UE20CS312: Data Analytics Project Phase Section G

Project Title : Intent Recognition

Project Team No: 1

Member Names : Shreya S Adiga, *PES1UG20CS401*

Shubangi Saxena, *PES1UG20CS418*

Srushti S Hampannavar, *PES1UG20CS437*

Advisor : Prof. K S Srinivas

Abstract & Scope

What is the problem we are solving?

- Intent Recognition (IR) is considered a key area in Natural Language Processing (NLP). It has crucial usage in various applications.
- Voice user interfaces are becoming an essential part of everyday life. For these systems to carry out effective dialogue, they must be able to determine the intent behind a user's spoken utterance.

What is the scope of the project?

- We have used BERT as our baseline model.
- For comparison purposes, we have applied primarily used Machine Learning techniques, namely Naive Bayes, Random Forest and SVM as well as Deep Learning Technique - Gated Recurrent Unit on the same dataset and evaluated the accuracy.

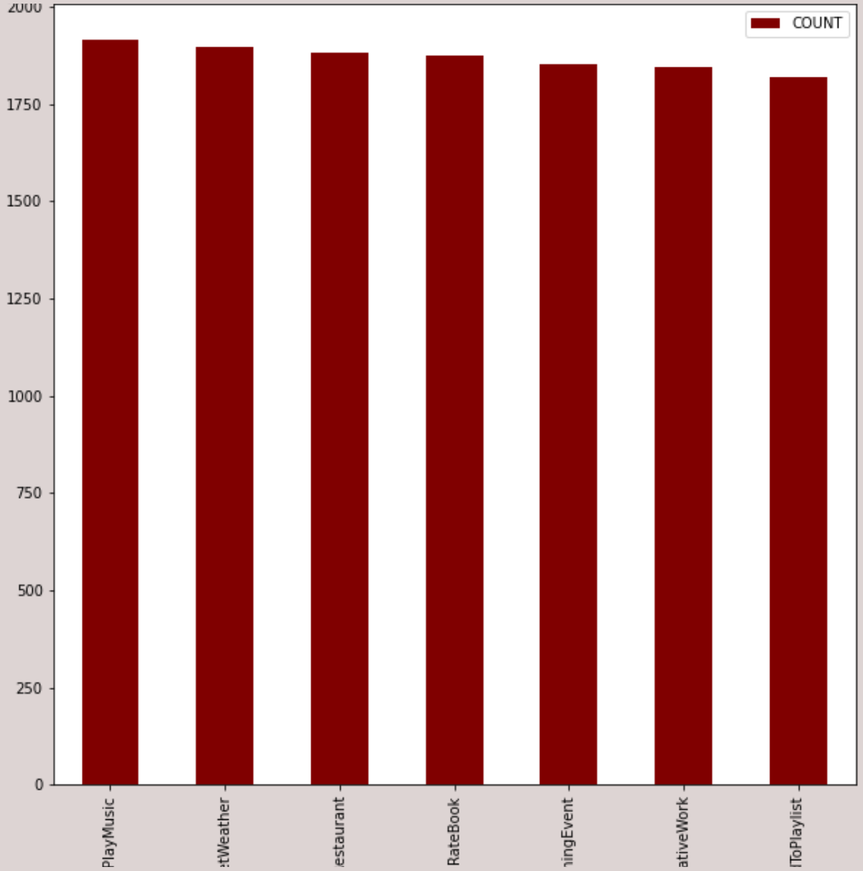
Abstract & Scope

Dataset we have used:

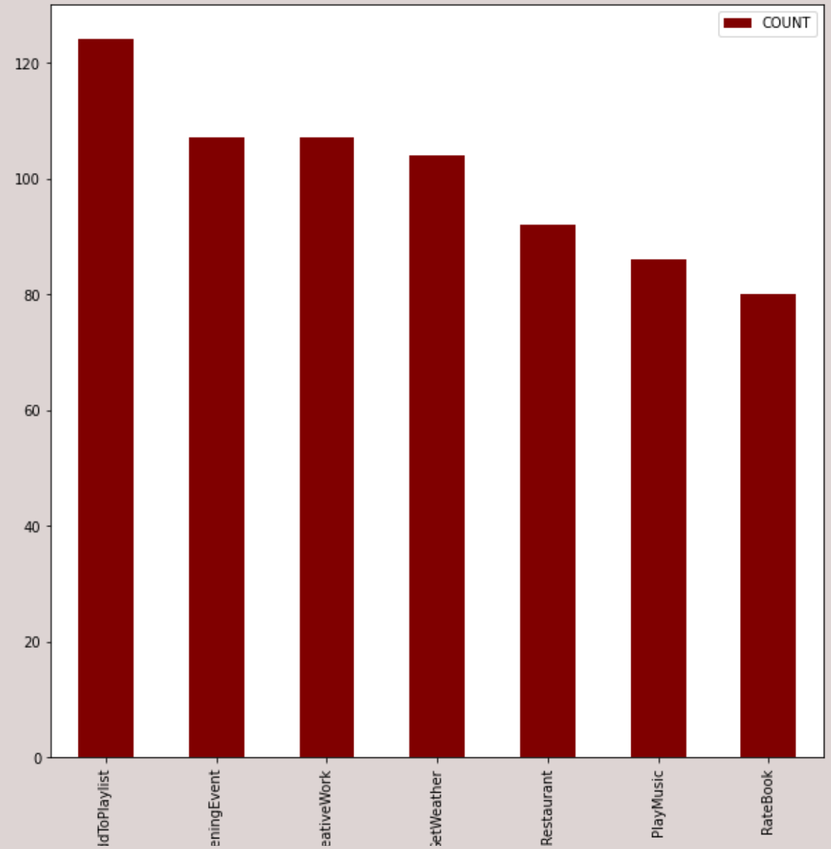
- For our intent recognition model, we're using the **Snips** dataset, which was collected through crowdsourcing for the Snips personal voice assistant. There are **7** unique intent classes for the training set, on a variety of topics :
  1. SearchCreativeWork (For instance: Can you show me some fine artwork)
  2. GetWeather (For instance: It seems Bhopal will have heavy rain today)
  3. BookRestaurant (For instance: I am in a mood to have Chinese food)
  4. PlayMusic (For instance: Let us listen to Bollywood songs)
  5. AddToPlaylist (For instance: This track should be present while you are traveling)
  6. RateBook (For instance: In contrast to the previous one, I like Sydney Sheldon's recent addition very much)
  7. SearchScreeningEvent (For instance: I want you to see the timings of Sharukh Khan's show in Canada next month)

Abstract & Scope

Train Data



Test Data



Module Implementations

Module 1- BERT (Bi-Directional Encoder Representation from Transformers)

Technology Used: BERT, transformers, PyTorch

Code Explanation:

---PREPROCESSING:

- In order to input the intent labels into our model we create a dictionary mapping each intent name to an integer ID.
- Tokenize the sentence.
- Normally our model expects that each text sequence (each training example) will be of the same length (the same number of words/tokens). We can control this using the 'maxlen' parameter.
- In addition to this, messages containing less than three words have been eliminated from the dataset.

Module Implementations

Module 1- BERT (Bi-Directional Encoder Representation from Transformers)

--Hyper-parameters used in our BERT model:

Learning Rate: 1e-5

Optimizer: Adam

Activation Function: Tanh

Batch Size: 16

Loss Optimization: Cross Entropy loss

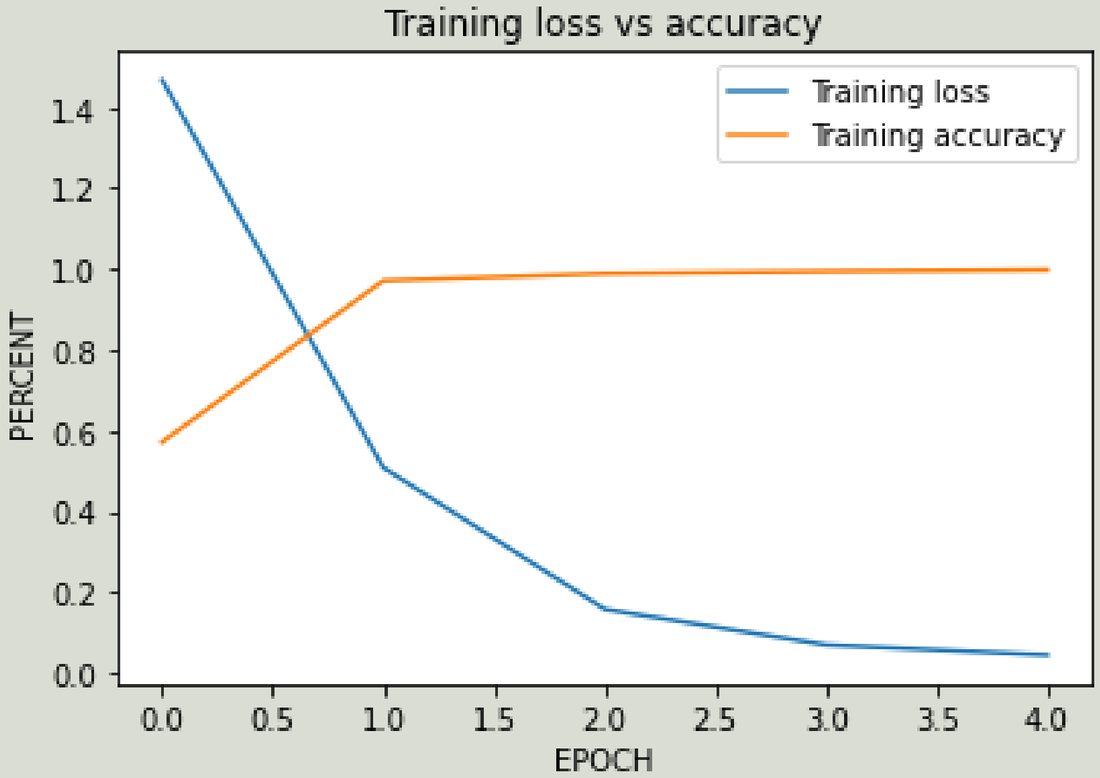
- In order to convert the output vector to a vector of probabilities, the **softmax function** is used. Without softmax, a multi-class classification model will simply return a vector of numbers, with the greatest number in the entry that represents the best intent guess, the second greatest entry the second best intent guess, etc. These raw output values are called **logits**.

Module Implementations

Module 1- BERT (Bi-Directional Encoder Representation from Transformers)

Model Summary

```
Layer (type:depth-idx)                   Param #
==========================================================================
├─BertModel: 1-1                        --
│    └─BertEmbeddings: 2-1               --
│    │    └─Embedding: 3-1               23,440,896
│    │    └─Embedding: 3-2               393,216
│    │    └─Embedding: 3-3               1,536
│    │    └─LayerNorm: 3-4               1,536
│    │    └─Dropout: 3-5                 --
│    └─BertEncoder: 2-2                  --
│    │    └─ModuleList: 3-6              85,054,464
│    └─BertPooler: 2-3                   --
│    │    └─Linear: 3-7                  590,592
│    │    └─Tanh: 3-8                    --
├─Dropout: 1-2                           --
├─Linear: 1-3                            5,383
==========================================================================
Total params: 109,487,623
Trainable params: 109,487,623
Non-trainable params: 0
==========================================================================
```



Received a test accuracy of 95.85%.

Module Implementations

Module 2- BiLSTM(Bidirectional LSTM)

Technology Used: Python, keras, nltk, one-hot encoding

Code Explanation:

---PREPROCESSING:

- Remove stop words using nltk.
- Stemming using LancasterStemmer().
- Tokenizing the sentences and finding the maxlength of the words vector to be 35.
- Padding the rest of the word vectors.
- Using one hot encoding to encode the intents.
- Splitting into train and validation sets. X_train, val_test gets a shape of (vocab size, max_length) and Y_train, val_train gets a shape of (vocab size, #intents).

Module Implementations

Module 2- BiLSTM(Bidirectional LSTM)

--Hyper-parameters used in model:

Optimizer: Adam

Activation Function: softmax

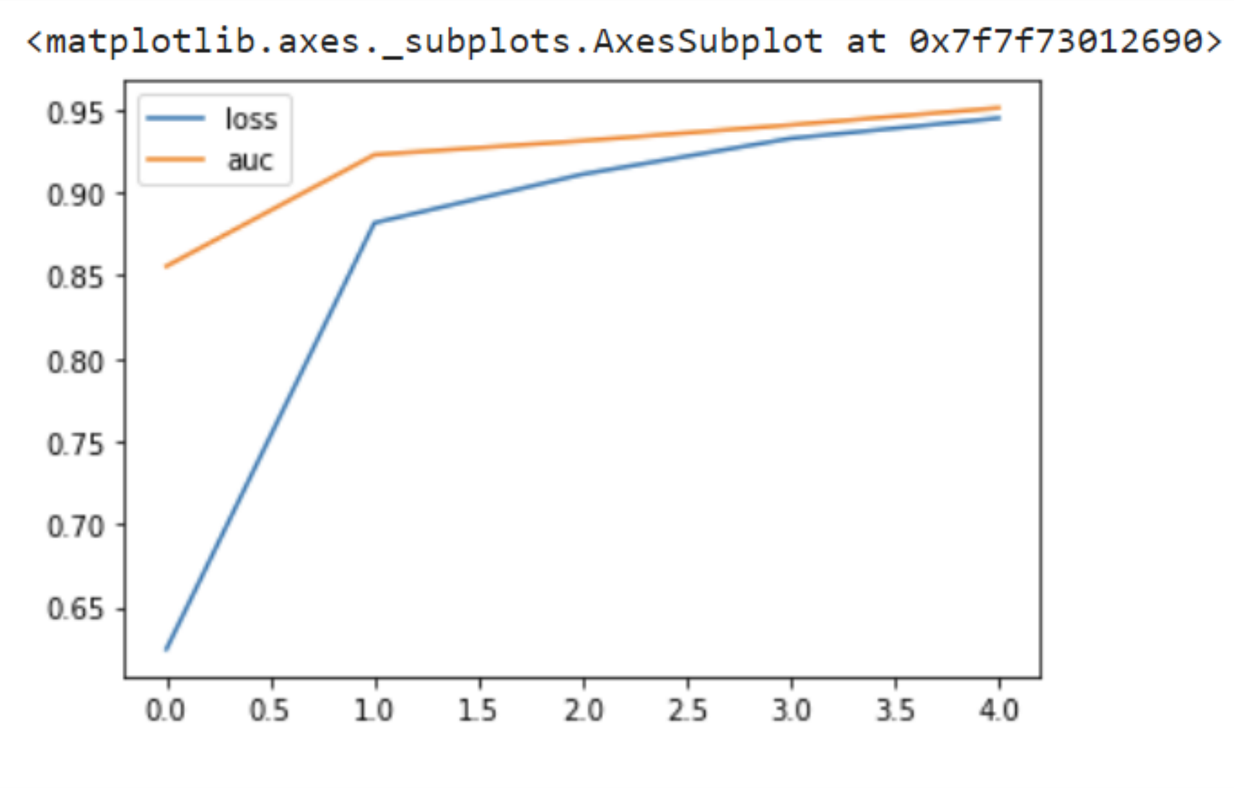
Batch Size: 16

Loss Optimization: Categorical Cross Entropy loss

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_9 (Embedding)     (None, 35, 128)           1415552

 bidirectional_11 (Bidirecti  (None, 35, 256)          263168
 onal)

 bidirectional_12 (Bidirecti  (None, 128)              164352
 onal)

 dense_17 (Dense)            (None, 32)                4128

 dropout_11 (Dropout)        (None, 32)                0

 dense_18 (Dense)            (None, 7)                 231

=================================================================
Total params: 1,847,431
Trainable params: 431,879
Non-trainable params: 1,415,552
_________________________________________________________________
```

Module Implementations

Module 2- BiLSTM(Bidirectional LSTM)



Training Accuracy: 94.49%

Module Implementations

Module 2- BiLSTM(Bidirectional LSTM)

```
text = "please search the young warriors game"
pred = predictions(text)
get_final_output(pred, unique_intent)
```

```
['please', 'search', 'the', 'young', 'warriors', 'game']
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_pr
1/1 [==============================] - 2s 2s/step
[1.6157503e-03 8.1890129e-04 3.3570162e-03 2.7376590e-02 9.6438873e-01
 7.2113017e-04 1.7218018e-03]
SearchCreativeWork has confidence = 0.9643887
SearchScreeningEvent has confidence = 0.02737659
RateBook has confidence = 0.0033570162
PlayMusic has confidence = 0.0017218018
BookRestaurant has confidence = 0.0016157503
GetWeather has confidence = 0.0008189013
AddToPlaylist has confidence = 0.00072113017
```

The predictions() function outputs the probabilities and the get_final_output() gives the confidence that the given sentence belongs to the class.

Testing accuracy: 95.6%

Module Implementations

Module 3,4,5- SVM, Naive Bayes and Random Forest

Technology Used: SVM, Naive Bayes, Random Forest, sklearn

Code Explanation:

---PREPROCESSING:

- Tokenization- done using word_tokenize from nltk.tokenize library
- Stop words are removed and only alphabets are taken into consideration.
- Lemmatization is done using WordNetLemmatizer and pos_tag function is used to tag the words as adjective,verb,adverb or noun
- The final processed set of words is stored in a separate column of the dataframe
- This is then converted to TF-IDF vectors.
- Train data and the test data are converted to the vectors and these vectors are now fit to different machine learning models like SVM, Random forest classifier and Naive Bayes classifier.

Module Implementations

Module 3,4,5- SVM, Naive Bayes and Random Forest

SVM hyper-parameters used:

C= 1.0--used to control the error

kernel= linear

gamma=auto

- For multiclass classification in svm, One-vs-Rest (OvR) method is used.
- In this method, one of the classes is kept as positive class and the other classes are marked negative and this is done for each class.
- The probabilities of each class is the outputed and the class with best probability is predicted.
- For implementing the One-vs-Rest (OvR) method the scikit-learn libraries provide a method called OneVsRestClassifier under the multiclass package.

Module Implementations

Module 3,4,5- SVM, Naive Bayes and Random Forest

Random Forest hyper-parameters used:

max_depth=10

max_features='sqrt'

random_state=1

Bootstrap= True

n_estimators=100

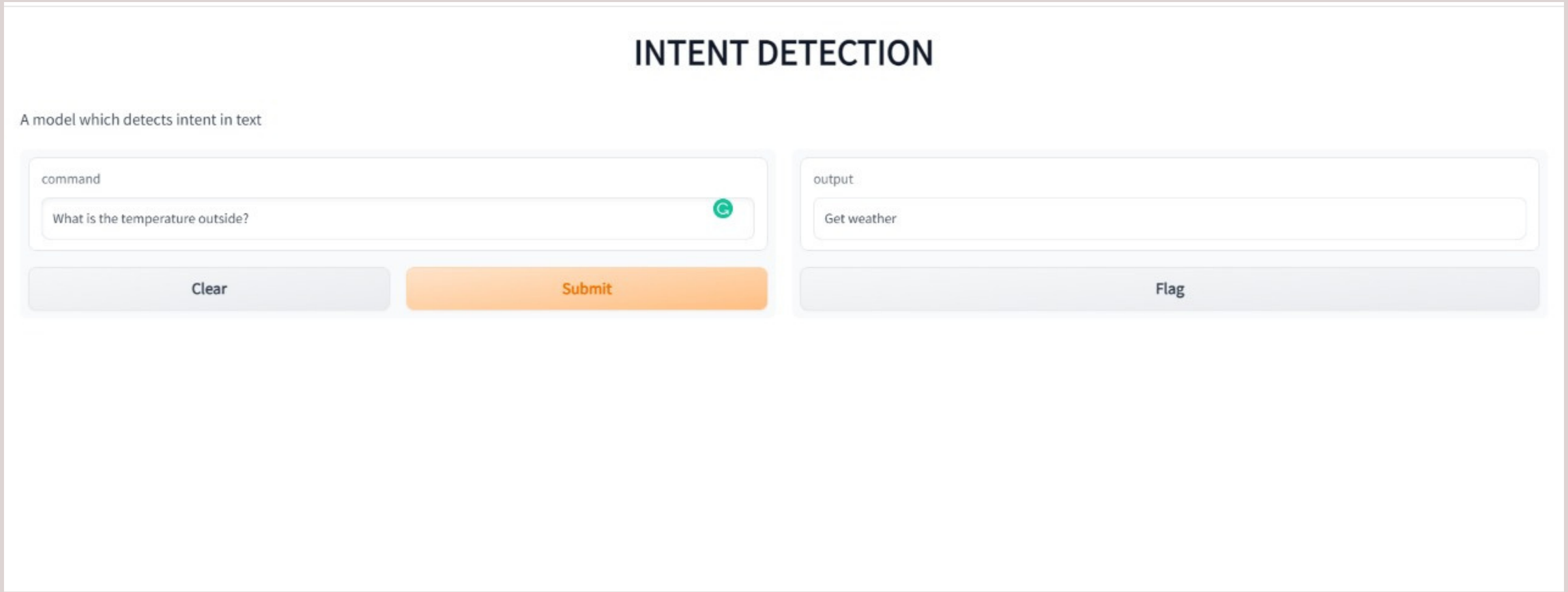
- Random forest is an ensemble algorithm that combines multiple decision trees.
- Many relatively uncorrelated models(trees) operating as a committee will outperform any of the individual constituent models.
- While some trees may be wrong, many other trees will be correct, so the trees can move in the right direction as a group.

Module Implementations

Module 3,4,5- SVM, Naive Bayes and Random Forest

- The naive Bayes model doesn't have any specific hyperparameters to tune,
- There are 2 types of Naive Bayes models- Gaussian and Multinomial, multinomial being the best.
- The multinomial model provides an ability to classify data, that cannot be represented numerically.
- Its main advantage is the significantly reduced complexity.
- The main Naive Bayes classifier in sklearn is called MultinomialNB and exists in the naive_bayes module.

User Interface



Final Results and Conclusion

Finally, summarizing the results of all the models,
BERT: 95. 86%
BiLSTM: 95. 6%
SVM: 96.43%
Naive Bayes: 96%
Random Forest: 93%

Future Scope:
We plan to use the BERT model and add more functionalities so as to be able to classify open intents.

References

Vasima Khan, Tariq Azfar Meenai, "**Pretrained Natural Language Processing Model for Intent Recognition (BERT-IR)**"

Tulika Saha, Aditya Prakash Patra, Sriparna Saha, Pushpak Bhattacharyya, "**A transformer based approach for identification of tweet acts**"

Sreelakshmi Ka, Rafeeque P C,, Sreetha S, Gayathri E S , "**Deep Bi-Directional LSTM Network for Query Intent Detection**"

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "**Attention Is All You Need**"

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova,"**BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**"

Thank you!