Named Entity Recognition on Kannada Low Resource Language using Deep Learning Models

MSc Research Project
MSc DATA ANALYTICS

Pavan Kulkarni
Student ID: x19231075

School of Computing
National College of Ireland

Supervisor
Prof Dr. Christian Horn

Named Entity Recognition on Kannada Low Resource Language using Deep Learning Models

Pavan Kulkarni
x19231075@student.ncirl.ie
MSc in Data Analytics
National College of Ireland

Abstract

Entity extraction with Kannada language recognition is a difficult task that requires adept knowledge of the literature. The language is influenced by other languages. Around 60 million people in Karnataka, India, can understand and speak. Kannada is classified as a low-resource language. Because the shortage of data makes it a difficult assignment to complete. Many tests have been conducted and have shown important results in the English language. In contrast to English, the Kannada language does not capitalize words. For contribution to Kannada Named Entity Recognition (NER) Bi-directional long short-term memory (Bi-LSTM), Bi-directional Encoder Representations from Transformers (BERT) and Random Forest Classifier (RFC) models are used. The accuracy obtained from Bi-LSTM is 0.9624 and 0.7985 accuracy and precision of 0.78 from RFC for the B-DATE entity.

Keywords:- Bi-LSTM, BERT, Random Forest Classifier, Entity extraction.

1 Introduction

The goal of NER is to characterize and label different entities. Typically, entities such as an individual, company, location, date, and so on. NER is the first step in performing many Natural Language Processing (NLP) tasks such as information extraction, retrieval of information, feature extraction, and so forth. To train and apply named entity tags to individual tokens in a sentence, models treat this as a sequential labelling problem. The performance is primarily determined by the part of speech, lexical features, surrounding words, and so on. Pre-trained sentence vectors aid in the recognition of each. (Jin and Yu 2021) the Bi-LSTM output will be used as a NER feature, labelling each word/token. Kannada is a language that is rarely used nowadays, and only a few people in Karnataka, India, can understand and speak it. Kannada is classified as a low-resource language. The syllable sequence is concatenated by Random Forest before being fed into BI-LSTM in sentence form. These methods use syllable-level LSTM to reduce Out of vocabulary (OOV).

NER is a critical step in locating as well as categorizing entities. NER is a subbranch of Information Extraction (IE). NER is becoming increasingly important in Natural Language Processing (NLP) application. The rule-based approach, according to the author (Melinamath 2014), necessitates months of research, as well as extensive experience, grammar. Morphology is the study of words in which small changes in grammar alter the meaning. Kannada nouns agglutinate. The meaning of a sentence is not altered by the addition of words. These words are difficult to distinguish both morphological and logically. Kannada words can be written in a variety of orthographic styles. There are only a few NER, chunking taggers, and parts of speech taggers reported. Natural language processing revolves around NER.

Entity With a Name One of the most important and difficult tasks in Natural Language Processing is recognition and classification. Kannada is an inflectional language with complex word forms. Kannada is primarily a suffixing language, and it is a poor resource language. Parts of Speech (POS) tagging, annotated corpora, name dictionaries, and other resources are not yet available in the required quantity, and little work has been reported for this language. The goal of this research is to create a new model for NER in Bi-LSTM and BERT. Significant progress has been made in the English language, but work on NER in Kannada has yet to be reported. Capitalization is useful in English for identifying entities, but the Kannada language is short of capitalization of words adding to the complexity.

NER research has recently become one of the most researched areas. Work on Kannada NER is minimal. Kannada language research is significant because sixty million people in the state of Karnataka interact in this language. Because Karnataka state in India shares borders with neighbouring states and the Kannada language has been heavily influenced. Kannada has many dialects and is heavily influenced by the English language. Because of demographic change from other Indian states, Kannada is used less frequently. The paper goes over the issues that were encountered while developing the model. The steps for performance evaluation and implementation are discussed in detail.

1.1 Motivation

In NER, supervised and unsupervised learning is used extensively for English. (Melinamath 2014) proposed a rule-based algorithm for NER with a precision of 86 percentage.

Kannada NER (KNER) is a difficult task due to the lack of a large, labelled dataset, the lack of capitalization, spelling variations, and the lack of standardization. Spellings and variations are not normalised. Because the country is multilingual, a new NER is not going to function in a different language. KNER cannot be used directly with a well-developed English NER system. As a result, there is a need to create a Kannada NER system to be present on the internet. Kannada is one of the twenty-two scheduled languages. (Amarappa and Sathyanarayana 2015) Kannada is the world's 33rd most spoken language, with over sixty-seven million native speakers. The primary goal is to retrieve specific data. The KNER annotated data is not readily available over the internet. The large corpus Named Entities aids in the development of NLP. To overcome the difficulties in annotating named entities, Kannada words must be morphologically joined and annotated. Language chunking, parts of speech, and morphology all influence sentence structure. With a large corpus of 73,676 records, the author (Pallavi, Sobha and Ramya 2018) proposed a novel application to determine named entities using rule base.

1.2 Research Question

How much can the Kannada named entity recognition system be improved by using a deep neural network to accurately tag and classify the named entities?

1.3 Application of the model

To classify the contents from Kannada news providers. Newspaper publishers generate articles in bulk. NER can scan entire articles and generate famous people, popular places, organizations, and important dates from the articles. This will help in content discovery. Internal algorithm searches, here NER will help to separate the tags from millions of articles. The most effective content recommendation system, for example, a news article recommendation system can effectively recommend news articles and magazines to its customers. Better customer support, here customer complaints can be extracted and passed to the responsible team within an organization. And NER systems can extract the entities from the research papers.

2 Related Work

(Wintaka, Bijaksana and Asror 2019) previously work Indonesian NER tweets are cumbersome because of grammatical errors and improper tweets styles make it a challenging task to perform. Wintaka and the team used 250 different tweets and applied Bi-LSTM deep learning model and CRF machine learning model. The results were obtained with an f1 score of 86.13 per cent. For this study, Wikipedia articles are used and converted into vector representations. The aim was to identify a person name, location, and organization from the tweets. Here, the username of the tweeter holders is taken as a personal name. In this proposed model, the Bi-LSTM layer acts as an encoding layer and CRF acts like an encoding layer. Before training the model, each word embedding is represented in the form of vectors. The backbone of this study is comparing word embeddings with Glove, FastText, Word2Vec. This step was taken to improve the performance. Here, 300 embedding dimensions are used for the experiments.

Bi-LSTM can scan through past and future context data, but sequential labelling poses a drawback to this model because of the generalization of correlation among output labelling. In CRF, the softmax layer is used at the top and this layer manages multi-class grouping. The f1-score of FastText and Bi-LSTM-CRF model has shown 86.13 percentage. F1- score of entity organization showed less score of 76.19%. A study suggests that with more corpus data the performance of the model can be improved (Wintaka, Bijaksana and Asror 2019).

(Aras et al. 2021) proposed Bi-LSTM as well transformer-based network to extract Turkish named entities. And the transformer-based network has CRF as a top layer for a novel approach to Turkish NER. Turkish is also an agglutinative language. Joining suffixes many Turkish words can be derived. In the Bi-LSTM model, every word is taken as a single token. The drawback is this will not capture the word location. Transformer-based models replace RNN cells with connected and self-attention layers. This combination helps in encode content information as well as dependencies. CRF being the top layer combines the last concealed state and basic network. The dataset used is newspaper content between 1997 and 1998. During building the Bi-LSTM model, represent encoder input embeddings as vector representations. Character level morphological analysis, word chunks, unigram tokenizers are referred for Turkish tokenizers. A stochastic downward gradient optimizer and a study rate of 0.05 have been employed. A model trained with an epoch of fifty and 0.9 momentum optimizer.

(Sreeja and Pillai 2020) in this study named entity recognition is performed on Malayalam. Malayalam is one of the Dravidian languages and is mostly spoken in the southern state of India i.e., Kerala. Malayalam is also a low resource language. The study goes on working on RNN and LSTM, a custom model with few stop words and a supervised approach. Feature extraction model extracted POS taggers and determined the suffixes and root words. RNN model has less information retaining on long sentence information. LSTM has retained information because of sigmoid operation, vector concatenation, tanh and pointwise addition cells.

(Todi, Mishra and Sharma 2018) the classic learning paradigm, in which the lengths of the input and output sequences are the same, is ideal for POS Tagging. We used character embedding to get around the problem of OOV terms. With the help of a Bi-LSTM network, the character embeddings were learned. For a morphologically rich language like Kannada, combine word embedding with character embedding, where a stronger syntactic representation was acquired to gain character-level information. The CRF system achieved an overall accuracy of 92.31 per cent in the experiment. In the CRF model, an F1-score of 0.92 was obtained.

(Aras et al. 2021) used transformer-based BERT model, deployed multilingual cased BERT as well as trained word model. In this study, BERT multilingual tokens do not match with morphological results. There was a mismatch of 20 to 40 per cent in the sub-word obtained and vocabulary used while training. Adam optimizer along with fixed weight decay and learning rate of 5e-5 is used to train the model. For model evaluation precision, recall, f1 score and accuracy are used. The highest accuracy was obtained from BERTurk with an accuracy of 99.41%.

(Zhang et al. 2021) in the health-preserving field, a NER approach based on BERT can generate many vectors for the same phrase while fully exploiting context semantics. Google released the BERT model14 in October 2018, which has since been widely used and the record of different NLP tasks. Based on CNN-LSTM-CRF, the BERT-CNN-LSTM-CRF fusion model is presented. When compared to Word2vec, the BERT pre-trained language model can transmit acquiring rich knowledge while at the same time. BERT generates vector representation using the superposition of word vectors, phrase vectors, and position vectors as input and transformer for coding.

(Zhang et al. 2021) multi-layer Transformer encoders are at the heart of BERT, and Transformer is reliant on the self-attention mechanism without the need of a convolutional neural network or a recursive neural network. Transformer compares the attention given to each word in the input sentence to the attention given to each word in the sentence. The goal is to capture the interrelationships between words, the underlying structure of sentences, and the importance and relevance of certain terms.

Attention (Q, K, V) = softmax (QKT/sqrt(dk)) V

(Zhang et al. 2021) here K is key vector, v is value vector and Q is query vector. BERT uses two unsupervised tasks, Masked LM and Next Sentence prediction, to build its input vector, which is generated by overlaying Token embedding, Segment embedding, and Position embedding. The former seeks to improve context semantics embedding, whereas the latter seeks to acquire sentence-level features by evaluating the connections between neighbouring sentences.

Based on a supervised machine learning model, (Atmakuri et al. 2018) proposed a portion of speech for the Kannada language. Based on considerable research, the author employed CRF and Pos tagging. Three very large corpora will be used for comparison, and the optimal approach will be decided based on the outcomes of these 3 corpora. A rapid, accurate pos tagger is a key part of any language's NLP toolkit, and it lays the groundwork for further research in the field.

(Atmakuri et al. 2018) in his 5th century BC work, Nirukta, the Sanskrit grammarian Yaska identified four classes. Verbs, nouns, and adjectives are among them. These definitions are simplified for modern linguists. Pos tag sets with additional word classes have been adopted by English treebanks. The Penn Treebank has forty-five tags from 1997 and 146 tags from 1998. There are three types of adjectives in the Japanese language, but there is only one type in English. With the pos, we can determine the number of morphemes associated with a word and its nature.

(Atmakuri et al. 2018) on the basis of topic, the Kannada treebank is separated into 3 corpora. For training and testing, a 70:30 ratio was adopted. To annotate the dependence relationships, the author employed a voice type. Using a suffix, this approach achieves the best accuracy of 89.1 per cent. The number of words that have been accurately labelled is quite high. The author employed 3 corpora, with each model improving the accuracy. We can deduce from this test that the general corpus has higher accuracy than the others. The corpus' size is also essential.

A process of identifying nouns and classifying them like name, organization, time, date, and location. Using CRF, (Amarappa and Sathyanarayana 2015) suggested a unique technique for Kannada NER. The report outlines the difficulties encountered, as well as the performance and implementation processes. The test was carried out on a large data of 95,127 tags, with 5,000 tag sets being chosen for testing. Based on this approach, this model has a better f1-measure, precision and recall of 85 per cent, 82 per cent, and 85 per cent, respectively.

(Amarappa and Sathyanarayana 2015) there are 49 phonemics in Kannada grammar, as well as characters. Kannada is an agglutinating language with a rich heritage. The task of extracting named entities and processing them is extremely difficult. The first NLP study began in 1940. Until 1950, the NLP system was based on complicated hand-written rules. Machine learning algorithms transformed the face of NLP after 1980, and it began to focus on most spoken languages and high-resource languages like English. However, due to a lack of categorization of named elements, Indian languages were left behind, and they are still in the early stages today. The phrase named entity was first used at the Sixth Message Understanding Conference (MUC-6).

(Amarappa and Sathyanarayana 2015) in the Kannada language, there is no capitalization of words. For NER, we can process data using the Brahmi script. Names are the most commonly used nouns. The processing and extraction of named entities become complicated due to a lack of annotated data. To overcome this problem, supervised learning techniques such as Decision trees and Support Vector Machines are frequently used. The CRF model is the subject of this article. It's also known as the Hidden Markov Model's discriminative counterpart. The posterior conditional probability will be determined using this model. When compared to the rule-based approach, machine learning boosts efficiency and may be employed more conveniently. The construction of a corpus is the initial stage in completing the CRF test. Kannada NERC is difficult to understand without labelled corpora. A corpus of about 100,000 Kannada words was retrieved from web articles, Enabling Minority Language Engineering (EMILLE), and books. This corpus is divided into two parts: development and testing. The corpus was tokenized into words and tags in the second step of pre-processing.

(Amarappa and Sathyanarayana 2015) individual tags are allocated to the appropriate class tags. Words are denoted by X and tags are marked by Y. The following stage in the training step of the innovative CRF analysis is to extract the feature set from X and Y, such as edge and node features. Determine the CRF parameters using both features from the training corpus. The fourth step is to validate the parameters, which involves taking the average of several assessments on the dev-test set. The Viterbi algorithm is used to assess the probability for the rest of the corpus. The fifth step is to decode the test data corpus, execute the validation process, and calculate the findings. The forward Viterbi method determines the transition path's highest probability. Individual words are tagged with the proper entity in this stage. According to the CRF, the training model required less duration 215.562 sec, whereas testing the model required 0.266 sec. The model was able to process 95,127 training and 5,000 test corpora with ease. The accuracy of the model was 85 per cent. The CRF technique received an f1-measure of 84.7 per cent, recall of 86 per cent, and precision of 85.4 per cent. This calculation is based on the appraisal of twenty-two separately identified entities. The new tag "NONE" identifies a term that is not a NE in testing corpora. The test results are very similar to those of the training set. When comparing f1-measure results to other Indian languages, this model scored better.

(Todi, Mishra and Sharma 2018) part of Speech (POS) tagging is a fundamental application of Natural Language Processing (NLP) in any language. It's the practice of giving each word in a sentence a tag. In any language, POS is used as a pre-processor for tasks like chunking, dependency parsing, and named entity recognition. Unknown or Out-of-Vocabulary (OOV) words are one of the most common issues in POS tagging, however, they can be avoided by using character embeddings. DeiTY, the Indian government, supported the dataset annotation.

(Todi, Mishra and Sharma 2018) the author used neural networks and a machine learning approach. CRF and SVM were employed in the machine learning approach. The Yamcha SVM model is used to create an SVM model with several binary classifiers. Deep learning approaches began with structured perceptron, in which we used the same set of characteristics as CRF and produced results that were equivalent to CRF. For word embedding studies, Adam optimizer was used with a batch size of 32, and RMSPROP was utilized with a batch size of 64 for joint character and word embedding experiments.

(Melinamath 2014) rule-based technique is used, as well as Machine Learning. The data was gathered from the Kannada daily ‘Prajavani.’ There are 5000 distinct proper nouns in the dataset. The accuracy of the model is 86 percent. Annotated texts are used to train the machine learning model. Support vector machines (SVMs), decision trees, the Maximum Entropy Model (MaxEnt), and random field methods are all used in this application. The system is built using gazetteer data, which has been found to improve accuracy. However, months of hard work and experience are required to train this model. The Rule-based method, which is handcrafted, produces better results.

(Melinamath 2014) the following is an explanation of how to handle NER problems: reading transliterated texts, tokenization, and dictionary method. The raw corpus is transformed into the transliterated corpus in the transliteration stage. The Unicode text is converted to a transliterated file in this step. The recognizer converts the input ‘named entity into a human name. A dictionary is used to look up tokens. A total of 5000 words are manually entered into an electronic dictionary. A total of 12.34 per cent of false positives were discovered during the experiment. In addition, 8.56 per cent of words went unnoticed.

(Chowdhury et al. 2018) on Chinese EMR, a multitask bi-directional RNN system is used. For vector representation of each word, the concentration of character embeddings and word embeddings was used. Bi-directional RNN is well-suited to extracting context information from Chinese sentences. The system is evaluated using the micro average F-score, the macro average, and the model’s accuracy. The accuracy of the micro average F-score, 2.41 per cent, and the macro average F-score increased by 5.66 per cent, 2.41 per cent, and 4.16 per cent, respectively.

Conclusion: The majority of the progress has already occurred in English. Because the English language capitalizes nouns and people’s names, identifying and classifying entities becomes a little easier. Also, having a lot of resources to train the models, such as annotated tags and large data sets, is a great help in getting higher accuracy ratios and f1-measure. There have been numerous attempts to contribute to KNER. The most prevalent machine learning methods are used, including CRF and Rule-based techniques.

Although the HMM approach is utilized, the model’s accuracy and f1-measure are low when compared to other models such as CRFs. The data set utilized in the experiment is also important in terms of overall score improvement. Since RNN does not gather the data and keep it in memory for a longer period, it scores below CRF. This is a critical region that requires attention to increase model scores. In China, NER has grown in prominence, and numerous works have already been completed. Kannada NER still requires a significant amount of work, and new models must be tried.

2.1 Recent work

(Santoso et al. 2021) part-of-Speech (POS) Tagger and NER are the two tasks that make up the model. The Bi-LSTM is then described. The key reason we chose Bi-LSTM as our model is that it can handle sequence classification tasks in both forward and backward directions by recognizing the context of the input. In our research, a bi-LSTM is made up of two LSTMs with two directions, forward LSTM (fw) and backward LSTM (bw). For each time step t, this model accepts input from a word vector x(t). Two hidden states will be produced by each LSTM cell. Concatenating the two hidden vector outputs yields the Bi-LSTM output for each timestep of every input in the sequence.

(Santoso et al. 2021) author used the Skip-Gram model for Word2Vec. To build

Word2Vec, we employ a negative sampling optimization approach, which is more efficient than hierarchical softmax. Using Indonesian Wikipedia Articles, Word2Vec Skip-Gram embedding is created. The suggested methodology outperforms previous models by an average of 1.21 per cent when compared to CRF, Bi-LSTM using Word, and Bi-LSTM using Word and POS. The model's weakness was that it couldn't extract relationships between things in a document. Experiments with various sorts of Embedding, such as labels and other types of word Embedding, will be conducted to see if there is a way to improve performance.

3 Research Methodology

The Knowledge Discovery in Databases (KDD) approach is used in this research. This method lays out a series of steps. These include data set gathering, cleaning the dataset, and feature engineering, data extraction, information modelling, evaluation, and selection.

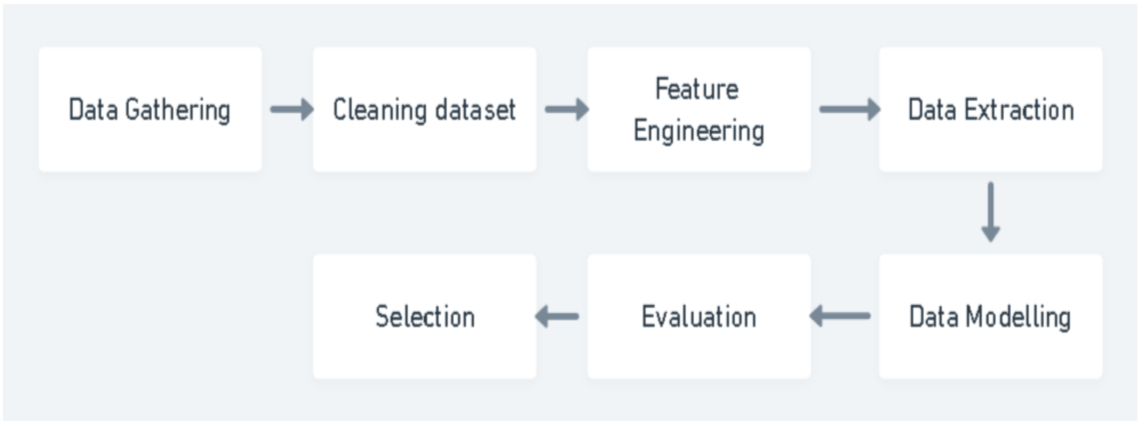


Figure1. Steps involved in the KDD process

3.1 About Dataset

One of the challenging tasks in performing NER on low-resource language is the availability of a reasonable amount of annotated datasets available. Since there was a shortage of datasets, the dataset must be created by own and tagged manually. The dataset contains sentence numbers – indicate each word with its sentence number. This helps identify which word belongs to which sentence. The second column in the dataset is a word – indicates the individual word and the original language is Kannada. This column is one of the important in performing the NER. The third column is parts of speech tagger – it indicates whether the word is a proper noun/ noun, it will help in distinguishing between a living thing and a non-living thing. And the last dependent variable in the dataset is tag this is also an important variable, this will help in understanding whether the word is a company or not, or location. Here, word and tag play an important role throughout performing the NER. The shape of the dataset is 2070 * 5. And this dataset had to be created manually.

3.2 Data Processing Step

Cleaning the dataset is vital to get an optimal result. Since outliers and void values pose an effect on the result. This dataset contains data types like serial number is an integer, sentence number is float, word, POS, Tag is a data type object. Sentence number is having no missing and not null () is applied before processing. No null values are found in the dataset. Dataset is manually created and attention to detail is given while creating.

3.3 Visualization of the dataset

Visualizations such as line plots for each of the variables are done after data cleaning to obtain a better understanding of the data. This allows us to observe the pattern over time. To examine the distribution of the data variables, the graphical representation is done. Using histogram plot and 'ggplot' style, the sentence length is represented. From the plot, we can observe that the maximum sentence length is 15. And the minimum length of the sentence is 1. The average length of the sentence lies between 7 to 15 sentences. This distribution follows a partial normal distribution curve. This shows that data is evenly distributed.

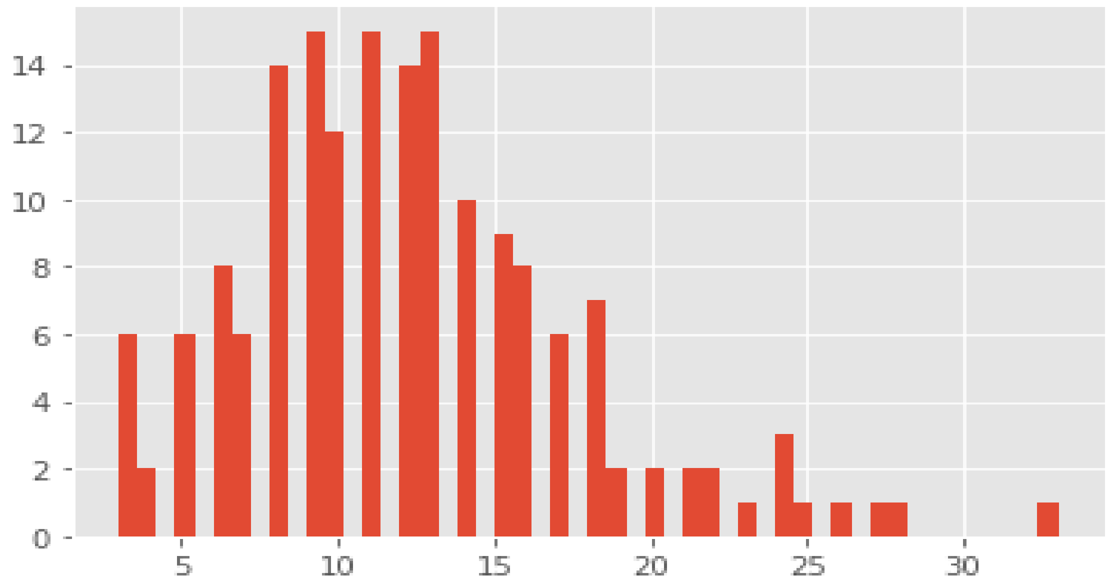


Figure 2. Histogram plot of length of sentences in the data using 'ggplot style'

3.4 Bi-LSTM, BERT, and Random Forest Classifier model

The KNER was not processed using the Bidirectional Long Short-Term Memory (Bi-LSTM) - Random Forest Classifier or the BERT (Bidirectional Encoder Representations from Transformers) algorithm. Because the algorithm learns both forwards and backwards, this model is appropriate for this experiment. Is it possible to figure out whom he is using the Bi-LSTM model? In NER, holding data in memory for a longer period is also significant. This will aid in the tagging of entities. Using a mix of Bi-LSTM and BERT, a novel strategy to automatically tag word and character level features has been developed. Feature engineering is not required using this method. Long-distance dependencies are easily learned by LSTM. By accounting for an endless amount of context, the Bi-LSTM and BERT take it to the next level. When the Bi-LSTM and BERT models come across a word or sentence, the named entities are tagged by the Bi-LSTM model. Feature vectors are extracted by the Random Forest Classifier (RFC). Individual words are then created by combining both vectors. RFC can also be utilized for NLP tasks at the character level. For feature extraction and POS labelling, (Bharadwaj et al. 2016) used CNN. However, the Bi-LSTM and BERT solutions were not used.

Each sentence's identification and classification were separate. The position of a sentence had no bearing on the context of the sentence. The information in the corpus was ordered at random. To investigate the NER, the author divided the data corpus into five folds and ran five tests. Using POS, feature functions, and other techniques, a probabilistic model based on sequence dependencies and structured sentences was developed. The major challenge in impacting the accuracy was feature selection. Unigrams represent individual features, while bigrams represent groups of features. Data from the tests are analysed for classification and identification.

Random Forest Classifier: It's a tree-based model. It's proved that Random Forest Classifier (RFC) deliver decent results in named entity recognition. RFC is one of the favourite tree-based models based on understanding the rule to learn based on tagged words.

Bi-LSTM model: (Bansal 2021) Recurrent neural networks come in a variety of types, including LSTMs. RNNs are designed to learn the structure of sequences and to handle them. RNN accomplishes this by generating the next output by combining the output from the previous item in the sequence with the current item. Mathematically this can be represented as,

$$f_t(x_t) = f(f_(t-1)(x_(t-1), x_t; \theta))$$

This equation states that the output at time t-1, as well as the input data xt at the same time step, are used as inputs to compute the output at time t. This is accompanied by a set of parameters or learning weights, denoted by. This RNN has a one-of-a-kind formulation.

RNN architecture: On the left, you can see the basic cell. To generate an activation component, the input vector is multiplied by a weight vector, represented in the diagram as U, at a certain time or sequence step t. A weight vector, shown by V in the diagram, is multiplied by the output of a previous step and added to the activation. The output of that step can be obtained by multiplying this activation by another weight vector, W, as illustrated at the top. This network can be unrolled in terms of sequence or time steps. This is a virtual unrolling. It is, however, shown on the figure's right side. At time step t, activation can be expressed mathematically as:

$$a_t = U.xt + V.(a_t - 1)$$

The output can be represented as:

$$o_t = W.a_t$$

Bi-LSTM model: (Bansal 2021) Because it is a single unit, the network is relatively simple structurally. Weight vectors U, V, and W are shared across time steps to exploit and learn the structure of inputs going through. In contrast to fully connected convolutional networks, this network does not include layers. It can be viewed as having as many layers as the steps in the input sequences because it is unrolled over time steps. This structure allows for the processing of sequences of any length.

While pre-processing, words and tags are used as input for training the named entity recognition model. The model used a single layer of bi-LSTMs. Hidden units are set to 100 and for optimization 'Adam' optimizer and for time distribution 'softmax' activation layer is used. For training, the batch size is set to 64, epochs set to 100. Initially, epochs were set to 10, after trial and error, epoch set to 100. There was little improvement in the accuracy.

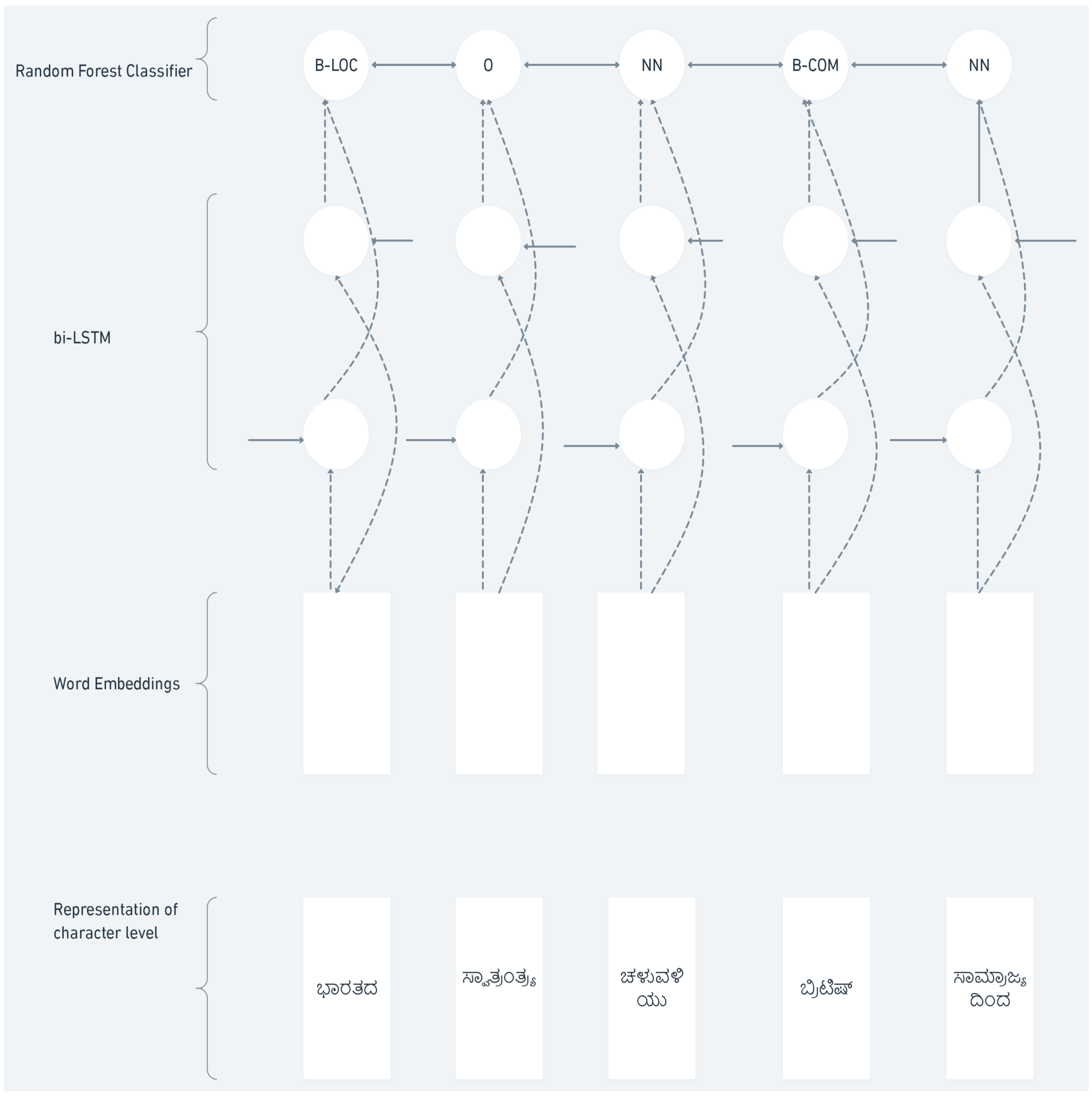


Figure 3. The architecture of Bi-LSTM model pipeline for the Named entity Recognition

BERT model: A stacked-layer structure was used to build the baseline transformer encoder. A multi-head self-attention sub-layer is present in each layer. The output is passed to the feed-forward network.

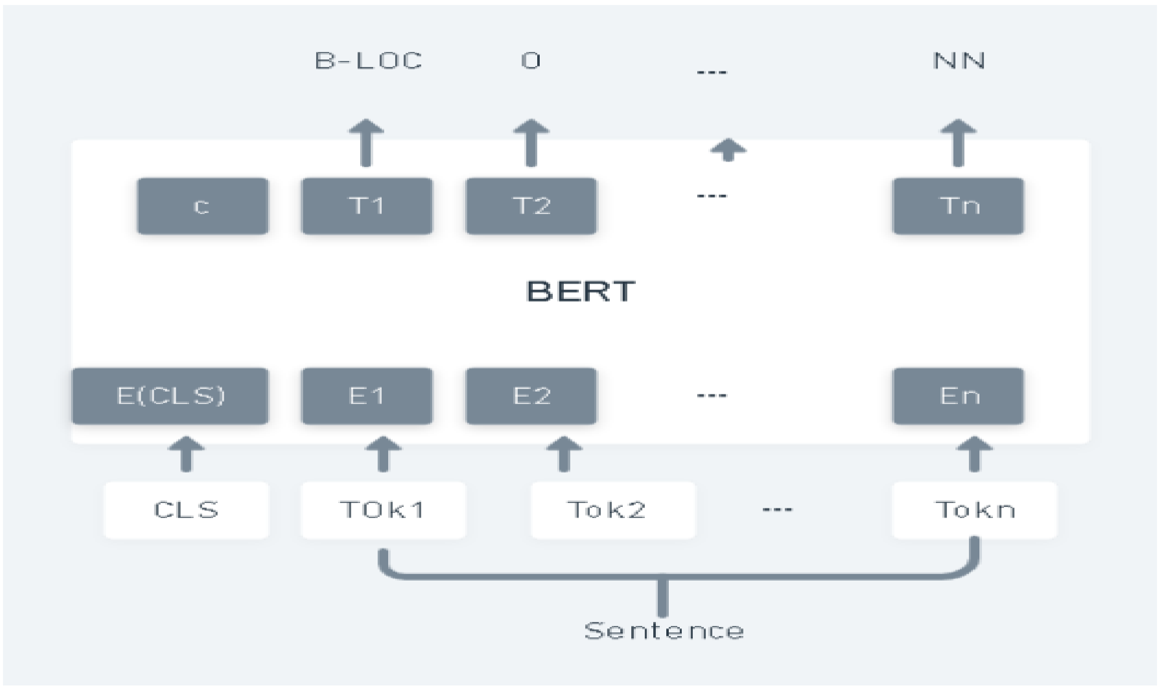


Figure 4. The architecture of BERT tuning for named entity recognition

3.5 Evaluation of the Model

This part entails the model evaluation steps for performing performance evaluation of the named entity recognition model. The proposed system has data related to Kannada words, and their related parts of speech and tag denoting the type of the entity. The aim is to identify the better suitable model to fit for the Kannada Named Entity Recognition model. The models we used are Bi-LSTM, BERT, and Random Forest Classifier. Furthermore, for evaluation of the model Bi-LSTM accuracy and value loss evaluation metrics are used. For Random Forest Classifier model evaluation metrics like precision, recall, f1-score and accuracy of the model are considered. For the BERT model, the evaluation metrics used are precision, evaluation loss, recall and f1-score of the model.

The model's accuracy offers the model assessment metrics. Generally speaking, accuracy is called a proportion of our model's properly predicted entities. The general precise formula is.

$$Accuracy = \frac{Correctly\_Predicted}{Total\_Predictions}$$

Precision is the proportion of relevant instances among the examples found, whereas recall is the fraction of the instances found.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

The F1 score can be regarded as the weighted averaged accuracy and recall when an F1 mark reaches best at 1 and worse at 0. The F1 mark is compared to a specific positive class.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

4 Implementations

This section explains the steps involved to process the processed approach. The steps carried out are given as follows.

4.1 Model Setup

Firstly, the workstation computer is a Lenovo laptop with 12 GB RAM, intel i3 core 8th generation with 6 GB intel UHD 620 GPU and 1024 GB HDD. And used Google Collab with 12.69 GB RAM and 68.35 GB SSD storage. Anaconda 3 64-bit environment and its Jupyter Notebook 3 (Anaconda 3) help run the python 3.7 version codes. And along with python following libraries are used i.e., os, NumPy, pandas, seaborn, sklearn, matplotlib, TensorFlow and Keras.

4.2 About Dataset

The most challenging task in performing named entity recognition is the availability of annotated dataset. This challenge has to be overcome by manually tagging each word in a sentence and the format of the file used in txt format. Each dependent variable is separated by a space and has five dependent variables. The variables used in the dataset are explained as follows:

Table 1. Details of KNER dataset

Variables in the dataset	Description of variables
Sentence#	Indicates the sequence of sentence number
Word	Original Kannada words used in the literature
POS	Indicates grammatical structure of word
Tag	Indicates type of the entity belong

4.3 Data Cleaning

The dataset is created manually by taking words from Wikipedia and time takingly tagged each word with respective 'POS' and 'Tag'. The dataset has no missing values, no new columns are added and altered data. And, some words are neither an entity nor a proper word are filled with 'na'.

4.4 Visualization of data

To better understanding data visualizations shows pictorial representation and KNER data is not a time series data, hence the visualizations are limited to the histogram.

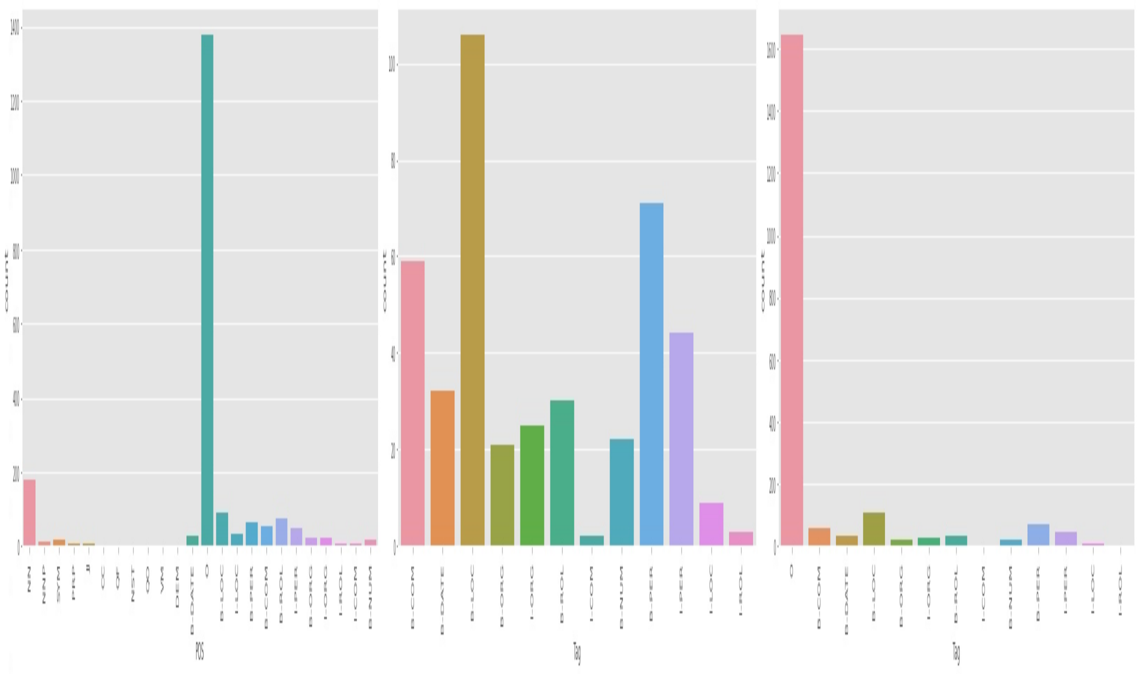
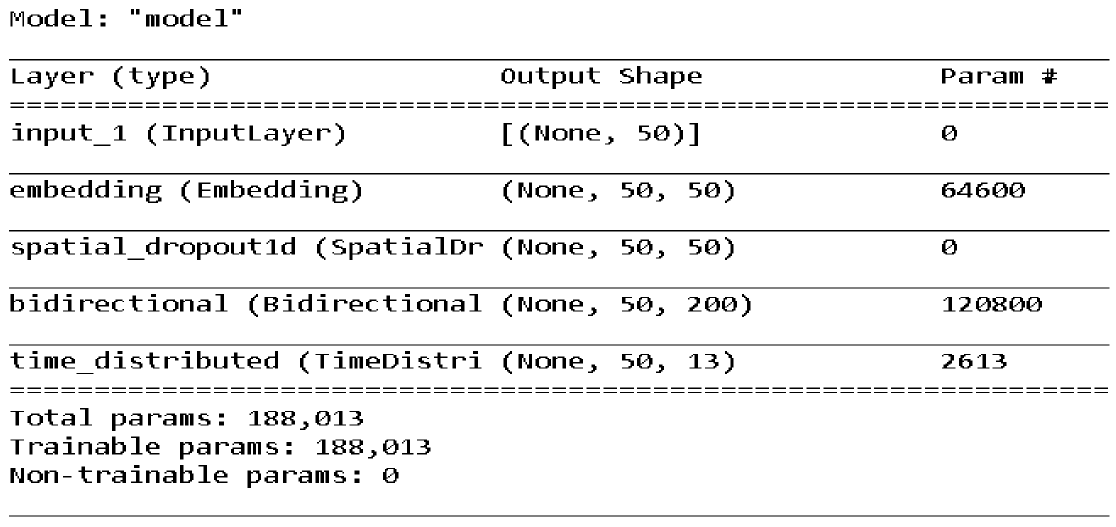


Figure 5. Count plot of labels with POS, types of tags without label and types of tags used respectively.

4.5 Modelling

An important model investigated in this research study is the Bi-LSTM model. Bi-LSTM can be able to predict entities present in the test dataset more accurately and comparatively. As the name says, two layers of LSTM are used. The description of the Bi-LSTM model for KNER goes this way, the shape and maximum length of the sentence is getting from the input word, to overcome overfitting, the dropout is set to 0.1. The learning rate is set to 100. Time distributed function and 'softmax' activation are performed on the dense operation. The rest parameters are calculated using the model function and the model architecture is shown in figure 6.



```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 50)]              0
_________________________________________________________________
embedding (Embedding)        (None, 50, 50)            64600
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 50, 50)            0
_________________________________________________________________
bidirectional (Bidirectional (None, 50, 200)           120800
_________________________________________________________________
time_distributed (TimeDistri (None, 50, 13)            2613
=================================================================
Total params: 188,013
Trainable params: 188,013
Non-trainable params: 0
_________________________________________________________________
```

Figure 6. Parameters implemented in the Bi-LSTM KNER model

Table 2. KNER model hyperparameters

Model	Hyperparameters	Values
Random Forest Classifier	n_estimators	20
Random Forest Classifier	CV	5
Bi-LSTM	spatialDropout1D	0.1
Bi-LSTM	recurrentDropout	0.1
Bi-LSTM	activation	softmax
Bi-LSTM	metrics	accuracy
Bi-LSTM	monitor	val$_{a}ccuracy$
Bi-LSTM	batch_size	64
Bi-LSTM	epochs	100
BERT	model	bert_base_multilingual_cased
BERT	train, eval batch_size	32
BERT	epochs	3

4.6 Train and Test Data

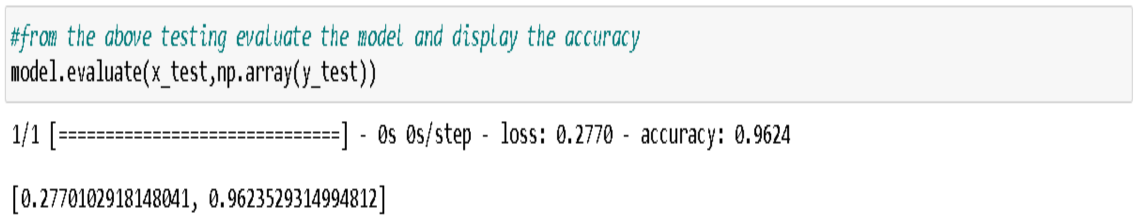
The data set contains a total of 2070 records. For analysing the Bi-LSTM model for training purposes first 10 per cent of the dataset is selected. The ratio of training and the testing dataset is 10:90. This aspect ratio has given good outcomes and this testing and training data split to fit them better for or research studies. During training, the model is supplied with 90 percent of the dataset and the model has accurately predicted 96.23 per cent accuracy. For the BERT model, the test data is split into 20 per cent. And train data is split into 80 per cent. The learning rate is set to 0.0001, train and evaluation batch is 32.

5 Result and Evaluation

This section explains the model outcome and evaluation metrics of this study. There are three models applied on this dataset, the evaluation metrics are based on two deep learning models and one machine learning model. The result of these models is explained as follows.

5.1 Model 1: Bi-LSTM deep learning model

In this Bi-LSTM KNER model, the values used to build the model are, maximum length = 50, x = word_idx, y = tag_idx. The outcome of the model with an accuracy of 0.9623 and value loss of 0.2770.



```
#from the above testing evaluate the model and display the accuracy
model.evaluate(x_test,np.array(y_test))
```

```
1/1 [==============================] - 0s 0s/step - loss: 0.2770 - accuracy: 0.9624

[0.2770102918148041, 0.9623529314994812]
```

Figure 7. Accuracy obtained on Bi-LSTM KNER model.

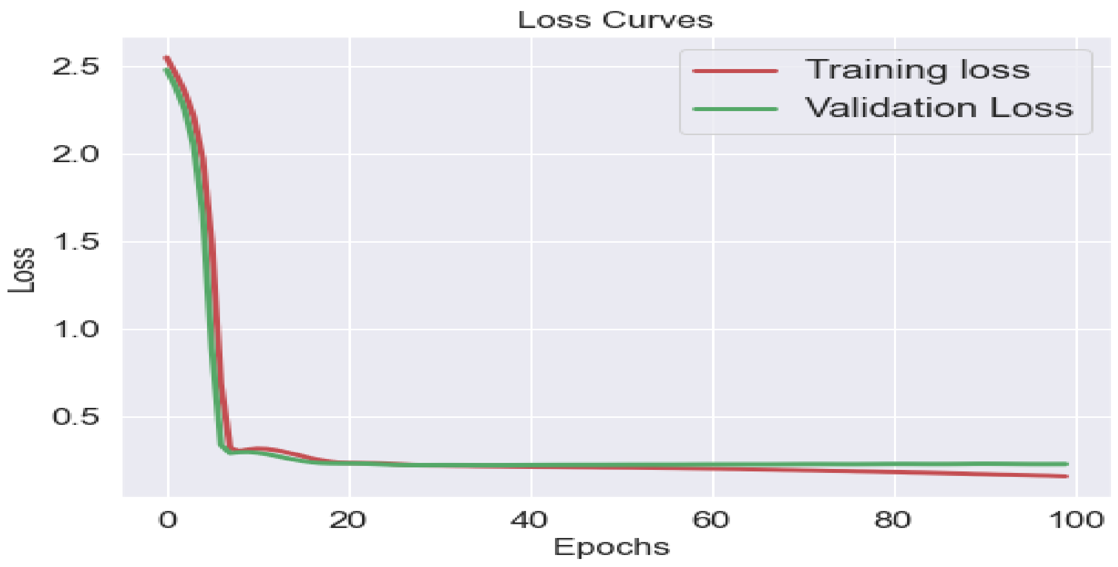
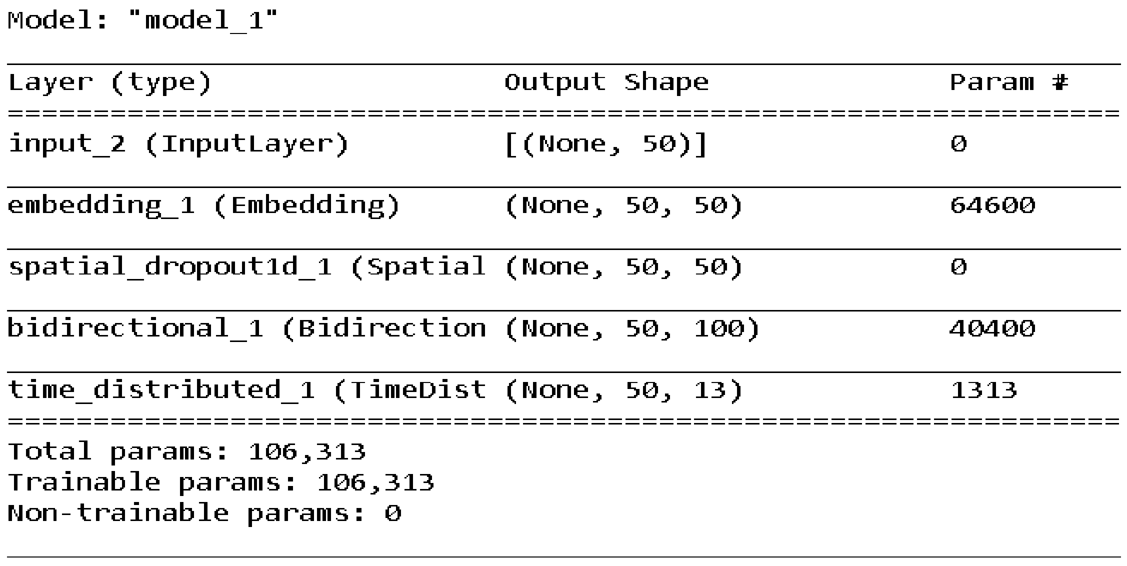


Figure 8. Plot of model performance.

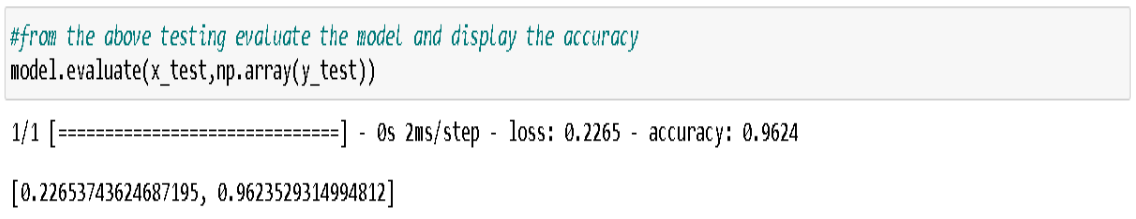
5.1.1 Experiment 1 on Bi-LSTM model to improve accuracy

This experiment aims to improve the accuracy and improve the loss. Hyperparameters are modified to get the best model parameter for this model. The hyperparameters modified are dropout, learning batch, train and test split size, batch size, epochs. The values of the above-mentioned parameters are 0.5, 50, 0.5, 32, 50. The results from these parameters are accuracy of 0.9623, loss value of 0.2265.



```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 50)]              0
_________________________________________________________________
embedding_1 (Embedding)      (None, 50, 50)            64600
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 50, 50)            0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 50, 100)           40400
_________________________________________________________________
time_distributed_1 (TimeDist (None, 50, 13)            1313
=================================================================
Total params: 106,313
Trainable params: 106,313
Non-trainable params: 0
_________________________________________________________________
```

Figure 9. The values of hyperparameters modified.



```
#from the above testing evaluate the model and display the accuracy
model.evaluate(x_test,np.array(y_test))
```

```
1/1 [==============================] - 0s 2ms/step - loss: 0.2265 - accuracy: 0.9624

[0.22653743624687195, 0.9623529314994812]
```

Figure 10. Accuracy of the modified model.

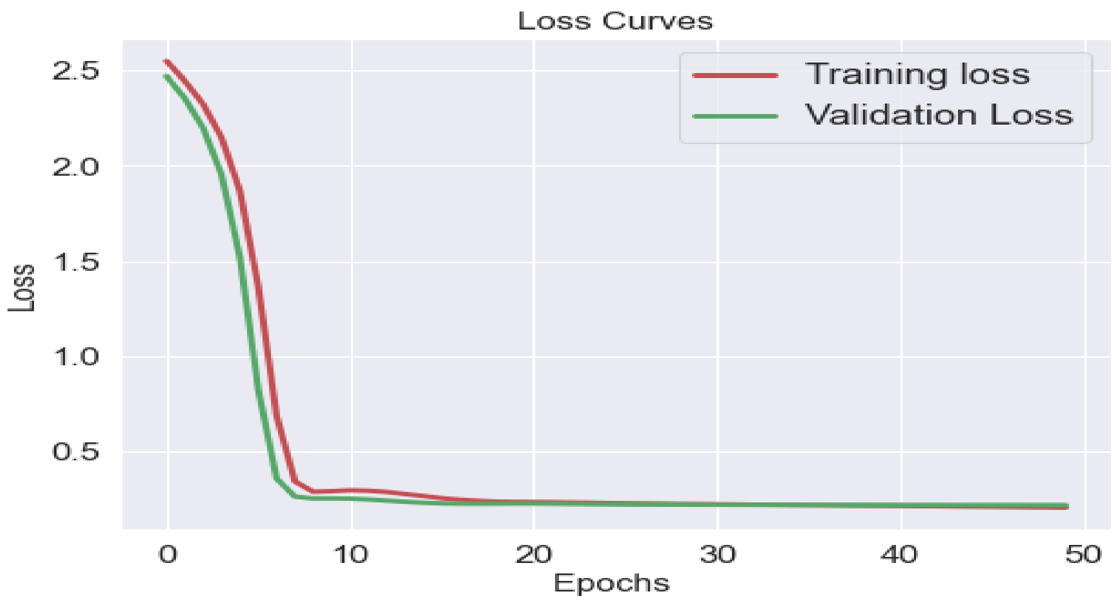
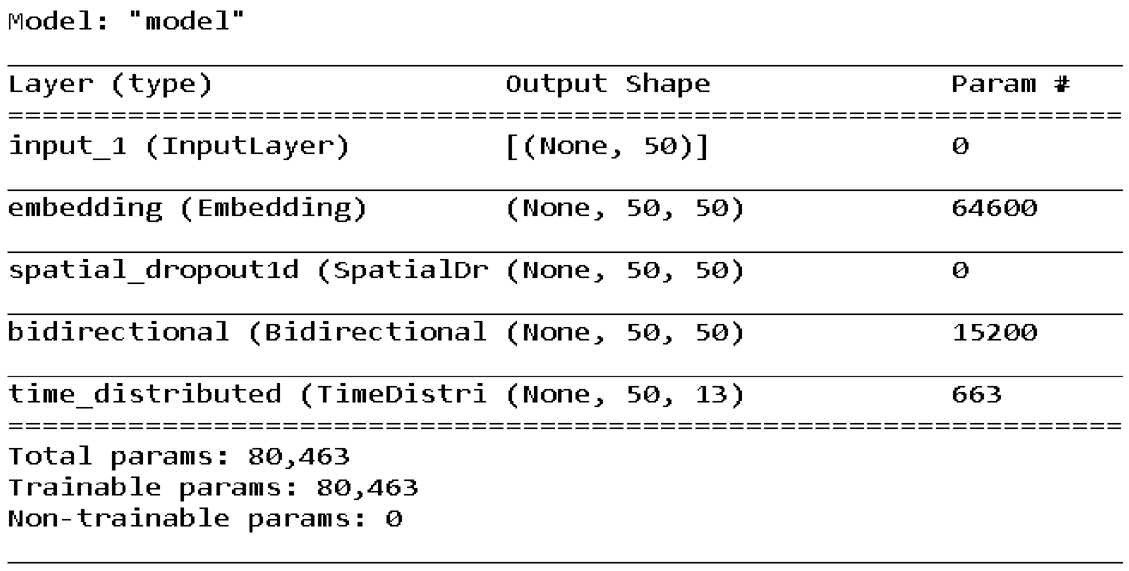


Figure 11. The graphical representation of model's epochs versus loss.

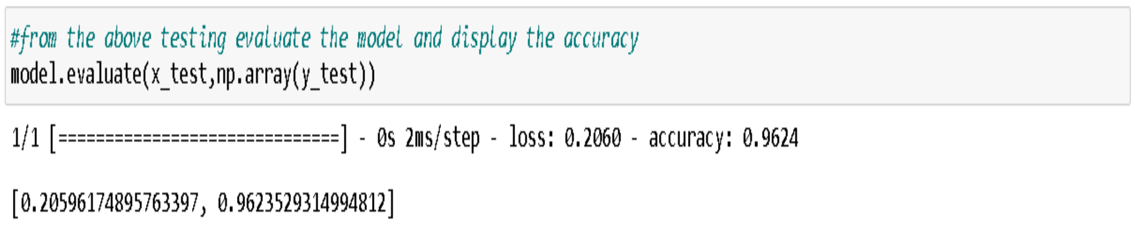
5.1.2 Experiment 2 on Bi-LSTM model to improve accuracy

The hyperparameters used in the experiment 2 are, dropout = 0.2, learning size = 25, validation split = 0.2, batch size = 10, epochs = 30. The result from this test is an accuracy of 0.9635 and loss 0f 0.2059. The hyperparameters used, the accuracy obtained, and the training and validation loss plot is given as follows.



```
Model: "model"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_1 (InputLayer)        [(None, 50)]              0
_________________________________________________________________
embedding (Embedding)       (None, 50, 50)            64600
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 50, 50)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 50, 50)           15200
_________________________________________________________________
time_distributed (TimeDistri (None, 50, 13)           663
=================================================================
Total params: 80,463
Trainable params: 80,463
Non-trainable params: 0
_________________________________________________________________
```

Figure 12. The tuned hyperparameters of the model.



```
#from the above testing evaluate the model and display the accuracy
model.evaluate(x_test,np.array(y_test))
```

```
1/1 [==============================] - 0s 2ms/step - loss: 0.2060 - accuracy: 0.9624

[0.20596174895763397, 0.9623529314994812]
```

Figure 13. Screenshot of the accuracy of the experiment 2 model.

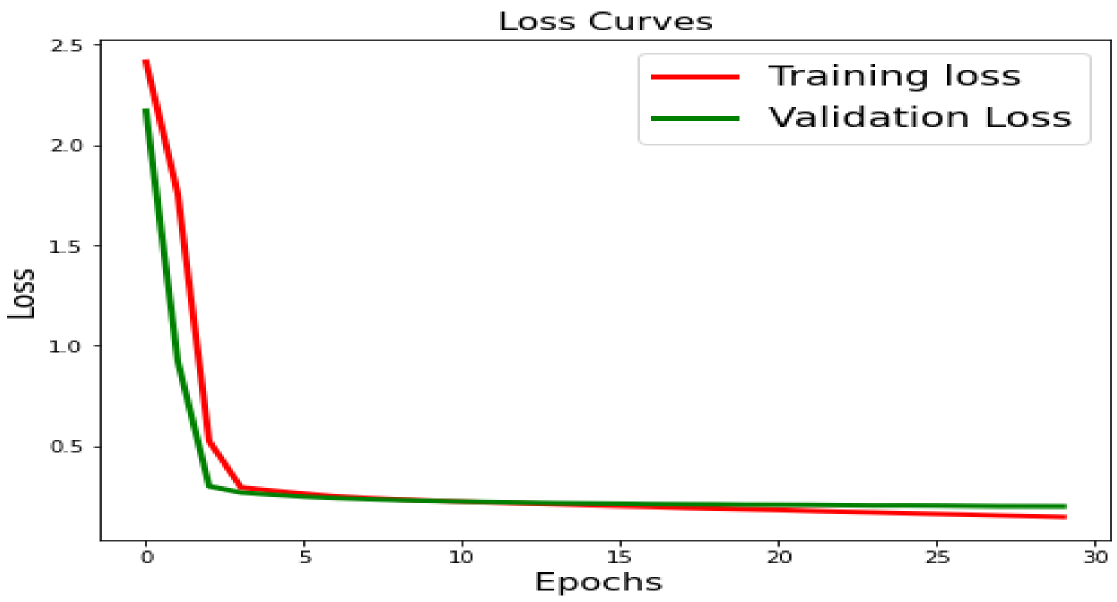
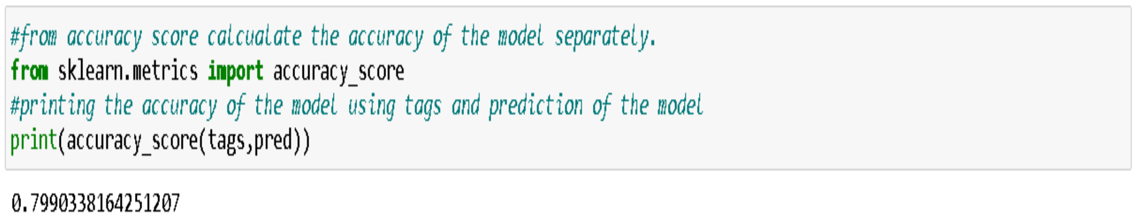


Figure 14. The plot representation of validation loss versus training loss.

5.2 Model 2: Random Forest Classifier Machine Learning model

In Random Forest Classifier KNER model, the values used are, x = words, y = tags, c.v = 5. The output of the RFC model with an accuracy of 0.7990, precision of 0.73 for B _DATE, f1_score of 0.37.



```
#from accuracy score calcualate the accuracy of the model separately.
from sklearn.metrics import accuracy_score
#printing the accuracy of the model using tags and prediction of the model
print(accuracy_score(tags,pred))
```

```
0.7990338164251207
```

Figure 15. Accuracy obtained from Random Forest Classifier model.

5.3 Model 3: BERT Deep Learning model

From the BERT KNER model, the values obtained are precision of 0.3617, recall of 0.1770 and f1-score of 0.2377. This indicates for this dataset BERT model is not performing efficiently. By comparing above mentioned three methods, it is seen that the Bi-LSTM model has outperformed the other two models with an accuracy of 0.9623 and the second-best model that fits well with the dataset is Random Forest Classifier with an accuracy of 0.7990.

5.4 Discussion on Models

In this experiment, two deep learning models and one machine learning model is used to extract the named entities in the given dataset. For this purpose, first a deep learning model i.e., the Bi-LSTM model is applied, and the accuracy obtained is 0.9623. From experiments 1 and 2, the accuracy of the model increased by 0.0001 and improvement in the loss was found to be 0.071. This improvement is small but, attempts were made to improve the primary model. Following the test on Random Forest Classifier with an accuracy of the model 0.7990 and precision of 0.73 for B_DATE. And final model of BERT with the precision of 0. 3617. By comparing the models, work on BI-LSTM and Random Forest Classifier has yielded substantially greater accuracy compared to BERT on the given dataset. This indicates 'softmax' and 'adam' optimizers for the hyperparameters is a better approach. This also states that Random Forest Classifier model this approach will predict more entities such as B_DATE more effectively than others.

6 Conclusion and Future work

This research study aims to achieve a novel deep learning model to extract named entities like person name, location, organisation, and date from the given sentence. The models applied have given a satisfactory result. The prominent model is the Bi-LSTM deep learning model with an accuracy of 0.9624. And another machine learning model called Random Forest Classifier to classify the named entities has an accuracy of 0.7990. And precision obtained from the RFC model for an entity B_DATE is 0.730. This work on the Kannada named entity recognition has given a fruitful result. Future effort to enhance precision by providing a wide range of named entities to strengthen the BERT model. And investigate a deep learning hybrid model for improved accuracy and precision.

7 Acknowledgement

I would like to take this opportunity to thank Professor Dr Christian Horn for his assistance and helpful supervision. And for constant support to me throughout the thesis duration. Valuable inputs and help in improving the thesis results. And also thankful for motivating me to push myself. Furthermore, I am thankful to my parents Mr Ramachandra S Kulkarni and Mrs Vanishree R Kulkarni with their encouragement and support I accomplished my master's study.

References

Aras, Gizem et al. (2021). “An evaluation of recent neural sequence tagging models in Turkish named entity recognition”. In: *Expert Systems with Applications* 182, p. 115049.

Bansal, SAshish (2021). “ADVANCED NATURAL LANGUAGE PROCESSING WITH TENSORFLOW 2 :” in: *Packt Publishing Limited*, p. 1290121.

Jin, Guozhe and Zhezhou Yu (2021). “A Korean named entity recognition method using Bi-LSTM-CRF and masked self-attention”. In: *Computer Speech & Language* 65, p. 101134.

Santoso, Joan et al. (2021). “Named entity recognition for extracting concept in ontology building on Indonesian language using end-to-end bidirectional long short term memory”. In: *Expert Systems with Applications* 176, p. 114856.

Zhang, Qiang et al. (2021). “Named entity recognition method in health preserving field based on BERT”. In: *Procedia Computer Science* 183, pp. 212–220.

Sreeja, PS and Anitha S Pillai (2020). “Towards an efficient Malayalam Named Entity Recognizer Analysis on the Challenges”. In: *Procedia Computer Science* 171, pp. 2541–2546.

Wintaka, Deni Cahya, Moch Arif Bijaksana and Ibnu Asror (2019). “Named-Entity Recognition on Indonesian Tweets using Bidirectional LSTM-CRF”. In: *Procedia Computer Science* 157, pp. 221–228.

Atmakuri, Shriya et al. (2018). “A comparison of features for POS tagging in Kannada”. In: *International Journal of Engineering & Technology* 7.4, pp. 2418–2421.

Chowdhury, Shanta et al. (2018). “A multitask bi-directional RNN model for named entity recognition on Chinese electronic medical records”. In: *BMC bioinformatics* 19.17, pp. 75–84.

Pallavi, KP, L Sobha and MM Ramya (2018). “Named Entity Recognition for Kannada using Gazetteers list with Conditional Random Fields.” In: *J. Comput. Sci.* 14.5, pp. 645–653.

Todi, Ketan Kumar, Pruthwik Mishra and Dipti Misra Sharma (2018). “Building a kannada pos tagger using machine learning and neural network models”. In: *arXiv preprint arXiv:1808.03175*.

Bharadwaj, Akash et al. (2016). “Phonologically aware neural model for named entity recognition in low resource transfer settings”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1462–1472.

Amarappa, S and SV Sathyanarayana (2015). “Kannada named entity recognition and classification (nerc) based on multinomial na\” ive bayes (mnb) classifier”. In: *arXiv preprint arXiv:1509.04385*.

Melinamath, Bhuvaneshwari C (2014). “Rule based Methodology for Recognition of Kannada Named Entities”. In: *International Journal of Latest Trends in Engineering and Technology* 3.4.