greatlearning
Learning for Life

Final Interim report group 5 - Industrial safety NLP based Chatbot

Team mentor:
Madhan Sedhuraman

Team members:
1.Leelavathi G

2.Teepan Krishna N

3.J.V.S Srinivas

4.Sandep Kumar

5.Shashank S

Table of Contents

I. Summary of problem statement, data and findings:

A. Problem statement:

One of the Biggest challenges for Mining/Metal industries are to save their employees from Accidents/Injuries in workplace. Unfortunately, sometimes employees also die in such environment. Using AI techniques, we can help the professionals to highlight the safety risk as per the incident description.

B. Data Collection:

The database comes from one of the biggest industries in Brazil and downloaded from Kaggle.

C. Data description:

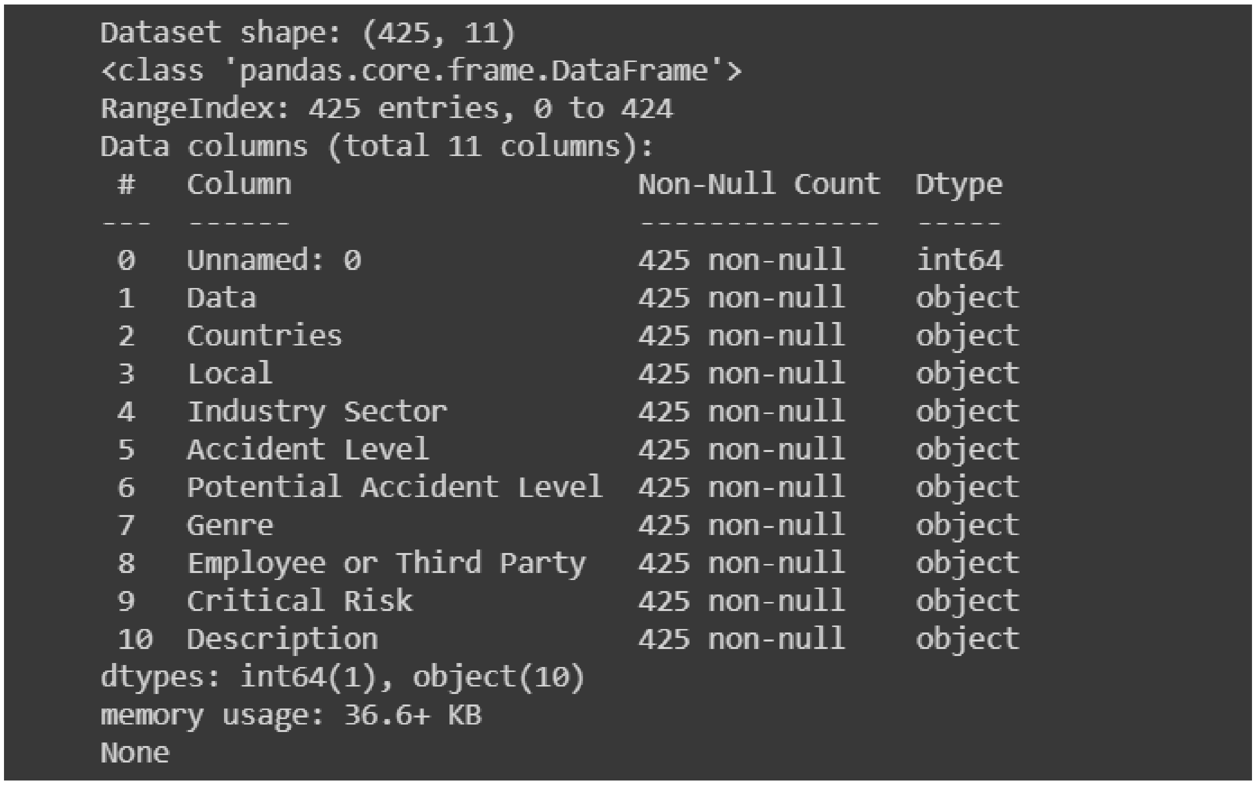
This The database is basically recording of accidents from 12 different plants in 03 different countries which every line in the data is an occurrence of an accident.

D. Columns description:

- Data: timestamp or time/date information
- Countries: which country the accident occurred (anonymized)
- Local: the city where the manufacturing plant is located (anonymized)
- Industry sector: which sector the plant belongs to
- Accident level: from I to VI, it registers how severe was the accident (I means not severe but VI means very severe)
- Potential Accident Level: Depending on the Accident Level, the database also registers how severe the accident could have been (due to other factors involved in the accident)
- Genre: if the person is male of female
- Employee or Third Party: if the injured person is an employee or a third party
- Critical Risk: some description of the risk involved in the accident
- Description: Detailed description of how the accident happened.

E. Understanding the dataset in detail:

Received dataset in excel(.xlsx) format used panda's library to read the dataset. Checking the null values and column data types.



```
Dataset shape: (425, 11)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 425 entries, 0 to 424
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                425 non-null    int64
 1   Data                      425 non-null    object
 2   Countries                 425 non-null    object
 3   Local                     425 non-null    object
 4   Industry Sector           425 non-null    object
 5   Accident Level            425 non-null    object
 6   Potential Accident Level  425 non-null    object
 7   Genre                     425 non-null    object
 8   Employee or Third Party   425 non-null    object
 9   Critical Risk             425 non-null    object
 10  Description               425 non-null    object
dtypes: int64(1), object(10)
memory usage: 36.6+ KB
None
```

From the shape we could understand that there are a total of 425 rows and 11 attributes in the dataset, there are all columns have categorical data and no null values.

Here Potential Accident Level is a target.

Please refer Image 1: Count of potential Accident level for visualization.

F. Count of Class and Target attributes:

Explanation:

This plot shows the count of accidents in each of the class levels which are again categorized into individual targets.

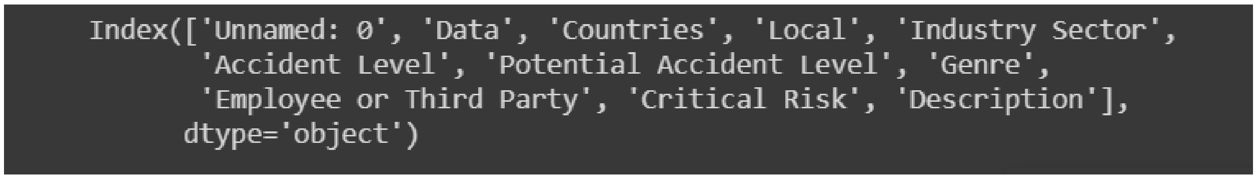
Observations:

- It's a multi class classification problem.

- The dataset is quite unbalanced, here we have only one record for class 'VI' and huge difference in the count of records between one another.
- Dataset has **class imbalance problem,** and we will be using few of the techniques to balance the classes distributions.
- Used **Up-sampling,** it is the process of randomly duplicating observations from the minority class using sklearn library resample function.
- Implemented **SMOTE** technique, it is an oversampling technique that generates synthetic samples from the minority class.
- Upweight the **down sampled** class, it the process of randomly taking the observations from the majority class.
- Among these Up-sampling technique works well to get the good accuracy.

G. Analysis on Dependent variables:

- We have 11 dependent variables in our dataset. Below are the column names.

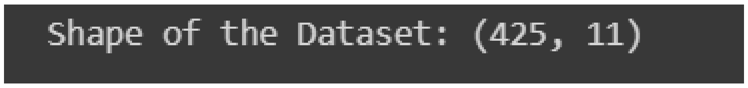


```
    Index(['Unnamed: 0', 'Data', 'Countries', 'Local', 'Industry Sector',
           'Accident Level', 'Potential Accident Level', 'Genre',
           'Employee or Third Party', 'Critical Risk', 'Description'],
          dtype='object')
```

- After performing univariate and bivariate analysis on the dependent columns, considering "Description"," Employee"," Industry Sector" is most useful data to predict potential accident level.
- After further analysis we have identified "Description" would be more weighted to predict the target.

H. Size of the dataset:

In this dataset we have only 425 records, it's a very small dataset to apply any Machine learning and deep learning algorithms.



```
Shape of the Dataset: (425, 11)
```

II. Overview of the final process:

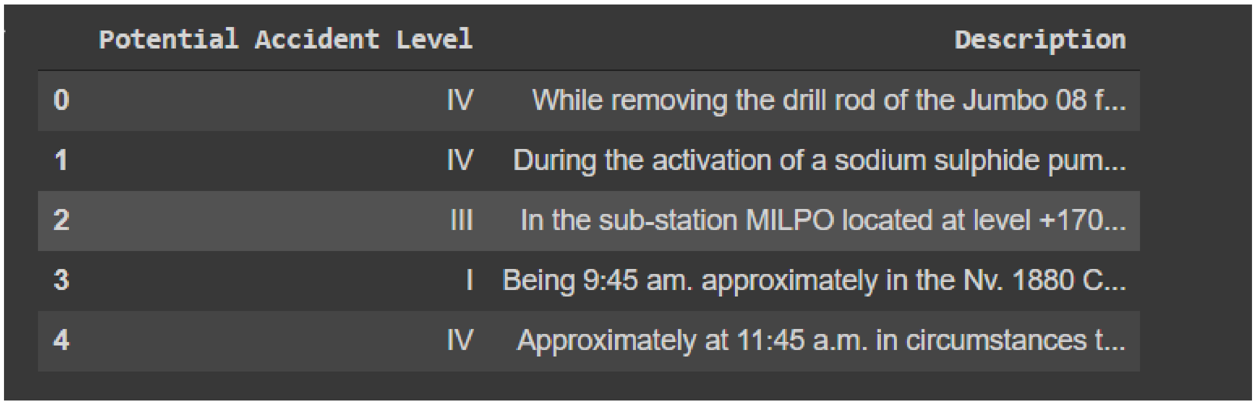
A. Text preprocessing:

1. Data Augmentation Techniques:

- Built a new dataset using Data augmentation techniques like **random deletion of verbs/random swapping of words.**
- Used **EDA** (Easy data Augmentation) algorithm for synonym replacement, deleting few words from the original dataset.
- We tried using one of the hugging face **library Pegasus paraphrase algorithms** to generate new descriptions from the original descriptions.
- These methods were not useful for the present dataset for accuracy.

2. Cleaning the text data

In dataset we have observed very few special characters and numeric values in text data, we have removed those things from accident description, updated string characters into lowercase and removed stop words as well to reduce the noise/vocabulary size of the text.



	Potential Accident Level	Description
0	IV	While removing the drill rod of the Jumbo 08 f...
1	IV	During the activation of a sodium sulphide pum...
2	III	In the sub-station MILPO located at level +170...
3	I	Being 9:45 am. approximately in the Nv. 1880 C...
4	IV	Approximately at 11:45 a.m. in circumstances t...

In the data preprocessing, we are only using the description column.

We have detailed accident information in description column as text data.

We have used **Tokenization** to Split each word into tokens from the sentences and applied **lemmatization** technique to bring the root word from the tokens.

We also tried **ngram, bigram and trigram** function to know most frequent words in the dataset.

We have tried **wordcloud** and **ImageColorGenerator** and displayed the most frequent words in the corpus.

Please refer Image 2: Most frequent words in the dataset for visualization.

We have cleaned the data and converted the text into word embeddings using the nltk vectorization methods.

We have used the frequency based TFIDF vectorizer and texts to sequence vectorizer to convert word embeddings.

In texts to sequence vectorizer only the most frequent words will be taken into account. Only the words known by the tokenizer will be considered and we have used this nltk function in final model.

We have taken max words as 6000 and max length as 100 and we have applied padding technique to give the same length of the input.

After applying multiple techniques and methodologies to up sample the classes, below mentioned dataset helps for better accuracy.

Please refer Image 3: final dataset with up sampling technique for visualization.

Final modified dataset has 800 records and 2 columns.

We had split the dataset into train and validation sets as 70:30 ratio. Applied all the preprocessing steps separately to the dataset.

We have encoded the target column into array using "to_categorical" function to fit the model.

B. Model building:

Below are the few Machine learning classifications models we have tried for the given problem statement

1. Logistic regression:

We tried logistic regression model to predict the potential accident level and we got around 82% accuracy and we have observed that the class-3 F1 score was too low and most of the class 3 accidents predicting as a class 0 or class 1.

2. KNN Classifier:

We tried K-Neighbors Classifier with the k range from 1 to 49, got the good accuracy where neighbors = 1, observed same problem as logistic. Most of the class 3 accidents predicted as a class0 and class 1.

3. SVC Classifier:

We tried Support vector classifier, Implemented different sets of kernel parameters like linear, polynomial and RBF.

We observed that RBF kernel gave around 85% accuracy, and the recall, f1 score was not that good.

4. Multinominal Naive Bias:

We tried Multinominal Naive Bayes classifier and suits to predict in multi-label classification. This model is not able to predict our target properly and accuracy was too low.

5. Decision Tree Classifier:

We tried Decision tree classifier,we observer this model gave around 66% accuracy and most of the target is predicted as class 1.

6. Random Forest Classifier:

We tried Random forest classifier, the random sampling technique used in the selection of the optimal splitting feature lowers the correlation and It improves the predictive capability of distinct trees in the forest.

Random Forest Classifier shows better results with the criterion parameter “entropy”, with “max_depth” of “100” and “n_estimators” as “200”. F1 score and recall scores are much better.

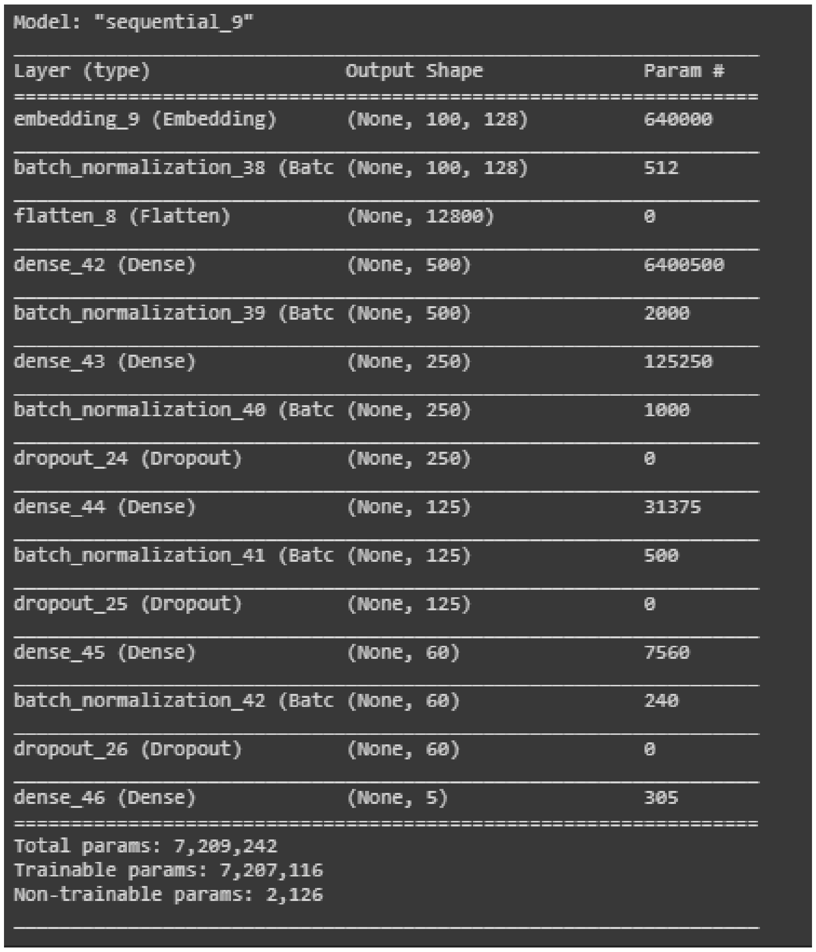
7. Voting Classifier:

We tried voting classifier, it simply aggregates the findings of each model and passes it into the Voting Classifier and it predicts the output of the class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class, and we have got 75% accuracy by using the voting classifier.

8. Neural network classifier:

We have trained the dataset using Vanilla neural network sequential model with 5 hidden layers, activation function as “RELU”, output layer with 5 neurons, SoftMax activation function and we observed overfitting, to avoid overfitting we have further added the Batch Normalization and dropout layers into the Vanilla neural network model.

We have compiled the model using “categorical_crossentropy” as loss and optimizer as “Adagrad”.



```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_9 (Embedding)      (None, 100, 128)          640000
_________________________________________________________________
batch_normalization_38 (Batc (None, 100, 128)          512
_________________________________________________________________
flatten_8 (Flatten)          (None, 12800)             0
_________________________________________________________________
dense_42 (Dense)             (None, 500)               6400500
_________________________________________________________________
batch_normalization_39 (Batc (None, 500)               2000
_________________________________________________________________
dense_43 (Dense)             (None, 250)               125250
_________________________________________________________________
batch_normalization_40 (Batc (None, 250)               1000
_________________________________________________________________
dropout_24 (Dropout)         (None, 250)               0
_________________________________________________________________
dense_44 (Dense)             (None, 125)               31375
_________________________________________________________________
batch_normalization_41 (Batc (None, 125)               500
_________________________________________________________________
dropout_25 (Dropout)         (None, 125)               0
_________________________________________________________________
dense_45 (Dense)             (None, 60)                7560
_________________________________________________________________
batch_normalization_42 (Batc (None, 60)                240
_________________________________________________________________
dropout_26 (Dropout)         (None, 60)                0
_________________________________________________________________
dense_46 (Dense)             (None, 5)                 305
=================================================================
Total params: 7,209,242
Trainable params: 7,207,116
Non-trainable params: 2,126
_________________________________________________________________
```

for “batch_size”: ‘8’ and “epochs” as ‘50’ we have got validation a ccuracy as 85% and the validation loss as 0.29. We have observed so many fluctuations in accuracy, and even after adding the regularizations methods we did not see much difference in accuracy.

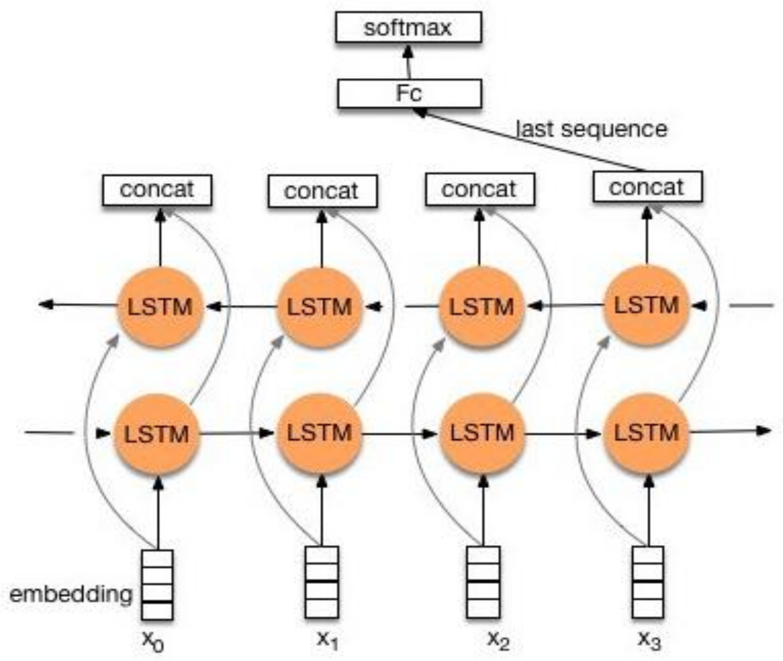
Please refer Image 4: Accuracy results for Vanilla Neural network for visualization.

III. Step-by-step walk through the solution:

A. Final Model (Bi-Directional LSTM Network):

Long Short-Term Memory (LSTM) networks are capable of learning order dependence in sequence prediction problems.

We have built the LSTM Network, observed high training loss, to avoid these we have added Bidirectional LSTM layer into the neural network. Bi-directional layer can do a reverse flow of information as well. In problems where all timesteps of the input sequence are available, Bidirectional LSTM train two instead of one LSTMs on the input sequence. This can provide additional context to the network the first on the input sequence as is and the second on a reversed copy of the file and result in faster and even fuller learning on the problem.



First, we must reshape the input and output sequences to be 3-dimensional to meet the expectations of the LSTM. The expected structure has the dimensions [samples, timesteps, features]. In an input layer giving the input 3D shape.

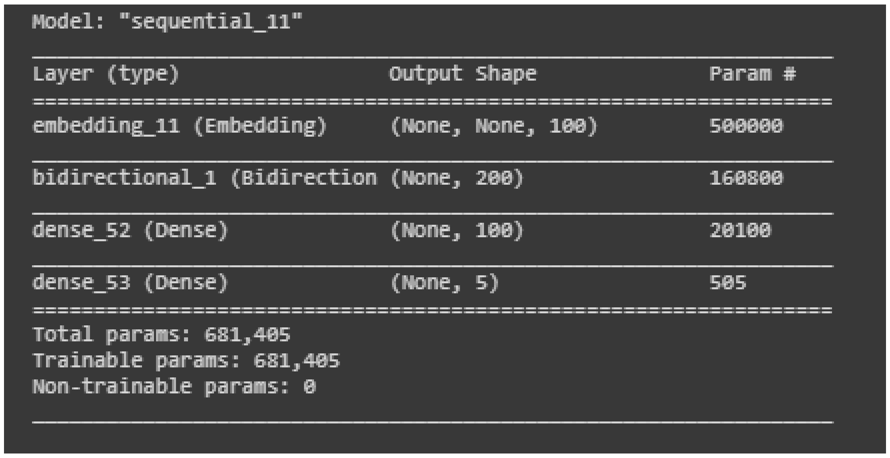
The first hidden layer will have 5000 memory units and the output layer will be a fully connected layer that outputs five value per timestep as it’s a multi class prediction problem. A sigmoid activation function is used on the output to predict the binary values.

The Embedding layer is initialized with random weights and will learn embedding for all the words in the training dataset. Here It is learning along with the model itself.

wrapping the LSTM hidden layer with a Bidirectional layer, as follows:

```
lstm_model_2.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(100, return_s
tate=False,recurrent_dropout=0.1)))
```

This will create two copies of the hidden layer, one fit in the input sequences as-is and one on a reversed copy of the input sequence. By default, the output values from these LSTMs will be concatenated.



```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_11 (Embedding)     (None, None, 100)         500000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 200)               160800
_________________________________________________________________
dense_52 (Dense)             (None, 100)               20100
_________________________________________________________________
dense_53 (Dense)             (None, 5)                 505
=================================================================
Total params: 681,405
Trainable params: 681,405
Non-trainable params: 0
_________________________________________________________________
```

Added dense layer which means each neuron in the dense layer receives input from all neurons of its previous layer and its basically used for changing the dimensions of the vector.

The dense layer function of Keras implements following operation:

output = activation (dot (input, kernel) + bias)

RELU activation is used for performing element-wise activation and the kernel is the weights matrix created by the layer, and bias is a bias vector created by the layer.

Finally, as it is a binary classification problem, the binary log loss ("binary crossentropy" in Keras) is used. The efficient ADAM optimization algorithm is used to find the weights and the accuracy metric is calculated and reported at each epoch.

The LSTM will be trained for 20 epochs. A new random input sequence will be generated each epoch for the network to be fit on. This ensures that the model does not memorize a single sequence and instead can generalize a solution to solve all possible random input sequences for this problem.

While fitting the model Used train dataset as train and test data.

We have got Bidirectional LSTM Model Accuracy as 0.8625 and Validation loss of 0.422.

B. Front end desktop application for Chatbot (Tkinter):

For front end we have created a desktop Chatbot application using Tkinter,

- We have deployed the final LSTM model, model weights, tokenizer, and required columns as a. json and .h5 files
- We have updated the python codes to load the required model data from the pre saved model deployment files
- We have created a desktop application using Tkinter and integrated the same to the LSTM Model to predict the potential accident level from the given Description
- We have built the solution as a chatbot, where the application asks for the name first, and then asks for the accident description and then replies the predicted potential accident level to the user
- We have provided the option to change the font, clear chat, exit and also other options to change the backend color of the application
- We have also created the respective .exe extension file for the .py file using the pyinstaller
- Please check below video, for walkthrough of Tkinter Desktop Chatbot Application

NLP-2 Chatbot (Tkinter).mp4

IV. Model evaluation:

Choosing an appropriate metric is challenging and also it is difficult for imbalanced classification problems.

There are standard metrics that are widely used for evaluating classification models, such as classification accuracy or classification error.

- **Accuracy** = Correct Predictions / Total Predictions
- **Error** = Incorrect Predictions / Total Predictions

This challenge is made even more difficult when there is a skew in the class distribution. The reason for this is that many of the standard metrics become unreliable when classes are imbalanced.

Imbalanced classification problems typically rate classification errors with the minority class as more important than those with the majority classes. As such performance metrics may be needed the focus on the minority classes, which is made challenging because it is the minority classes where we lack observations required to train an effective model.

The F1-Score is a popular metric for imbalanced classification.

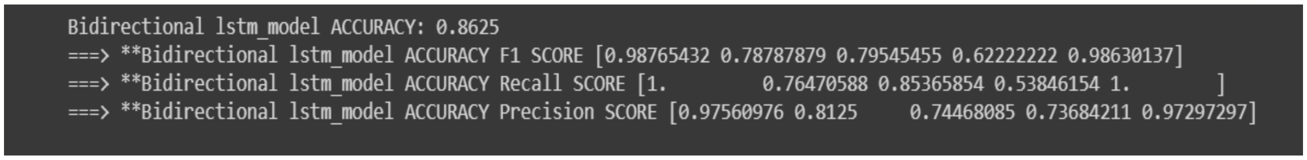
- **F-Measure** = (2 * Precision * Recall) / (Precision + Recall)

Always Observes the Confusion matrix to identify how many classes are incorrect predictions, those incorrect predictions are predicted as which class.

The below table is the confusion matrix for the Bi-directional LSTM model.

Please refer Image 5: Performance of a classification model for the LSTM Model for Visualization.

The below are the F1 score, recall and precision for the Bi-directional LSTM model.



```
Bidirectional lstm_model ACCURACY: 0.8625
===> **Bidirectional lstm_model ACCURACY F1 SCORE [0.98765432 0.78787879 0.79545455 0.62222222 0.98630137]
===> **Bidirectional lstm_model ACCURACY Recall SCORE [1.         0.76470588 0.85365854 0.53846154 1.        ]
===> **Bidirectional lstm_model ACCURACY Precision SCORE [0.97560976 0.8125     0.74468085 0.73684211 0.97297297]
```

We achieved the good f1 core for Bi-directional LSTM model. For class 0(means accident level1), and class 4 have pretty good score, and even other class scores are good.

Observed very less difference in train and validation accuracy, as well as log loss. among all the models and also Bidirectional LSTM model have No fluctuations on accuracy and log loss.

Please refer Image 6: Model accuracy and loss for the LSTM Model for visualization records.

V. Comparison to benchmark:

After analyzing all the Machine learning and deep learning models evaluation met rics and by considering accuracy and f1 score, we have
finalized the Bidirectional LSTM model and also it is not as overfitted when compared to the other model and it has less modulations in the loss.

Model	**Accuracy**	**F1_score**	**Comments**
Logistic Regression	0.875	[0.97560976 0.72222222 0.89411765 0.60465116 0.97297297]	Accuracy is good,f1 score for all classes are not good
KNN Classifier	0.80625	[0.86956522 0.75862069 0.83333333 0.42424242 0.87804878]	Accuracy is 80, class3 f1 score is too less
SVC Classifier	0.875	[0.97560976 0.73333333 0.83870968 0.68292683 0.97297297]	Accuracy is good
Multinominal Naive bais Classifier	0.49375	[0.62686567 0.3 0.51485149 0.34782609 0.54545455]	Accuracy is poor
DecisionTreeClassifier	0.65	[0.78571429 0.10526316 0.64516129 0.25641026 0.82352941]	Accuracy is poor
RandomForestClassifie r	0.86875	[0.95238095 0.78571429 0.8125 0.66666667 0.98630137]	Accuracy is good
VotingClassifier hard voting	0.733333 3	[0.81690141 0.59574468 0.65714286 0.6122449 0.92063492]	Accuracy is not good
VotingClassifier soft voting	0.76	[0.86567164 0.64 0.69565217 0.68 0.875]	Accuracy is not too good
Vanila Neural Network	0.85	[0.96385542 0.7027027 0.80519481 0.66666667 0.97222222]	Accuracy is good
Bi directional Neural Network	0.81875	[1. 0.63414634 0.73563218 0.53658537 0.98591549]	Accuracy and f1 score is good

Note: We have a separate python Notebook for all the models, kindly refer (Industrial safety - NLP (EDA and Models).ipynb)

VI. Visualizations:

A. Count of potential Accident level:

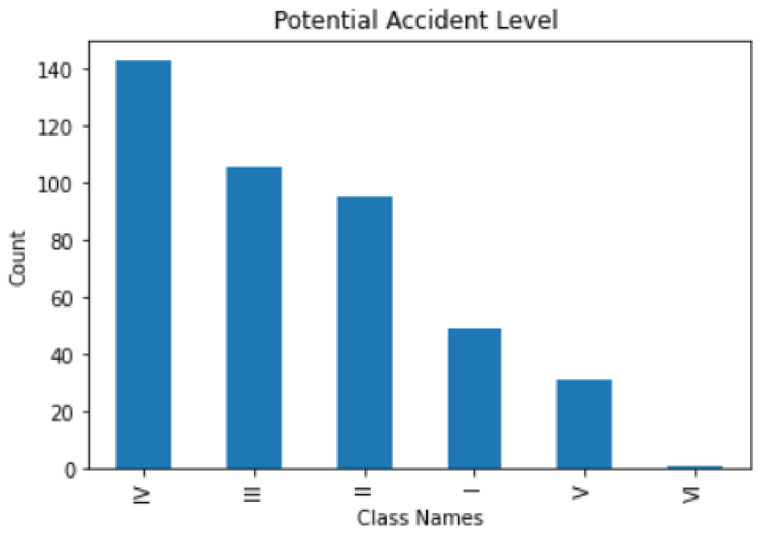


Image 1: Count of potential Accident level

B. Most frequent words in the dataset:

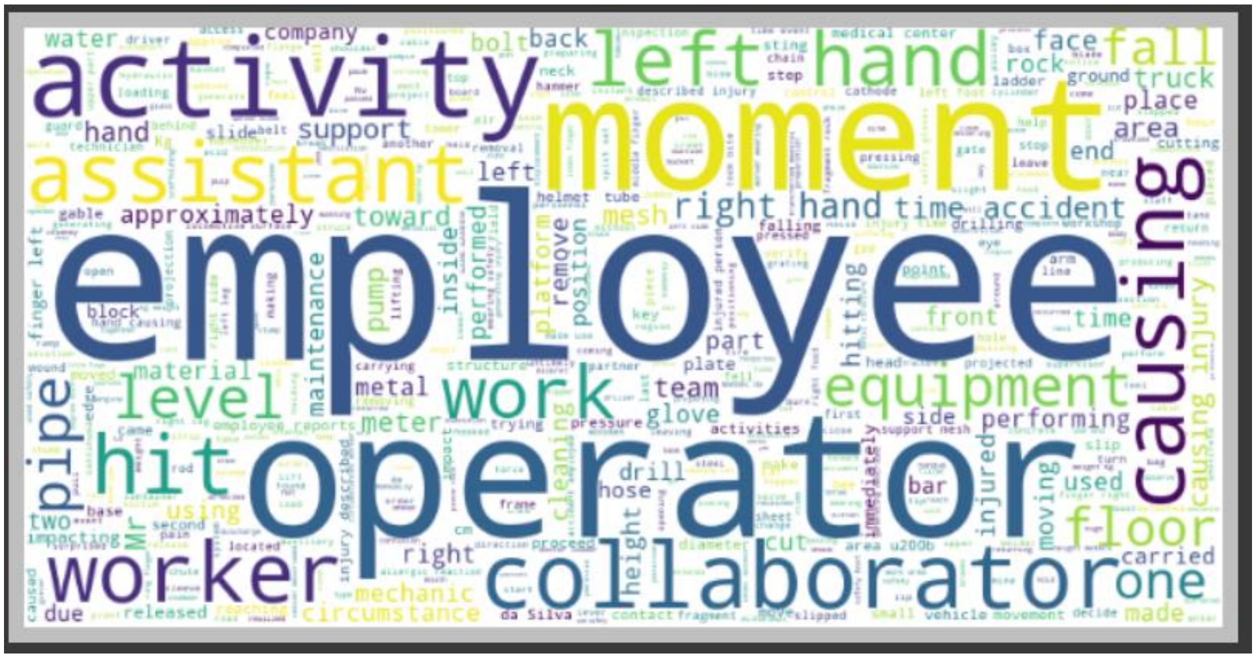


Image 2: Most frequent words in the dataset

C. Up sampled dataset:

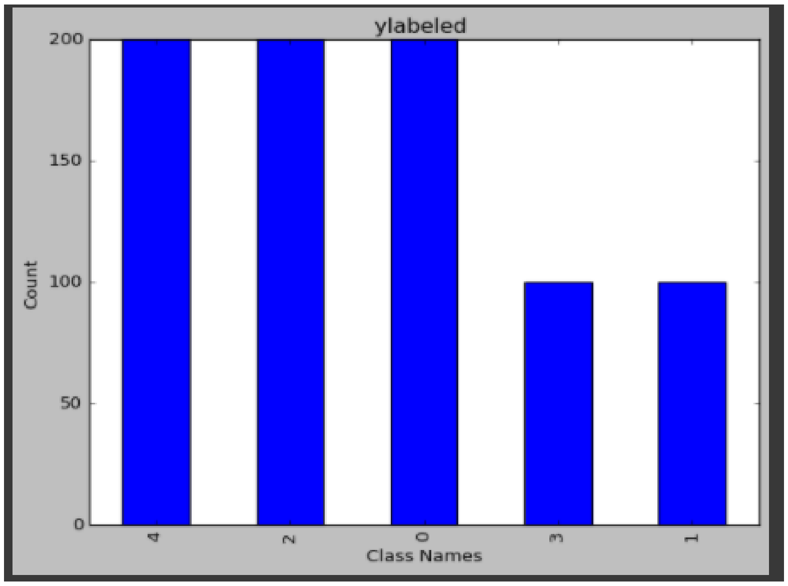


Image 3: final dataset with up sampling technique

D. Accuracy results for vanilla Neural network:

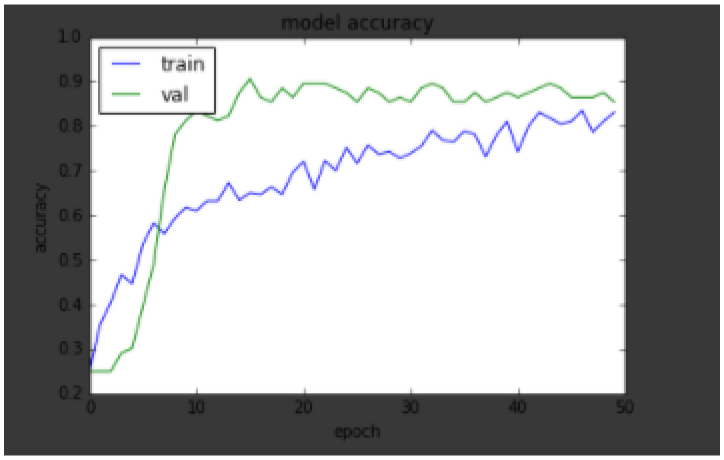
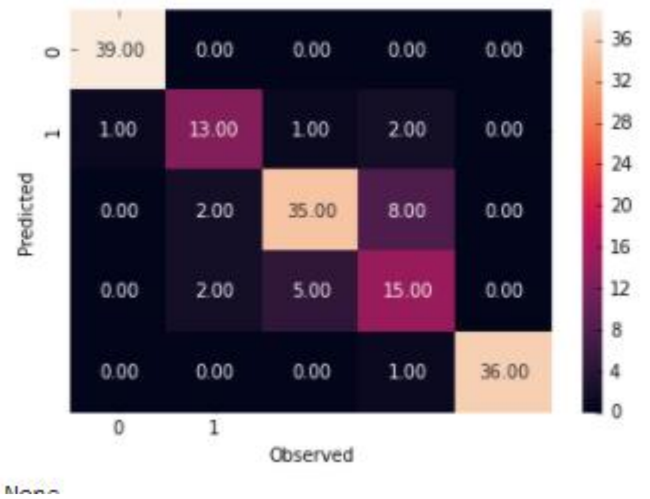


Image 4: Accuracy results for Vanilla Neural network

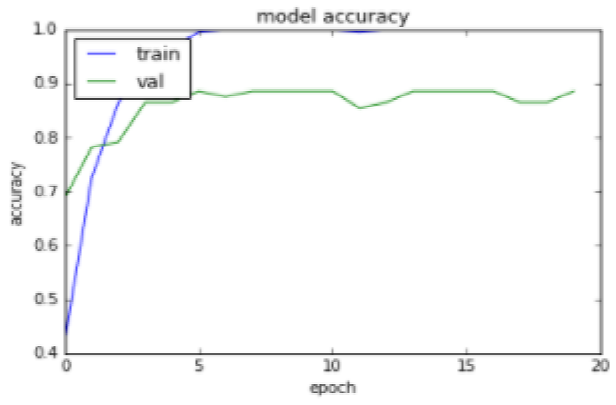
E. Performance of a classification model:



0	39.00	0.00	0.00	0.00	0.00
1	1.00	13.00	1.00	2.00	0.00
	0.00	2.00	35.00	8.00	0.00
	0.00	2.00	5.00	15.00	0.00
	0.00	0.00	0.00	1.00	36.00
	0	1			

Image 5: Performance of a classification model for the LSTM Model

F. Model accuracy and loss for the LSTM Model:



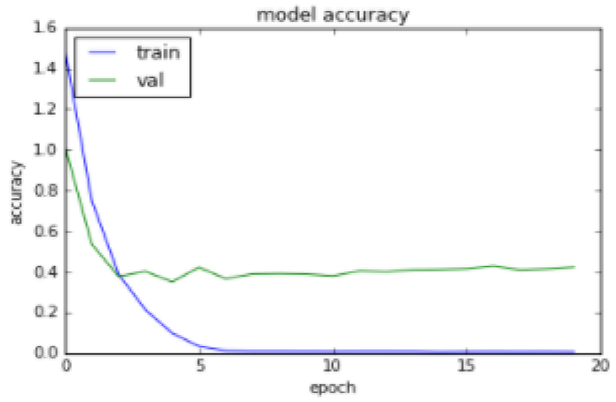


Image 6: Model accuracy and loss for the LSTM Model

VII. Implications:

- The model can predict the severity of the accident/incident on a real time basis without any human bias or error
- If we extend it, the model can give suggestions to the management as well as workers in that location with certain safety features that can be followed and also helps to eradicate such incidents or accidents in future
- By adding feedback propagation technique, we can help to improve the accuracy

VIII. Limitations:

- This model does not implement any feedback propagation technique which means the model does not learn from its mistake or does not adapt in a real time scenario.

  Enhancement: The feedback propagation technique can be implemented to improve the efficiency of the model over the time

- At this point of time, the model can only suggest safety measures to the management as well as workers.

  Enhancement: The application can be fully automated by integrating a system that can monitor using computer vision techniques that way the model can give warning to an appropriate executive if the measures are not followed by the worker.

- At this point of time, the model does not inform the incident to any emergency provider for getting immediate help.

  Enhancement: The model can be extended to look for emergency help that can be provided to the accident location on a priority basis while collecting the information. This can help in timely treatment of the injured.

IX. Closing Reflections:

A. Model Building:

- Understanding each and every component in the data set, it helps to recognize and identify patterns in it.
- We must be more careful when we are dealing with Imbalanced classification data, it has a chance to be overfit the results.
- Data Preprocessing, this step plays an important role in improving the accuracy. Different vectoring techniques gives different results.
- Removing special characters, numeric numbers and stop words, this cleaned data gives rich information to the model.
- While changing the text into vector space of matrix fitting the train and test data separately reduces the overfitting and helps increasing the accuracy.
- Applying data augmentation may help when we have less records, but in this case, we have an overfitting issue.
- Up sampling helps us to deal with this class imbalanced dataset.
- It’s Multiclass classification problem. Initially we have seen overfitting results due to imbalanced multiclass classification problem.
- After adding regularization and dropout layers into Neural networks we have not seen any overfitting results in results.
- Bi-directional LSTM model gave the best results, as it has reverse flow of information
- Good to tune the model with different Loss, Optimization and batch sizes.
- In Evaluation matrix we have considering accuracy, F1 score mainly, always trying to reduce the false positives.
- Visualization helps to identify the oscillations, variations in the model at all the batches.

B. Front end Chatbot:

- For front end chatbot capability, we have built it using Tkinter desktop application and also tried using Microsoft bot emulator, we got opportunity to explore Model deployment part, chatbot framework logic, and also the Desktop application using Tkinter
- For chatbot we have tried it using the MS Bot emulator and able to complete the code and test it using the emulator and for deploy to azure since it was payable, I have stopped that approach to deploy it and then went for Tkinter desktop application.
- For Tkinter desktop application we were able to create a .exe file using Pyinstaller.