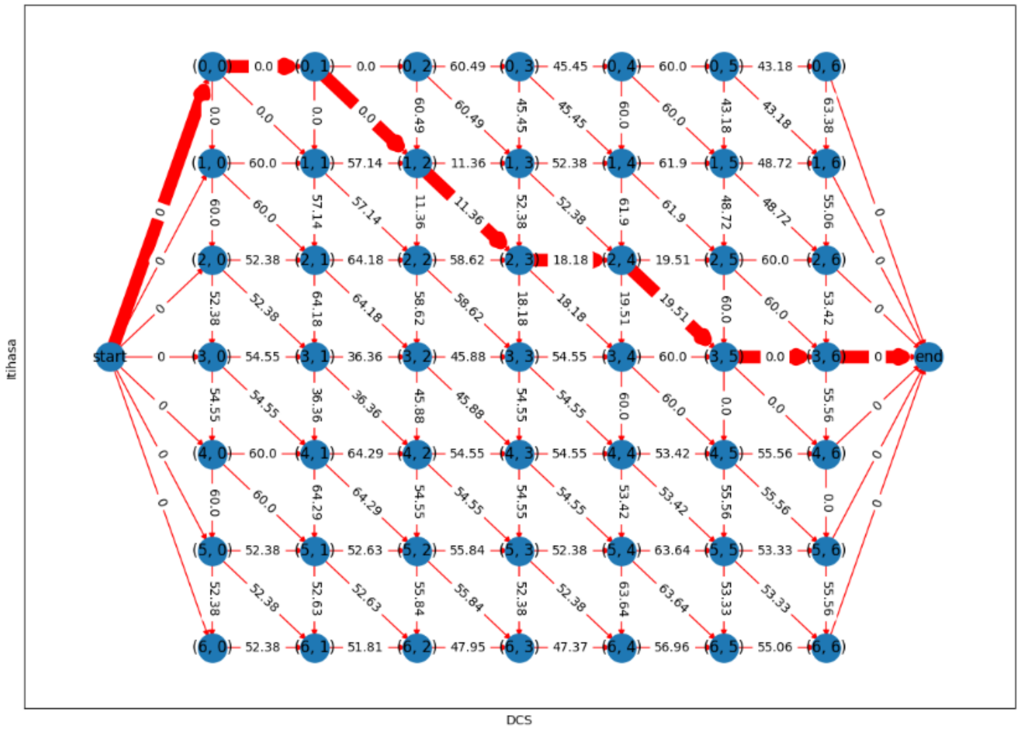


Figure 7.1: Data Matching

A threshold of similarity 80 was used for the final selection. This threshold was decided based on an inspection of the matchings.

Example Matching: Volume 1 Chapter 2

Itihasa Metrical Line Number	DCS Metrical Line Number
0	0, 1
1	2
2	3, 4
3	5, 6

Sample Partial Matchings:

Volume 1 (Adi Parva), Chapter 1 (Adivansavatarana Parva), Verse 48:

kvasTAH vIrAH pARqavAH te baBUvuH kutas ca etat SrutavantaH priyam te (Itihasa)
janamejayaH uvAca kvasTAH vIrAH pARqavAH te baBUvuH (DCS)
Similarity: 78.57
Distance: 100 - 78.57 = 21.43

Anushasana Parva (Volume 13), Chapter 8, Verse 67:
Itihasa: *annAdya BUtAH lokasya iti api SrUyate SrutiH*
DCS: *anAdi BUtA BUtAnAm iti api SrUyate SrutiH*
Similarity: 75.95
Distance: 100 - 75.95 = 24.05

7.4.2 Statistics

Total number of metrical lines	106913
Total number of tokens	566524
Token number of tokens got segmented	128261 22.64 % of total tokens
Total number of tokens after segmentation	727047
Total number of tokens contains entity and got segmented	13403
Total number of entities in tokens that got segmented	15043
Total no of tokens that are entity and not segmented	44074
Total no of entities	59117
Total percentage of tokens that contains entities	10.15 % of total tokens
Total percentage of entities that are segmented	25.45 % of total entities
Multi entities tokens	1421
Number of entities	3061

Table 7.7: Statistics of NER and Segmentation Data

Person	**Location**	**Miscellaneous**
54479	2336	2302

Table 7.8: Statistics for entity class

Chapter 8

Experiments and Methodology

8.1 Testing Existing NER Models

8.1.1 LSTM

The family of neural networks known as recurrent neural networks (RNNs) are designed to process sequential data by accepting a series of vectors $(x_1, x_2, ..., x_n)$ as input and output a different series $(h_1, h_2, ..., h_n)$ that contain information about the input sequence at each step. While RNNs can theoretically learn long dependencies, in practice they tend to be biased toward the most recent inputs in the sequence and struggle with capturing long-range dependencies. To address this issue, Long Short-term Memory Networks (LSTMs) were developed. LSTMs include a memory cell that allows the network to selectively forget or remember previous inputs. This is accomplished through gates that regulate the flow of information into and out of the memory cell. LSTMs have been shown to be capable of capturing long-range dependencies, making them useful for tasks such as speech recognition, machine translation, and natural language processing. In addition to LSTMs, there are other types of recurrent neural networks that have been developed to address the issue of capturing long-range dependencies, such as Gated Recurrent Units (GRUs). These networks also use gating mechanisms to regulate the flow of information through the network and have been shown to be effective for various sequential data processing tasks.

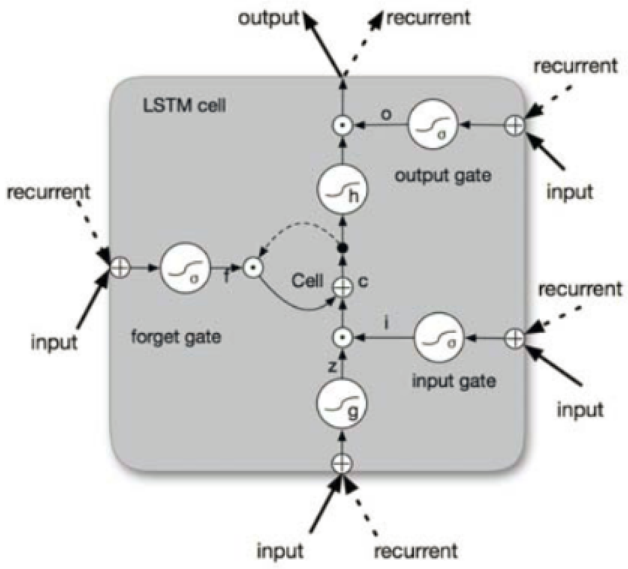


Figure 8.1: Lstm Unit

Formula:

$$i_t = \sigma(\mathbf{W}_{xi}x_t + \mathbf{W}_{hi}h_{t-1} + \mathbf{W}_{ci}c_{t-1} + \mathbf{b}_i)$$

$$c_t = (1 - i_t) \odot c_{t-1} + i_t \odot \tanh(\mathbf{W}_{xc}x_t + \mathbf{W}_{hc}h_{t-1} + \mathbf{b}_c)$$

$$o_t = \sigma(\mathbf{W}_{xo}x_t + \mathbf{W}_{ho}h_{t-1} + \mathbf{W}_{co}c_t + \mathbf{b}_o)$$

$$h_t = o_t \odot \tanh(c_t)$$

where σ is the element-wise sigmoid function, and $\odot$ is the element-wise product. An LSTM computes a representation h_t of the left context of the phrase at each word t for a given sentence $(x_1, x_2, ..., x_n)$ having t words, each represented as a d-dimensional vector. Naturally, creating a representation of the appropriate context should also contribute valuable information. A second LSTM that reads the same sequence backward can do this. The former will be referred to as the forward LSTM, while the latter will be the reverse LSTM. There are two separate networks here with various characteristics. Bidirectional LSTM describes this pair of forward and reverses LSTMs. The representation of a word using this model is obtained by concatenating its left and right context representations,$h_t = [h_t; h_t]$. For many tagging purposes, these representations effectively incorporate a representation of a word in context [3].

8.1.2 BiLSTM

Having access to both past (left) and future (right) contexts is helpful for many sequence labeling tasks. However, the LSTM's hidden state h_t only uses data from the past and lacks any knowledge of the future. One of the challenges with BLSTM models is their computational complexity, as they require processing the sequence in both forward and backward directions. However, there have been various optimization techniques proposed to reduce the computational cost, such as layer pruning and parameter sharing. Additionally, recent advancements in hardware technology, such as GPUs and TPUs, have made it easier to train and deploy BLSTM models efficiently. Overall, BLSTM models have proven to be a valuable tool in the field of natural language processing and continue to be a subject of active research and development. Their ability to incorporate both past and future context information has led to significant improvements in sequence labeling tasks, and they are expected to remain an important component of many NLP systems in the future [16].

8.1.3 CRF Tagging Model

A very simple—but surprisingly effective—tagging model is to use the h_t's as features to make independent tagging decisions for each output y_t [17]. Although this model performs well in straightforward tasks like POS tagging, its ability to make independent classification decisions is constrained when there are significant relationships between the output labels. One such challenge is NER because it must account for the "grammar" that distinguishes interpretable sequences of tags. This "grammar" imposes a number of hard limitations that are impossible to represent using independent assumptions. Therefore, using a conditional random field, we simulate tagging decisions jointly rather than separately. For an input sentence

$$X = (x_1, x_2, ..., x_n)$$

We consider $\mathbf{P}$ to be the matrix of scores output by the bidirectional LSTM network. $\mathbf{P}$ is of size $n \times k$, where k is the number of distinct tags, and $P_{i,j}$ corresponds to the score of the jth tag of the ith word in a sentence. For a sequence of predictions

$$y = (y_1, y_2, ..., y_n)$$

we define its score to be

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=0}^{n} P_{i, y_i}$$

where A is a matrix of transition scores such that $A_{i,j}$ represents the score of a transition from the tag i to tag j. y_0 and y_n are the start and end tags of a sentence, that we add to the set of possible tags. A is therefore a square matrix of size $k + 2$. A softmax over all possible tag sequences yields a probability for the sequence y:

$$p(y|X) = \frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y_x} e^{s(X,\tilde{y})}}$$

During training, we maximize the log-probability of the correct tag sequence:

$$\log(p(y|X)) = s(X, y) - log(\sum_{\tilde{y} \in Y_x} e^{s(X,\tilde{y})})$$

It is clear from the preceding formulation that we encourage our network to provide a legitimate series of output labels.

8.1.4 BiLSTM-CNN-CRF

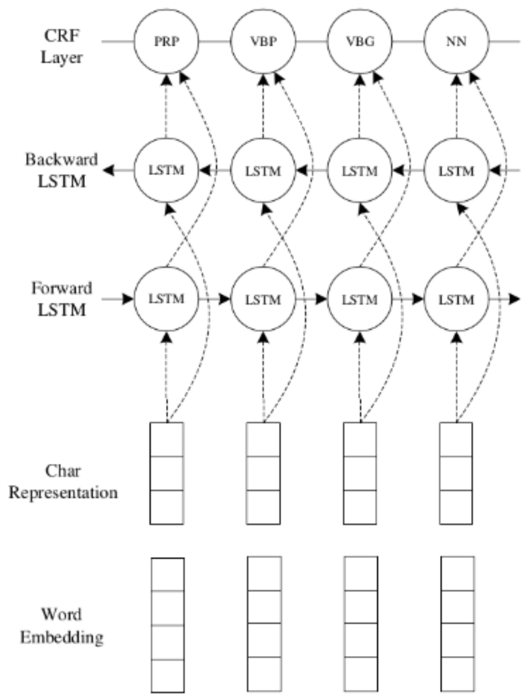


Figure 8.2: Our neural network's primary architecture. The CNN in this Figure computes the character representation for each word. Prior to feeding the word embedding and character representation vector into the BiLSTM network, they are joined. Dropout layers were applied to both the input and output vectors of the BiLSTM as indicated by the dashed arrows.

We applied our Sanskrit dataset to a neural network model by feeding the output vectors of BiLSTM into a CRF layer. Figure 8.2 illustrates the architecture of our network in detail. CNN uses character embeddings as inputs to determine the character-level representation for each word. The word embedding vector and the character-level representation vector are then concatenated and sent into the BiLSTM network. Finally, the CRF layer receives the output vectors from the BiLSTM in order to jointly decode the ideal label sequence.

As shown in Figure, dropout layers are applied on both the input and output vectors of BiLSTM. Experimental results show that using dropout significantly improves the performance of our model. We used three different word embedding models (GloVe, Albert, and fastText) to train our model. We used Datasets in two different modes: Random Split wise dataset and Less Overlap Split.

8.2 Proposed Model

Our new neural model for Sanskrit named entity recognition (NER) integrates conditional random fields (CRF), long short-term memory (LSTM) networks, and convolutional neural networks (CNN). In order to improve the performance of NER, the model also includes word segmentation as a support job. The suggested model is assessed using our Mahabharat datasets and contrasted with a number of cutting-edge techniques. In order to assess the contribution of each model component, our work also includes an ablation investigation. Overall, by utilizing a variety of neural network topologies and support tasks to enhance performance, the suggested approach provides a potential solution for Sanskrit NER.

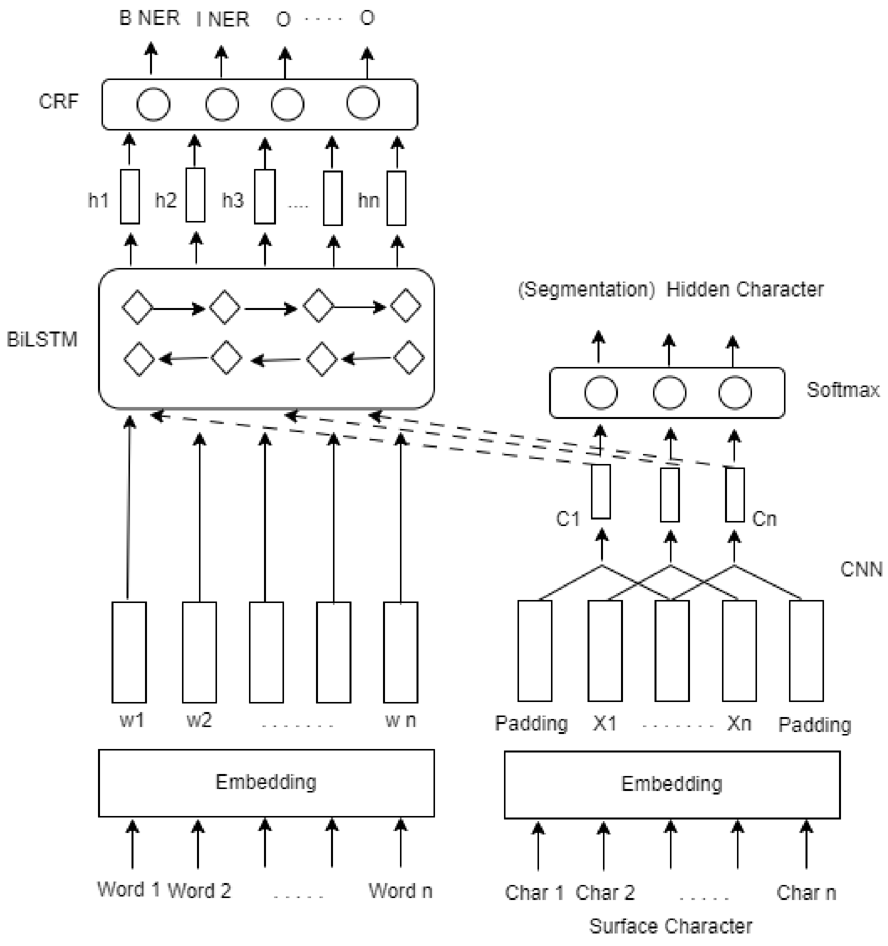


Figure 8.3: The framework of our Approach Model.

Our proposed CNN-LSTM-CRF neural architecture for Sanskrit-named entity recognition is illustrated in Fig. 8.3. Next we introduce each layer from bottom to top in detail. The first layer is embedding, which aims to convert a sentence from a sequence of characters into a sequence of dense vectors.
The second layer is a CNN network. CNN has been widely used in computer vision to extract local information from images. The local context of sentences is also very important for Sanskrit named entity recognition. We propose to use CNN to capture the local context information for Sanskrit NER.

The third layer is a Bi-LSTM network. LSTM is a special type of recurrent neural network (RNN) that can capture long-distance sequence information and is powerful in modeling sequential data. LSTM can generate hidden states for tokens in a sequence using all previous contexts, which is beneficial for Sanskrit-named entity recognition. In addition, both left and right contexts can be useful for recognizing named entities. Thus, we use bidirectional LSTM (Bi-LSTM) to learn hidden representations of characters from global contexts.

The fourth layer in our model is the Conditional random field (CRF). In NER, neighboring labels usually have strong dependencies. For example, I-PER label usually follows B-PER and I-PER, but it cannot follow B-LOC or I-LOC. Thus, it is beneficial to jointly decode the labels of characters in a sequence rather than decode them independently. In this paper, we use first-order linear chain CRF to jointly decode the labels of characters.
The Loss of our proposed model is: $Loss(NER) + Loss(SWS)$

Chapter 9

Results

9.1 Results of NER models

For applying to our NER model, we used our existing (Itihasa) dataset in two situations. Those are Random Split and Less Overlap Split.
During our training process, we utilized three distinct word embedding models and trained our model using a Random Split wise Dataset. Additionally, we employed the CNN Character Embedding Mode.

After Random Split our existing (Itihasa) dataset:
Train size: 58893 verses
Valid size: 7362 verses
Test size: 7362 verses

Parameter	**GloVe**	**Albert**	**fastText**
Epoch	50	50	50
Accuracy	96.98%	96.71%	97.59%
Precision	84.93%	88.74%	86.65%
Recall	78.85%	70.87%	84.38%
FBI	81.78	78.80	85.50%

Table 9.1: When Character Embedding Mode is: CNN

As part of our training process, we employed the same word embedding models and utilized a Random Split wise Dataset. Furthermore, we incorporated the LSTM Character Embedding Mode during that period.

Parameter	**GloVe**	**Albert**	**fastText**
Epoch	50	50	50
Accuracy	97.35%	97.17%	97.37%
Precision	85.18%	84.92%	82.21%
Recall	82.46%	80.28%	86.88%
FBI	83.80	82.54	84.48%

Table 9.2: When Character Embedding Mode is: LSTM

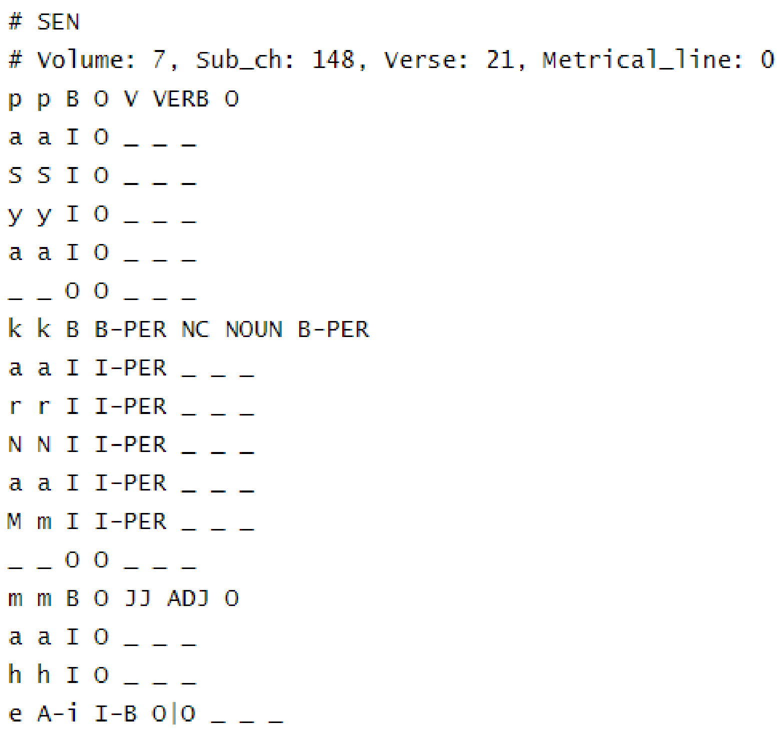
As the last part of our training methodology, we utilized three distinct word embedding models and trained our model using a Less overlap Split wise Dataset. During this phase, we implemented the CNN Character Embedding Mode.
<u>After Less Overlap Split:</u>
Train size: 58680
Valid size: 7455
Test size: 7455

Parameter	**Glove**	**Albert**	**fastText**
Epoch	50	50	50
Accuracy	93.11%	92.8%	93.15%
Precision	75.81%	74.44%	75.07%
Recall	43.32%	40.8%	44.29%
FB1	55.13	52.7	55.71

Table 9.3: When Character Embedding Mode is: CNN

9.2 Result of Proposed Model

Our Dataset ratio is 80:10:10 for Train, Valid and Test. In dataset we mentioned Surface character, Hidden Character, Segmentation Boundary, NE Boundary, Character level, and POS. The dataset looks like:



```
# SEN
# Volume: 7, Sub_ch: 148, Verse: 21, Metrical_line: 0
p p B O V VERB O
a a I O _ _ _
S S I O _ _ _
y y I O _ _ _
a a I O _ _ _
_ _ O O _ _ _
k k B B-PER NC NOUN B-PER
a a I I-PER _ _ _
r r I I-PER _ _ _
N N I I-PER _ _ _
a a I I-PER _ _ _
M m I I-PER _ _ _
_ _ O O _ _ _
m m B O JJ ADJ O
a a I O _ _ _
h h I O _ _ _
e A-i I-B O|O _ _ _
```

Figure 9.1: Pattern of Dataset

We applied four types of experiments. In every experiment, we applied 20 epochs. Our every experiment model, predicts whether a Token contained Name Entity or Not and we processed 56461 tokens with 5732 NE for every model.
Our experiment result is shown below in the table.

	Accuracy	**Precision**	**Recall**	**FB1 Score**	**Found NE**	**Correct NE**
BiLSTM	74.35%	70.46%	74.35%	72.35	6049	4262
BiLSTM + CNN Character Embedding	68.28%	78.66%	68.28%	73.10	4976	3914
BiLSTM+CNN+Joint	53.72%	90.88%	53.72%	67.52	3388	3079
BiLSTM+CNN+Joint (WS and NER Boundary Prediction)	45.52%	92.03%	45.52%	60.91	2835	2609

Table 9.4: Result of Our Approch Model

9.3 Observations from Proposed Model

- Precision has increased with the joint model, but overall performance did not improve as the recall is becoming less.
- Need Hyperparameter tuning
- Model only predicting NE, NE classification (PER, LOC, MISC not tested). Requires changes in the model.
- No pre-trained word embedding was used.
- Further analysis required in identified NE boundary at character level.

Chapter 10

Future Work

Till now, we worked on our Sanskrit dataset in word-level segmentation. We used three different word embedding models (GloVe word embedding, fastText, and Albert word embedding model). After that, We used Bi LSTM and CRF to detect NER from the Sanskrit language. Then, we approach a framework to jointly train Sanskrit NER and word segmentation models in order to enhance the ability of the Sanskrit NER model in identifying entity boundaries.

In the future, we will try to increase our dataset, and apply our dataset in more efficient models with better accuracy. Sanskrit-Corpus dataset continues to grow, it is likely to become an increasingly valuable resource for researchers interested in Sanskrit natural language processing.

Transformer models, such as BERT, RoBERTa, and GPT, have been shown to be effective in NER tasks in various languages, including English, Spanish, Chinese, and others. These models use attention mechanisms to process text and learn contextual relationships between words and phrases, making them well-suited for tasks like NER. We will try to implement any transformer model for detecting NER from the Sanskrit language which will perform better than now.

Chapter 11

Conclusion

The paper proposes a neural approach to recognize named entities in Sanskrit. The proposed method uses a neural network architecture for sequence labeling and does not rely on any task-specific resources, feature engineering, or data pre-processing. The model can detect named entities at the word level boundary.

Additionally, the paper proposes a framework to train the Sanskrit NER and word segmentation models together to improve the Sanskrit NER model's ability to identify entity boundaries.

However, creating an effective NER model for the Sanskrit language requires a large, high-quality labeled dataset of Sanskrit text with annotated named entities, which is currently not widely available. Training a NER model on a small or low-quality dataset may result in poor performance. Therefore, while transformer models show promise for Sanskrit NER, further research, and resources are needed to develop effective models for this task.

Bibliography

[1] S. Sujoy, A. Krishna, and P. Goyal, "Pre-annotation based approach for development of a Sanskrit named entity recognition dataset," in *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, (Canberra, Australia (Online mode)), pp. 59–70, Association for Computational Linguistics, Jan. 2023.

[2] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[3] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[4] F. Wu, J. Liu, C. Wu, Y. Huang, and X. Xie, "Neural chinese named entity recognition via cnn-lstm-crf and joint training with word segmentation," in *The World Wide Web Conference*, pp. 3342–3348, 2019.

[5] C. Chen and F. Kong, "Enhancing entity boundary detection for better chinese named entity recognition," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 20–25, 2021.

[6] Z. Chen, K. Wang, J. Zheng, Z. Cai, J. He, and J. Gao, "Opdai at semeval-2022 task 11: A hybrid approach for chinese ner using outside wikipedia knowledge," in *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pp. 1488–1493, 2022.

[7] A. Dey, A. Paul, and B. S. Purkayastha, "Named entity recognition for nepali language: A semi hybrid approach," *International Journal of Engineering and Innovative Technology (IJEIT) Volume*, vol. 3, pp. 21–25, 2014.

[8] A. Kulkarni and D. Shukl, "Sanskrit morphological analyser: Some issues," *Indian Linguistics*, vol. 70, no. 1-4, pp. 169–177, 2009.

[9] S. G. Thottempudi, "A visual narrative of ramayana using extractive summarization, topic modeling and named entity recognition,"

[10] L. Mills, "Ii. the ahuna-vairya from yasna xxvii, 13, with its pahlavi and sanskrit translations," *Journal of the Royal Asiatic Society*, vol. 42, no. 1, pp. 57–68, 1910.

[11] N. Timalsina, *IndoLib: A Natural Language Processing Toolkit for Low-Resource South Asian Languages*. PhD thesis, Harvard University, 2022.

[12] S. Bawa and M. Kumar, "A comprehensive survey on machine translation for english, hindi and sanskrit languages," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–34, 2021.

[13] S. Sharma and M. Mohania, "Comparative analysis of entity identification and classification of indian epics," in *Proceedings of the 2022 International Conference on Multimodal Interaction*, pp. 404–413, 2022.

[14] R. Aralikatte, M. de Lhoneux, A. Kunchukuttan, and A. Søgaard, "Itihasa: A large-scale corpus for sanskrit to english translation," *arXiv preprint arXiv:2106.03269*, 2021.

[15] O. Hellwig and S. Nehrdich, "Sanskrit word segmentation using character-level recurrent and convolutional neural networks," in *Conference on Empirical Methods in Natural Language Processing*, 2018.

[16] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional lstm feature representations," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 313–327, 2016.

[17] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso, "Finding function in form: Compositional character models for open vocabulary word representation," *arXiv preprint arXiv:1508.02096*, 2015.