International Journal of Data Science and Analytics (2022) 14:127–140
https://doi.org/10.1007/s41060-021-00305-w

REGULAR PAPER

Telugu named entity recognition using bert

SaiKiranmai Gorla[1] · Sai Sharan Tangeda[1] · Lalita Bhanu Murthy Neti[1] · Aruna Malapati[1]

Received: 16 April 2021 / Accepted: 16 December 2021 / Published online: 18 January 2022

Abstract
Named entity recognition (NER) is a fundamental step for many Natural Language Processing tasks that aim to classify words into a predefined set of named entities (NE). For high-resource languages like English, many deep learning architectures have produced good results. However, the NER task has not yet achieved much progress for Telugu, a low resource Language. This paper performs the NER task on Telugu Language using Word2Vec, Glove, FastText, Contextual String embedding, and bidirectional encoder representations from transformers (BERT) embeddings generated using Telugu Wikipedia articles. These embeddings have been used as input to build deep learning models. We also investigated the effect of concatenating handcrafted features with the word embeddings on the deep learning model's performance. Our experimental results demonstrate that embeddings generated from BERT added with handcrafted features have outperformed other word embedding models with an F1-Score 96.32%.

Keywords Named entity recognition · Telugu · Word2vec · Glove · FastText · Contextual string embeddding · BERT

1 Introduction

The global web generates an enormous amount of unstructured text content every day. This content is not helpful if there are no tools and techniques for searching and indexing the text. The main aim of Information Extraction (IE) is to extract meaningful information from unstructured data. Named Entity Recognition (NER) is a subtask of IE which identifies noun phrases and classifies them into pre-defined semantic categories like the person, location, organization, etc. In 1995, the 6th Message Understanding Conference (MUC-6) [1] had coined the term Named Entity (NE). NER is crucial to many natural language processing applications such as knowledge base construction, information retrieval, automatic summarization, question answering system, machine translation, etc. Most of the traditional methods to build the NER model depend on statistical models such as the Hidden Markov Model (HMM), Conditional Random Field (CRF), Support Vector Machine (SVM), etc. These methods use labeled NER corpus to construct statistical models that take input as hand-crafted features such as lexical feature, orthographic pattern, character level affixes of the target and surrounding words, part-of-speech, etc.

Unlike traditional statistical methods, deep learning methods do not rely on feature engineering but automatically learn features where each word is represented by a vector trained on large unannotated corpora capturing semantic and syntactic information. These methods take input as a word vector, encode the entire sentence through a neural network, use each unit's output as a NER feature, and finally output each word's NE label. NER for the English language based on the deep learning model has been widely researched. However, for South-East Asian languages (especially Telugu), there has not been much progress. Since Telugu is a highly inflectional and rich language, many linguistic challenges occur while training models. Table 1 provides examples that more specifically illustrate the real challenges of the Telugu language.

Unlike English, there are no indicators for tags like capitalization in the Telugu language. For instance, *Capitalization* in the table, the words *cennail* and *tenAli* refer to *Chennai* and

✉ SaiKiranmai Gorla
p2013531@hyderabad.bits-pilani.ac.in

Sai Sharan Tangeda
f20170241@hyderabad.bits-pilani.ac.in

Lalita Bhanu Murthy Neti
bhanu@hyderabad.bits-pilani.ac.in

Aruna Malapati
arunam@hyderabad.bits-pilani.ac.in

[1] Department of Computer Science and Information Systems, Birla Institute of Technology and Science Pilani, Hyderabad Campus, Hyderabad, India

Tenali, names of locations. The English words *Chennai* and *Tenali* are capitalized, and clearly understand that they are location names. Therefore, they may be labeled as a location through NER models trained in the English language. On the other hand, by observing the Telugu sentences, it is unclear which part-of-speech (POS) each word means by looking at its appearance. While neural networks no longer rely on capitalization, we still find this issue valid for Telugu as there are no indicators. Telugu's typical characteristics are that the same word may refer to multiple meanings based on the context. In *Multiple Meaning* in the table, both sentences have the same highlighted Telugu word *pUja*, which refers to *pooja* as worship a common noun in the first sentence and *Pooja* the name of a person in the second sentence as a proper noun. It is possible to detect in English as it can distinguish proper nouns from common nouns using capitalization. Telugu is a relatively free word order language. The primary word order is SOV (subject-object-verb), but the word order of subject and object is mostly free. Internal changes or position swaps among words in sentences or phrases will not affect the sentence's meaning, as shown in the table in *Sentence Structure*. Besides, Telugu has highly inflected words. Depending on the context of the sentences, various prepositions are appended with proper names. For example, in sentences in *Inflection* in the table, the word Chennai is referred to different word shapes in Telugu, where both words mean Location. There are many words with different affixes in Telugu that add to the root word causing complex inflections.

Most Telugu NER models are Machine Learning (ML) models built with hand-crafted features represented in sparse dimensionality. However, the recent popularity of neural-based models where words are characterized with low dimensional and distributed representations that capture semantic and syntactic information has achieved superior results on various language-related tasks compared to traditional machine learning models.

Traditional word embeddings such as Word2Vec [2], Glove [3], and FastText [4] are context-independent (static). These models output just one vector for each word, combining all the word's different meanings into one vector. They fail to capture polysemy as they generate the same embedding for the same word in a different context. They fail to capture higher-level information like anaphora, long-term dependencies, agreement, negation, and many more. To address polysemous, some architectures used contextualized word (dynamic) embeddings built from language models to capture the semantics information of words in different contexts. The dynamic word embedding output is the trained model and vectors, which can be used in any downstream task. Flair embeddings [5], Elmo [6], BERT [7], etc., are contextualized word embeddings.

BERT (Bidirectional Encoder Representations from Transformers) [7] is a contextualized word representation constructed using bidirectional transformers [8] by making use of masked-language model. The bidirectional representation of words is pre-trained from unlabeled data by joining both the left and right contexts. Previous language models are trained unidirectional (i.e., left-to-right and right-to-left). The bidirectionally trained nature of the BERT model can have a more profound sense of the language context. Particularly for NER, contextual embeddings can help a model understand each word's context based on its neighbors in the sentences, thus helping to understand the difference in NER tag classes.

In this paper, we would build Telugu NER models by applying deep learning techniques. We have generated the embeddings using Telugu Wikipedia articles from scratch. Besides these embeddings, we also incorporated handcrafted features (e.g., gazetteer features, word-level features) into the final representations of words before feeding them into context encoding layers. The contributions of our paper are as follows:

– We build word embeddings like Continuous Bag-of-Words (CBOW), Skip-gram model, Glove, FastText, and Contextual String embedding using Telugu Wikipedia articles.
– We present a neural sequence labeling architecture for Telugu NER using above-mentioned embeddings. It consists of three layers, i.e., a character sequence representation layer, a word sequence representation layer using LSTM/CNN, and a Final layer. We also incorporated handcrafted features such as gazetteer features, word-level features, along word embeddings.
– Extensive experiments were conducted and investigated the effect of adding handcrafted features to word embeddings, and results show that FastText has outperformed other embeddings.
– We present the Telugu BERT-NER model pretrained using Telugu Wikipedia articles using the masked language model and finetuned to the Telugu NER model.
– Different models were built using varied network architectures of Telugu BERT-NER with and without handcrafted features and compared the performance of the models.
– A comparative study was performed using different word embeddings and investigated the effect of adding handcrafted features. The results show that BERT has outperformed other models.

The remaining section is structured as follows: Related work is covered under Sect. 2. Section 3 illustrates the architecture used to construct the Telugu NER. Details on the dataset, evaluation metrics, and experiments are presented in Sects. 4 and 5. The conclusion and focus of future work are discussed in Sect. 6.

2 Related work

This paper describes NER in the Telugu language using BERT. Thus, in this section, we first discuss some related work on Telugu NER, then some recent English NER work using deep learning and different word embeddings. NER models can be built using handcrafted rules or machine learning approaches. Previous works explored machine learning techniques for the Telugu language. Srikanth and Murthy [9] were some of the first authors to explore NER in Telugu. They built a two-stage classifier tested using the LERC-UoH (Language Engineering Research Centre at the University of Hyderabad) Telugu corpus. In the early stage, they created a noun identifier classifier using CRF, which was trained on manually tagged data of 13,425 words and tested on 6223 words. They then developed a rule-based NER system for Telugu, where their primary focus was on identifying the name of a person, location, and organization. A manually verified NE-tagged corpus of 72,157 words was used to develop this rule-based tagger through boot-strapping. They then created a CRF-based NER system for Telugu using manually generated features such as prefix\suffix, orthographic information, and gazetteers and reported an F1-score 88.5%.

Praneeth et al. [10] proposed a CRF-based NER model for Telugu using contextual word of length three, prefix\suffix of the current word, POS, and chunk information. They conducted experiments on data released as a part of the NER for South and South-East Asian Languages (NERSSEAL) competition with 12 classes. The best performing model gave an F1-Score of 44.91%. Raju et al. [11] created a 30,000 words training data set using news articles with the probable names of four NEs, person, location, organization, and miscellaneous. The Telugu NER model is built using CRF using the following features like suffix list, context features, POS, and a gazetteer list of person, location, and organization constructed manually. Sasidhar et al. [12] proposed a two-phase ruled-based NER model in Telugu. In the first phase, noun identification using Telugu dictionaries, morphological noun stemmer, and noun suffixes. The second phase identifies NEs with transliterated lists of NEs, various suffix features, contextual features, and morphological attributes.

SaiKiranmai et al. [13] built a Telugu NER model using three classification learning algorithms (i.e., CRF, SVM, and ME) on the data set provided as a part of the NER for South and South-East Asian Languages (NERSSEAL) competition. The model is built using contextual information, POS tags, morphological information, word length, orthogonal information, and sentence information. The results show that the SVM achieved the best F1-Score of 54.78%.

SaiKiranmai et al. [14] developed a NER model that classifies textual content from online Telugu newspapers using a well-known generative model. They used generic features like contextual words and their POS tags to build the learning model. Understanding the Telugu language's syntax and grammar introduced some language-dependent features like post-position features, clue word features, and gazetteer features to improve the model's performance. The model achieved an overall average F1-Score of 88.87% for a person, 87.32% for location, and 72.69% for organization identification. SaiKiranmai et al. [15] attempted to cluster NEs based on semantic similarity. They used vector space models to build a word-context matrix. The row vector was constructed with and without considering the different occurrences of NEs in a corpus. Experimental results show that the row vector considering various occurrences of NEs enhanced the clustering results.

SaiKiranmai et al. [16] attempts to improve Telugu NER performance using gazetteer-related features, which are generated dynamically using Wikipedia articles. To build NER models, they use these gazetteer features and other well-known features like contextual, word-level, and corpus features. Using three well-known classifiers-conditional random field (CRF), support vector machine (SVM), and margin infused relaxed algorithms (MIRA), NER models are built. The experimental results on two data sets show that the gazetteer features improved the NER models' performance. With the proposed gazetteer features, the performance (F1-score) built using MIRA, SVM, and CRF increased by 14.72%, 15.95%, and 17.13%, respectively.

Reddy et al. [17] describe an LSTM-CRF-based approach for NER in Telugu. They use pre-trained fastText embeddings and character-level embeddings generated using a Bi-LSTM to learn contextual word embeddings using another Bi-LSTM. A linear chain CRF is finally used to perform the tagging based on the contextual word embeddings. Various word embeddings and a CRF allow the classifier to capture more contextual information and tagging dependencies, respectively. The overall F-measure achieved using the LSTM-CRF classifier is 85.13%.

Neural network systems have become popular, starting with Collobert et al. [18] with minimal feature engineering and contribute to higher domain independence. Santos and Guimaraes [19] extended the work of Collobert et al. citesjmlr2011 where a convolutional layer extracts the character level features of each word and concatenates with pre-trained word embeddings for sequential classification. The commonly used deep learning architecture for the NER task [17,20,21] is LSTM-CRF [22]. The model consists of bi-directional LSTM networks that extract word and character level features. In the final layer, CRF was used for sequence classification.

Initial work on LSTM-CRF [22] architecture for the NER task used static embedding [4,23,24], which cannot be dynamically optimized for specific tasks. The words and vectors are one-to-one relationships that do not solve the problem of polysemous words. Recent work explored dynamic embedding (contextual embeddings) from language models along with LSTM-CRF architecture. Akbik et al. [5] trained words as sequences of characters by their surrounding text, which have different embeddings depending on its context for the same word. These embeddings are concatenated with pre-trained word embeddings and fed to a BiLSTM-CRF model. Peters et al. [25] uses ELMo, a deep contextualized word representation that uses complex characteristics of a word (e.g., syntax and semantics). These word vectors are learned using a bidirectional language model (biLM). This model is pre-trained on large corpora, easily added to existing models to improve performance.

In addition to word and character representations, some studies also incorporate supplementary information (e.g., gazetteers and lexical similarity [26]) into final word representations before feeding to encoding layers. In Huang et al. [27] BiLSTM-CRF model, four features spelling features, context features, word embeddings, and gazetteer features for the NER task. The experimental results show that adding gazetteer features improved the performance. Strubell et al. [28] concatenated five-dimensional word shape vectors such as all capitalized, not capitalized, first-letter capitalized, or contained a capital letter with 100-dimensional embeddings. Aguilar et al. [29] utilize a CNN to capture orthographic features and word shapes at the character level. At the word level, they capture syntactical, contextual information and word embeddings using LSTM architecture.

In recent years, researchers have shown that transfer learning, i.e., pre-training a neural network model on a known task and then performing fine-tuning using the trained neural network, has improved many natural language tasks. One of the popular models using transfer learning is BERT. Lee et al. [30] introduces BioBERT, which is a pre-trained language representation model for the biomedical domain. They first initialized BioBERT with BERT weights, pretrained on public domain corpora (English Wikipedia and BooksCorpus). Then, BioBERT is pre-trained on biomedical domain corpora (PubMed abstracts and PMC full-text articles). Souza et al. [31] built Portuguese BERT models and used a BERT-CRF architecture for the NER task in the Portuguese language, combining BERT's transfer capabilities with CRF's structured predictions. They investigated the training strategies based on functionality and fine-tuning for the BERT model. Their approach obtains new state-of-the-art results on the HAREM-I dataset. De et al. [32] developed and evaluated a monolingual Dutch BERT model called BERTje. It is based on a large and diverse dataset of 2.4 billion tokens. BERTje constantly surpasses the multilingual BERT model of equal size for downstream NLP tasks. Li et al. [33] developed the Clinical Named Entity Recognition (CNER) which identifies and classifies clinical terms in electronic Chinese medical records. They pretrained the BERT model on unlabeled Chinese clinical documents using different layers such as Long Short-Term Memory (LSTM) and Conditional Random Field (CRF) to extract the text features and decode the predicted tags. In addition, they proposed a new strategy to consider dictionary features. Radical features such as Character-based joint segmentation and pos tagging of Chinese characters are used to improve the model performance.

3 Methodology

The Neural sequence labeling architecture is discussed in Sect. 3.1 followed by BERT architecture in Sect. 3.2 for Telugu NER. The handcrafted features used in the present work are described in Sect. 3.2.2.

3.1 Basic end-to-end model for Telugu NER

In this section, we briefly describe different word embeddings generated using Telugu Wikipedia articles in Sect. 3.1.1 and neural sequence architecture build using these embeddings for Telugu NER is described in Sect. 3.1.2.

3.1.1 Word embeddings

To build machine learning models for text applications, we need to convert text into a numeric representation. One of the most straightforward approaches is one-hot encoding, where every single word will have a vector, and a binary value indicates whether the word is present or not. However, the computation of this representation is challenging when dealing with thousands of words in the vocabulary. In comparison, word embeddings represent words and sentences into vectors with smaller and denser dimensions. The words with the same meaning have a similar representation. The various word representations used in this work are explained below. These representations are built using 71,785[1][2] Telugu Wikipedia articles with 33,854,720 tokens.

1. ***Word2Vec*** Tomas Mikolov et al. [23] developed one of the popular algorithms in the word embedding is Word2Vec. It proposes two different architectures: The Continuous-Bag-Of-Words model (CBOW) and the Skip-gram model. The architecture of the CBOW model attempts to predict the current target word based on the context of surrounding words.
   Considering a sentence,
   ఈ సమావేశం న్యూయార్క్ లో జరుగుతుంది (I samAvESam nyUyArk lO jarugutumdi) this can be pairs of (context_window, target_word) if we consider a context window of size

[1] https://dumps.wikimedia.org/tewiki/.

[2] https://meta.wikimedia.org/wiki/List_of_Wikipedias.

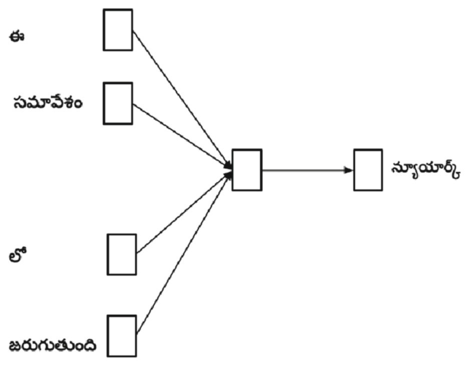

Fig. 1 Continuous-Bag-Of-Words model

2, we have examples like ([samAvEsam, lO], nyUyArk), ([I, nyUyArk], samAvEsam), etc. where the model tries to predict the target_word based on the context_window words as shown in Fig. 1.
The architecture of the Skip-gram model typically tries to achieve the inverse of what the CBOW model does. It tries to predict the surrounding words, i.e., context words given a target word. Considering a simple sentence, ఈ సమావేశం న్యూయార్క్ లో జరుగుతుంది (I samAvESam nyUyArk lO jarugutumdi) the skip-gram model predict the context [samAvESam, lO] given target word nyUyArk or [I, nyUyArk] given target word samAvESam, etc. where the model predicts the context_window words based on the target_word as shown in Fig. 2.

2. ***Glove*** GloVe [24] stands for Global Vectors for Word Representation. This model combines two major model advantages in the literature: global matrix factorization and local context window methods. This model effectively leverages statistical information by training only nonzero elements in the word-word-co-occurrence matrix rather than the entire sparse matrix or separate context windows in a large corpus.
3. ***FastText*** FastText [4] is an effectively used library for word representations and sentence classification. It supports CBOW or Skip-gram training models using negative sampling, softmax, or hierarchical softmax loss functions. FastText embeddings for the Telugu language were built using the Skip-gram model using negative sampling. The significant difference between fastText and the skip-gram model is that fastText uses character-level information. Each word in the sentence is presented in the form of a bag of character n-grams. For example, with n = 3, the character n-grams are <app, ppl, ple> for the word apple. It helps preserve the meaning of short words and captures the meaning of suffixes/prefixes.
4. ***Contextual String Embeddings*** Contextual Embedding [5] differs from classical word Embeddings like Word2 Vec, Glove, and FastText. This embedding captures the

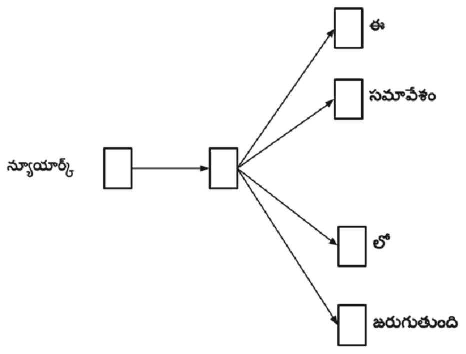

Fig. 2 Skip-gram model

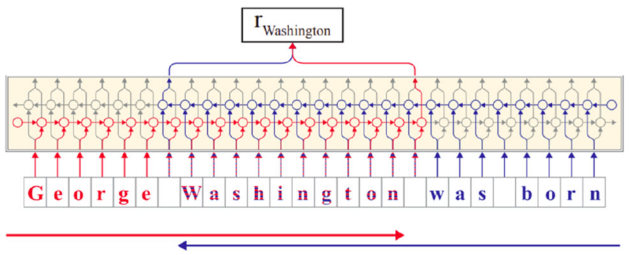

Fig. 3 Contextual embeddings [5]

semantic of words in a context in which the same word will have a different vector based on the context. Classical word embeddings produce a single vector for each word, combining all the different meanings of the word into a single vector. The contextual Embeddings model is based on character level tokenization. It converts sentences into a sequence of characters and uses the language model to learn word representation using bi-directional LSTM, as shown in Fig. 3.
Taking "Washington" as an example: bi-directional LSTM model allows "Washington" to retrieve information from previous words (i.e., George) and following words (i.e., was born) such that it can compute the vectors under a sentential context.

3.1.2 Neural sequence labeling architecture for Telugu NER

The Neural sequence labeling architecture for Telugu NER is designed with three layers: a character sequence layer, a word sequence layer, and the final layer. An example of the sentence నేను పూణే లో ఉన్నాను (*nEnu pUNE lO unnAnu*) using neural sequence labeling architecture is shown in Fig. 4. For every sequence of input words, words are represented by word inserts. The character sequence layer may automatically extract the word level characteristics by encoding the character sequence into the word. Concatenation of word embeddings (red circles), the sequence of characters encoding hidden vectors (yellow circles), and handcrafted features

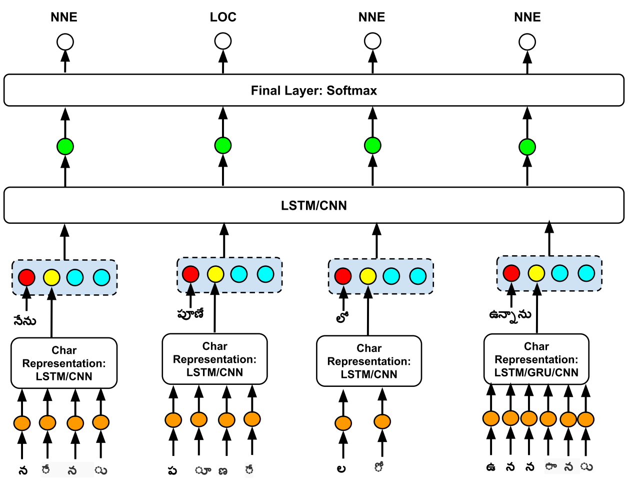
NNE
LOC
NNE
NNE
Final Layer: Softmax
LSTM/CNN
Char Representation: LSTM/CNN
Char Representation: LSTM/CNN
Char Representation: LSTM/CNN
Char Representation: LSTM/GRU/CNN

Fig. 4 Neural sequence labeling architecture

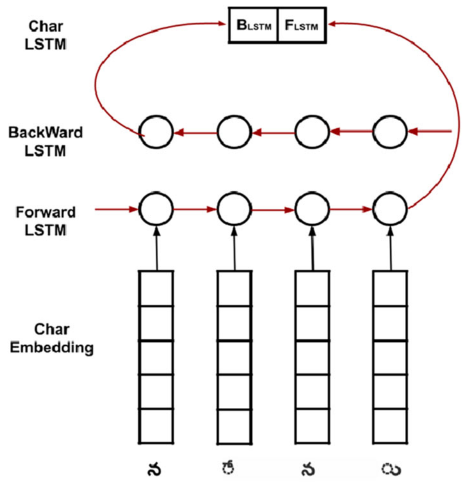

Fig. 5 Character LSTM

(blue circles) are word representations. The word sequence layer takes the word representations as input, extracts the sentence level features, and then feeds into the final layer to assign a label to each word.

Character sequence layer

The character sequence layer combines multiple neural encoders for character sequence information, like RNN and CNN. Characters are represented by character embeddings (yellow circles in Fig. 4) that act as input to the character sequence layer.

Character RNN The bidirectional RNN is used to capture sequence information from left to right and right to left. Both masked RNNs are concatenated and given as input to the character sequence layer as shown in Fig. 5.

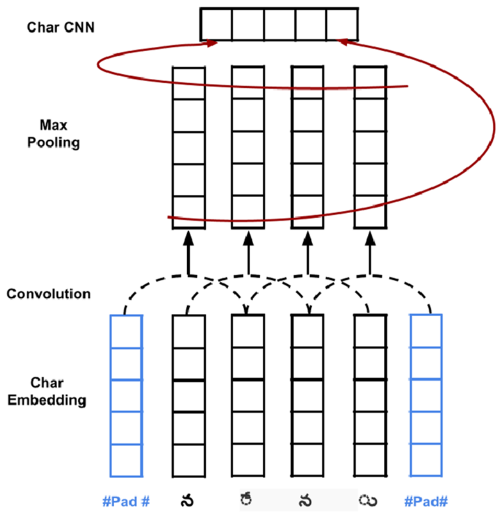

Fig. 6 Character CNN

Character CNN It requires a sliding window to capture the features and then uses maximum pooling in the character sequence's aggregate encoding as shown in Fig. 6.

We use "CLSTM" and "CCNN" to simplify the description to represent character LSTM and character CNN encoder.

Word sequence layer

In the character sequence layer, we can use RNN or CNN as the word sequence feature extractor. The word sequence layer input is a word representation, including word embeddings as explained in Sect. 3.1.1, character sequence representations, and handcrafted features. To simplify the description, we "WLSTM" and "WCNN" represent word sequence using LSTM and CNN.

Final layer

The Final layer takes the representations extracted from the word sequence as features and assigns tags to the word sequence. This layer first maps the input sequence representations to label vocabulary size scores, which are used to model each word's label probabilities through simple softmax or calculate the whole sequence's label score using CRF. In this work, we use softmax, which maps the label scores into a probability space. The neural sequence model for Telugu NER has experimented with different combinations of features such as only word embedding, word embedding + character sequence representation, word embedding + character sequence representation + handcrafted features. The experimental results are shown in Table 6.

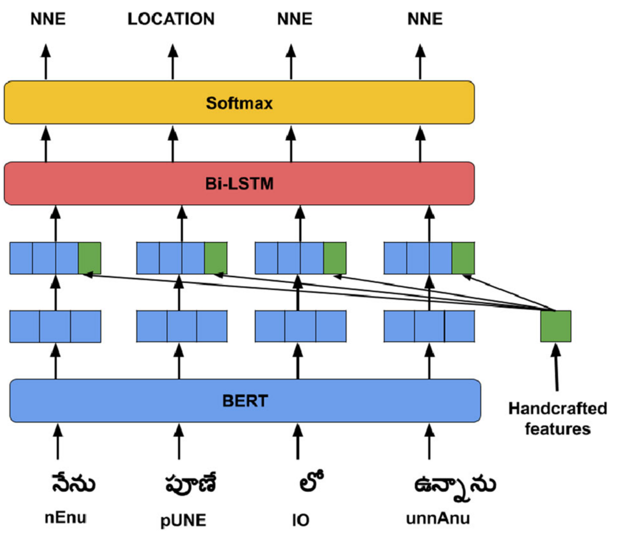

Fig. 7 The architecture of the proposed BERT model for Telugu named entity recognition

3.2 Telugu BERT model

The overall architecture of our Telugu BERT-NER model is shown in Fig. 7. Sentences taken from the Telugu NER dataset are first inserted into the BERT model to extract feature representations. These representations are then sent to a Bidirectional LSTM network. The outputs of the BiLSTM network are then sent to a Fully Connected (FC) layer to finally produce tag sequences. In the subsequent sections, the components of the architecture are described separately.

3.2.1 Bidirectional encoder representation from transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT). Devlin et al. [7] is a recently proposed architecture is based on transformer [8]. It produces deep bidirectional representations of words using unlabeled text based on contextual relationships with the words around them. In this work, we pre-train the BERT model on Telugu Wikipedia articles. An input sentence S that consists of n tokens w_i , where $i \in [1, n]$ is passed to BERT, where word embedding are generated for each word. It calculates each token's input embedding by summing over the token, position, and segment embedding, as shown in Fig. 8. Token embedding for each token is generated using Word Piece Tokenization [34] where each word is segmented into characters, and the most common combinations of characters are added iteratively to the vocabulary. In the Word Piece Tokenization method, the way the word Telugu is segmented into characters is described in the example below: For example the word in Telugu కిరణ్ ⇒ క ##ి ##ర## ణ##్. Position embedding includes the positional information of each token

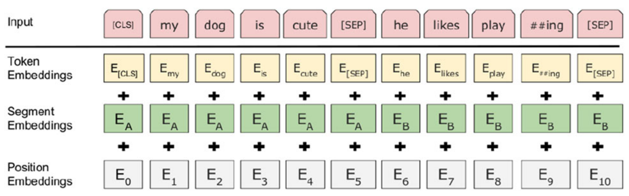

Fig. 8 BERT input representation [7]

in the sentence. Segment embedding provides the same label to the tokens that belong to one sentence.

Telugu BERT is pre-trained using a masked language model (MLM), where random words are masked, and the model attempts to predict those masked words. The BERT model is built using 12 layers with 768 hidden layers, 12 headed attention layers, and 110M parameters. For example, the sentence consists of words $W1$, $W2$, ..., $W5$. The word $W4$ is randomly masked using the [MASK] token. The output of the block is the sentence along with the predicted value of the masked token $O1$, $O2$, ..., $O5$.

In Fig. 7, BERT represents a pre-trained BERT model for Telugu, where we extract our input sentence representation for our Telugu NER task. For example, in the sentence S, the input sequence X is constructed as:

For a sentence S, the input sequence X is constructed as:

$$X = ([CLS], w_1, w_2, ...w_n, [SEP])$$

Here, $[CLS]$ and $[SEP]$ are special tokens that indicate the beginning and end of the sequence, respectively. Constructed sequence X of length $n+2$ is passed to BERT, where word embedding of each token is generated and then encoder layer outputs the hidden representation $H \in \mathbb{R}^{(n+2)\times d}$ for all the items of X.

$$H = BERT(X)$$

Here, d is the size of the hidden dimension of BERT word embeddings. We use the final hidden state h_{CLS} of $[CLS]$ token as the aggregate sequence representation of the input sentence S.

3.2.2 Handcrafted features

We extract the following handcrafted features for the given sentence S.

Word-level features Word-level features are related to the individual orthographic nature and structure of each word.

- *Word length* Kumar et al. [35] found that short words are most probably not NEs and predefined the threshold to be less than or equal to three. So, we considered

word length as a binary feature if the current word length ≥ 3.
- *Position of word* In a sentence, the position of a word acts as a good indicator for named entity identification, as NEs tend to appear in the first position of the sentence. In Telugu, verbs typically appear in the last position of the sentence, as it follows a subject-object-verb structure. So, we considered two binary features FirstWord and LastWord.
- *Part-of-speech (POS) of a word* Previous works in Telugu NER used POS features as a binary feature (i.e., whether a word is a noun or not a noun). The study by SaiKiranmai et al. [36] suggests that other part-of-speech tags like postposition, quantifiers, demonstratives, cardinal/ordinal, NST (noun denoting spatial and temporal expression), and quotative are helpful in identifying whether a given word is a named entity or not. So, in our work we used the TnT [37] POS tagger, which classifies a Telugu word into one of 21 POS tags.

Gazetteer features Gazetteers or entity dictionaries play an essential role in improving the performance of the NER task. Person, Location and Organization gazetteer are generated using Telugu Wikipedia Categories [16].

- *Person gazetteer* If the current word (w_i) is present in the person gazetteer, then the **Person** feature is set to 1.
- *Location gazetteer* If the current word (w_i) is present in the location gazetteer, then the **Location** feature is set to 1.
- *Organization gazetteer* If the current word (w_i) is present in the person gazetteer, then the **Organization** feature is set to 1.
- *Surname gazetteer* Surnames occur at the start of person names. For example, in అల్లు అరవింద్ (allu aravimd), అల్లు (allu) is the surname. If the current word (w_i) is present in the surname gazetteer, then the **Surname** feature is set to 1.
- *Person suffix gazetteer* The person suffix occurs at the end of a person's name. For example, in రామచంద్ర రెడ్డి (rAmacmdra reDDi), రెడ్డి (reDDi) is the person suffix. If the current word (w_i) is present in the person suffix gazetteer, then the **PerSuffix** feature is set to 1 for the current (w_i) and previous two words (w_{i-1}, w_{i-2}).
- *Designation gazetteer* Designation words represent the formal and official status of a person. For example, ప్రధానమంత్రి(pradhAnama mtri). If the current word (w_i) is present in the designation gazetteer, then the **Desig** feature is set to 1 for the next word (w_{i+1}).
- *Person prefix gazetteer* Person prefixes help in identifying person names (e.g., శ్రీమతి (SrImati). If the current word (w_i) is present in a person prefix gazetteer, then the **PerPrefix** feature is set to 1 for the current (w_i) and next two words (w_{i+1}, w_{i+2}).
- *Month gazetteer* The month gazetteer consists of the names of months of both English and Telugu calendars. There are 24 entries in this list. If the current (w_i) word is present in the month gazetteer, then the **Month** feature is set to 1.
- *Location clue gazetteer* The location clue gazetteer consists of the words that give clues about location names-for example, clue words like: -pur, -puram, -gunTa, -nagar, -paTnam (తిరువంతపురం (tiruvamta**puram**), కాన్పూర్ (Kan**pur**))). If the current (w_i) word contains any of the suffixes listed in the location clue gazetteer, then the **LocClue** feature is set to 1.
- *Organization clue gazetteer* Organization names tend to end with one of a few suffixes, such as సం□ష్థ (Company), సంఘం (Community). These were collected manually. The feature **OrgClue** is set to 1 for the current (w_i) and previous two words (w_{i-1}, w_{i-2}) if the current word (w_i) is present in the organization clue gazetteer.

Using above manually extracted features we create a feature vector $F \in \mathbb{R}^{34}$ of 34 dimensions and then combine it with hidden state h_{CLS} to generate new sentence representation $V = h_{CLS} \oplus F$.

3.2.3 Bidirectional LSTM

LSTM (Long-Short Term Memory) [38] is an effective network architecture for sequential information tasks as it can study long-term dependencies. LSTM network performs the computation in either one direction, whereas a variant of LSTM, i.e., BiLSTM, performs the computation in both directions of a sequence. Since the BiLSTM network can store both backward and forward hidden state information, it is advantageous in sequence tagging tasks like NER [2,3,22].

The BiLSTM Layer takes a sequence of the input sentence. In our case, for the Telugu NER task, the input representations are the concatenation of BERT embedding and handcrafted features $V = (V_1, V_2, ..., V_n)$. The network outcome is a set of hidden states for each input vectors $(h_1, h_2, ..., h_n)$. The final hidden state is the concatenation of the backward and forwards hidden states. Here, v_i denotes the i^{th} token of input V_i. Hence, the output becomes,

$$
\begin{aligned}
h_i^f &= LSTM(v_i, h_{i+1}^f) \\
h_i^b &= LSTM(v_i, h_{i-1}^b) \\
h_i &= concat(h_i^f, h_i^b)
\end{aligned}
$$

where h_i^f ,h_i^b denotes forward and backward hidden states of the LSTM network.

In NER task, the information regarding the preceding and following words of the current word is very much useful and the same is achieved by having BiLSTM as encoder layer as explained in Sect. 3.1.2

3.2.4 Linear layer

The BERT model is fed with an input sequence that produces contextual string embedding of each token. Then the contextual string embedding is concatenated with 34 handcrafted features. In our experiments, we forwarded these embeddings to a BiLSTM layer. The output of the BiLSTM is then projected to 5 neurons (one for each target class) using fully connected layers. The fully connected layer's output is then passed through a softmax activation function to get the labels' probability distribution. We also performed experiments where the BERT embeddings concatenated with 34 handcrafted features were projected down to 5 neurons using a fully connected layer with and without using a BiLSTM layer.

4 Experiments

4.1 Dataset

In this work, we have used the benchmarked data-set[3] provided by the Forum of Information Retrieval and Evaluation (FIRE-2018). The data consists of 767,603 tokens out of which 200,059 are NEs. The size of data-set is given in Table 2.

The data set is annotated with nine NE tags. A tag conversion routine has been implemented on the corpus to scale down nine initially available tag-set to the intended four tag-set namely name, location, organization, and miscellaneous as shown in Table 3.

4.2 Baseline methods

We perform a comparison of Telugu BERT-NERT with two groups of baselines. The first group of models is based on the statistical machine learning approach and uses CRF, SVM, and MIRA for classification. The model description is given below:

- **SaiKiranmai et al.** [16] It uses gazetteer-related features, which are automatically generated using Telugu Wikipedia pages. They use these gazetteer features and other well-known features like contextual, word-level, and corpus to build NER models. NER models are developed using three well-known classifiers-conditional random field (CRF), support vector machine (SVM), and margin infused relaxed algorithms (MIRA).

The second group of models is based on a basic end-to-end neural sequence network that uses pre-trained different word embeddings models, as explained in Sect. 3.1.1 and the architecture is explained in Sect. 3.1.2.

[3] http://fire.irsi.res.in/fire/2018/home.

Table 1 Examples of challenges in Telugu Language

Challenges	**Sentences**
Capitalization	నేను చెన్నైలో నివసిస్తున్నాను (nEnu cennailO nivasistunnAnu) I live in Chennai
	నా జన్మస్థలం తెనాలి (nA janmasthalam tenAli) My birthplace is Tenali
Multiple meaning	నా పేరు పూజ (nA pEru pUja) My name is Pooja
	నేను పూజ చేస్తున్నాను (nEnu pUja chEstunnAnu) I am doing pooja
Sentence structure	రాము సీతకు హారాన్ని పంపాడు (rAmu sItaku hAranni pampAdu) Ramu sent the necklace to Sita
	రాము హారాన్ని సీతాకు పంపాడు (rAmu hArAnni sItAku pampAdu) Ramu sent the necklace to Sita
Inflection	నేను బెంగళూరులోనే ఉన్నాను (nEnu bemguLUrunE unnAnu) I stay in Bengaluru
	నేను బెంగళూరుకు వస్తున్నాను (nEnu bemgaLUruku vastunnAnu) I am coming to Bengaluru

Table 2 Size of the data-set

Dataset	Train	Test	Total
No. of tokens	537,510	230,093	767,603
No. of entities	139,999	60,060	200,059

Table 3 Named entity tagset

Named Entity tag	Meaning	Example
Person	person name	ముకేష్ (Mukesh)
Location	location name	చైనా (China)
Organization	organization name	మైక్రోసాఫ్ట్ (Microsoft)
Miscellaneous	number, date, events, things, year and occupation	2021

The third group has variations of our proposed Telugu BERT-NER model, which compares the handcrafted features effect. Following are the variants of our model:

- **BERT** This model uses BERT embeddings, which are projected to a fully connected layer. The output from the fully connected layer is then passed through a soft-max activation function.
- **BERT + Word-Level features** This model uses BERT embeddings concatenated with Word-Level features, which are projected to a fully connected layer. The output from the fully connected layer is then passed through a soft-max activation function.
- **BERT + Word-Level features + Gazetteers features** This model uses BERT embeddings concatenated with Word-Level features and gazetteer features, which are projected to a fully connected layer. The output from the fully connected layer is then passed through a soft-max activation function.
- **BERT + BiLSTM** This model uses BERT embeddings, which is forwarded to a BiLSTM layer. The output of the BiLSTM is then projected to fully connected layers. The output from the fully connected layer is then passed through a softmax activation function.
- **BERT + Word-Level features + BiLSTM**: This model uses BERT embeddings concatenated with Word-Level features, which are forwarded to a BiLSTM layer. The output of the BiLSTM is then projected to fully connected layers. The output from the fully connected layer is then passed through a softmax activation function.
- **BERT + Word-Level features + Gazetteers features + BiLSTM** This model uses BERT embeddings concatenated with Word-Level features and gazetteer features, which is forwarded to a BiLSTM layer. The output of the BiLSTM is then projected to fully connected layers. The output from the fully connected layer is then passed through a softmax activation function.

4.3 Experimental settings

The data-set is split into training (70%) and testing (30%) randomly. We performed experiments on Ubuntu 18.04os, pytorch0.4.1, python 3.6, and Nvidia T4 Tensor Core GPU with 55GB of memory. Our proposed model utilizes BERT BASE to encode the given sentence, generating a 768-dimensional hidden state. We use 34 handcrafted auxiliary features. For the optimization, the AdamW optimizer is used with a learning rate of 0.001. We train our model for 30 epochs and a batch size of 32. The results reported for all models are the average over five runs.

Table 4 Total number of named entities in the test set

Named entity	Number
Person	18,141
Location	28,856
Organization	725
Miscellaneous	12,338
Total	60,060

Table 5 Performance of first group of models based on the statistical machine learning approaches

Approach	Precision	Recall	F1-Score
CRF [16]	95.87	83.88	88.80
SVM [16]	95.45	88.54	91.63
MIRA [16]	96.05	89.91	**92.66**

4.4 Evaluation metrics

The standard evaluation measures like Precision (P), Recall (R), F1-Score (F1) are considered to evaluate our NER model.

$$Precision(P) = \frac{c}{r}$$

$$Recall(R) = \frac{c}{t}$$

$$F1{-}Score = \frac{2 * P * R}{P + R}$$

where r is the number of NEs predicted by the system, t is the total number of NEs present in the test set and c is the number of NEs correctly predicted by the system.

5 Results and discussion

The data consisted of 767,603 tokens out of which 200,059 were NEs, and we trained the model with 70% of the data and tested on the remaining 30%. Five sets of training and testing data were generated using the annotated corpus. This split was done randomly and sentences were not repeated in the training and testing data. The total number of NEs in the test set are shown in Table 4.

The performance of the first group of various statistical models is shown in Table 5. MIRA model outperforms other models with an F1-Score of 92.66% using contextual, word-level, corpus, and gazetteer features.

The second group of deep learning models is built using LSTM and CNN using CBOW, Skip-gram, Glove, FastText, and Contextual String Embeddings. We conducted the first experiment using the CBOW model with a comprehensive range of comparisons for word sequence representations,

Table 6 Performance of various models in F1-Score using CBOW, Skip-gram, Glove, FastText and Contexual string embeddings

Model	Features	Embeddings				
		CBOW	Skip-gram	Glove	FastText	Flair
WLSTM	Embedding	74.81	79.51	82.93	89.986	86.84
	Embedding + Word Level	75.21	80.06	83.444	90.58	87.25
	Embedding + Word Level + gazetteer	76.21	80.65	83.84	**91.25**	87.71
	Embedding + CLSTM	76.84	82.33	84.96	89.15	89.30
	Embedding + CLSTM + Word Level	77.77	82.88	85.58	89.75	89.67
	Embedding + CLSTM + Word Level + gazetteer	**78.58**	**83.68**	**85.99**	90.04	**90.04**
	Embedding + CCNN	75.38	79.81	83.28	89.03	87.12
	Embedding + CCNN + Word Level	75.79	80.29	83.82	88.55	87.65
	Embedding + CCNN + Word Level + gazetteer	76.06	81.01	84.27	88.90	88.03
WCNN	Embedding	71.64	77.23	79.67	86.96	84.67
	Embedding + Word Level	72.44	77.63	80.752	87.49	84.95
	Embedding + Word Level + gazetteer	73.17	78.12	81.32	87.86	85.50
	Embedding + CLSTM	72.81	78.74	81.65	86.88	85.80
	Embedding + CLSTM + Word Level	73.32	79.20	82.22	87.40	86.04
	Embedding + CLSTM + Word Level + gazetteer	73.68	79.87	82.7	87.74	86.71
	Embedding + CCNN	72.99	78.91	82.12	86.24	85.78
	Embedding + CCNN + Word Level	73.71	79.33	82.44	86.77	86.17
	Embedding + CCNN + Word Level + gazetteer	74.41	79.69	82.89	86.83	86.63

including all combinations of character CNN\LSTM and the word CNN\LSTM structures. If a rare word occurs in the vocabulary CBOW model cannot handle it. To improve the performance, we have used character embedding in combination with CBOW embedding. We have even included handcrafted Word-level and gazetteer features to improve the performance. The best performing model was word sequence representation with LSTM, character sequence representation with LSTM, Word-Level, and gazetteer feature with an F1-Score 78.58%, as shown in Table 6.

We conducted the second experiment using the Skip-gram model with a comprehensive range of comparisons for word sequence representations, including all combinations of character CNN\LSTM and the word CNN\LSTM structures. The Skip-gram model predicts the context words for a given window size with the given center word. To handle rare and out-of-vocabulary (OOV) words, we have use character embedding with the combination of Skip-gram embedding. The best performing model was word sequence representation with LSTM, character sequence representation with LSTM, Word-level, and gazetteer feature with an F1-Score 83.68% as shown in Table 6. The main drawback of CBOW and Skip-gram models is that they are static and cannot be dynamically optimized for specific tasks. Words and vectors are one-to-one relations that do not solve polysemous words.

We conducted the third experiment using the Glove model with a comprehensive range of comparisons for word sequence representations, including all combinations of character CNN\LSTM and the word CNN\ LSTM structures. GloVe builds word embeddings so that a combination of word vectors relates directly to the probability of these words co-occurrence in the corpus. The Glove model's disadvantage is it has relatively low information on the sentence or the word's context as training is done at the word-level co-occurrence matrix that does not handle unknown and rare words. The embeddings are static like CBOW, Skip-gram model. To improve the performance and identify rare words, we have used character embedding with the combination of Glove Embeddings. The best performing model was word sequence representation with LSTM character sequence representation with LSTM, Word-level, and gazetteer feature with an F1-Score 85.99%, as shown in Table 6.

We conducted the fourth experiment using the FastText model with a comprehensive range of comparisons for word sequence representations, including all combinations of character CNN\LSTM and the word CNN\LSTM structures. FastText was created by Facebook, an extension of the Skip-gram (Word2Vec) model by treating each word as a "character n-grams". The word vector is the sum of all its n-grams. As a result, the word embeddings tend to be better for less frequent words (given they share some n-grams). The model can also generate embeddings for unknown words (contrarily to Word2Vec and GloVe), given that it decomposes them by their n-grams. The best performing model was word sequence representation with LSTM, Word-level, and gazetteer feature with an F1-Score 91.25% as shown in

Table 7 Performance of various models of our approach using BERT

Models	Precision	Recall	F1-Score
BERT	94.67	91.57	93.09
BERT + BiLSTM	96.20	94.57	**95.37**
BERT + wordlevel features	94.98	92.10	93.52
BERT + word level features + BiLSTM	96.87	95.07	**95.96**
BERT + word Level features + gazetteer features	95.17	93.21	94.18
BERT + word Level features + gazetteer features + BiLSTM	97.02	95.62	**96.32**

Table 8 Handling of different challenges of Telugu using fastText and BERT based model

Challenges	Sentences	fastText Prediction	BERT Prediction
Capitalization	నేను చెన్నైలో నివసిస్తున్నాను (nEnu cennailO nivasistunnAnu) I live in Chennai	O LOC O	O LOC O
	నా జన్మస్థలం తెనాలి (nA janmasthalam tenAli) My birthplace is Tenali	O O LOC	O O LOC
Multiple meaning	నేను పూజ చేస్తున్నాను (nEnu pUja chEstunnAnu) I am doing pooja	O PER O	O O O
	నా పేరు పూజ (nA pEru pUja) My name is Pooja	O O PER	O O PER
Sentence structure	రాము సీతకు హారాన్ని పంపాడు (rAmu sItaku hAranni pampAdu) Ramu sent the necklace to Sita	PER O PER O	PER PER O O
	రాము హారాన్ని సీతాకు పంపాడు (rAmu hArAnni sItAku pampAdu) Ramu sent the necklace to Sita	PER O O O	PER O PER O
Inflection	నేను బెంగళూరులోనే ఉన్నాను (nEnu bemguLUrunE unnAnu) I stay in Bengaluru	O O O	O LOC O
	నేను బెంగళూరుకు వస్తున్నాను (nEnu bemgaLUruku vastunnAnu) I am coming to Bengaluru	O O O	O LOC O

Table 6. There was no improvement when we added character embedding as fastText stores information of "character n-grams". The main disadvantage is that it is trained at the word-level like Skipgram and Glove, which doesn't consider context, and co-occurrence is also ignored, which cannot solve polysemous words.

Contextual String embedding captures word semantics in a context such that the same wordwill have different representations under different senses. The model is based on character-level tokenization rather than word-level tokenization. In other words, it will convert sentences to a sequence of characters and go through the language model to learn word representation. It addresses the issue of polysemous and the context-dependent nature of words. The best performing model was word sequence representation with LSTM character sequence representation with LSTM, Word-level, and gazetteer feature with an F1-Score 90.04%, as shown in Table 6.

The first group of models, based on the statistical machine learning approach, outperforms the second group, based on various deep learning models and this could be due to the presence of OOV words. The embeddings used in the second group are static in nature and they are not dynamically optimized for specific tasks.

The third group is the Telugu BERT-NER model, as explained in Sect. 3. We compare different combinations of BERT embeddings with handcrafted features, as shown in Table 7. The first model uses only BERT embeddings and obtained an F1-Score of 93.09%. The second model used the BERT embeddings + BiLSTM layer and obtained an F1-Score of 95.37%. The third model uses BERT embeddings concatenated with word-level features and obtained an F1-Score of 93.52%. The fourth model uses BERT embeddings concatenated with word-level features + BiLSTM layer and obtained an F1-Score of 95.96%. The fifth model uses BERT embeddings concatenated with word-level features + gazetteer features and obtained an F1-Score of 94.18%. The sixth model uses BERT embeddings concatenated with word-level features + gazetteer features + BiLSTM layer and obtained an F1-Score of 96.32%. The models have outperformed when we add the BiLSTM layer as it captures long-term dependencies and solves the exploding/vanishing gradient problem.

Table 8 illustrates the results on sentences for both fastText and BERT models based on problems demonstrated in Sect. 1, Table 1. The predictions of the underlined Telugu words are highlighted using red and green, indicating incorrect and correct predictions, respectively. For the Capitalization problem, both fastText and BERT models detected the highlighted words correctly as Location. In the Multiple Meaning section, the fastText model predicts the word 'pUja' as a Person entity which is wrong, and the BERT model predicts it correctly as Others. In the second sentence, both models detected correctly 'pUja' as a Person entity. For the problem of Sentence Structure, the fastText based model failed to predict the highlighted word 'sItaku' correctly in both of the sentences. At the same time, the BERT model was able to predict the word correctly as Person even though it appears in different locations in the sentences but with the same meaning. As for the Inflection problem, the fastText-based model failed to predict the highlighted word 'bemgaLUrulOnE' and 'bemgaLUruku' correctly in both of the sentences. In contrast, the BERT model was able to predict the word correctly as Location even though the suffix attached to the word is different.

Table 9 Performance comparison of various word embeddings

Model	Precision	Recall	F1-Score
CBOW	82.67	74.87	78.58
Skip-gram	88.94	79.01	83.68
Glove	90.79	81.68	85.99
FastText	94.51	88.21	91.25
Contextual string	93.21	87.08	90.04
BERT	97.02	95.62	96.32

BERT has outperformed other word embeddings, as shown in Table 9. BERT trains a language model that takes both previous and next tokens while predicting, capturing the context of a word, syntactic and semantic natures of the language than other embeddings. BERT embeddings are dynamic, which can be modified while using for any downstream task. Embeddings like CBOW, Skip-gram, Glove, and FastText could not perform well because they are static, which cannot be modified for downstream tasks. These models output one vector for each word, combining all the word's different senses into one word.

The architecture explained in Figs. 4 and 7 are language-independent and can be used for NER model of any language where as the word embeddings and handcrafted features are language-dependent.

6 Conclusion

Telugu is a morphologically rich language, and it has very few annotated data sets for NLP research. Recent state-of-the-art architectures are seen to use pre-trained language models to gain embeddings that do not require an annotated data-set for training. These generic models are effective in training specific tasks like Named Entity Recognition. This paper builds a BERT model that is pre-trained on Telugu Wikipedia articles and performed the NER task on the FIRE data-set. We also generated and experimented with different word embeddings like Word2Vec, Glove, FastText, and Contextual string embedding for the same data set. These embeddings could not perform well because they are static, which cannot be modified for downstream tasks. Experimental results show that the BERT model outperformed other embedding models as it is trained bidirectionally, taking both the previous and next tokens while predicting and capturing the context of a word. It also captures the syntactic and semantic nature of the language. BERT embeddings are dynamic, which can be modified while using for any downstream task. We aim to perform experimentation with language models other than BERT, e.g., ELMO, ALBERT, etc., to compare the results across all those language models for future works.

References

1. Grishman, R., Sundheim, B.: Message understanding conference-6: a brief history. In: COLING 1996, The 16th International Conference on Computational Linguistics, vol. 1 (1996)
2. Chiu, J.P.C., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNs. Trans. Assoc. Comput. Linguist. **4**, 357–370 (2016)
3. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, vol.1 (Long Papers), Berlin, pp. 1064–1074 (2016)
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
5. Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., Vollgraf, R.: Contextual string embeddings for sequence labeling. In: Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA, pp. 1638–1649 (2018)
6. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Christopher, C. (2018)
7. Devlin, J., Chang, M.-W., Lee, K., Kristina, T.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long and Short Papers), Minneapolis, Minnesota, pp. 4171–4186 (2019)
8. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, Bengio, U.V., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.), Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., pp. 5998–6008 (2017)
9. Srikanth, P., Murthy, V.: Named entity recognition for Telugu. In: Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages (2008)
10. Shishtla, P.M., Gali, K., Pingali, P., Varma, V.: Experiments in Telugu NER: a conditional random field approach. In: Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages (2008)
11. Raju, G.V., Srinivasu, B., Raju, S.V., Balaram, A.: Named entity recognition for Telugu using conditional random field. Int. J. Comput. Linguist. (IJCL) **1**(3), 36 (2010)
12. Sasidhar, B., Yohan, P.M., Babu, A.V., Govardhan, A.: Named entity recognition in Telugu language using language dependent features and rule based approach. Int. J. Comput. Appl. **22**(8), 30–34 (2011)
13. Gorla, S., Bhanu Murthy, N. L., Malapati, A.: A comparative study of named entity recognition for telugu. In: FIRE'17, New York, NY, pp. 21–24 (2017)
14. Gorla, S., Velivelli, S., Bhanu Murthy, N.L., Malapati, A.: Named entity recognition for Telugu news articles using naïve bayes classifier. In: Albakour, D., Corney, D., Gonzalo, J., Martinez-Alvarez, M., Poblete, B., Valochas, A. (eds.), Proceedings of the Second International Workshop on Recent Trends in News Information Retrieval co-located with 40th European Conference on Information Retrieval (ECIR 2018), Grenoble, France, March 26, 2018, CEUR Workshop Proceedings, vol. 2079, pp. 33–38 (2018)
15. Gorla, S., Chandrashekhar, A., Bhanu Murthy, N.L., Malapati, A.: Telneclus: Telugu named entity clustering using semantic similarity. In: Verma, N.K., Ghosh, A.K. (eds.), Computational Intelligence: Theories, Applications and Future Directions, vol. II, Singapore, pp. 39–52 (2019)

16. Gorla, S., Neti, L., Bhanu, M., Malapati, A.: Enhancing the performance of Telugu named entity recognition using gazetteer features. Information **11**(2), 8 (2020)
17. Adusumilli, M., Gorla, S.K., Neti, L.B.M., Reddy, A.J., Malapati, A.: Named entity recognition for telugu using lstm-crf. In: Jha, G.N., Bali, K., Sobha, L., Ojha, A.K. (eds.), Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Paris, France (2018)
18. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(76), 2493–2537 (2011)
19. dos Santos, C.N., Guimarães, V.: Boosting named entity recognition with neural character embeddings. CoRR. arXiv:1505.05008 (2015)
20. Kaur, K.: Khushleen@iecsil-fire-2018: Indic language named entity recognition using bidirectional lstms with subword information. In: Parth, M., Paolo, R., Prasenjit, M., Mandar, M, (eds.), Working Notes of FIRE 2018—Forum for Information Retrieval Evaluation, Gandhinagar, India, December 6–9, CEUR Workshop Proceedings, vol. 2266, CEUR-WS.org, pp. 153–157 (2018)
21. Bhattu, S.N., Krishna, N.S., Somayajulu, D.V.: idrbt-team-a@iecsil-fire-2018 named entity recognition of Indian languages using bi-lstm', booktitle =
22. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. CoRR. arXiv:1603.01360 (2016)
23. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Yoshua, B., Yann, L. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, Workshop Track Proceedings (2013)
24. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1532–1543 (2014)
25. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of of NAACL (2018)
26. Ghaddar, A., Langlais, P.: Robust lexical features for improved neural network named-entity recognition (2018)
27. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging (2015)
28. Strubell, E., Verga, P., Belanger, D., McCallum, A.: Fast and accurate entity recognition with iterated dilated convolutions (2017)
29. Aguilar, G., Maharjan, S., López Monroy, A.P., Solorio, T.: A multitask approach for named entity recognition in social media data. In: Proceedings of the 3rd Workshop on Noisy User-generated Text (2017)
30. Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics **36**(4), 1234–1240 (2020)
31. Souza, F., Nogueira, R., Lotufo, R.: Portuguese named entity recognition using BERT-CRF. arXiv preprint arXiv:1909.10649 (2019)
32. de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., Nissim, M. Bertje: a dutch bert model. arXiv preprint arXiv:1912.09582 (2019)
33. Li, X., Zhang, H., Zhou, X.-H.: Chinese clinical named entity recognition with variant neural structures based on BERT methods. J. Biomed. Inform. **107**, 103422 (2020)
34. Yonghui, W., Schuster, M., Chen, Z., Le, Q.V., et al.: Bridging the gap between human and machine translation, Google's neural machine translation system (2016)
35. Bharadwaja Kumar, G., Muthy, Kavi Narayana, Chaushri, B.B.: Statistical analyses of Telugu text corpora. IJDL Int. J. Dravid. Linguist. **36**(2), 71–99 (2007)
36. Gorla, S., Velivelli, S., Satpathi, D. K., Bhanu Murthy, N.L., Malapati, A.: Named entity recognition using part-of-speech rules for Telugu. In: Elçi, A., Sa, P.K., Modi, C.N. Olague, G., Sahoo, M.N., Bakshi, S. (eds.), Smart Computing Paradigms: New Progresses and Challenges, Singapore, pp. 147–157 (2020)
37. Reddy, S., Sharoff, S.: Cross language POS taggers (and other tools) for Indian languages: an experiment with Kannada using Telugu resources. In: Proceedings of the Fifth International Workshop On Cross Lingual Information Access, Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pp. 11–19 (2011)
38. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)