Named Entity Recogintion

Internship Project

SANDEEP KUMAR GANDRAKOTA
VIVEK REDDY BANDI

OSLO METROPOLITAN UNIVERSITY
STORBYUNIVERSITETET

Photo: Ronny Østnes / OsloMet

Table of contents

Introduction

Named Entity Recognition (NER) also called entity identification or entity extraction is a Natural Language processing (NLP) technique that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person,organization,location, medical codes, time expressions, quantities, monetary values,percentages etc.

Objective.

The objective of this project The aim of this project is to classify the tokens in a Telugu data set into different categories like person, organization, location and miscellaneous.

Introduction(contd..)

Telugu is a highly inflectional and agglutinating language with a complex word formations.Also challenges like the unavailability of annotated corpora, no gazetteer lists,no capitalization, spelling variations, free word order, etc.Hence NER in Telugu is highly challenging task

Overview.

In this Project we develop and evaluate the Long Short-Term Memory-Conditional Random Field(LSTM-CRF) classifier model to perform NER classification for Telugu language data The output and performance of this Long Short-Term Memory-Conditional Random Field(LSTM-CRF) model is compared with the already existing models

Related Work

In 2008, Srikanth and Murthy explored NER in telugu using a 2 stage classifier using CRF in the 1st stage to identify the nouns and a rule based model to identify named entities and obtained F-measures ranging from 80% to 97% on Telugu UoH News corpus

A Hybrid approach using Maximum Entropy model to identify the nouns and then identify Named Entities using rule based approach was proposed by Khanum (2017)

Ekbal and Bandyopadhyay (2008) developed an SVM based model for NER in Bengali using various features and gazetteer list features as an input featureset and reported an F-measure of 91.8% on Bengali News Corpus

A language independent Hidden Markov Model (HMM) based classifier was proposed by Gayen and Sarkar at ICON 2013 and reported F-measure of 40.03% for Telugu

Related Work(contd..)

Shishtla in 2008 also built a CRF based approach for Telugu NER with language independent and dependent features but reported less F-measure of 44.89%.

NER in kannada using Multinomial Naive Bayes (MNB) classifier was proposed by Amarappa and Sathyanarayana (2015) and reported F-measure of 81% for Kannada

A Bi-LSTM and a CNN based hybrid model that automatically detects word and character-level features was proposed by Chiu and Nichols (2015). The model obtained F-measures of 91.62% on CoNLL-2003 and 86.28 on CoNLL- 2012 datasets

Corpus Description

We have used the corpus provided by Aniketh and monica (2019) which was accumulated by crawling Telugu newspaper websites and then manually tagged by with NEs belonging to four classes using the standard IOB scheme. It was described as Golden Telugu Data and it contains train and test sets with various number of entities. It was used to train model in Experiment 1.

Additionally, we have obtained data from web sources which was termed as NER Telugu data. Then it was added to Golden Telugu Data and model to train model with more data in Experiment 2. The details of number of entities in individual datasets and combined dataset is shown in tables below:

Statistics of Golden Telugu Data

No.Training sets	Organization	Location	Person	Miscellaneous	Total
set-1	1180	1630	1977	2298	7085
set-2	1118	1624	1905	2250	6897
set-3	1165	1587	1959	2226	6937
set-4	1182	1608	1902	2275	6967
set-5	1161	1660	1961	2215	6997
set-6	1222	1661	1895	2234	7012
set-7	1134	1629	1930	2288	6981
set-8	1171	1625	1869	2300	6965
set-9	1168	1637	1892	2256	6953
TOTAL	10501	14661	17290	18342	62794

Table: 1

Statistics of NER Data

No.Training sets	Organization	Location	Person	Miscellaneous	Total
set-1	1198	1592	1894	2255	6934

Table: 2

Statistics of Combined Data

No.Training sets	Organization	Location	Person	Miscellaneous	Total
TOTAL	11699	16253	19184	20597	69728

Table: 3

Model and Implementation:

We introduce LSTMs, CRFs and other relevant concepts before proceeding to explain the architecture of the classifier.

Long Short term Memory (LSTM):

RNNs are capable of learning long term dependencies between data points.

So they are used for building classifiers which operate on sequential data.

But there exists a vanishing gradient problem in them

To overcome this, LSTMs can be used which are a class of RNNs and are capable of learning long term dependencies

LSTM(contd..

It takes sequential data as input and gives a sequential output.

It has a memory cell and 3 gates: Forget gate, Input gate and Output gate.

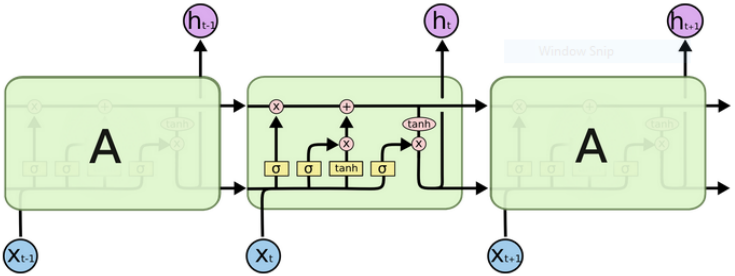


Figure: Internal network in LSTM

LSTM (contd..)

Forget Gate:

It takes previous state output and present input and outputs a number between 0 and 1 for each number in the previous cell state.

It is used to selectively forget the information when the context of the data changes or when the data is not needed

$$f_t = \sigma(W_f \times [h_t - 1, x_t] + b_t$$

Input Gate:
The input gate used to selectively read the input and decides about what new information to be stored in the new cell state

LSTM (contd..)

The sigmoid layer decides which values to update and tanh layer creates a vector of new candidate values. Then these are combined to give an update to the state.

$$i_t = \sigma(W_i \times [h_t - 1, x_t] + b_i$$

$$\widetilde{C}_t = \tanh(W_c \times [h_t - 1, x_t] + b_c$$

$$C_t = f_t \times C_t - 1 + i_t \times \widetilde{C}_t$$

Output Gate:

The ouput gate used to selectively write a filtered version of the cell state to the output

LSTM (contd..)

The sigmoid layer decides which parts to output from the previous state and then tanh layer bounds the cell state values between –1 to +1. Both are multiplied and given as output h

$$o_t = \sigma(W_o \times [h_t - 1, x_t] + b_o$$

$$h_t = o_t \times tanh(C_t)$$

Thus an LSTM retains, forgets and stores information according to context

Model and Implementation contin..

Bi-directional LSTM

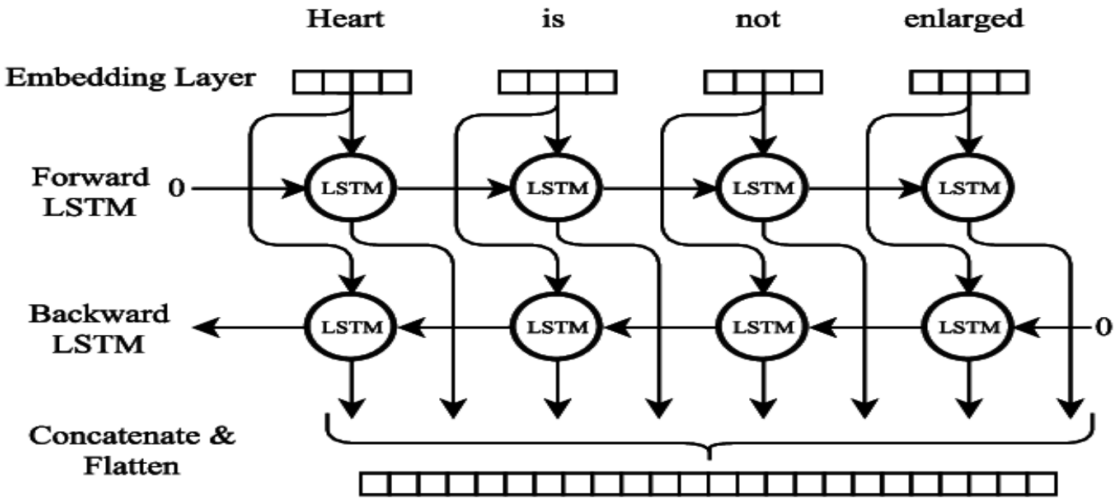
A Bidirectional LSTM, or biLSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backward direction

Since one of the 2 LSTMs run from past to future and other from future to past, at any point in time we preserve information from both past and future

This structure allows the networks to have both backward and forward information about the sequence at every time step

Bi LSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm

Bi-directional LSTM (contd..)



Bi-directional LSTM (contd..)

Character-Level Word Embedding:

We use character level word embeddings to determine named entities in the text

Here initially each character is assigned an n-dimensional vector.Then each token is broken up into its individual characters which are then mapped to their corresponding vectors

Thus, a token is converted to a sequence of vectors which is then fed to a Bi-LSTM.

The final states of both LSTMs are then concatenated to obtain the final character level embedding of the token

Bi-directional LSTM (contd..)

Contextual Representations of Words:

To perform NER, we need information about the tokens before the current token to find its contextual representation

Also it is useful if the context is stored from both past and future tokens rather than only from past

Hence we use Bi-LSTMs to get contexts from both sides and concatenate the internal states of both LSTMs to get a token's final contextual representation

Linear Chain Conditional Random Fields (CRFs)

Dependencies and patterns between tokens:

Entities can be spread across multiple tokens and Presence of separators between named entities

Occurrences of certain tags in a certain sequences

Therefore, a tagging decision must be made after taking into account the tagging decisions for all the other tokens in the sequence

Hence we use Linear Chain CRFs to make a tagging decision considering the dependencies between the adjacent word

Linear Chain CRFs (contd..)

Implementation of a CRF:

Each token ti in a sequence of the form (t0, t1, t2...tn) has a corresponding m dimensional vector si where m is the number of possible tags

To capture the linear dependencies between the tagging decisions, we use a mxm dimensional matrix A where Aij is probability of jth tag occuring after ith tag

Also we use 2 m-dimensional vectors s and e which represent the confidence of a tagging sequence starting and ending with a given tag

Then each tagging sequence of the form (y0, y1, y2...yn) assigned to the token sequence is scored as:

Linear Chain CRFs (contd..)

$$Score(y_0, y_1, y_2 \ldots y_n) = s[y_0] + \sum_{i=0}^{n} s_i[y_i] + \sum_{i=0}^{n-1} A[y_i, y_i + 1] + e[y_n]$$

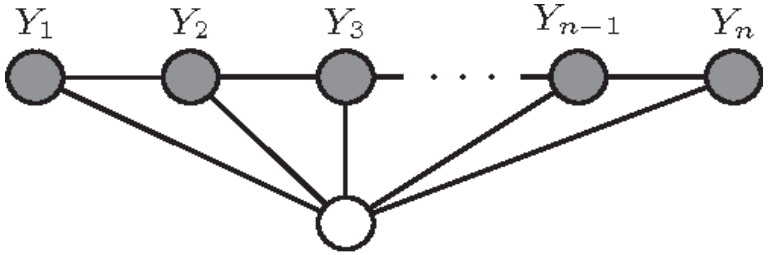


Figure: Structure of linear chain CRF

Linear Chain CRFs (contd..)

The tagging sequence which has the maximum Score is output by the CRF based classifier

In this classifier we minimized –ve log loss of the probability of correct tagging sequence Y while training

$$Loss = -log(P(\widetilde{Y}))$$

$$P(\widetilde{Y}) = e^{Score(\widetilde{Y})} / \sum_{i=o}^{n} e^{Score(y_0, y_1, y_2 \ldots y_n)}$$

During Backpropagation, various weights and embeddings of the model are adjusted to minimize the loss

Neural Network Architecture

To capture the context of words in unseen text, we use the 300-dimensional fastText pretrained Telugu word embeddings provided by Facebook Research

Then 200- dimensional character-level embeddings for each token are generated using Bi-LSTMs

Both these embeddings are then concatenated for each token to represent them as vectors to form a 600-dimensional contextual word embeddings are generated for each token

The contextual word embeddings are then used to compute the scores for a word using a $600 \times m$ weight matrix W and an m dimensional bias vector b where m is the number of classes

Neural Network Architecture (contd..)

The score for each word is an m dimensional vector given by

$$s = W \times h + b$$

where h is the contextual embedding of that word

After getting the scores for an entire sequence, we use a linear chain CRF to make the tagging decisions for the whole sequence

Neural Network Architecture (contd..)

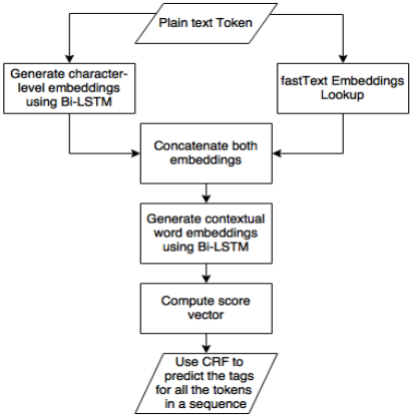


Figure: Architecture of LSTM-CRF classifier

Experiments and Results

We have done two experiments to evaluate the performance of the Long Short-Term Memory-Conditional Random Field(LSTM-CRF) model.These results are compared with other models.

Example (Experiment-1)

In this experiment we trained and evaluated the performance of the model by using Golden Telugu Data. We have trained the model nearly 20 Epoch.We have divided the Golden Telugu Data into Train set and Test set

1.Accuracy and F-measure of the model:
Model Accuracy=99.52
F-measure=97.61
Processed 9812 tokens with 1293 phrases and found 1238 phrases and 1103 correct words.

Evaluation Table for Experiment-1

Category	Precision	Recall	FB1	Correct
LOCATION	88.47%	88.95%	88.71	373
Organisation	74.48%;	67.50%	70.82	145
Person	88.40%	79.66%	83.80	319
Miscellaneous	95.51%	93.87%	94.68	401

Table: 4

Overall accuracy=96.48%
Overall Precision=89.10%
Overall recall=85.31%
Overall FB1=87.16

Experiments and Results(Contd..)

Example (Experiment-2)

In this experiment we trained and evaluated the performance of the model by adding the NER Telugu Data to the existing data. We have trained the model nearly 20 Epoch.We have Evaluated the performance of model using the combined data.The model built the vocabulary by using the Facebook Fasttext data

1.Accuracy and F-measure of the model:
Model Accuracy=99.62
F-measure=98.03
Processed 9812 tokens with 1293 phrases and found 1238 phrases and 1103 correct words.

Evaluation Table for Experiment-2

Category	Precision	Recall	FB1	Correct
LOCATION	90.17%	86.52%	88.31	356
Organisation	94.46%;	91.91%	93.17	397
Person	88.40%	77.12%	80.65	323
Miscellaneous	94.46%	91.91%	93.17	397

Table: 5

Overall accuracy=96.27%
Overall Precision=88.31%
Overall recall=83.53%
Overall FB1=85.85

Conclusion and Future Works

In this Project we described the LSTM-CRF model based approach for NER classification in Telugu.Named Entity Recognition(NER) have many applications in the field of Natural Language processing.

The accuracy and F-measure achieved by LSTM-CRF model are 99.52% and 97.61 respectively.

It can be noted that the accuracy and F-measure of LSTM-CRF model is high compared to other classifiers and models.

This project can be further extended to all Indian Languages. Using advanced tools from Machine Learning and Deep Learning many classifiers and models can be developed for NER identification in coming future.

Addition of extra data to the corpus can improve the performance of the model and we can get good results.

Conclusion and Future Works (contd..)

The LSTM-CRF classifier clearly outperforms the other classifiers based on all the metrics. In fact, its performance is greater by approximately 7% based on overall F-measure. This is because of three main reasons.

Firstly, the use of CRFs makes tagging more accurate because the classifier is able to discern the dependencies which exist between the individual tags. These dependencies can not be captured by an SVM and other classifiers thereby lowering its performance.

Secondly, character-level embeddings allow the classifier to learn the general form of an NEs. Hence, our classifier is able to handle unknown NEs because it is able to identify them based on their structure.

Finally, Bi-LSTMs are great at learning long term dependencies and further augment the classifier's ability to learn dependencies between tokens.

Refrences

1.Named Entity Recognition for Telugu using LSTM-CRF by Aniketh Janardhan Reddy, Monica Adusumilli, Sai Kiranmai Gorla, Lalita Bhanu Murthy Neti and Aruna Malapati

2.Srikanth, P. and Murthy, K. N. (2008). Named entity recognition for telugu. In IJCNLP, pages 41–50.

3.KANNADA NAMED ENTITY RECOGNITION AND CLASSIFICATION (NERC) BASED ON MULTINOMIAL NAÏVE BAYES (MNB) CLASSIFIER by S Amarappa 1 Dr. S V Sathyanarayana

4.Experiments in Telugu NER: A Conditional Random Field Approach by Praneeth M Shishtla, Karthik Gali, Prasad Pingali and Vasudeva Varma

5.Vijayakrishna, R. and Devi, S. L. (2008). Domain focused named entity recognizer for tamil using conditional random fields. In IJCNLP, pages 59–66.

6.Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. CoRR, abs/1603.01360.