Individual Project 3 Report

Sourav Panda – sbp5911@psu.edu

**Task 1 - Implementation of two preprocessing steps choosing from the following (4 pts)**

**1.1 - Turn to lowercase 1.2 - Remove hashtags start with @**

A screen shot of a computer program

Description automatically generated A black screen with white text

Description automatically generated

**1.3 - Remove special characters** **1.4 – Lemmatization**

A black background with white text

Description automatically generated A computer screen with white text

Description automatically generated

**Task 2 - Implementation of transforming test texts into BOW representation (1 pt)** A black background with white text

Description automatically generated

* RandomForest and GridSearchCV, both were implemented to make predictions. Accuracy, Recall, and F1 all three parameters were used for refitting the GridSearchCV.

**Task 3 - Top-10 most similar words to “bomb” based on word2vec embeddings (2 pts)**

A screen shot of a computer code

Description automatically generated

**Task 4 - Tune the arguments of Word2Vec and show the top-10 similar words to “bomb” based on updated embeddings (1 pt)**

A screen shot of a computer code

Description automatically generated

* The Word2Vec (w2v) model was tuned with different hyper parameter values like vector\_size, window, negative, and min\_count. Implemented a Manual Parameter search algorithm that evaluates based on average of the sum of similarity scores of top 10 words similar to “bomb” generates best parameters.

**Task 5 - Implementation of document/tweet embedding (2 pts)**

**5.1 – Implementation of Choice 1**

A computer screen shot of a program

Description automatically generated

**5.2 – Implementation of Choice 2**

A screen shot of a computer program

Description automatically generated

**Task 6 - Implementation of using ML on word2vec embeddings (2 pts) - RandomForest with GridSearchCV**

**Choice 1 Choice 2**

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

* Different models were used on both choices including Logistic Regression, RandomForest, RandomForest with GridSearchCV and Support Vector Machines. RandomForest with GridSearchCV yields the best results as compared to other models. SVM yields poorer results compared to RandomForest indicating differences in how each model interacts with the dataset's characteristics.

**Task 7 - 1 Bonus point: implementation and result of RNN training**

**Implementation: -**

* **Tokenization and Vocabulary Building** - Utilized a basic English tokenizer. Built a vocabulary from the training dataset's text iterator. Special tokens like "<unk>" (unknown) were included.
* **Model Architectures** - A basic RNN model with embedding, dropout, and linear layers is used. An LSTM-based model, more advanced than the basic RNN, capable of capturing long-range dependencies in text is also used.
* **Hyperparameters** – Different hyperparameter values are used for different model architectures, like embedding dimension, hidden size, learning rate, dropout, and weight decay.
* **Training and Validation** - Data is split into training and validation sets. Loss values, accuracies, recalls, and F1 scores are tracked.
* **Metrics Calculation** - Accuracy, Recall, and F1 score are used for performance evaluation.
* **Visualization** - Plotting training and validation loss over epochs, provides a visual understanding of the learning process.

**Results of RNN training: -**

A graph with orange line and blue line

Description automatically generated

**Observations on RNN training: -**

* The use of different architectures (basic RNN, LSTM, bidirectional LSTM) provides flexibility in model selection based on the complexity of the dataset.
* The variations in hyperparameters suggest experimentation to optimize model performance.
* Dropout and weight decay help to combat overfitting.
* The choice of accuracy, recall, and F1 score offers a comprehensive view of model performance.
* The approach to tokenize and convert text data into tensors is crucial for text-based models.
* The inclusion of both training and validation phases ensures that the model's performance is evaluated on unseen data, mitigating overfitting.
* Loss plots help in monitoring the training process and diagnose issues like overfitting or underfitting.

**Further Improvements for RNN training: -**

* Implementing k-fold cross-validation could provide a more robust evaluation.
* Automated methods like grid search or random search could be employed to find the best hyperparameters.
* Post-evaluation, an analysis of misclassified examples could offer insights for further improvements.
* Exploring more advanced architectures like GRU or Transformer-based models could yield better results for complex datasets.