OSE PROJECT

Name: Sneha Roy
Matriculation Number: 50078578

Introduction:

In this project, my primary focus was on the task of intent classification using the "nlu_evaluation_dataset" sourced from the HuggingFace Library. This dataset consists of short conversational utterances, each annotated with corresponding intents and scenarios. The central objective of this project was to develop models capable of accurately classifying these intents. The project was structured into several phases to achieve this goal, ensuring a systematic approach. Additionally, I maintained consistency by using the same test set and custom text across all models to facilitate rigorous benchmarking.

The primary metric used for benchmarking different models was the **accuracy score**, which suited my dataset and my task at hand.

The custom text which was used was, ‘Why do I not feel well today’ . This is done intentionally, as it is not one of the regular commands given to the NLU systems, but something unorthodox which might confuse the models. Indeed so, I got different predictions for my custom text from different models.

All the numerical values mentioned in the following section of the report are based on the most recent execution of my notebook in Google Colab. I incorporated seed settings wherever I anticipated potential sources of randomness, and I also established and maintained a consistent environment for running the notebook. When it comes to the scikit-learn models, the results are entirely reproducible(to the last decimal point). However, for the transformer models, the numerical values may exhibit slight variations (not exceeding 1 percentage point) due to differences in GPU configurations across various devices and other non-deterministic factors. Nevertheless, I designed the code to consistently select the best-performing model (the one with the highest accuracy) when making predictions on the test data and custom text.

Data Preprocessing:

To ensure a robust evaluation of my models, I initiated the project with the "nlu_evaluation_dataset," which initially contained 25,715 training examples. To maintain an appropriate balance between training and evaluation, I partitioned the data into three sets: training, cross-validation, and test, using an 18:1:1 ratio. This resulted in 23,143 training examples, with 1,286 examples each for cross-validation and testing.

Project Report Part 1: Intent Classification Using Scikit-Learn Models

Feature Extraction and Scikit-Learn Models:

I employed the Term Frequency-Inverse Document Frequency (TF-IDF) method for feature extraction to vectorize the dataset. These vectorized features served as inputs to various scikit-learn models known for their proficiency in intent classification, which was my motivation for using the following models.

Logistic Regression: Logistic Regression after feature extraction seemed the most plausible method for me to start training my dataset, to help me classify the intents. For the Logistic Regression model, I utilized TF-IDF vectorization and performed hyperparameter tuning via Grid Search (codes provided in markdown for run time efficiency). The best hyperparameters yielded an accuracy of 83.43% on cross-validation and 85.92% on the test set.
The predicted label for the custom text is general_quirky

Decision Tree: Similarly, for the Decision Tree model, I applied TF-IDF vectorization and Grid Search for hyperparameter tuning. This model achieved an accuracy of 74.49% on cross-validation and 75.50% on the test set. The predicted label for the custom text is alarm_remove. This was my weakest model of the notebook not only performing poorly on the test set but also on the custom text.

Random Forest: The Random Forest model was optimized using TF-IDF vectorization and Grid Search. This model delivered an accuracy of 80.24% on cross-validation and 82.11% on the test set. The predicted label on the custom text was general_negate.

Gradient Boost: To fine-tune the Gradient Boost model, I applied TF-IDF vectorization and Grid Search. The Gradient Boost model achieved an accuracy of 81.18% on cross-validation and 83.51% on the test set, along with a general_negate predicted label on the custom text.

Results and Analysis of part 1 of the Project.

Among the scikit-learn models, the Decision Tree model demonstrated the weakest performance, producing unexpected predictions. In contrast, the Logistic Regression, Random Forest, and Gradient Boost models outperformed the Decision Tree. The Logistic Regression model achieved the highest accuracy score, though on the custom text, the Random Forest and Gradient Boost models exhibited greater accuracy. Interestingly, a real-world comparison with a home Alexa device revealed alignment with the predictions made by the Random Forest and Gradient Boost models. When my home Alexa was asked the same question, ‘why do I not feel well today’ and the answer was ‘Sorry, i do not know that one’, which indeed corresponds to ‘general_negate’

Conclusion:

In summary, the first part of this project focused on intent classification using scikit-learn models. While the Logistic Regression model displayed the highest accuracy, it's crucial to select a model tailored to the specific application, as evidenced by the performance variations on custom text. The unexpected behavior of the Decision Tree model warrants further investigation in future work. Part 2 of the project involves the utilization of advanced NLP models for intent classification

Project Report Part 2: Intent Classification Using Transformer Models:

In this phase, I employed four transformer models, namely 'bert-base-uncased,' 'roberta-base,' 'xlnet-base-cased,' and 'distilbert-base-uncased.' I created a consistent data dictionary containing the training, test, and cross-validation datasets. For each model, I conducted hyperparameter tuning by considering the number of epochs (3 or 4), batch size (64, 128), and learning rate (2e-5, 3e-5). I intentionally excluded the batch size option 32, as it did not suit my large dataset and yielded suboptimal results. I also omitted epoch 2 and learning rate 1e-5 due to their consistently poor performance across all models, along with significantly extending notebook execution time. The hyperparameter tuning was standardized across all models.

BERT:

I trained my BERT model on 'bert-base-uncased' since case sensitivity was not crucial for my dataset. After hyperparameter tuning based on cross-validation accuracy, my best BERT model achieved an accuracy of 91.29% on cross-validation and 90.82% on the test set. The test set and custom text remained consistent for rigorous benchmarking, with the custom text yielding the predicted label "general_quirky."

ROBERTA:

I tested the 'roberta-base' model, known for its larger training corpus and dynamic masking compared to BERT. The best ROBERTA model attained an accuracy of 90.7% on cross-validation and 90.0% on the test set.I expected the ROBERTA model to give significantly better than BERT model as it is trained on a larger vocabulary corpus and is able to capture more nuances of the text.
While my BERT model's performance was marginally superior to that of my ROBERTA model, the difference between them is quite minimal. Therefore, it's reasonable to conclude that my ROBERTA model performed on par with BERT. I believe that my dataset lacked the complexity that would necessitate the nuanced capabilities of the ROBERTA model. The task at hand was relatively straightforward and didn't demand an exceptionally deep comprehension of linguistic semantics, mainly because the dataset primarily consisted of everyday, informal, and concise utterances made to any Natural Language Understanding (NLU) device which led to the at- par performance of the 2 models. The custom text produced the predicted label "general_quirky."

DISTILBERT:

I trained my model on 'distilbert-base-uncased,' a distilled version of BERT. The best distilbert model achieved an accuracy of 90.9% on cross-validation and 91.05% on the test set, closely matching BERT and ROBERTA's performance. DiSTILBERT's lighter architecture(as the name suggests it is a distilled version of BERT) allowed for faster computations and lower resource requirements without any tradeoff for accuracy for my dataset. The predicted label was again general_quirky.

XLNET:

I opted for 'xlnet-base-cased' for my XLNet model, as no pre-trained model was available in an uncased version in HuggingFace. Despite case sensitivity not being a factor in my dataset, I wanted to run this model to see if the more sophisticated architecture of XLNet would be able to capture some of the text nuances that the variants of BERT models are failing to do. This is because XLNet captures the context in a permutation based way rather than the bidirectional way that the BERT variants employ.
The best XLNet model delivered an accuracy of 90.2% on cross-validation and 90.2% on the test set. While XLNet's sophisticated architecture excels at capturing complex text dependencies, it performed similarly to other models in my opinion, because my dataset is mostly simple comprising of only short everyday colloquial utterances.

Conclusion:

In conclusion, I anticipated the predicted label for transformer models to be "general_negate" rather than "general_quirky" due to their substantial performance improvements over scikit-learn models (a nearly 4% increase for every model compared to the best model of sklearn- Logistic Regression). In my dataset, training the DistilBERT model proved most logical, as it provided high performance with reduced computational resources. The choice of model should align with the specific application and dataset complexity, as highlighted by my results.