Natural Language Processing: Chatbot Interface

Capstone Project: Jun' 24

Batch: June 23 B Capstone Project Group-6

Mentor: Jyant Mahara

Team members : Rohit, Ganesh, Ankita, Shweta, Ambrish

Date : 16th June 2024
Milestone 1 Submission
—---------------------------------------
Date : 07th July 2024
Milestone 2 Submission

Table of Contents

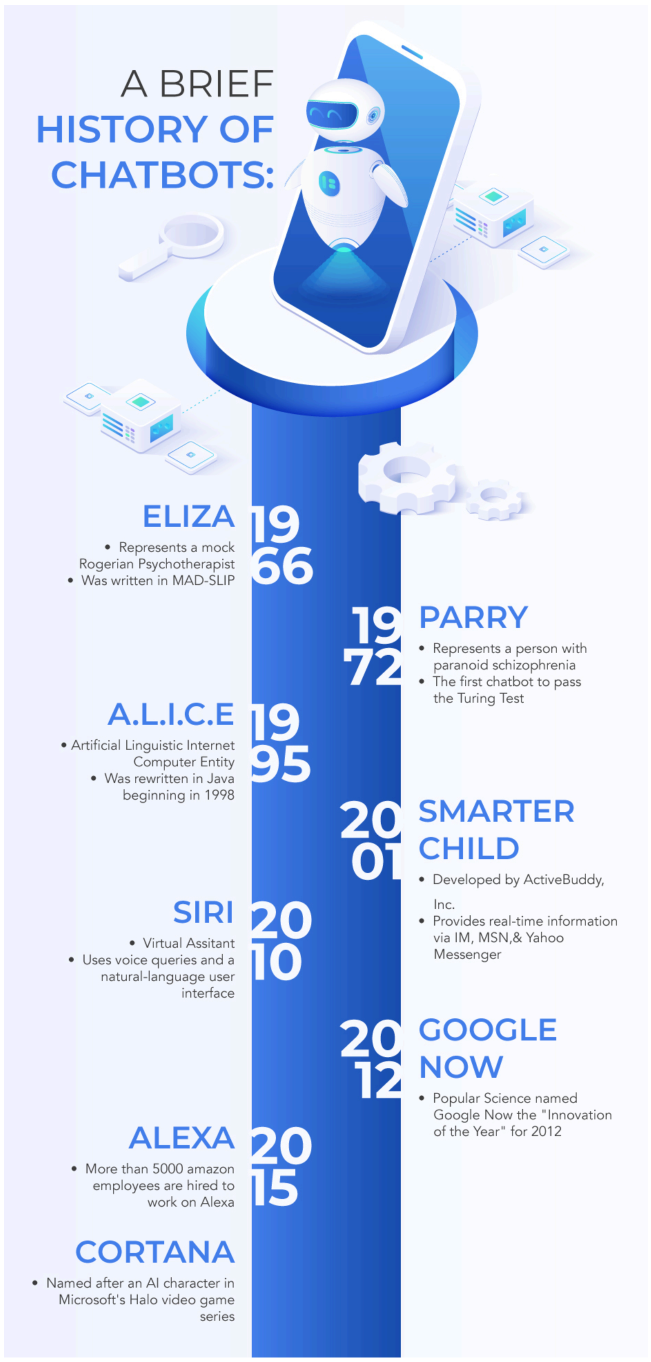
Introduction

As a part of the Capstone project for the AIML course, we need to design a chatbot that would help in improving industrial safety. As part of this project, we will use the database from one of the biggest industries in Brazil and in the world.
It is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants. Sometimes they also die in such a hazardous environment.

Overview

Chatbots are not a recent development. They are simulations that can understand human language, process it, and interact back with humans while performing specific tasks. For example, a chatbot can be employed as a helpdesk executive. Joseph Weizenbaum created the first chatbot in 1966, named Eliza.

It all started when Alan Turing published an article named “Computer Machinery and Intelligence” and raised an intriguing question, “Can machines think?” ever since, we have seen multiple chatbots surpassing their predecessors to be more naturally conversant and technologically advanced. These advancements have led us to an era where conversations with chatbots have become as normal and natural as with another human.


A BRIEF HISTORY OF CHATBOTS:
ELIZA 1966
• Represents a mock Rogerian Psychotherapist
• Was written in MAD-SLIP
PARRY 1972
• Represents a person with paranoid schizophrenia
• The first chatbot to pass the Turing Test
A.L.I.C.E 1995
• Artificial Linguistic Internet Computer Entity
• Was rewritten in Java beginning in 1998
SMARTER CHILD 2001
• Developed by ActiveBuddy, Inc.
• Provides real-time information via IM, MSN,& Yahoo Messenger
SIRI 2010
• Virtual Assitant
• Uses voice queries and a natural-language user interface
GOOGLE NOW 2012
• Popular Science named Google Now the "Innovation of the Year" for 2012
ALEXA 2015
• More than 5000 amazon employees are hired to work on Alexa
CORTANA
• Named after an AI character in Microsoft's Halo video game series

Identifying opportunities for an Artificial Intelligence chatbot

The first step is to identify the opportunity or the challenge to decide on the purpose and utility of the chatbot. To understand the best application of Bot to the company framework, you will have to think about the tasks that can be automated and augmented through Artificial Intelligence Solutions. The respective artificial intelligence solution broadly falls under two categories for each type of activity: “Data Complexity” or “Work Complexity”. These two categories can be further broken down into 4 analytics models: *Efficiency, Expert, Effectiveness, and Innovation.*

Types of Chatbots

There are many types of chatbots available. A few of them can be majorly classified as follows:

- Text-based chatbot: In a text-based chatbot, a bot answers the user's questions via a text interface.
- Voice-based chatbot: In a voice or speech-based chatbot, a bot answers the user's questions via a human voice interface.

There are mainly two approaches used to design the chatbots, described as follows:

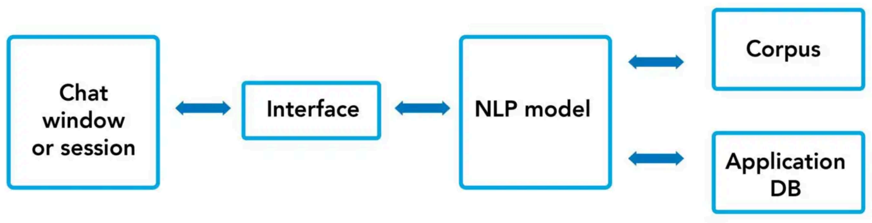
- In a Rule-based approach, a bot answers questions based on some rules on which it is trained on. The rules defined can be very simple to very complex. The bots can handle simple queries but fail to manage complex ones.
- Self-learning bots are the ones that use some Machine Learning-based approaches and are definitely more efficient than rule-based bots. These bots can be further classified into two types: Retrieval Based or Generative.

There are many types of chatbots available, depending on the complexity. A few of them can be majorly classified as follows:

- Traditional chatbots: They are driven by system and automation, mainly through scripts with minimal functionality and the ability to maintain only system context.
- Current chatbot: They are driven by back-and-forth communication between the system and humans. They have the ability to maintain both system and task contexts.
- Future chatbot: They can communicate at multiple levels with automation at the system level. They have the ability to maintain the system, task, and people contexts. There is a possibility of introducing master bots and eventually a bot OS.

The Architecture of chatbots

Typical chatbot architecture should consist of the following:



- Chat window/session/front end application interface
- The deep learning model for Natural Language Processing [NLP]
- Corpus or training data for training the NLP model
- Application Database for processing actions to be performed by the chatbot

Problem Statement

DOMAIN : Industrial safety. NLP based Chatbot.

CONTEXT : The database comes from one of the biggest industries in Brazil and in the world. It is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants. Sometimes they also die in such an environment.

DATA DESCRIPTION: The database is basically records of accidents from 12 different plants in 03 different countries which every line in the data is an occurrence of an accident.
Columns description:

- **Data:** timestamp or time/date information
- **Countries:** which country the accident occurred (anonymised)
- **Local:** the city where the manufacturing plant is located (anonymised)
- **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.

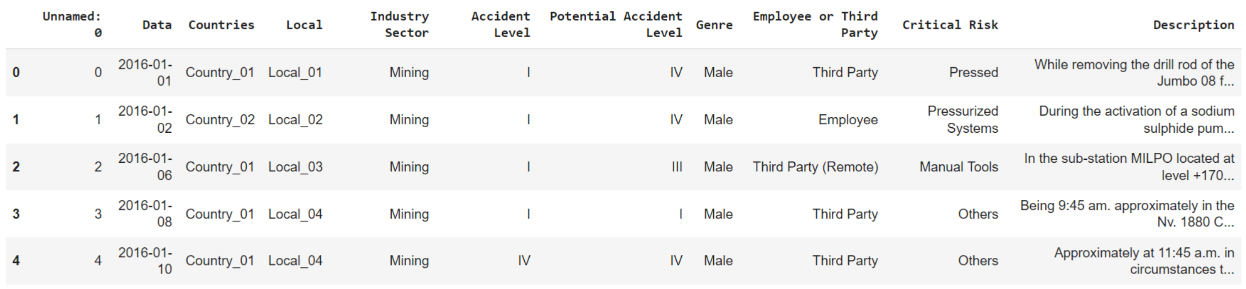
PROJECT OBJECTIVE: Design a ML/DL based **chatbot utility** which can help the professionals to highlight the safety risk as per the incident \ description.

Data Analysis

Exploratory Data Analysis

Reviewing the data and using univariate and bivariate analysis gives some insights about our data like identifying patterns, detecting anomalies, formulating hypothesis and also preparing our data for modeling.

Raw data sample



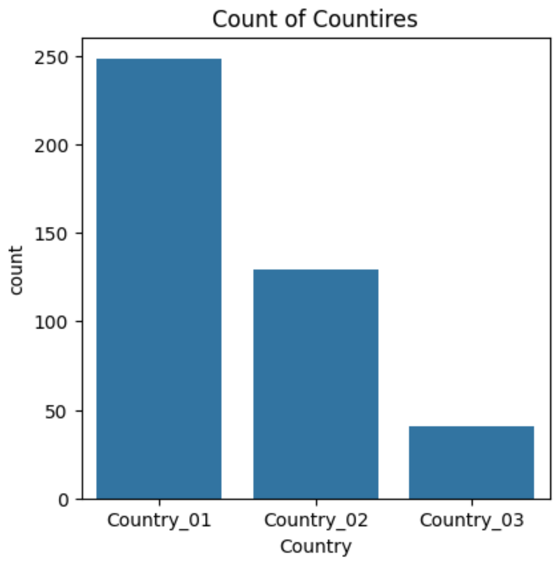
	Unnamed: 0	Data	Countries	Local	Industry Sector	Accident Level	Potential Accident Level	Genre	Employee or Third Party	Critical Risk	Description
0	0	2016-01-01	Country_01	Local_01	Mining	I	IV	Male	Third Party	Pressed	While removing the drill rod of the Jumbo 08 f...
1	1	2016-01-02	Country_02	Local_02	Mining	I	IV	Male	Employee	Pressurized Systems	During the activation of a sodium sulphide pum...
2	2	2016-01-06	Country_01	Local_03	Mining	I	III	Male	Third Party (Remote)	Manual Tools	In the sub-station MILPO located at level +170...
3	3	2016-01-08	Country_01	Local_04	Mining	I	I	Male	Third Party	Others	Being 9:45 am. approximately in the Nv. 1880 C...
4	4	2016-01-10	Country_01	Local_04	Mining	IV	IV	Male	Third Party	Others	Approximately at 11:45 a.m. in circumstances t...

Data Insights

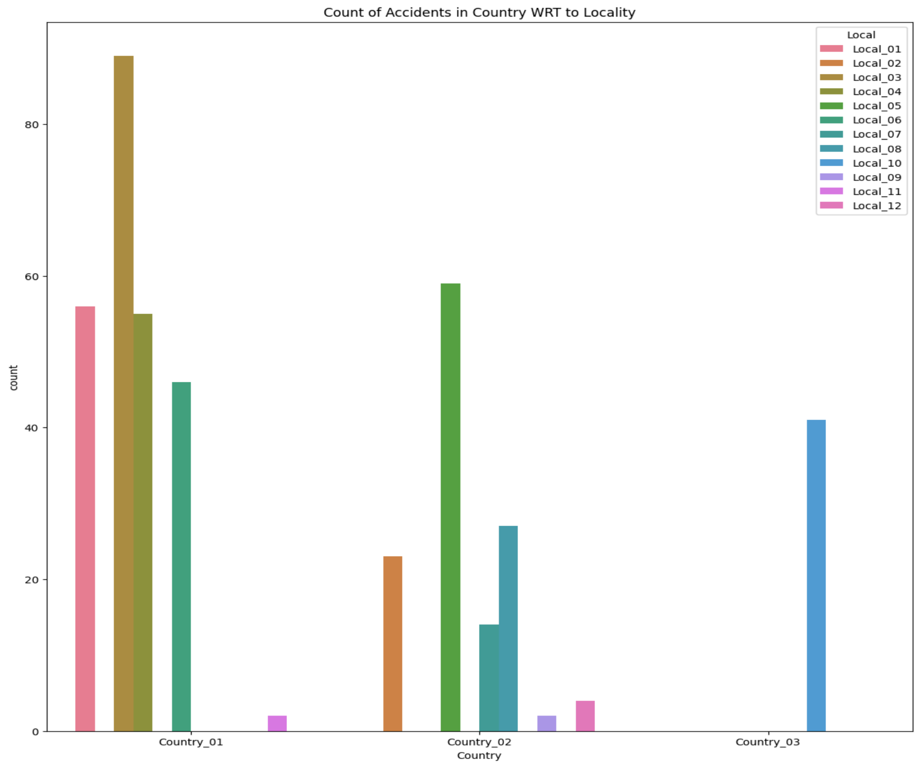
Dataset Size : 425 rows x 11 columns
Dataset type : DateTime: 1; int64: 1; Object: 9
Missing data : There are no rows/columns with missing data. Data quality seems to be good.
Location data has been kept anonymous

EDA insights

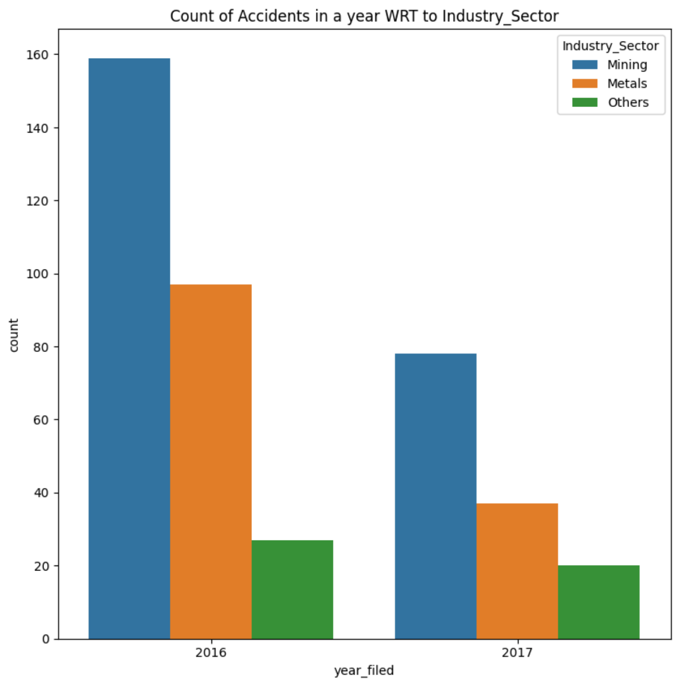
- **Country 01** is the country where most of the accidents happen (more than 50%).



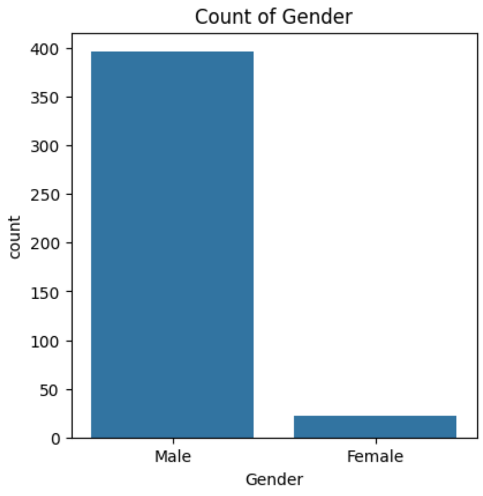
- **Local 03** (which also belongs to Country 01) is where most of the accidents happen.



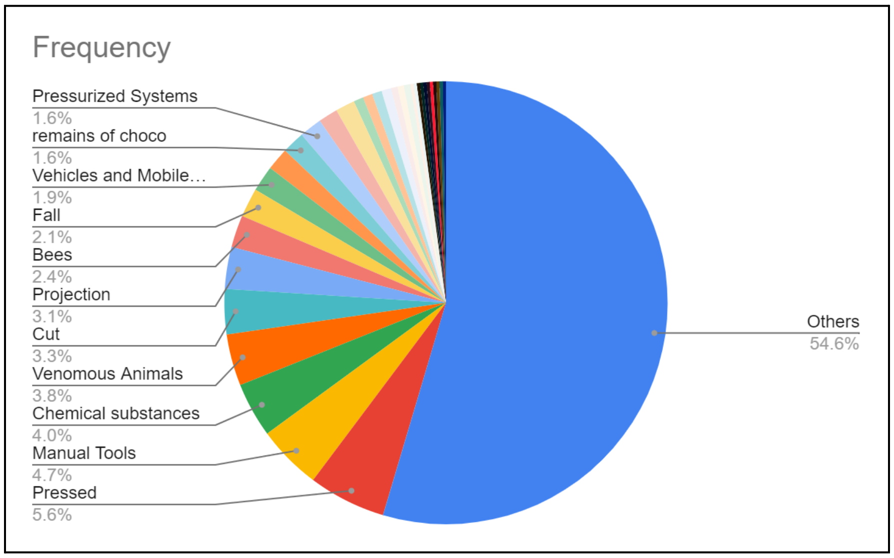
- **Mining** is also the most significant contributor to accidents. There is a decreasing trend in the number of accidents year over year in the mining sector



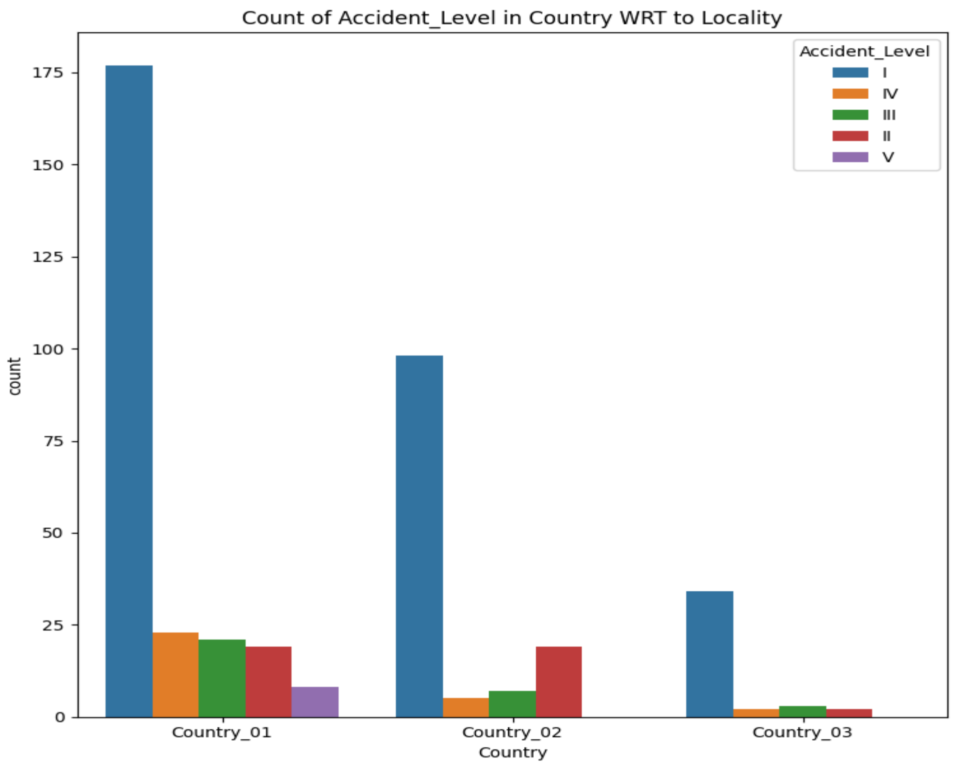
- **Male** (95%) count for the kind of people that suffer more accidents. It is possible that because more males are working in hazardous jobs, more men are suffering from accidents.



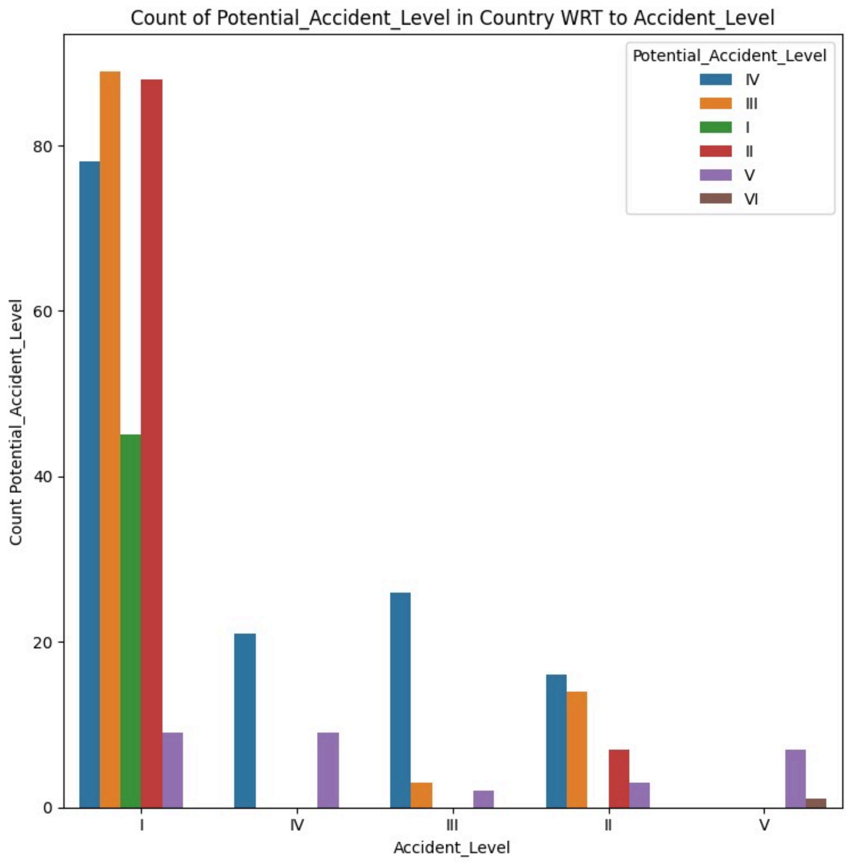
- **Critical Risk** : Pressed, Manual Tools, Chemical Substances, Venomous Animals and Cut are the top 5 risks. For more than 50% risks category is mentioned as Others (as good as not categorized).Which shows inaccuracy at the time of data entry.



- **Accident Level 1** is the highest occurring accident level. This means that there are possibly some safety measures that are implemented in the systems that help contain the severity of the accident level.

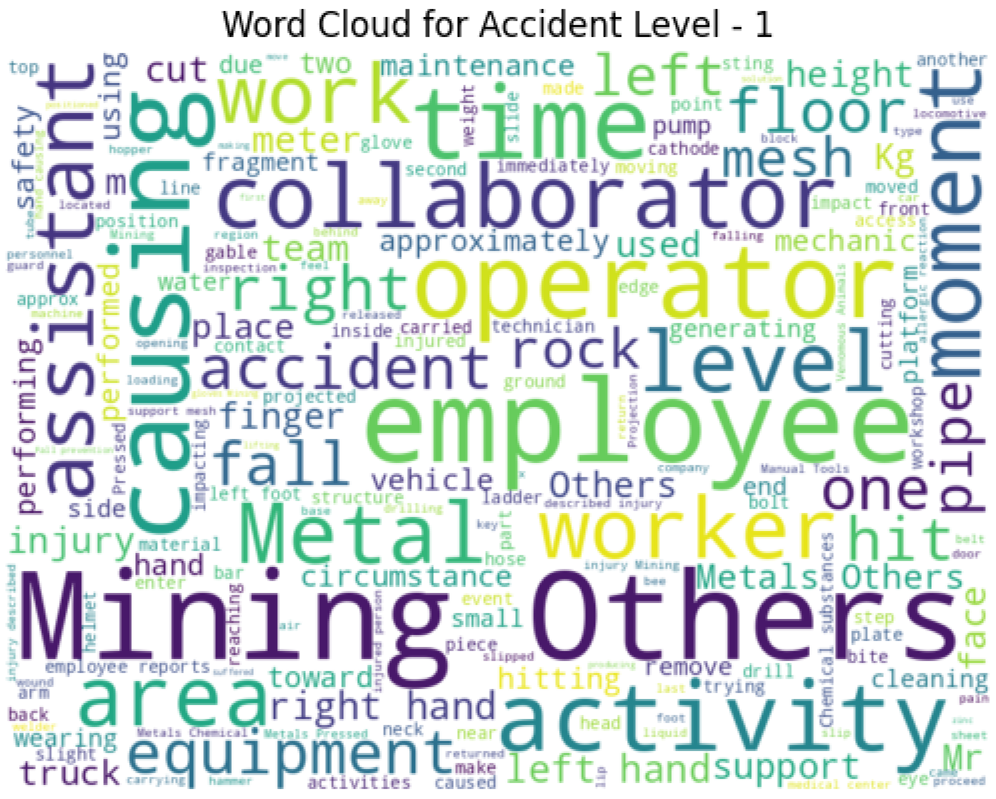


- Within Accident Level 1 we can see that the safety measures have helped in restricting the Potential Accident level III and IV. This shows that in most cases current safely measures have reduced the severity of the accidents.

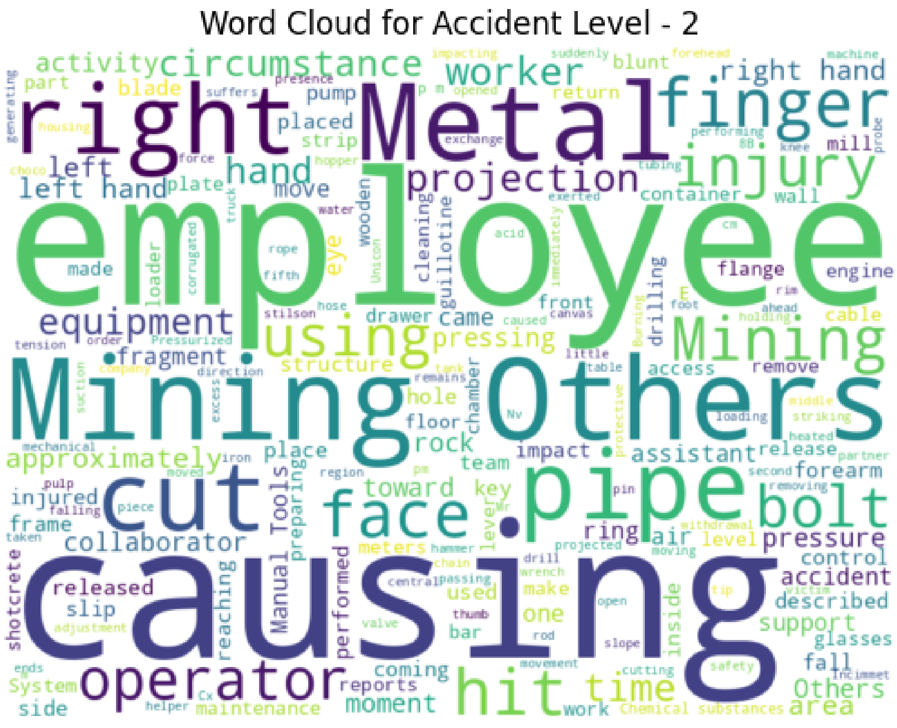


- Word Cloud per accident Level. The word cloud analysis for the accident levels will tell us what keywords are there for each accident level.

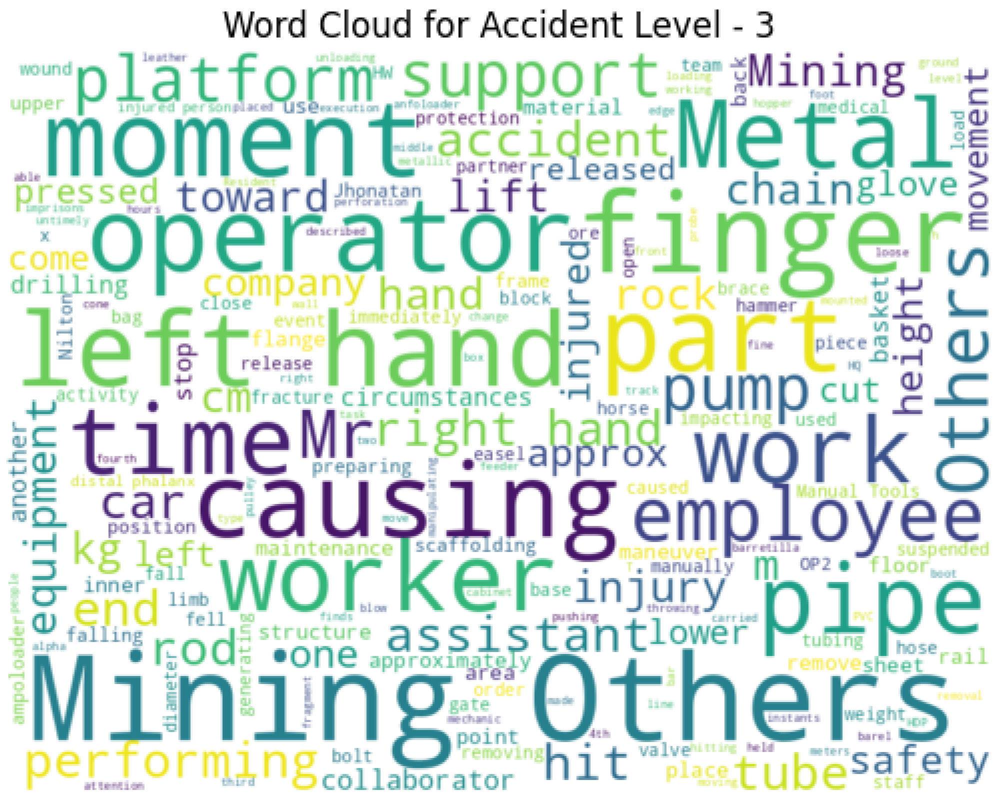
Accident Level I : Employees, Operators are the key words which do not tell us much about the type of injury but there are words like fall, rock that are also visible in the word cloud, indicating some causes.



Accident Level II : Most mentioned words are right , metal, finger, cut and Mining.

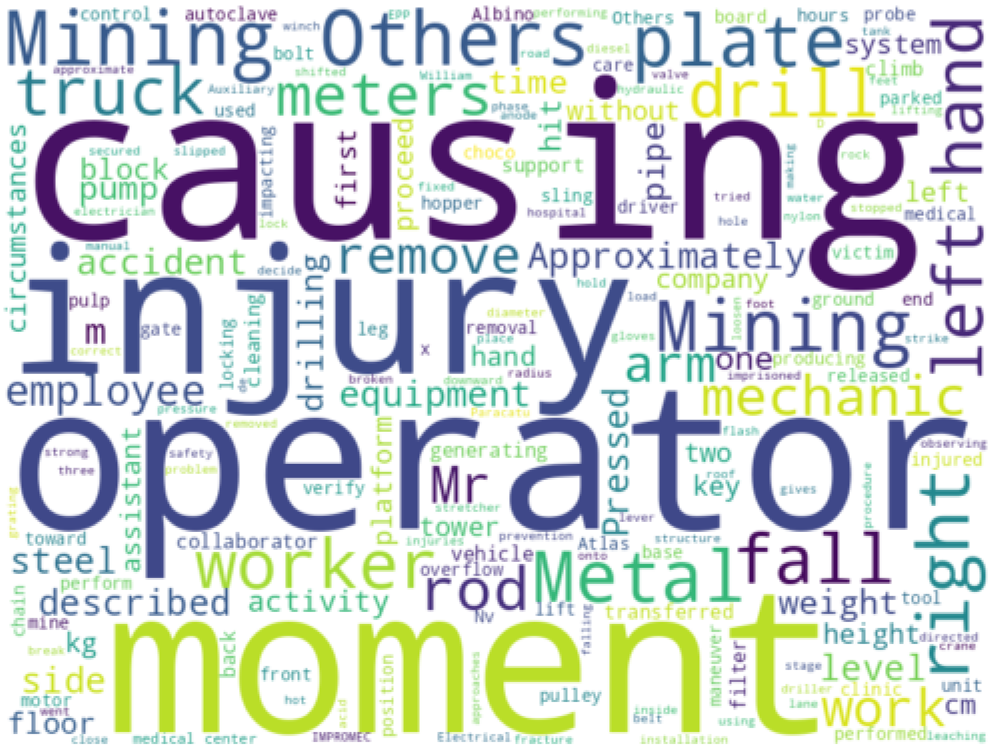


Accident Level III: It seems that the most prevelent impacted areas are finger in the level III.



Accident Level IV : Left hand seems to be mentioned very often in level 4

Word Cloud for Accident Level - 4



Overall, there seems to be **finger/hand** injuries that are most prevalent in the system. There can be some safety actions that can be taken at the finger / hand level.

Data Pre-Processing

For using data to train our chatbot, we need to have data that can be formatted to create a good model. For this, the first step is to import the data. After importing required libraries and data, we started checking the profile of the data and took actions accordingly.

Pre-processing steps included

- five point summary,
- detecting null values,
- removal of unnecessary columns,
- data types,
- unique values for each column,
- changing a few variables name,
- feature engineering,
- stopwords removal,
- stemming and tokenization.

ML Models and Neural Network

Leveraging word embeddings and ML techniques enhances the capability of chatbots to understand and respond to user queries effectively, making the approach ideal for designing intelligent conversational agents across different domains and applications.

Following is a summary of the tools deployed by us:

Word Embeddings:

- **CountVectorizer**: Converts text into a matrix of token counts, capturing frequency information of words in documents.
- **FastText**: Provides word embeddings that encode semantic meanings and context by considering sub-word information, useful for capturing nuances in language.
- **TF-IDF**: Weighs the importance of words in documents based on their frequency and rarity across the corpus, enhancing the relevance of features for machine learning models.
- **Word2Vec:** A neural network-based model that learns word embeddings by predicting context words from a target word (or vice versa). It uses either the Continuous Bag of Words (CBOW) or Skip-gram approach, capturing semantic relationships between words based on their co-occurrence in a corpus.
- **GloVe (Global Vectors for Word Representation):** A model that creates word embeddings by aggregating global word-word co-occurrence statistics from a corpus. It focuses on the ratios of co-occurrence probabilities, producing vectors that capture both local and global statistical information, resulting in embeddings that reflect semantic relationships between words.

Machine Learning Techniques:

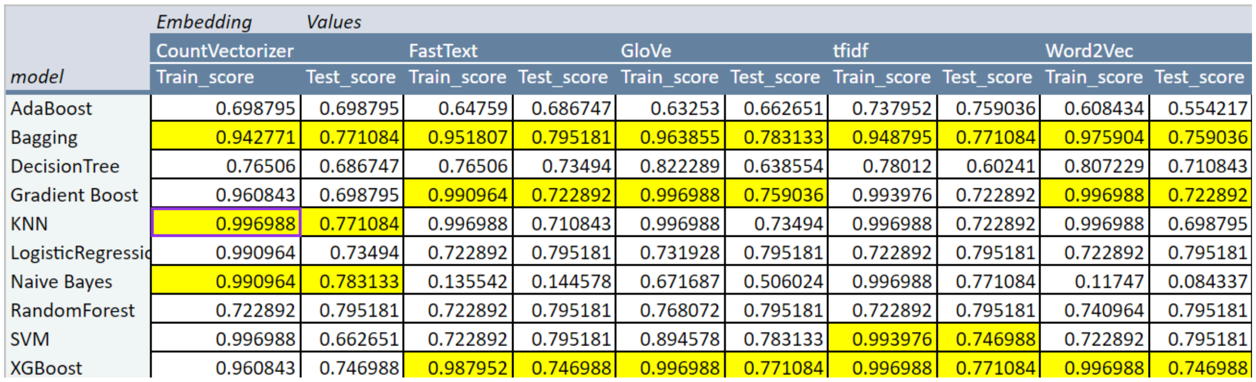
- **Logistic Regression, Naive Bayes**: Simple yet effective for text classification tasks, leveraging probabilistic approaches and linear decision boundaries.
- **K-Nearest Neighbors (KNN)**: Uses similarity metrics to classify new instances based on the nearest neighbors in the training data.
- **Support Vector Machines (SVM)**: Constructs hyperplanes in a high-dimensional space to separate classes, effective in high-dimensional feature spaces.
- **Decision Tree, Random Forest, Bagging, AdaBoost, Gradient Boost, XGBoost**: Ensemble methods that combine multiple models for improved accuracy and robustness, handling non-linear relationships and capturing complex patterns in data.

By deploying the above tools, we will achieve

- **Flexibility**: The choice of word embeddings allows capturing different aspects of text semantics and frequency, catering to diverse chatbot requirements.

- **Performance**: Machine learning techniques provide flexibility in modeling chatbot responses, enabling effective handling of varying user inputs.
- **Scalability**: These methods scale well with large datasets and can handle the complexity of natural language interactions, ensuring robust performance in real-world scenarios.
- **Integration**: By combining word embeddings with a range of ML algorithms, the approach facilitates the creation of chatbots that are versatile, responsive, and capable of learning from interactions to improve user engagement and satisfaction.

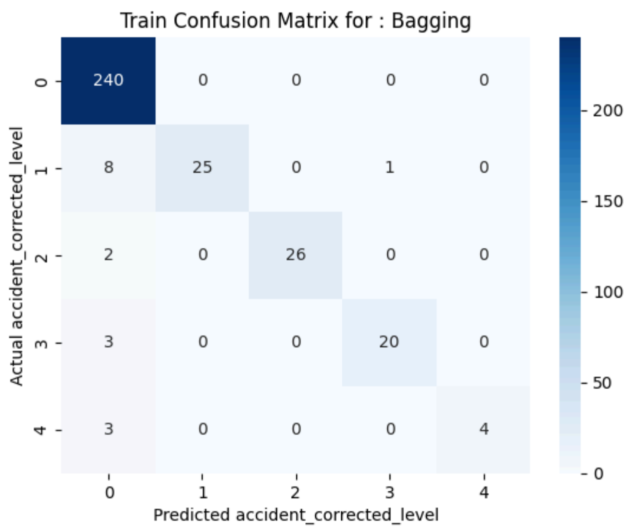
The **Test vs. Train** results for .the above are below:

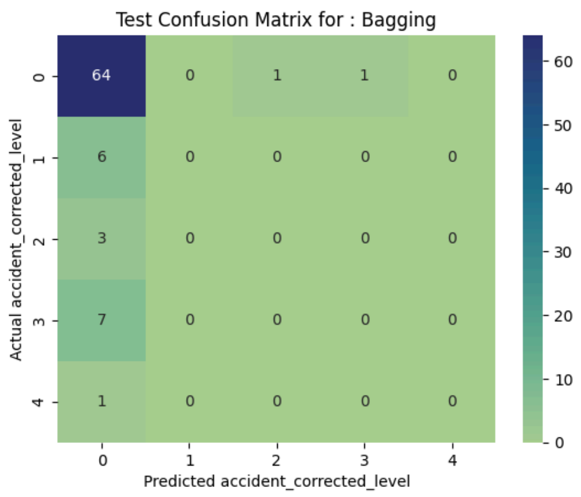


	Embedding	*Values*								
	CountVectorizer		FastText		GloVe		tfidf		Word2Vec	
model	Train_score	Test_score	Train_score	Test_score	Train_score	Test_score	Train_score	Test_score	Train_score	Test_score
AdaBoost	0.698795	0.698795	0.64759	0.686747	0.63253	0.662651	0.737952	0.759036	0.608434	0.554217
Bagging	0.942771	0.771084	0.951807	0.795181	0.963855	0.783133	0.948795	0.771084	0.975904	0.759036
DecisionTree	0.76506	0.686747	0.76506	0.73494	0.822289	0.638554	0.78012	0.60241	0.807229	0.710843
Gradient Boost	0.960843	0.698795	0.990964	0.722892	0.996988	0.759036	0.993976	0.722892	0.996988	0.722892
KNN	0.996988	0.771084	0.996988	0.710843	0.996988	0.73494	0.996988	0.722892	0.996988	0.698795
LogisticRegressic	0.990964	0.73494	0.722892	0.795181	0.731928	0.795181	0.722892	0.795181	0.722892	0.795181
Naive Bayes	0.990964	0.783133	0.135542	0.144578	0.671687	0.506024	0.996988	0.771084	0.11747	0.084337
RandomForest	0.722892	0.795181	0.722892	0.795181	0.768072	0.795181	0.722892	0.795181	0.740964	0.795181
SVM	0.996988	0.662651	0.722892	0.795181	0.894578	0.783133	0.993976	0.746988	0.722892	0.795181
XGBoost	0.960843	0.746988	0.987952	0.746988	0.996988	0.771084	0.996988	0.771084	0.996988	0.746988

Sample Train and Test Confusion matrix for Bagging.

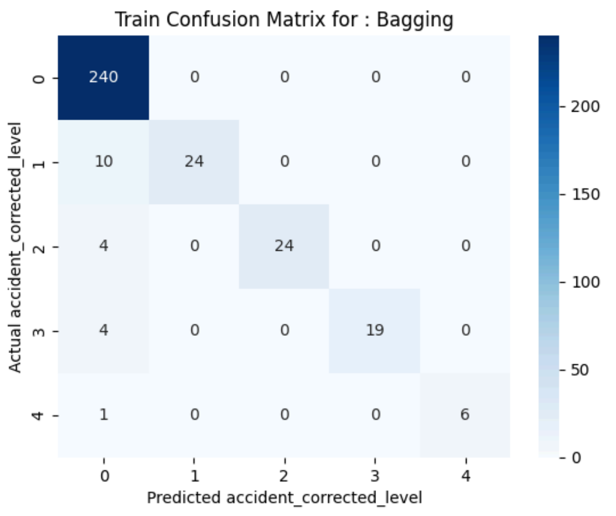
TFIDF

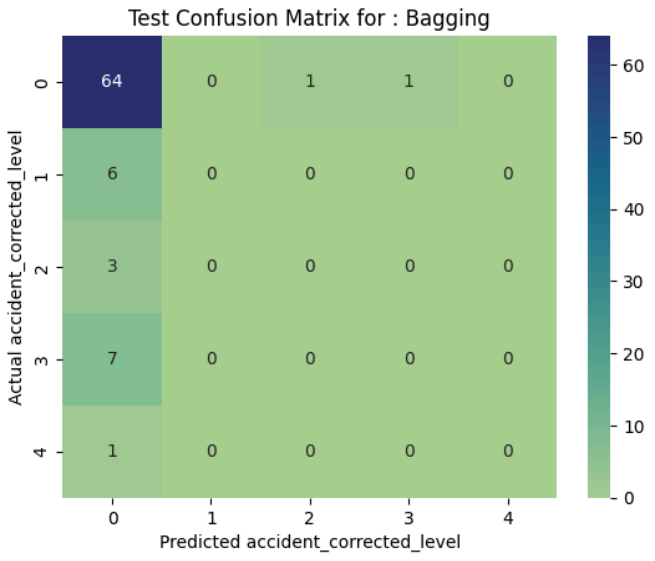




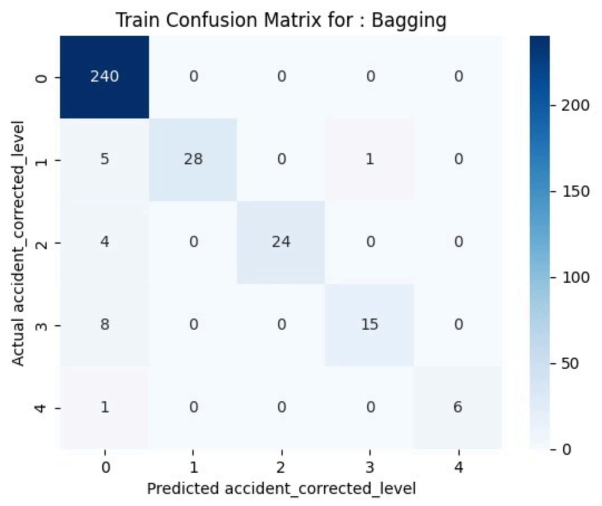
NLP Project 1

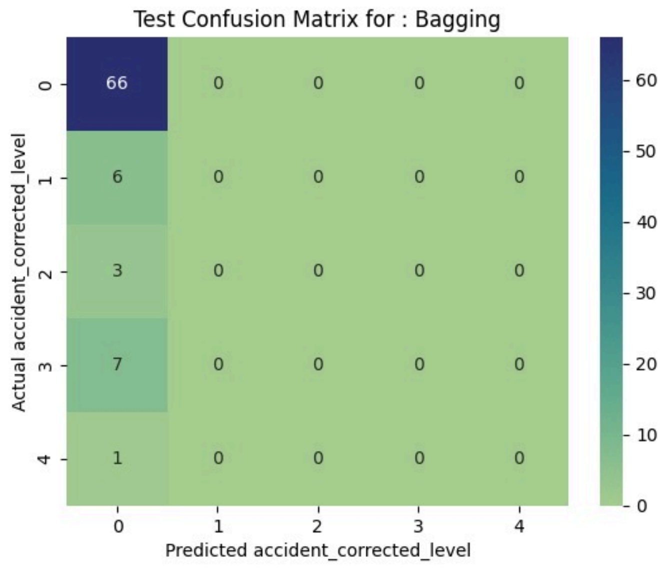
CountVectorizer



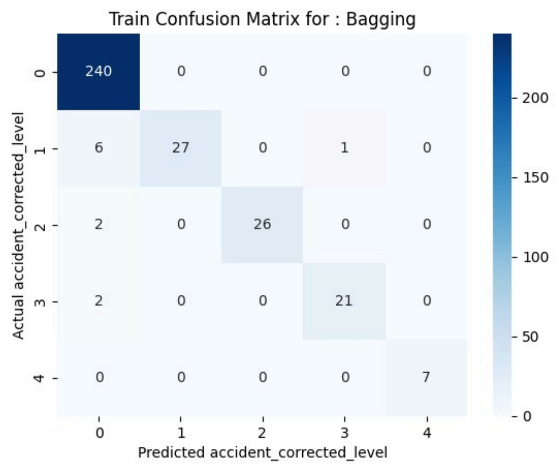


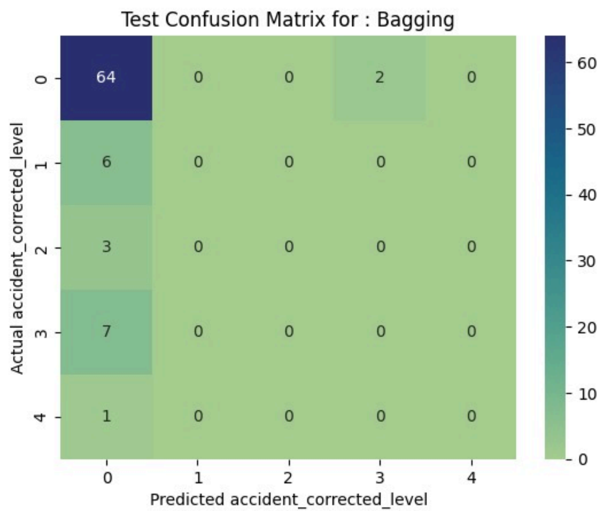
Fasttext



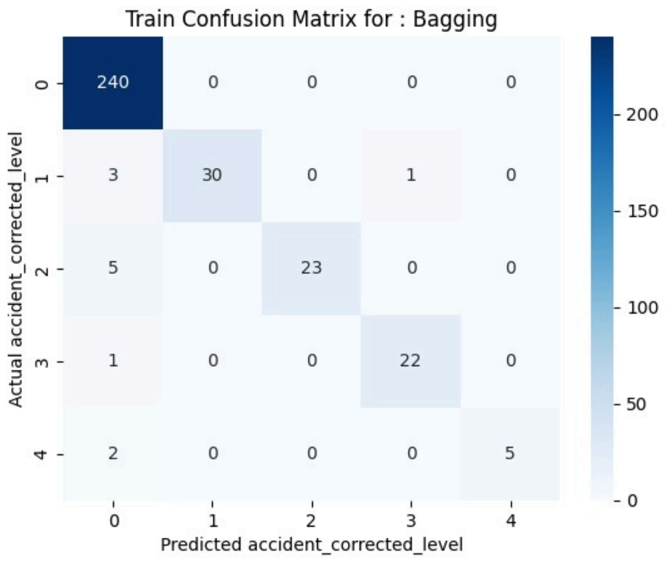


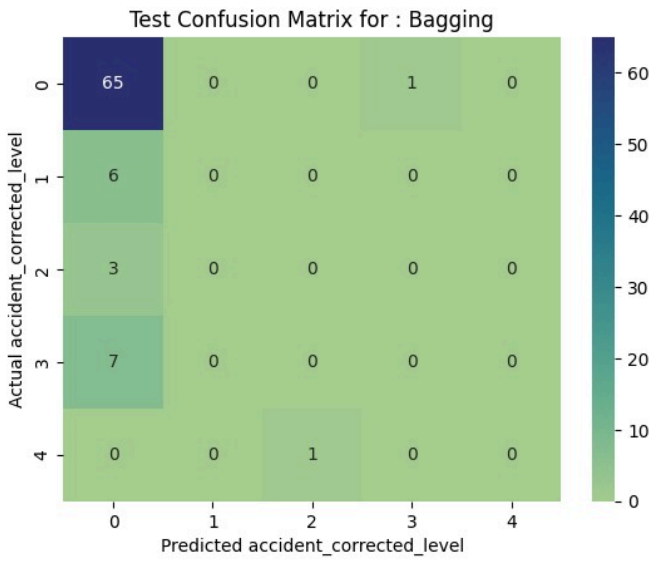
Word2Vec





GloVe





Key Observations

1. Bagging is the best Model across embeddings
2. For** Random forest** and AdaBoost, accuracy on training data is less than testing data (under fitting).
3. Bagging, XG boost and gradient boost have best scores across embeddings
4. For Bagging Model, Fasttext embedding has the best Test Score

Going forward, we will focus on the following steps to optimize and train the models

1. Hyperparameter tuning
2. Reduce data skewness and data balancing using SMOTE
3. Design, train and test Neural networks classifiers
4. Design, train and test RNN or LSTM classifiers
5. Choose the best performing classifier and pickle it

—----------------------------------- MileStone 1 Submission —------------------------------—-----------

NLP Project 1

—---------------------------- Milestone 2 —-----------------------------

Improving the Performance of ML models

As the Data is highly imbalanced, we need to balance the data to remove any biases from the data. Once the biases are removed, we would need to re-run the classification models to identify the best model for the updated balanced dataset.

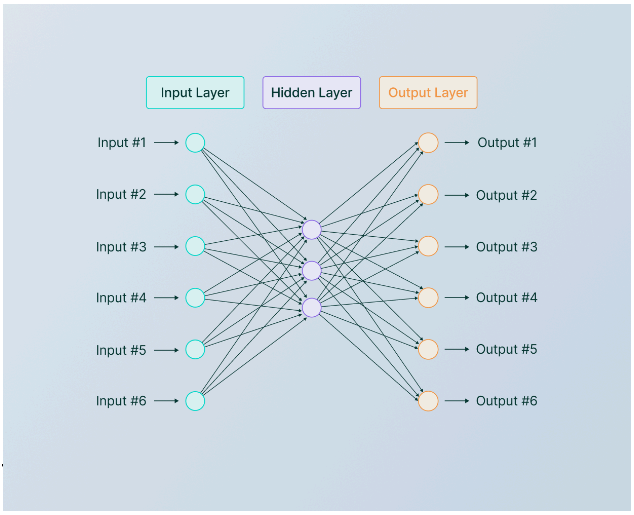
For doing this, we tried using SMOTE and Stratify to balance the dataset.
SMOTE (Synthetic Minority Oversampling Technique): In machine learning, classification is a task that aims to assign a category or label to data points. This task can involve classifying emails as spam or non-spam or medical images as showing disease or not, and it revolves around making predictions (decisions) based on data characteristics.Ideally, in a classification problem, the dataset is balanced. This means we have approximately the same number of data points in each class. Such proportional representation assists algorithms in learning and generalizing more effectively, as they have access to a similar number of examples for each category. This class imbalance can occur for various reasons, such as one class representing rare events.

It's a data augmentation that mitigates the effects of class imbalance by generating synthetic examples for the minority class. Ultimately, this enhances the performance and reliability of machine learning models.

Design, train and test NN, RNN or LSTM classifiers

For creating a better chatbot, we need to create a Deep learning model. The Machine learning models will not be the best models to create a Chatbot. Reinforcement learning is required to create a better chatbot. Therefore we will use neural networks to create a reinforcement model for our chatbot.

NN (Neural Networks):

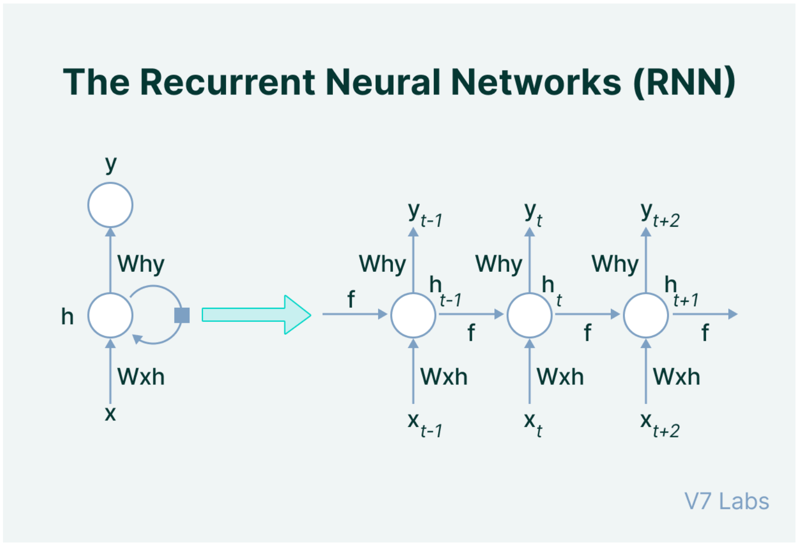


Neural Networks are the functional unit of Deep Learning and are known to mimic the behavior of the human brain to solve complex data-driven problems.

The input data is processed through different layers of artificial neurons stacked together to produce the desired output.

The Neural Network architecture is made of individual units called neurons that mimic the biological behavior of the brain.
From speech recognition and person recognition to healthcare and marketing, Neural Networks have been used in a varied set of domains.

RNN (Recurrent Neural Networks) :



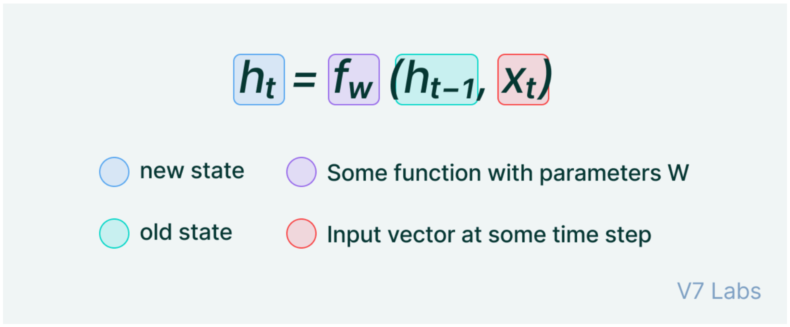
The basic deep learning architecture has a fixed input size, and this acts as a blocker in scenarios where the input size is not fixed. Also, the decisions made by the model were based on the current input with no memory of the past.

Recurrent Neural Networks work very well with sequences of data as input. Its functionality can be seen in solving NLP problems like sentiment analysis, spam filters, time series problems like sales forecasting, stock market prediction, etc.

The input is in the form of sequential data that is fed into the RNN, which has a hidden internal state that gets updated every time it reads the following sequence of data in the input.

The internal hidden state will be fed back to the model. The RNN produces some output at every timestamp.

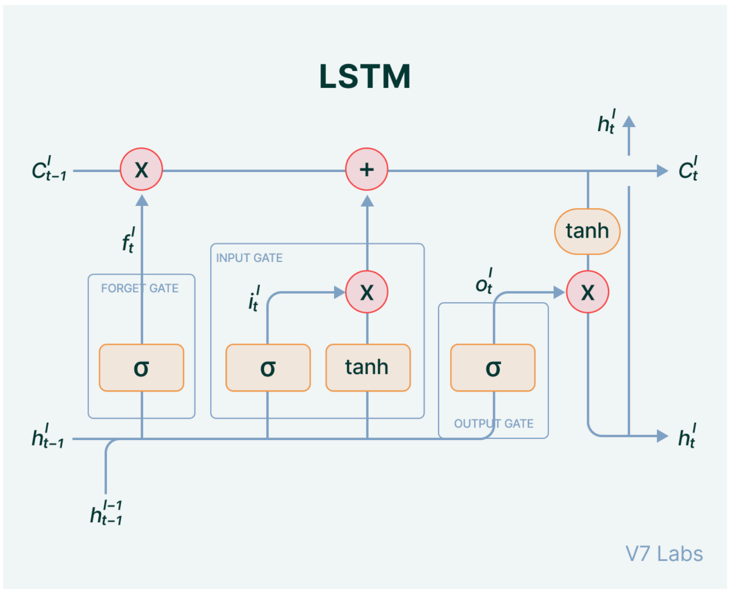
The mathematical representation is given below:



$$h_t = f_W(h_{t-1}, x_t)$$

new state
Some function with parameters W
old state
Input vector at some time step
V7 Labs

LSTM (Long Short Term Memory) :



In RNN each of our predictions looked only one timestamp back, and it has a very short-term memory. It doesn't use any information from further back.

To rectify this, we can take our Recurrent Neural Networks structure and expand it by adding some more pieces to it.

The critical part that we add to this Recurrent Neural Networks is memory. We want it to be able to remember what happened many timestamps ago. To achieve this, we need to add extra structures called gates to the artificial neural network structure.

HyperParameter Tuning on Balanced Data:

In addition to the neural network models, we tuned the hyperparameters for the ML models again with the same embeddings to train and test the classifiers on the balanced dataset. This is important to understand if there is any improvement or depreciation in the accuracies of the model-embedding pairs. Tweaking the hyperparameters gives us an understanding of which set of parameter values is the best fit for our model.

Please see below the accuracy scores for the embeddings:

	Embedding																			
	CountVectorizer				FastText				GloVe				tfidf				Word2Vec			
Model	train_score	test_score	train_before Smote	test_before Smote	train_score	test_score	train_before Smote	test_before Smote	train_score	test_score	train_before Smote	test_before Smote	train_score	test_score	train_before Smote	test_before Smote	train_score	test_score	train_before Smote	test_before Smote
AdaBoost	0.37	0.39	0.70	0.70	0.48	0.31	0.65	0.69	0.63	0.66	0.63	0.66	0.43	0.19	0.74	0.76	0.49	0.27	0.63	0.64
Bagging	0.99	0.53	0.95	0.77	1.00	0.43	0.95	0.78	0.97	0.78	0.96	0.78	1.00	0.72	0.96	0.76	1.00	0.63	0.97	0.78
DecisionTree	0.59	0.22	0.77	0.69	0.70	0.28	0.77	0.73	0.82	0.64	0.82	0.64	0.81	0.43	0.78	0.60	0.86	0.45	0.79	0.71
Gradient Boost	0.94	0.54	0.95	0.70	0.93	0.41	0.99	0.73	1.00	0.76	1.00	0.72	1.00	0.73	1.00	0.73	1.00	0.64	1.00	0.73
KNN	0.99	0.06	1.00	0.77	1.00	0.18	1.00	0.71	1.00	0.73	1.00	0.73	1.00	0.07	1.00	0.72	1.00	0.04	1.00	0.71
LogisticRegressi	0.96	0.60	0.99	0.73	0.28	0.10	0.72	0.80	0.73	0.80	0.73	0.80	1.00	0.69	0.72	0.80	0.30	0.08	0.72	0.80
LSTM	0.20	0.08	0.00	0.00	0.20	0.04	0.00	0.00	0.26	0.12	0.00	0.00	0.23	0.07	0.00	0.00	0.20	0.80	0.00	0.00
Naive Bayes	0.81	0.78	0.99	0.78	0.33	0.17	0.14	0.14	0.67	0.51	0.67	0.51	1.00	0.77	1.00	0.77	0.32	0.06	0.11	0.08
Neural Network	0.99	0.63	0.00	0.00	0.20	0.07	0.00	0.00	1.00	0.75	0.00	0.00	1.00	0.69	0.00	0.00	0.32	0.05	0.00	0.00
RandomForest	0.83	0.61	0.72	0.80	0.63	0.23	0.73	0.80	0.76	0.80	0.76	0.80	0.93	0.59	0.72	0.80	0.96	0.45	0.74	0.80
RNN	0.15	0.07	0.00	0.00	0.20	0.06	0.00	0.00	0.58	0.17	0.00	0.00	0.30	0.08	0.00	0.00	0.18	0.04	0.00	0.00
SVM	0.98	0.57	1.00	0.66	0.29	0.04	0.72	0.80	0.89	0.78	0.89	0.78	1.00	0.75	0.99	0.75	0.28	0.07	0.72	0.80
XGBoost	0.96	0.58	0.96	0.75	0.98	0.41	0.99	0.75	1.00	0.77	1.00	0.77	1.00	0.72	1.00	0.77	1.00	0.61	1.00	0.75

CountVectorizer :

1) Train performance improved on SOMTE in Bagging,Random Forest
2) Test performance scores decreased on SMOTE across all embeddings

Fast Text:

1) Test performance score improved in SMOTE for Naïve Bayes
2) Train performance score increased onSMOTE for Bagging, Naïve Bayees
3) All scores remain same for Glove

tfidf:

1) Training scores increased for Bagging, Gradient Boost, KNN, NaiveBayees, SVM
2) Testing Score remain same for GradientBoost, SVM

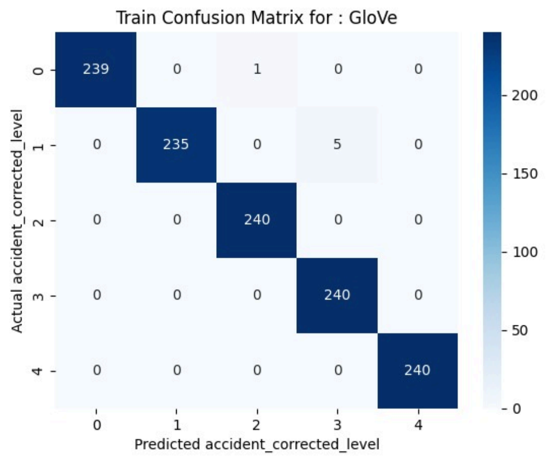
Word2vec:

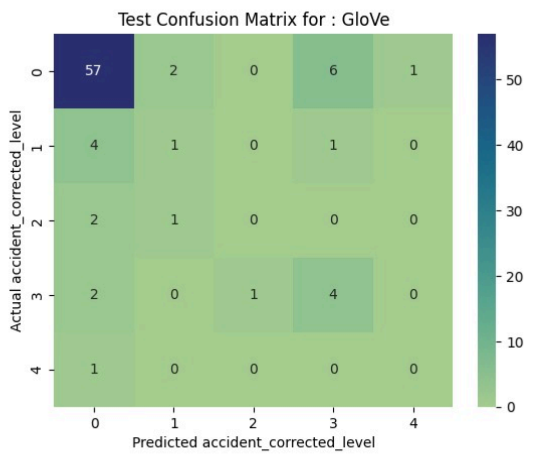
1) Training scores increased for Bagging, Decision Tree, Naïve Bayes, Random Forest, XGBoost after using SMOTE
2) All testing scores have fallen after SMOTE

Overall, after using SMOTE for balancing the dataset, the combination of Naive Bayes | CountVectorizer, Naive Bayes | FastText, Gradient Boost | tfidf, Naive Bayes | tfidf, SVM | tfidf, bettered on SMOTE

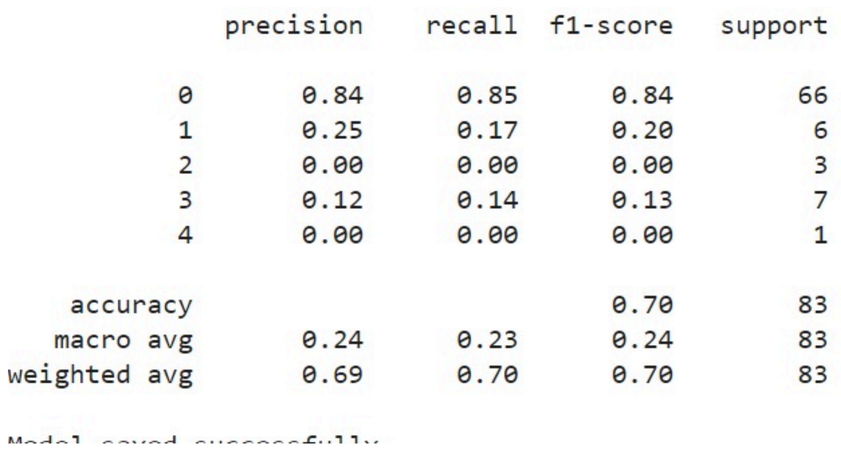
There are multiple models and embeddings that are having good results, but we would like to go for "*Neural Network* Classifier using *GloVe embedding*" as for most NLP tasks, neural networks, especially those based on transformer architectures, are superior to bagging algorithms. They offer better performance, scalability, and the ability to capture complex patterns and context within text data. The advancements in pre-trained models like BERT and GPT have significantly pushed the boundaries of what is achievable in NLP, making neural networks the go-to choice for modern NLP applications.
Therefore although we have better models like Bagging, we will go ahead and use Neural networks.

Confusion Matrix for *Neural Networks Model with Glove Embeddings:*





Classification Report for Neural Network using Glove embedding:



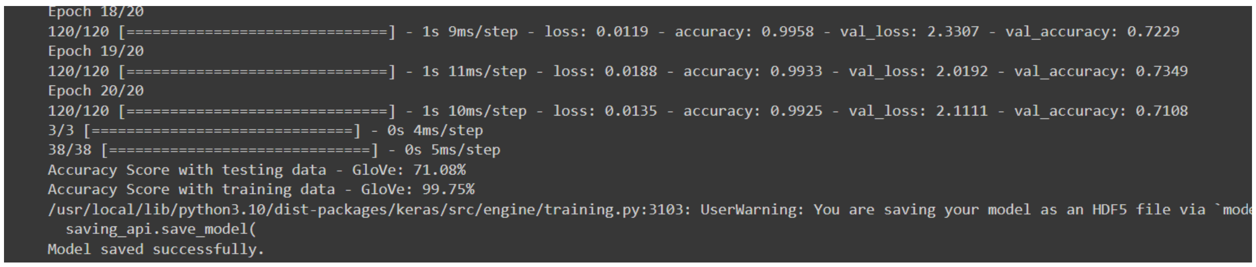
```
              precision    recall  f1-score   support

           0       0.84      0.85      0.84        66
           1       0.25      0.17      0.20         6
           2       0.00      0.00      0.00         3
           3       0.12      0.14      0.13         7
           4       0.00      0.00      0.00         1

    accuracy                           0.70        83
   macro avg       0.24      0.23      0.24        83
weighted avg       0.69      0.70      0.70        83

Model saved successfully.
```

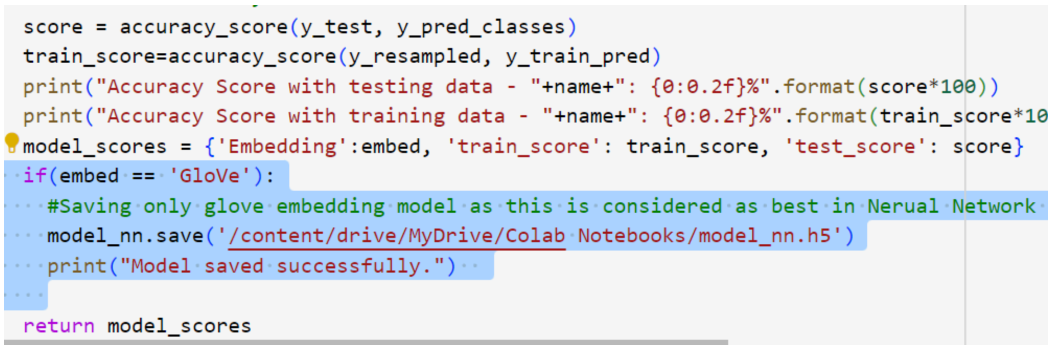
<u>Neural network - Glove Epoch.</u>



```
Epoch 18/20
120/120 [==============================] - 1s 9ms/step - loss: 0.0119 - accuracy: 0.9958 - val_loss: 2.3307 - val_accuracy: 0.7229
Epoch 19/20
120/120 [==============================] - 1s 11ms/step - loss: 0.0188 - accuracy: 0.9933 - val_loss: 2.0192 - val_accuracy: 0.7349
Epoch 20/20
120/120 [==============================] - 1s 10ms/step - loss: 0.0135 - accuracy: 0.9925 - val_loss: 2.1111 - val_accuracy: 0.7108
3/3 [==============================] - 0s 4ms/step
38/38 [==============================] - 0s 5ms/step
Accuracy Score with testing data - GloVe: 71.08%
Accuracy Score with training data - GloVe: 99.75%
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `mode
  saving_api.save_model(
Model saved successfully.
```

Choose the best performing classifier and pickle it

Once we have finalized the model, we need to pickle it to use the trained model in other applications. We will use this pickled file for creating a basic chatbot.



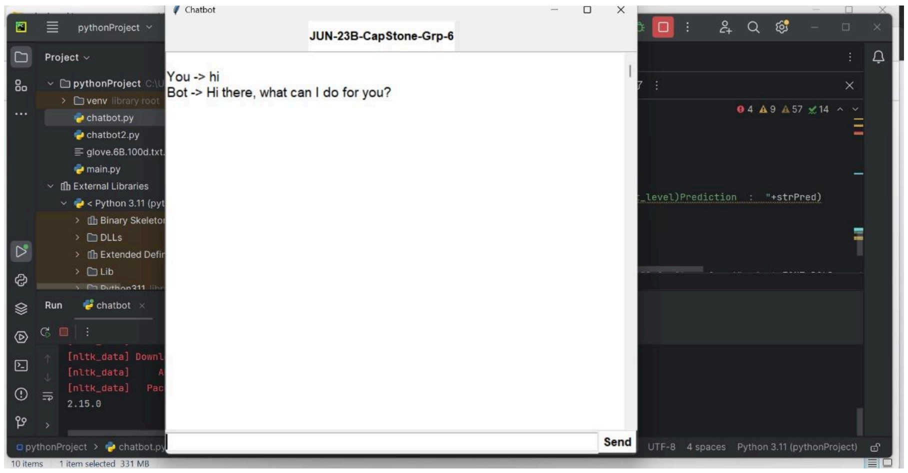
```
score = accuracy_score(y_test, y_pred_classes)
train_score=accuracy_score(y_resampled, y_train_pred)
print("Accuracy Score with testing data - "+name+": {0:0.2f}%".format(score*100))
print("Accuracy Score with training data - "+name+": {0:0.2f}%".format(train_score*10
model_scores = {'Embedding':embed, 'train_score': train_score, 'test_score': score}
if(embed == 'GloVe'):
  #Saving only glove embedding model as this is considered as best in Nerual Network
  model_nn.save('/content/drive/MyDrive/Colab Notebooks/model_nn.h5')
  print("Model saved successfully.")

return model_scores
```

Design a clickable UI based chatbot interface

For creating a sample chatbot to use our trained model, we used pycharm and the tkinter — Python interface. The chatbot is a very basic chatbot that will tell us what is the prediction if we use words / sentences that are used in the dataset.

We used the pickle file as the input model for the same. Following are some screenshots of the chat interface

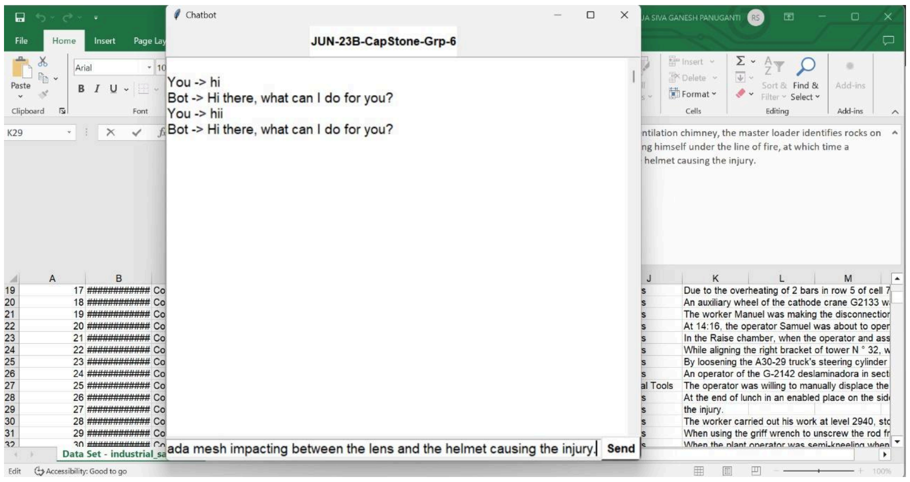


```
You -> hi
Bot -> Hi there, what can I do for you?
```

```
[nltk_data] Downl
[nltk_data]      A
[nltk_data]    Pac
2.15.0
```

Using the pickled model and the Tkinter library in pycharm, we ran the chatbot, we start interacting with the chatbot.

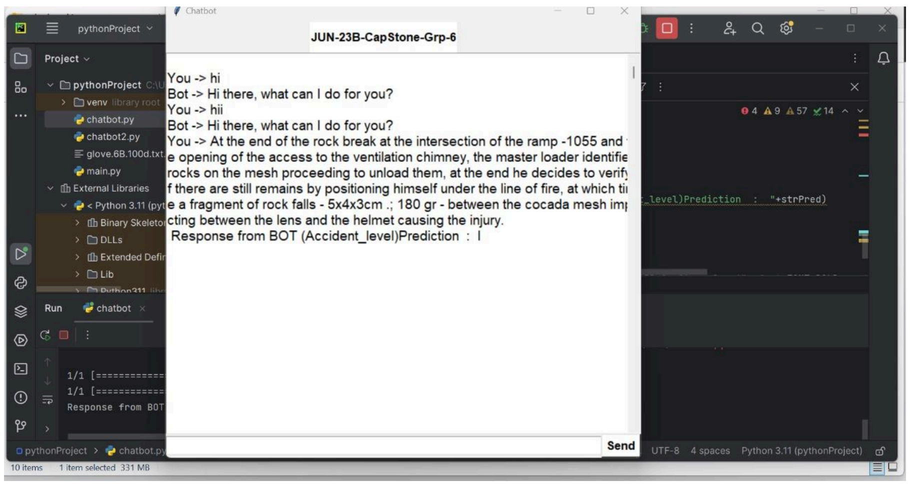
Chatbot is responding, which is good.


JUN-23B-CapStone-Grp-6

```
You -> hi
Bot -> Hi there, what can I do for you?
You -> hii
Bot -> Hi there, what can I do for you?
```

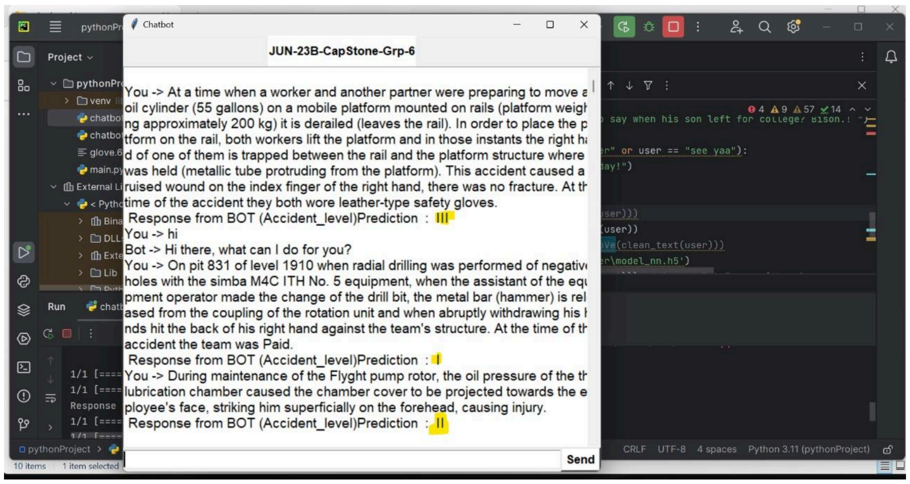
```
ada mesh impacting between the lens and the helmet causing the injury.
```

We then copy one accident information from the data, to see if the chatbot understands which category does this accident belong to.


JUN-23B-CapStone-Grp-6

```
You -> hi
Bot -> Hi there, what can I do for you?
You -> hii
Bot -> Hi there, what can I do for you?
You -> At the end of the rock break at the intersection of the ramp -1055 and
e opening of the access to the ventilation chimney, the master loader identifie
rocks on the mesh proceeding to unload them, at the end he decides to verify
f there are still remains by positioning himself under the line of fire, at which ti
e a fragment of rock falls - 5x4x3cm .; 180 gr - between the cocada mesh im
cting between the lens and the helmet causing the injury.
 Response from BOT (Accident_level)Prediction  :  I
```

The chatbot accident level prediction for this is Level I. This is a correct prediction and it looks good that our chatbot has some good level of accuracy in predicting. We tried entering some key words, that also works.



```
You -> At a time when a worker and another partner were preparing to move a
oil cylinder (55 gallons) on a mobile platform mounted on rails (platform weigh
ng approximately 200 kg) it is derailed (leaves the rail). In order to place the p
tform on the rail, both workers lift the platform and in those instants the right ha
d of one of them is trapped between the rail and the platform structure where
was held (metallic tube protruding from the platform). This accident caused a
ruised wound on the index finger of the right hand, there was no fracture. At th
time of the accident they both wore leather-type safety gloves.
 Response from BOT (Accident_level)Prediction  :  III
You -> hi
Bot -> Hi there, what can I do for you?
You -> On pit 831 of level 1910 when radial drilling was performed of negativ
holes with the simba M4C ITH No. 5 equipment, when the assistant of the equ
pment operator made the change of the drill bit, the metal bar (hammer) is rel
ased from the coupling of the rotation unit and when abruptly withdrawing his h
nds hit the back of his right hand against the team's structure. At the time of th
accident the team was Paid.
 Response from BOT (Accident_level)Prediction  : I
You -> During maintenance of the Flyght pump rotor, the oil pressure of the th
lubrication chamber caused the chamber cover to be projected towards the e
ployee's face, striking him superficially on the forehead, causing injury.
 Response from BOT (Accident_level)Prediction  : II
```

We tried some more statements, and we can see that it is able to predict the accident level correctly. This gives us a good confidence that we will be able to use this chatbot to predict the accident level.

Limitations:

1. The chatbot that we have created is a basic chatbot that will only predict the category of the accident based on the description
2. The dataset is very limited and therefore the model is not expandable to other applications

Key Observations

1. After SMOTE balancing, model performance degraded across most models.
2. Bagging model is showing good performance across embeddings
3. The score for the glove embedding is the same for almost all models before and after SMOTE balancing. This is unusual and not sure if bias does not affect glove models.
4. Other models' accuracy has substantially reduced after balancing
5. Glove - NN is the model that we should go ahead with

Next Steps

1. We can use LLM models to fine-tune our chatbot that will make it more responsive in understanding the language.
2. We can gather more data around accidents in other countries to make our model more robust

References

1. Tkinter: https://docs.python.org/3/library/tkinter.html
2. SMOTE: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
   https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/
3. RNN: https://www.tensorflow.org/guide/keras/working_with_rnns
4. LSTM: https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
5. NN : https://scikit-learn.org/stable/modules/neural_networks_supervised.html
6. Why NN.:
   https://www.techtarget.com/searchenterpriseai/answer/Machine-learning-vs-neural-networks-Whats-the-difference#:~:text=Key%20differences%20between%20machine%20learning%20and%20neural%20networks&text=ML%20algorithms%20make%20decisions%20based,from%20experience%20and%20previous%20errors.