Sentiment Analysis with Basic Neural Networks

1. Introduction:

This report summarizes the methodology, result for Homework 2, mainly focusing on a three-class text classification of news headlines. The main objective is text classification using simple neural networks, specifically Feedforward Neural Networks (FNN) and Recurrent Neural Networks (RNN). Here I have used Word2Vec embeddings to represent words as dense vectors. I have extracted 15000 records i.e. 5000 records from each category (Politics, business, and Sports) from the HuffPost News Category Dataset.

2. Data Preprocessing:

From the three classes: **"POLITICS"**, **"BUSINESS"**, and **"SPORTS"**. 5000 records from each category were taken and created a balanced DataFrame named df_final. Then the DataFrame was shuffled to check that the data was not ordered specifically but any category.

First the test data was separated which as 20% of the entire dataset which consists of 3000 records. Then from the remaining 80% of the data, 70% of the data is used for training and 10% data is used for validation.

```
Training Set (70%): 10500 records
Validation Set (10%): 1500 records
Testing Set (20%): 3000 records
```

2.1. Removal of Missing words:

After these splits, I have applied a text cleaning step where I have involved a custom function named ‘remove_missing’ with the Word2Vec vocabulary to clear-out the missing values.

remove_missing(text)=’ ’.join(word for word in text.split() if word in w2v)

3. Text Processing:

After the data split and removing the missing words, the next crucial phase is vectorising the text and preparing the pre-trained Word2Vec embeddings.

3.1. Word2Vec:

I have used the pre-trained Word2Vec (Google News 300) model via genism library. This model contains 300-dimentional dense vectors for millions of words. Word2Vec is implemented as a two-layer neural network trained to reconstruct the linguistic context of words. I have used the vectors extracted from this pre-trained model.

Then I have created an embedding matrix which links the integers indices from tokenizer to the 300-dimensional Word2Vec vectors. The matrix was initialized with zeros, and the dimensions of the matrix are (voacb_size, 300). The matrix was filled by iterating through our vocabulary (word_index).

- If the word was present in the pre-trained Word2vec model, its 300D vector is inserted into the corresponding row of the matrix.
- If the word is not present in the pre-trained Word2vec model, The corresponding row remained zero.

3.2. Tokenization and Sequence Padding:

A keras tokenizer is fitted only on the training data to establish the word_index and voacb_size specific tom our task.

The maximum sequence length is manually calculated based on the longest headline found in the training and validation data.

All the three datasets i.e. training, validation, and testing were converted into sequence of integers using the tokenizer.

Then the integer sequences were padded using pad_sequence up to the maximum length.

3.3. Label and One-Hot encoding:

A label encoder was implemented on the training and validation data to convert the three category strings (“Politics”, “Business”, “Sports”) to discrete integer classes.

Now these integers classes are converted into one-hot encoded format (y_train_oh, y_val_oh, y_test_oh) to use in the models’ output layer.

4. Model Description:

4.1. Model 1: Feed Forward Neural Network (FFN)

Embedding Layer: The input sequence is processed by the embedding layer, which uses the previously calculated Word2vec embedding (300 dimensions) from the embedding matrix. This layer is non-trainable layer, keeping the weights constant and making the subsequent layers to learn feature extraction directly from the fixed semantic representations which is provided by Word2vec.

Flatten Layer: The output from the embedding layer, which is matrix with sequence of words is flatted into a single large vector before sending it to the next hidden layers.

Hidden Layers: Now, the flattened vector is passed through the fully connected dense hidden layer with ReLU activation function.

Output layer: It consists of 3 units and a SoftMax activation function.

4.2. Model 2: RNN (GRU) with FFN Classification Head

This is a type of recurrent neural network which is best for sequential data.

Embedding Layer: It is also like FFN which uses the non-trainable Word2Vec embeddings.

GRU Layer: It has 128 GRU nodes which processes the sequential input one timestep at a time. GRUs use update and reset gates to control the flow of information. GRU have few parameters compared to LSTM which makes them computationally lighter.

Output Layer: The output of the GRU layer is then applied SoftMax function and generates probability across the 3 output classes. The class with highest probability is chosen.

Model 3: LSTM with FFN Classification Head

Embedding Layer: The input layer consists of Word2Vec embeddings, which are non-trainable to preserve the pre-trained semantic features.

LSTM Layer: The single LSTM layer with 128 nodes processes the input sequence. It gains the ability to retain a memory state throughout the headline’s words. And for the final output layer it represents the contextual summary of the entire sequence.

Output Layer: This is same a model 2, which consists of 3 units and SoftMax activation function.

5. Evaluation:

The model gives the prediction probabilities of the three classes from the SoftMax layer. These probabilities are now converted into hard class labels by selecting the index corresponding to maximum probability. By using the label encoder both the true labels and predicted indices are converted into original string format (“politics”, “Business”, “sports”).

After that per-class precision, per-class recall and per-class F1 scores are calculated.

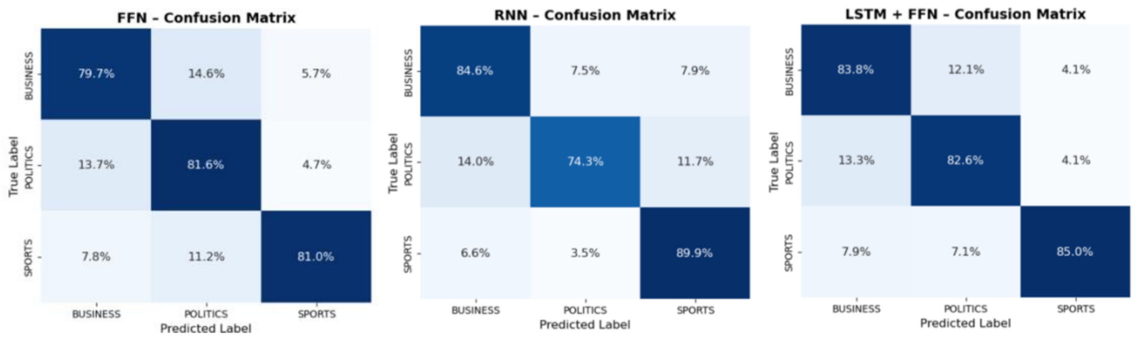
Precision: Out of all instances the model predicted as belonging to a class, how many were actually correct.

Recall: Out of all actual instances, how instances did the model predict correctly.

F1 Score: The mean of Precision and Recall which provides a single idea of both.

Confusion Matrix: A matrix that shows us the performance by number or percentage of correct and incorrect predictions for each class.

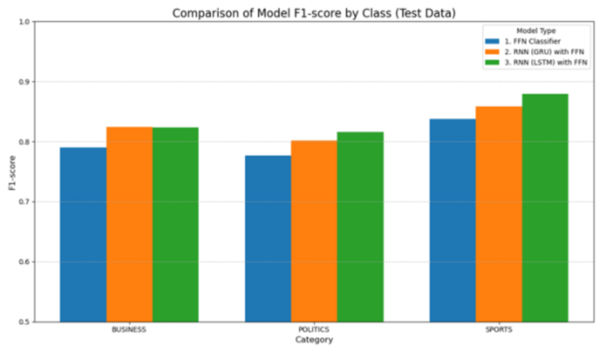
Category	FFN Precision	FFN Recall	FFN F1	GRU RNN Precision	GRU RNN Recall	GRU RNN F1	LSTM RNN Precision	LSTM RNN Recall	LSTM RNN F1
BUSINESS	0.773	0.808	0.790	0.804	0.846	0.825	0.812	0.835	0.823
POLITICS	0.797	0.757	0.776	0.871	0.743	0.802	0.830	0.802	0.816
SPORTS	0.836	0.840	0.838	0.821	0.899	0.858	0.877	0.882	0.879



Model 1: The diagonal values are high and sports as leading category. The model mainly struggles with politics, frequently misclassifying 228 actual headlines as business.

Model 2: The model performs well on sports and business categories. These misclassifications occur due to overlap in vocabulary between politics and business.

Model 3: The model performs badly while misclassifying politics (212 misclassifications) as business. Which says that the model is little bit recall bias towards business category.



Model 2 and Model 3 performs better than model 1 which shows that RNN’s ability to utilize word order and sequence context Is beneficial for this classification task.

Sports category yields the highest F1 score (0.879) in all models by that we can say that sports headlines use a distinct vocabulary that is easier to separate from others.

Politics has the lowest F1 score for all models (0.776) from that we can say vocabulary overlaps more significantly with other categories or that context is hard to capture.