Part of Speech Tagging using Bi-LSTM and K-NN

Shaimaa Koukaa

*aDepartment of Engineering and Information Technology, Ajman, Ajman University United Arab Emirates*

# Abstract

# Part of speech(POS) tagging is one of the initial steps implemented regularly as part of the Natural Language Processing (NLP) pre-processing steps after tokenizing, padding the text data and followed by lemmatizing and removing stop words. POS is an NLP problem solved by NLP researchers achieving high accuracy and overcoming language challenges. The main obstacles faced research in the past building POC taggers is the lack of open-source and free-of-charge treebanks especially for not common languages like Arabic, Italian and German and the expenses, and labor of build a treebank. Experimenting with the Arabic treebanks provided by universal dependencies PUD, we have implemented a sequence labeling classifier bi-directional long-short term memory(Bi-LSTM) neural network and a multi-classifier K-nearest neighbor. The experiments demonstrates the fundamental differences between the two models operation as they differ in the principle of digesting sequential information which is reflected in the execution time and accuracy.

*Keywords:* Artificial Intelligence, Natural Language Processing , Word Embedding, Machine Learning, LSTM, RNN, KNN

# Background

* 1. *K-Nearest Neighbor*

K-nearest neighbor is a machine learning algorithm that is a famous and effective algorithm for both classification and regression tasks. The segregation between the output is according to the features distance from each other in the solution space. The most popular distance metrics used for measuring distance between records/data are :

* Euclidean Distance (x, xi) = sqrt(sum(xj-xij)^2)
* Hamming distance
* Manhattan Distance
* Minkowski Distance

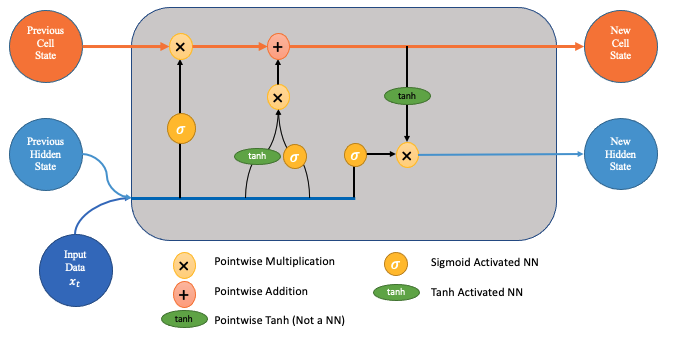
When the algorithm is used in regression, the prediction is computed as the mean or median of the top k similar instance. On the other hand, when used for classification problem the prediction is (via voting systme) the class with the highest frequency from the k-most similar instance.

Figure 1: K-nearest neighbor

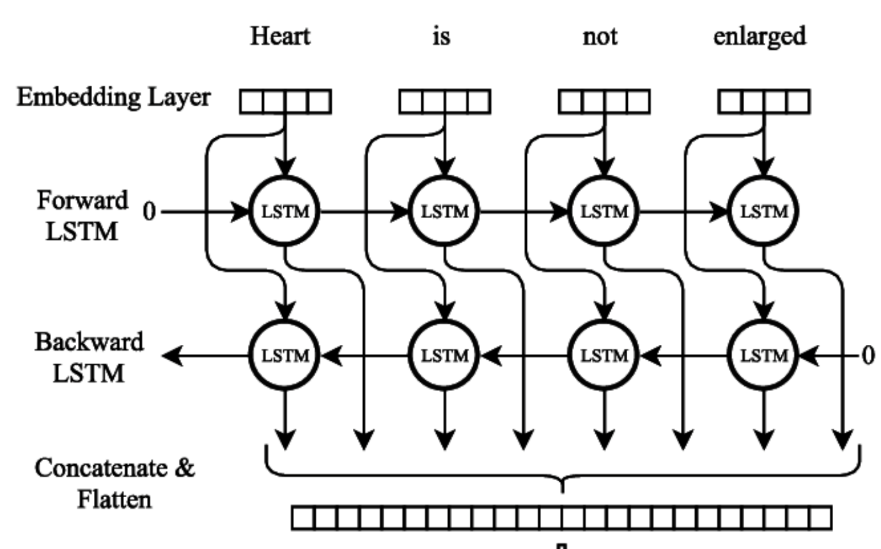
* 1. *Long Short-Term Memory*

Processing text-based data is different than regular tabular data, in a way that word’s order within a sentence is significant and highly influential on the overall meaning of the sentence. Some facts spread even as far as across several sentences or paragraphs, therefore processing each word independently, from the rest of the sentence causes losing of important information crucial for prediction.

As a solution, Recurrent Neural Networks are proposed as it has a feedback loop connecting the previous state of an RNN cell to current cell computations therefore having a memory of previous words. However, one of the main issue often encountered with Vanilla RNN is its vanishing gradient where the RNN does not have the ability to propagate the learnt information from the backpropagation process to the early stages of RNN, therefore it get rarely updated and eventually get stuck and stop learning.

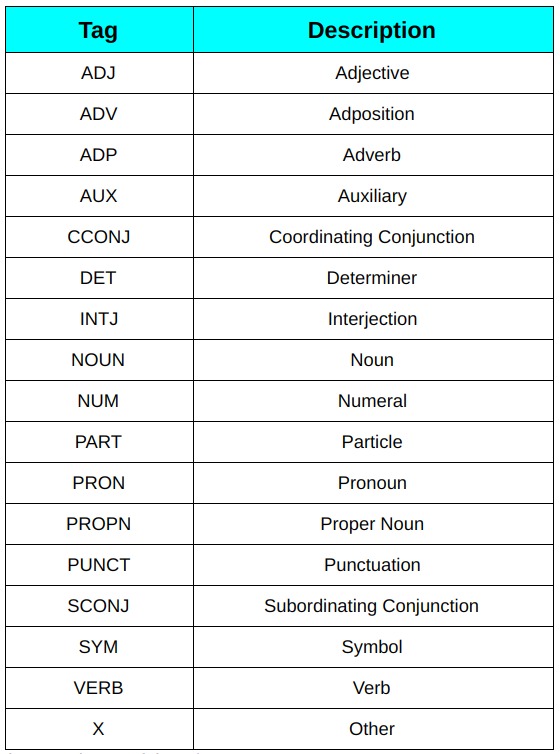
Many techniques are used to overcome the vanishing gradient problem of Deep neural networks, yet one of the most remarkable practices of using LSTM cells in RNN. Long Short-Term Memory or short LSTM is created with the purpose of reducing the vanishing gradient effect, and it has an excellent ability to save feedback/information from previous entries for longer distances using its unique structure and gates. The input for each LSTM cell is input value and a previous state (initially seed value) that get passed through a forget gate that decides whether to pass the information from previous stages or carry it still to upcoming stages. Another important gate is the update gate the computes the output of the cell according to the hidden state and the previous results. The update operations are in the form of addition/summation therefore reduce the vanishing gradient effect on the new update instead of the multiplication update operation.

Moreover, a version of LSTM cell outperformed the vanilla version “bidirectional” lstm that provides two-way scanning of information, forward and backwards which gave the RNN context as information from the past and the future



# Simulations and Experiments

The following experiments are Training was implemented using Skipgram word embedding models with window size 300 on the twitter data dump, resides in an open source github repository AraVec (<https://github.com/bakrianoo/aravec>). The development of the RNN model is done using keras and tensorflow libraries in the latest versions on Anaconda - Jupyter. The treebank data used are PUD from in the format CoNLL-U annotations which is pared using conllu python package and split to 70% training (10% is for validation during the training phase) and 30% is left for testing. In addition, all experiments in this report are executed on HP-Omen-15 Laptop with core i7 CPU, 16 GB of RAM, and 64-bit Windows 11 operating system.

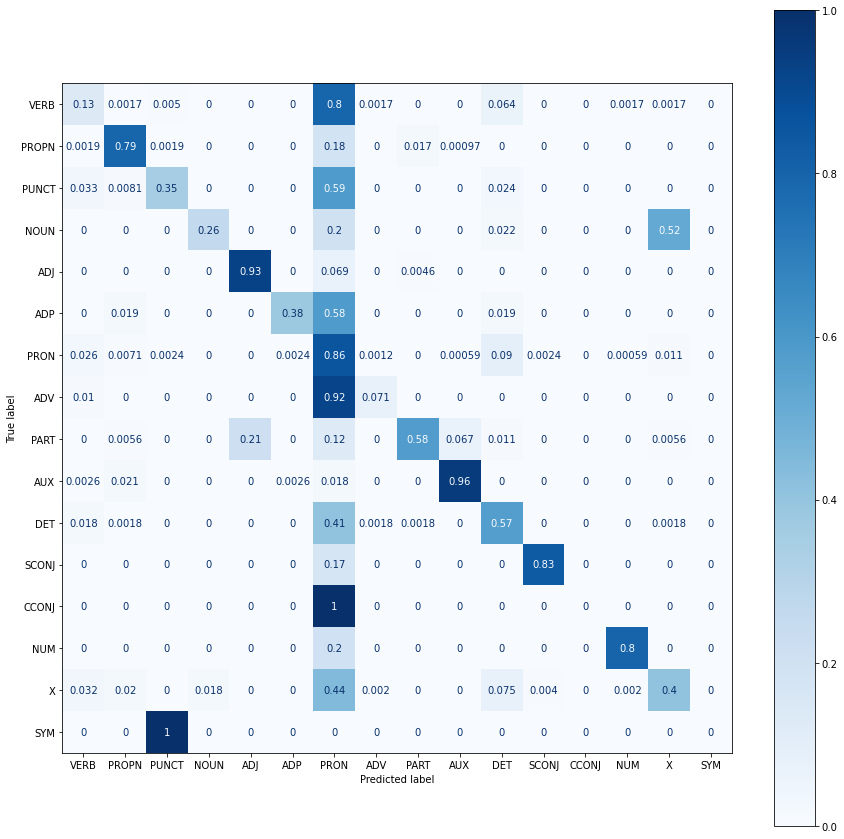
* 1. *Results and Discussion*
     1. *K-nearest Neighborhood*

As can be seen in the figure the number of tags are 16 in total where the distribution of each tag across the corpus can be seen in the figure.

The first implementation is designed to be multiclassification problem, with the absence of sequential processing and relying solely on the correct prediction of each word and neglecting the context of the word within the sentence the evaluation of the performance showed a midcore performance barely higher than a random chance distribution.

Chart, histogram

Description automatically generated

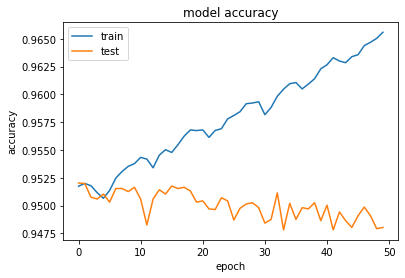


|  |  |
| --- | --- |
| metric | Value |
| accuracy | 0.67 |
| precision | 0.71 |
| recall | 0.67 |
| F1-Score | 0.69 |

* + 1. *Bi-LSTM*

Using a bi-directional LSTM recurrent neural network gave the model the advantage of processing and absorption the context of the word within the sentence therefore have better chance of deducing the part of speech tag for that certain word. The model consists of an input layer that takes the data in the form of padded numerical nested array of sentences where each sentence is array of numerical representation of the words within that sentence. Due the inflexibility of the RNN structure the sentences have unified length where shorter sentences are padded from the left side. Padding the sentences in usually not preferred as it can introduce a false high accuracy or overfit the model. Followed by layer of embedding using the word embedding matrix of aravec for the n-gram skipgram Aravec open-source project. Then the data is processed in series of 4 consecutive bidirectional LSTM layers, is them fed to a time distributed dense layer where each word is tackled at a time and for each word a prediction of a one-hot-vector is produced. Therefore, the need for padding and then one-hot encoding for each target tag. An assumption is made in the padding where we used the tag X as padding for the output since it was declared that tag X is assigned for “other” and for further information refer to the table above of each tag and its description. After 50 epochs of training, the model training and validation loss and accuracy are graphed depicting a consistent performance and minor(insignificant) difference in the performance , along with the performance metric which is computed as the following table:

|  |  |  |
| --- | --- | --- |
|  | Accuracy | Loss |
| Training | 0.9656 | 0.0963 |
| Validation | 0.9480 | 0.1661 |
| Testing | 0.9547 | 0.1510 |



# Conclusion and Future Work

During the experiments we processed the experiment with different structures of RNN, embedding types and machine learning classification modes and concluded the importance of choosing a suitable structure for the problem at hand. As expected, the multi-classification and as well as the multi-labeling classification are not a suitable classification models due to the lack of sequential prediction which is done in sequential labeling as the model Bi-LSTM where the labels are not predicted independently rather sequential as well as the processing of the words is sequential. One problem we are going to work on in the future is trying to incorporate larger treebanks (integrating PADT, PUD and NYUAD) and work on a more accurate performance metric, since the current metric counts padding prediction which is represents a high percentage of the whole predictions. Ignoring the padding prediction will give a more accurate measurement of the performance as well as innovating a new RNN structure that allows more flexibility in the input therefore drops the need for padding in the first place. Another improvement for the future is using the keras callback that stops the training once the vanishing gradient starts to spread and/or experiment with adding a batch normalization layer.