Final Project

Chatbot

MSDS 453: Natural Language Processing

Husein Adenwala

Northwestern University

1. Introduction & Problem Statement

The purpose of this research was to create a chatbot for an investment research firm's website. The purpose of the chatbot is to retrieve company-specific stock market data requested by the user and also be an instant glossary to find meaning of financial terms (Q&A). Searching for this data/information on a website would make a user parse through a lot dense data to get the specific desired information and, in some cases, could require opening a new webpage. This chatbot is intended to allow the user to bypass this search process.

The NLP pipeline to create such a chatbot uses a deep learning model to classify the user's intent. User intents were categorized based on domain knowledge and based on the type of stock data available on the website. If the user intent is to find meaning/definition of a financial term, the bot would use vector or embedding cosine similarity to retrieve information. If the intent is to get stock data/information, then it would use a pretrained NER model to extract the Entity i.e. the company name or stock symbol would be used as a look-up table to get the company identifier that would be used in the API function to get the stock data. The results of the experiment conducted in this research suggest that using a transformer-based model such as BERT for transfer learning can generalize well on classifying user intent and a pre-trained NER model performs well identifying entities. Also TF-IDF vectorizing outperformed word2vec and doc2vec in retrieving information based on cosine similarity.

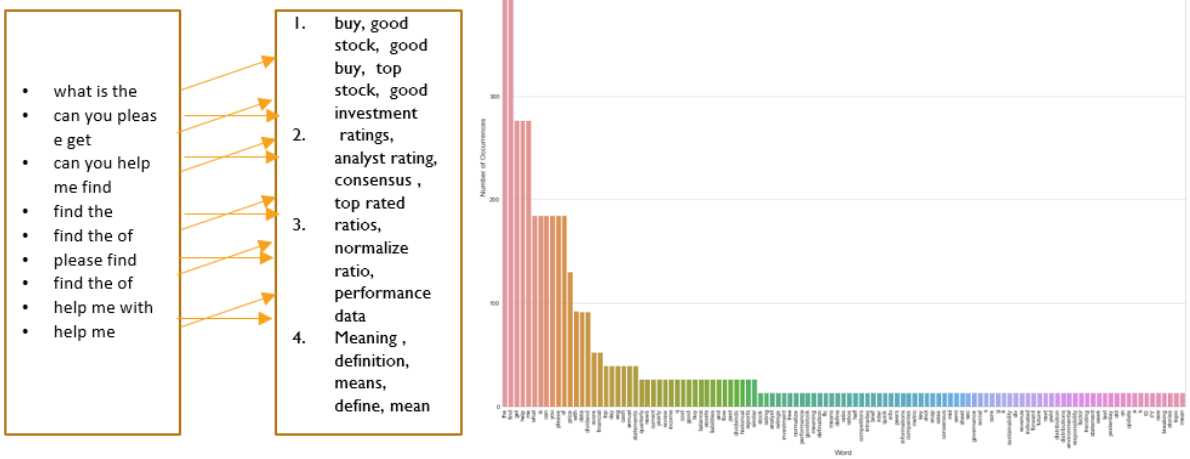
2. Data preparation, exploration, visualization

Classification data

Created by taking the combination of key terms (intent) and question start phrases to get 1196 documents for 18 intent classifications.

The text for classification consisted of short phrases that did not require any data wrangling.

Total number of words= 5050 : Total number of unique words= 115



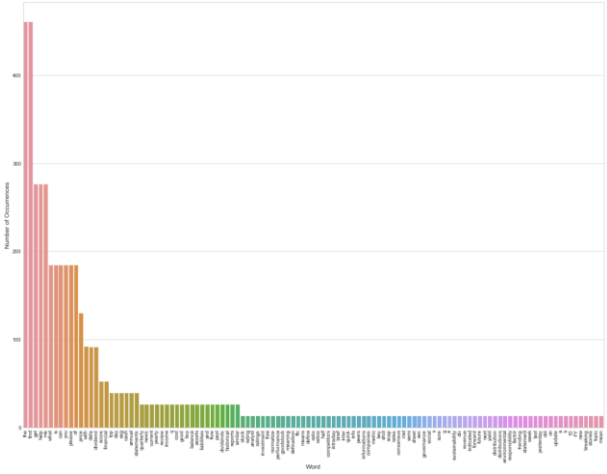
Information retrieval

I used a corpus of 170 documents for financial term definitions that would be the basis for the Q&A data retrieval. The Corpus was curated and did not require much preprocessing.

The document was tokenized by paragrah. For example:

['days inventory outstanding 360 days inventory turnover also known as inventory turnover in days it represents the number of days inventory sits in the warehouse in other words it measures the number of days from purchase of inventory to the sale
'accounts payable turnover net credit purchases ave accounts payable represents the number of times a company pays its accounts payable during a period a low ratio is favored because it is better to delay payments as much as possible so that the
'days payable outstanding 360 days accounts payable turnover also known as accounts payable turnover in days payment period it measures the average number of days spent before paying obligations to suppliers unlike dso and dio the longer the dpo
'operating cycle days inventory outstanding days sales outstanding measures the number of days a company makes 1 complete operating cycle i e purchase merchandise sell them and collect the amount due a shorter operating cycle means that the compar
'cash conversion cycle operating cycle days payable outstanding ccc measures how fast a company converts cash into more cash it represents the number of days a company pays for purchases sells them and collects the amount due generally like operat
'total asset turnover net sales average total assets measures overall efficiency of a company in generating sales using its assets the formula is similar to roa except that net sales is used instead of net income',
'debt ratio total liabilities total assets measures the portion of company assets that is financed by debt obligations to third parties debt ratio can also be computed using the formula 1 minus equity ratio',
'equity ratio total equity total assets determines the portion of total assets provided by equity i e owners contributions and the company s accumulated profits equity ratio can also be computed using the formula 1 minus debt ratio the reciprocal
'debt equity ratio total liabilities total equity evaluates the capital structure of a company a d e ratio of more than 1 implies that the company is a leveraged firm less than 1 implies that it is a conservative one',
'times interest earned ebit interest expense measures the number of times interest expense is converted to income and if the company can pay its interest expense using the profits generated ebit is earnings before interest and taxes']

After Stopword removal and lemmatization: Total number of words= 5086 : Total number of unique words= 939

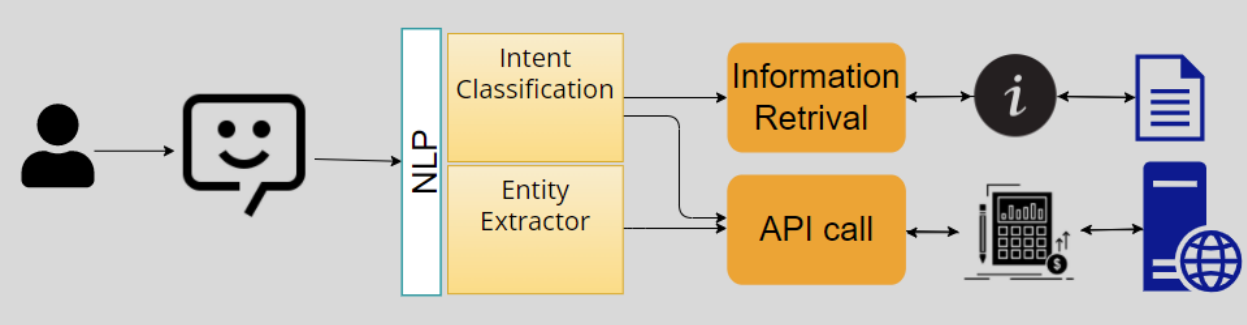


3. Research Design and Modeling Method(s)

Overview of the system design plan

NLP tasks for the project were:

- Intent classification
- Entity extraction
- Information retrieval



Ontology

I created the ontology in Protégé to gain understanding of the documents and select the number of intents for classification. The ontology helped me identify ambiguous entities that could either mean stock data retrieval or information retrieval. For example, if asked for “ratio”, it could mean getting price equity ratios for IBM or could be a request for what price to equity ratio means.

The verb prior and after the ambiguous term is important to understand the intent and based on this, I created the classification corpus:

LSTM and Bert transfer learning

For intent classification, I used a Long Short-Term Memory (LSTM) model and I finetuned a BERT- "bert-base-cased" model.

The LSTM model has an embedding layer to create embedding vectors. For the BERT model, I created input IDs and attention masks using the pretrained BERT encoder. The categories were one hot encoded. I used validation accuracy as the metric to evaluate the performance. Finally, I used categorical cross entropy to measure the loss between labels and predictions and SoftMax activation function for the output layer.

Model research was done using Python programming language and TensorFlow. TensorFlow is a free and open-source software library for machine learning and artificial intelligence and TensorFlow can be implemented in python by Keras. Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow which is extremely user-friendly.

I used the sci-kit learn library which has functions to create a confusion matrix and reports on precision, accuracy and F1 score.

Additionally, I used NumPy and Pandas library for data for EDA and finally, I also used seaborn package and matplotlib library to visualize the data and results.

Entity extraction

For Entity extraction is used the SPACY package in python with a pre-trained "en_core_web_lg" NER model which is trained on news and blog articles and has 685k keys and 685k unique vectors (300 dimensions).

Information Retrieval

The Corpus for information retrieval was tokenized to create the vocabulary that was used to create TF-IDF and Doc2Vec and Word2Vec embeddings. I experimented with Doc2Vec and Word2Vec embedding sizes of 100, 200 and 300 in order to find the best vectorization or embedding layer that could be used to retrieve information with cosine similarity score.

4. Results

LSTM and BERT Intent Classification:

The following 3 Experiments were done for classifying intent.

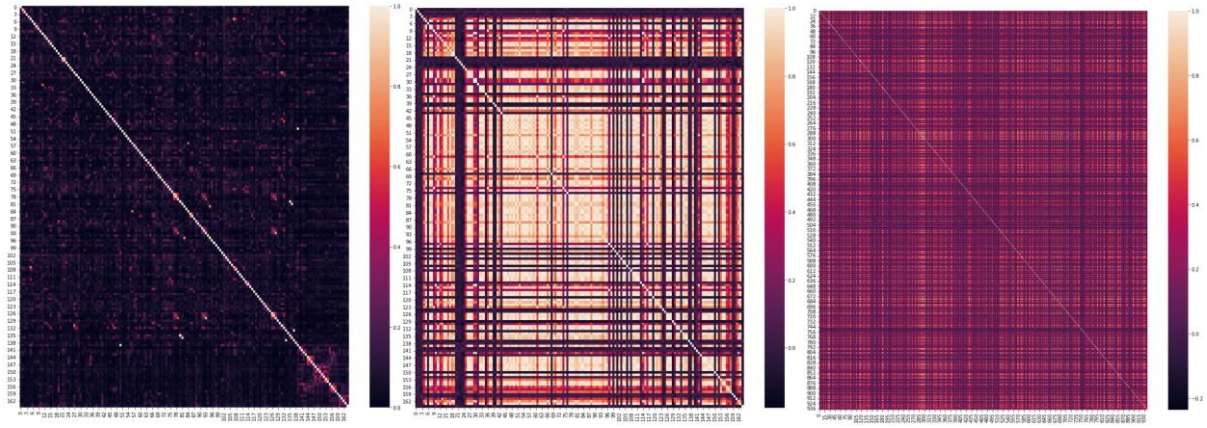
Name	Classification	Vocabulary size	Number of Intents	Type	Layers	optimizer	Train Acc	Val Acc	Test Acc
Model 1	Genre	max	18	LSTM	1 bidirectional 128 LSTM layer , 1 Dense layer 64	Adam	100	100	100
Model 2	Sentiment	max	18	BERT	Input_id , attention_mask 64 , Bert Transoformer layer, 1 Dense Layer 1024	Adam	31	41	32
Model 3	Sentiment	max	13	BERt	Input_id , attention_mask 64 , Bert Transoformer layer, 1 Dense Layer 1024	Adam	51	71	47

Information Retrieval

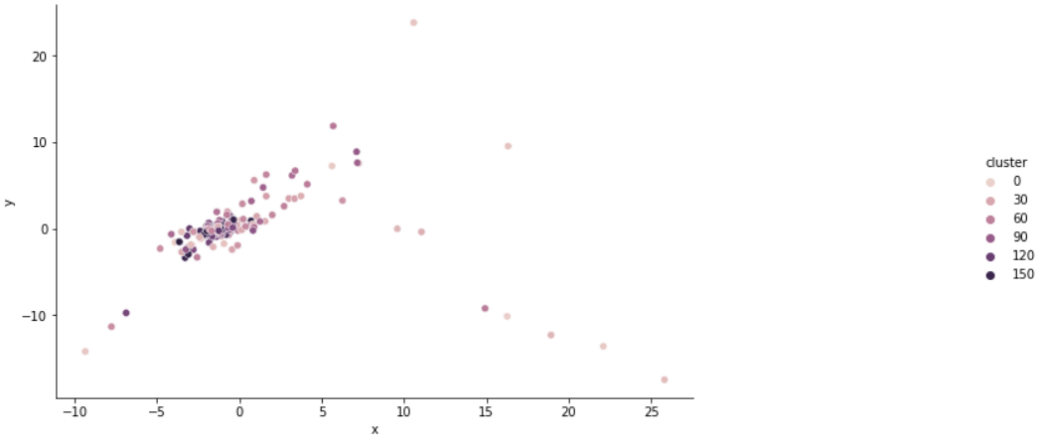
The experiments with TFIDF, Doc2vec and Word2vec give the following cosine-similarity heatmap.

The optimal size of Doc2vec and Word2vec embedding layer based on the cosine-similarity heatmap was 300.

TF-IDF Doc2vec Word2vec



K-means plot shows that there were no distinct clusters in the information retrieval document.



Entity Extraction

The following examples show cases where the SPACY model can extract Entities in text and cases where it fails to extract entities.

L is a stock symbol for Lowes Inc

```
1 displacy.render(doc, style= 'ent')
```

get price for Apple ORG

```
1 displacy.render(doc, style= 'ent')
```

get price for apple

```
1 displacy.render(doc, style= 'ent')
```

get price for microsoft ORG

```
1 displacy.render(doc, style= 'ent')
```

get price stock symbol L ORG

```
1 displacy.render(doc, style= 'ent')
```

get price for L

5. Analysis and Interpretation

Key Findings:

LSTM and BERT Models for intent classification

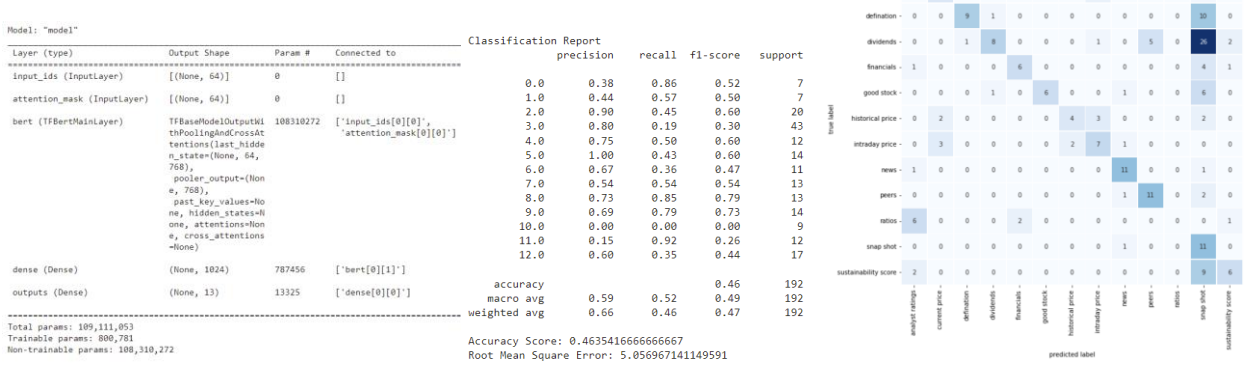
The accuracy on the small dataset that was used to train the LSTM model was 100% on train, test and validation data; however, the dataset was homogenous, and the training, testing and validation data was not distinct enough to provide an accurate result. Testing the chatbot using the LSTM model did not result in 100% accurate classification.

Fine-tuning a BERT-based model on the small dataset provided a realistic view of the model's performance. The model with a smaller number of intents performed with 50% accuracy, which was similar to what was reflected when using a chatbot. Using non-overlapping classes for intents and a richer dataset could significantly improve the model's performance.

The confusion matrix for the BERT model indicated that it could not distinguish between the overlapping intents of getting historical price, intraday price and current price. The same was true for ratios. It performed relatively poorly because there was ambiguity in the terms used to identify different intents.

Recommended Model

Name	Classification	Vocabulary size	Number of Intents	Type	Layers	optimizer	Train Acc	Val Acc	Test Acc
Model 3	Sentiment	max	13	BERt	Input_id , attention_mask 64 , Bert Transoformer layer, 1 Dense Layer 1024	Adam	51	71	47



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_ids (InputLayer)      [(None, 64)]              0         []

 attention_mask (InputLayer) [(None, 64)]              0         []

 bert (TFBertMainLayer)      TFBaseModelOutputWi       108310272 ['input_ids[0][0]',
                             thPoolingAndCrossAt                  'attention_mask[0][0]']
                             tentions(last_hidde
                             n_state=(None, 64,
                             768),
                              pooler_output=(Non
                             e, 768),
                              past_key_values=No
                             ne, hidden_states=N
                             one, attentions=Non
                             e, cross_attentions
                             =None)

 dense (Dense)               (None, 1024)              787456    ['bert[0][1]']

 outputs (Dense)             (None, 13)                13325     ['dense[0][0]']

__________________________________________________________________________________________________
Total params: 109,111,053
Trainable params: 800,781
Non-trainable params: 108,310,272
```

```
Classification Report
              precision    recall  f1-score   support

         0.0       0.38      0.86      0.52         7
         1.0       0.44      0.57      0.50         7
         2.0       0.90      0.45      0.60        20
         3.0       0.80      0.19      0.30        43
         4.0       0.75      0.50      0.60        12
         5.0       1.00      0.43      0.60        14
         6.0       0.67      0.36      0.47        11
         7.0       0.54      0.54      0.54        13
         8.0       0.73      0.85      0.79        13
         9.0       0.69      0.79      0.73        14
        10.0       0.00      0.00      0.00         9
        11.0       0.15      0.92      0.26        12
        12.0       0.60      0.35      0.44        17

    accuracy                           0.46       192
   macro avg       0.59      0.52      0.49       192
weighted avg       0.66      0.46      0.47       192

Accuracy Score: 0.4635416666666667
Root Mean Square Error: 5.056967141149591
```


Information Retrieval (Q&A)

The TF-IDF cosine similarity heatmap shows that diagonal clusters are more profound compared to word2vec and doc2vec. This indicates that TF-IDF was better at discerning between documents.

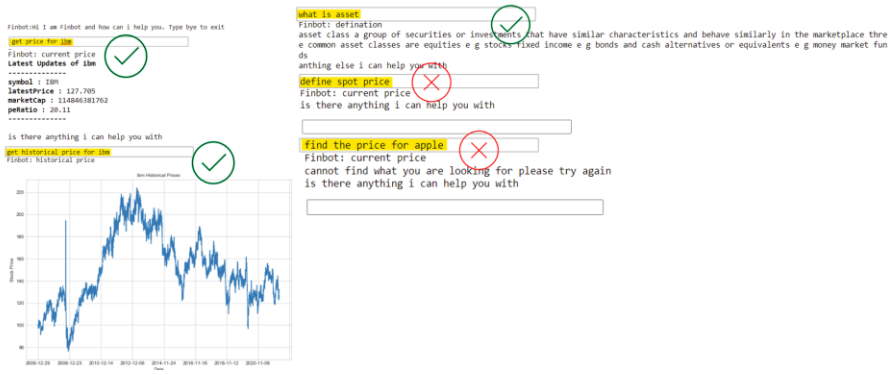
In theory, word2vec and doc2vec embedding layers can provide semantic meaning and therefore perform better than TF-IDF. However, the corpus was relatively small and could not extract any meaningful semantic information.

Entity Extraction

The Entity extraction examples above show the limitation of using a pretrained NER Model. The NER extraction is case sensitive for entities that share a name with a common noun and/or require context in the sentence.

Chatbot

The chatbot performed well if the user was very specific on its intent. The limitation was due to overlapping intents and because the training dataset was small and homogeneous. Also, the SPACY model had limitations in extracting entries.



6. Conclusion

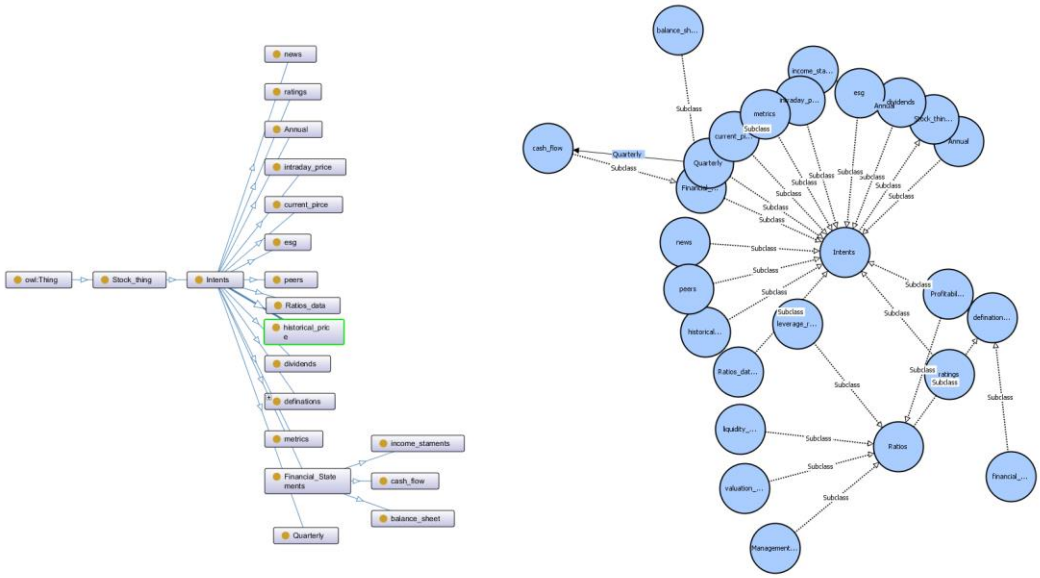
With limited training data for classification, I would recommend using transfer learning to train a deep neural net model. A BERT-based transformer model is ideal for such applications where there is limited training data. The BERT encoding layer can encode new data and can also create an embedding vector with input IDs and attention masks that can be fed into the BERT model. We can then train a dense input layer and output layer that can provide classification. The BERT model was limited based on the low quality of training data – I would recommend using the BERT model with richer/better data for future work.

Using an NER model for entity extraction was better than using a regular expression as the company names and stock symbols have common names or are represented by a single letter, such as “L” for Lowes. Adding entities to a pre-trained model or training an NER model can provide better results.

The corpus used for information retrieval (Q&A) was well-curated, as these documents are published on the company’s website. For the given corpus, TF-IDF provided better results compared to word2vec and doc2vec and it was computationally less expensive. In theory, word2vec and doc2vec embedding layers can provide semantic meaning and could therefore be better than TF-IDF. However, the corpus was relatively small and could not extract any meaningful semantic information.

Appendix:

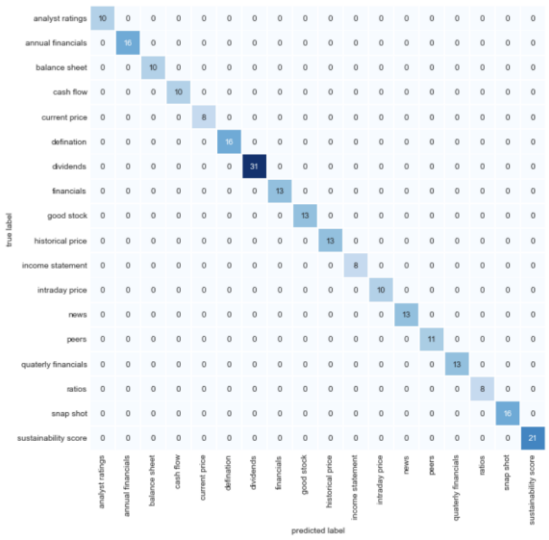
Appendix A



Appendix B

LSTM model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVec  (None, None)             0
 torization)

 embedding (Embedding)       (None, None, 32)          3712

 bidirectional (Bidirectiona  (None, 128)              49664
 l)

 dense (Dense)               (None, 64)                8256

 dense_1 (Dense)             (None, 18)                1170

=================================================================
Total params: 62,802
Trainable params: 62,802
Non-trainable params: 0
_________________________________________________________________
```



Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	16
2	1.00	1.00	1.00	10
3	1.00	1.00	1.00	10
4	1.00	1.00	1.00	8
5	1.00	1.00	1.00	16
6	1.00	1.00	1.00	31
7	1.00	1.00	1.00	13
8	1.00	1.00	1.00	13
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	8
11	1.00	1.00	1.00	10
12	1.00	1.00	1.00	13
13	1.00	1.00	1.00	11
14	1.00	1.00	1.00	13
15	1.00	1.00	1.00	8
16	1.00	1.00	1.00	16
17	1.00	1.00	1.00	21
accuracy			1.00	240
macro avg	1.00	1.00	1.00	240
weighted avg	1.00	1.00	1.00	240

Accuracy Score: 1.0
Root Mean Square Error: 0.0