Comparison based approach to Sequence Labeling for NLP tasks

We compare two different Machine Learning approaches and their performance and accuracy using pre-trained word vectors from Glove(Stanford) and CONLL-2003 Shared task dataset (English)

Samartha Kajoor Venkatramana
Siddhartha Sharma
Chinonso Larry Okeke

Contents

Introduction

Background

Approach

Results

Conclusion

Introduction

- ★ Basically, we are annotating every word in a sentence of text to its respective tag.
- ★ Each type of task, NER or POS or chunking have their own pre-defined tags.
- ★ This paper aims to help the future researchers to easily proceed with the problem at hand instead of spending endless hours in figuring out the correct parameters.
- ★ We provide in depth comparison between two famous approaches in machine learning
- ★ Using these studies, when one is developing a similar system, he can decide very confidently which parameters he would need to keep track of to increase the accuracy scores.
- ★ We have built a program to help us test various parameters, written in python.

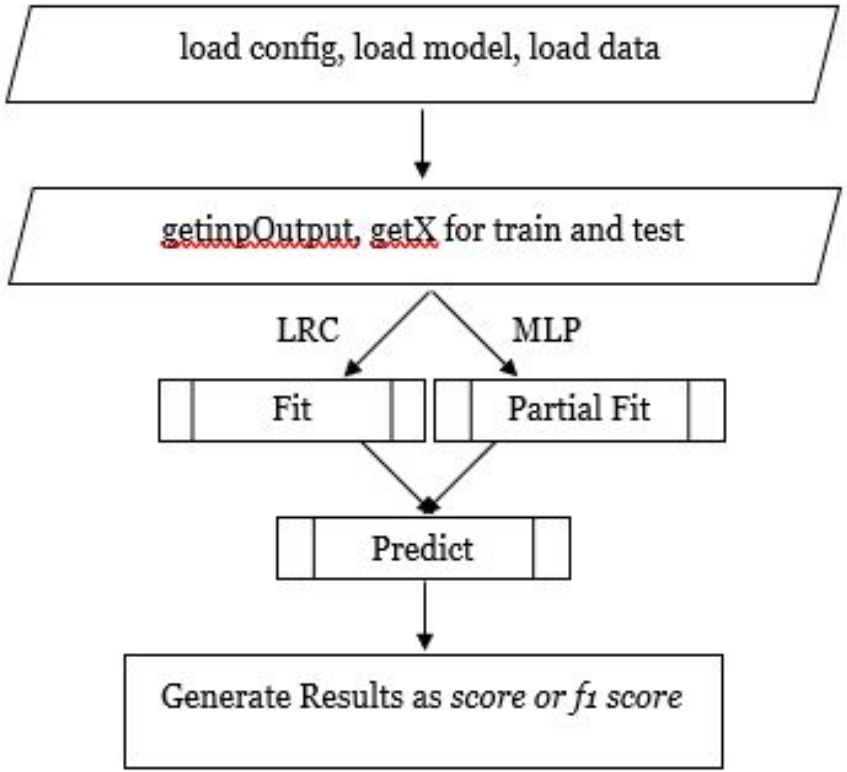
Approach

- ★ The model is built upon the approach – Classification of Window based concatenation of Word embeddings
- ★ Config.yaml:
  - We want to experiment with different tasks our system processes the correct column from the dataset depending on the specific task, whether it is NER, POS or chunking.
  - Our system loads the word embedding model from the path, i.e. GloVe word embeddings.
  - Then, the task is acquired and set from the options.
  - We use the load_data.load() method to load our data and the corresponding values from the labels column depending on the task.

Approach

- We use the getinpOutput() method to generate the word ids.
- We then use next method getX() to convert these lists of context windows into word embeddings.
- We do this for both the training and testing dataset and to simplify our task, we have extended the training set to include the validation set as well.
- Depending on the algorithm specified in config, we use either MLP classifier or LR classifier to fit and predict on the training and testing dataset.

Approach


load config, load model, load data
getinpOutput, getX for train and test
LRC
MLP
Fit
Partial Fit
Predict
Generate Results as score or f1 score

Results

Specifics of data used

- English data only.
- Training set merged with validation set.
- For faster processing, words converted into indices, then indices converted to list of indices based on context width, then these lists converted to lists of word embeddings for each of the index in list.
- Same procedure applied for labels of y set.

Results

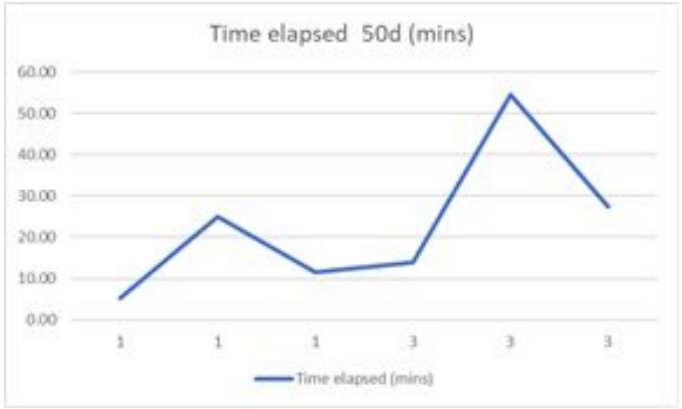
Experiments performed are by modification of config file and running the program, and are listed below

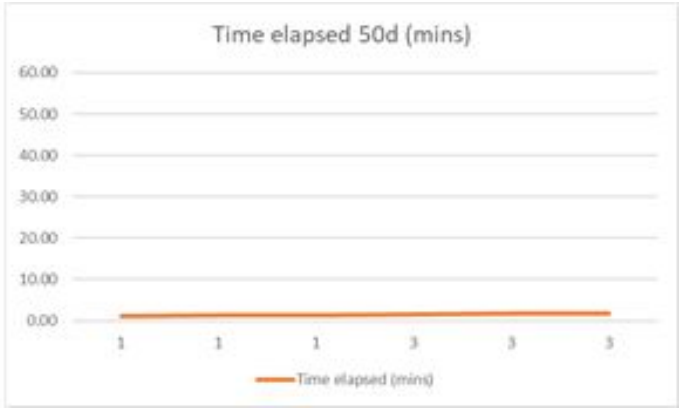
- Varying word embedding dimensions: 50d, 100d
- Varying context window size: 1,3
- Different tasks chosen from: [NER, POS, Chunk]
- Different algorithms: [Logistic regression classifier or MLP classifier]

Results

★ Logistic Regression Classifier VS MLP Classifier:

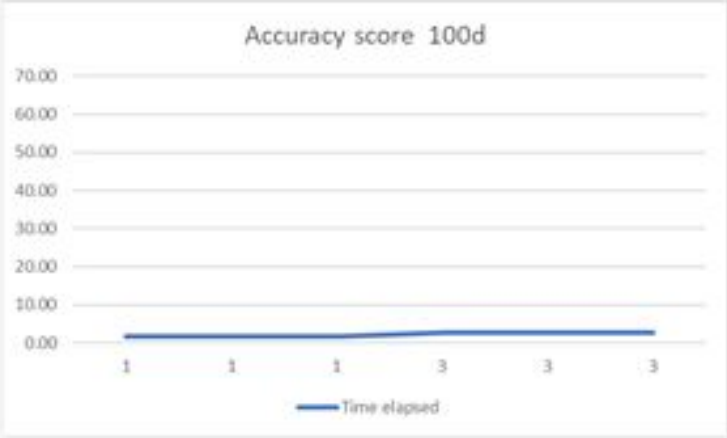
- 1. 50d VSM plotted with changing context width





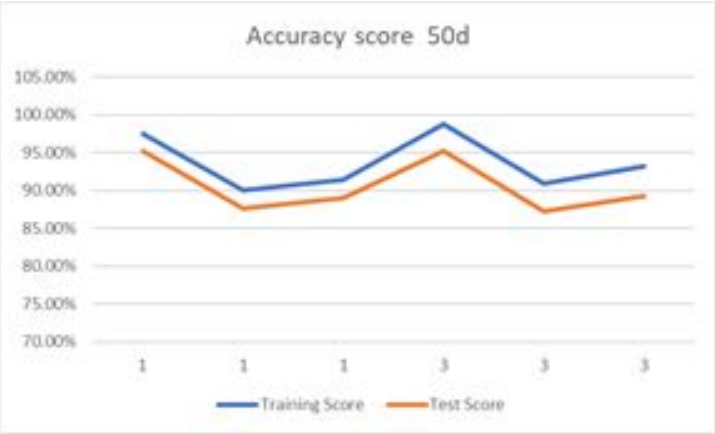
Results.

- 1. 100d VSM plotted with changing context width



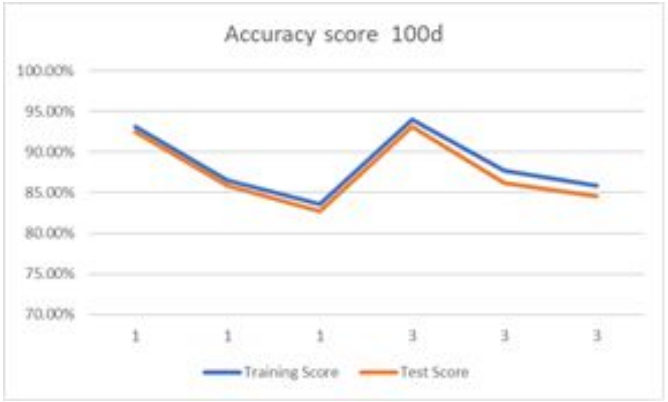
Result.

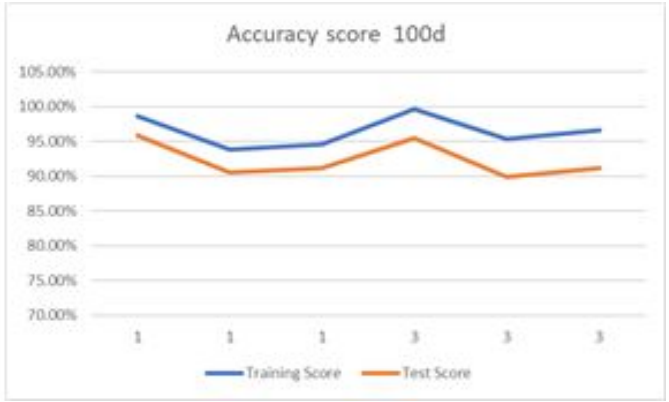
- Accuracy Score: 50d VSM plotted changing context width



Result

- 100d VSM plotted changing context width





Conclusion.

- ★ After running these tests, and varying the previously mentioned properties and parameters, we conclude that the MLP Classifier is better.
- ★ As the dimension is increased the time taken and accuracy also increases by 5% and 10% respectively.
- ★ We verify from the results that a higher context width might make sense to us since it adds more meaning to each word, but the graphs prove that it may not be a good idea, or further testing is required.
- ★ Further increasing the word embeddings seems to increase the accuracy, but the increase isn't quite evident in the margin of 5% only, whereas the running time increases by almost 10%.
- ★ Logistic regression classifier is very slow compared to neural networks.