[NN'23] Arabic Sentiment Analysis

24-12-2023

Artificial Neural Network and Deep Learning

Project stages

1- data preprocessing

2- data preparation (model entry)

3- build transformers model architecture.

4- find best model (hyperparameter tuning)

5- predict y-test in csv file.

6- build LSTM model architecture.

7- repeat stage 4, 5 for LSTM model.

Preprocessing

1) **Reading Data:**

- read file “train.xlsx” which consists of three columns[“ID”, “review_description”, ‘,“rating”].
- read file “test _no_label.csv” which consists of columns[“ID”, “review_description].

2) Handling Null Values:

- It replaces the null values in the "review_description" column of the training data with the string "مجهول" (meaning "unknown" in Arabic).
- Similarly, it attempts to replace null values in the test data's "review_description" column with the same Arabic string.

3) Data Cleaning:

- Converting the "review_description" column in both datasets to string type.Removing specific characters (#.][!XR) from the "review_description" column in both train_data and test_data.
- Replacing empty strings in train_data with NaN (missing value indicator using replace and inplace=True. Re-filling missing values in train_data's "review_description" column with "مجهول".

4) Techniques applied on "review_description" column:

- remove_punctuations: removes Arabic and English punctuations to create a list of punctuations such as['"`÷×؛<>_()*&^%][ـ“…”!|+¦~{}',.؟":/؛-"'] from the text.
- remove_repeating_char: removes repeating characters in the text.
- remove_stop_words: removes Arabic stop words such as {' 'مش','ليست','ليس','لا','نعم
,'واو','لسنا','لسن','لستن','لستما','لستم','لست','ليست','ليسا','ليس','أقبل','غير','ما'}
from the text.

- build_emoji_dictionary: reads an emoji dictionary from a CSV file, replaces emojis with corresponding text from the emoji dictionary then removes emojis from the text.

- Arabic Text Normalization: normalizes some Arabic characters to their standard forms such as ["ئ","ء","ؤ","ا", "[إأآا]"].

- remove_diactrics: removes diacritics (small signs above or below characters) from Arabic text using the pyarabic library.

- Tokenization: tokenize each sentence in the "review_description" column. The regular expression \w+ matches word characters (alphanumeric and underscores) to tokenize the text.

- Stemming: text processing task in which you reduce words to their root, After stemming, it rejoins the stemmed tokens into a single string per sentence.

- Prepare data to enter models:

1- Tokenization

- *Split texts into list of words*

2- To-sequence

- *Get the dictionary mapping each unique word to integer number.*
- *Text becomes sequence of numbers.*

3- Padding

- *Unit all texts length to one fixed size by getting the maximum length and adding 0's at the end of short texts.*
- *In case of longer ones remove from the start of text to adjust length*

Models building

Transformer Model

Model Architecture

- ***Embedding Layer***: Converts input tokens into dense vectors of fixed size ("d_model: dimension of embedding vector") .

- ***Reshape***: Output will be in shape of (batch size ,sequence length, numoffeatures "d_model: dimension of embedding vector").

- ***Multi-Head Attention Layer***: Computes self-attention, allowing the model to weigh different words in the input sequence when encoding information**.**

- ***Residual connections*:** are crucial in deep neural networks to address the vanishing gradient problem. enhance the information flow, ease the training of deep networks, and contribute to the model's ability to capture long-range dependencies. They are particularly important in architectures with multiple layers and sequential processing, such as recurrent neural networks (RNNs) with attention mechanisms.

  - ✓ After applying the attention mechanism (self.attention), there's an element-wise addition (x = x + attention_output). This is a residual connection.

  - ✓ The purpose is to allow the information from the input (x) to flow directly to the output, bypassing the attention mechanism if needed.

  - ✓ It helps with the flow of information during training and can mitigate issues like vanishing gradients.

- ***Sequential Layer***: Feedforward Neural Network (FFN) Two dense layers used for feature transformation within each transformer block. adjusting the weights to minimize the difference between the predicted output and the true target values.

  - ✓ The first layer (Dense(ff_dim, activation='relu')) applies a rectified linear unit (ReLU) activation function to the output of the layer. For

each element x in the output of the first dense layer, ReLU applies the following transformation.

- ✓ The second layer (Dense(d_model)) is another dense layer without an activation function specified, which means it applies a linear transformation. The output is a weighted sum of the inputs plus biases $y=\sum iWi \cdot xi+b$.

- ***GRU Layer***: type of RNN layer that can capture dependencies and patterns in sequential data "While transformers excel at capturing global dependencies through self-attention, the incorporation of recurrent layers like GRU allows the model to capture more fine-grained sequential context"

- ***Normalization Layer***: Used for normalization and regularization, by:

  - ✓ Normalizing the activations helps in stabilizing the training process. It can mitigate issues related to vanishing or exploding gradients by keeping activations within a similar range during training.

  - ✓ ovariate shift refers to the change in the distribution of inputs to a learning system. reduces this shift by standardizing the activations, making the optimization process more robust.

  - ✓ Parameters: epsilon=1e-6 This parameter is a small value added to the variance to prevent division by zero when normalizing

- ***Dropout Layer***: is a regularization technique that helps in reducing overfitting by randomly setting a fraction of input units to zero during each training step. This prevents units from co-adapting too much, forcing the model to learn more robust features.

  - ✓ Parameter: The rate parameter specifies the fraction of the input units to drop. In this case, rate=0.1 means 10% of the input units will be randomly set to zero during training.

- ***Pooling Layer***: pooling is a technique used to reduce the spatial dimensions of the input tensor, typically employed in convolutional or 1D sequence-based models.

- ***Dense layer***: is a fully connected neural network layer. It performs a linear operation on the input data, followed by ‘’softmax” activation to produce class probabilities.

Hyperparameters of the Transformer model

- ***vocab_size***: Represents the size of the vocabulary, usually derived from the tokenizer's word index.
- ***d_model***: Determines the dimensionality of the word embeddings.
- ***num_heads***: Specifies the number of attention heads to employ in the Multi-Head Attention layer.
- ***ff_dim***: Indicates the dimensionality of the feedforward network used within each Transformer block.
- ***num_transformer_blocks***: Defines the number of Transformer blocks or layers to stack.output_dim: Denotes the number of classes for the classification task.

Bi-LSTM Model

Model Architecture

- ***The Sequential***: The Sequential model in Keras is a linear stack of layers. You can add layers to the model one by one in a sequential manner.
- ***Embedding layer***: as used in transformer model with Parameters:
  - ✓ Get the length of dictionary of unique vocabularies in the dataset.
  - ✓ embedding dimension (size of vector representation of the word),
  - ✓ Input length: each input sequence.

- ***Bi-Lstm layer***: Processes the input sequence in both forward and backward directions. Enhances the model's ability to capture long-range dependencies in sequential data. It consists of two LSTMs, one processing the sequence from start to end, and the other from end to start. use three gates to control the flow of information:

  - ✓ Input Gate: Deciding What to Write. Uses a mechanism similar to deciding whether an event is important (close to 1) or not (close to 0).

  - ✓ Forget Gate: Cleaning Up Old Entries. Output values close to 0 for things you want to forget and close to 1 for things you want to remember.

  - ✓ Output Gate: Deciding What to Share. Filters the information in your diary to produce the output.

  - ✓ Each gate is implemented using a sigmoid activation function, outputting values between 0 and 1. These values act as gates, regulating the flow of information.

- ***Dropout layers***: Dropout layers are used for regularization to prevent overfitting by randomly dropping a fraction of input units.

Evaluation of models:
Using hyper tuning technique to get the best hyperparameters get best model.

Second model architecture evaluation

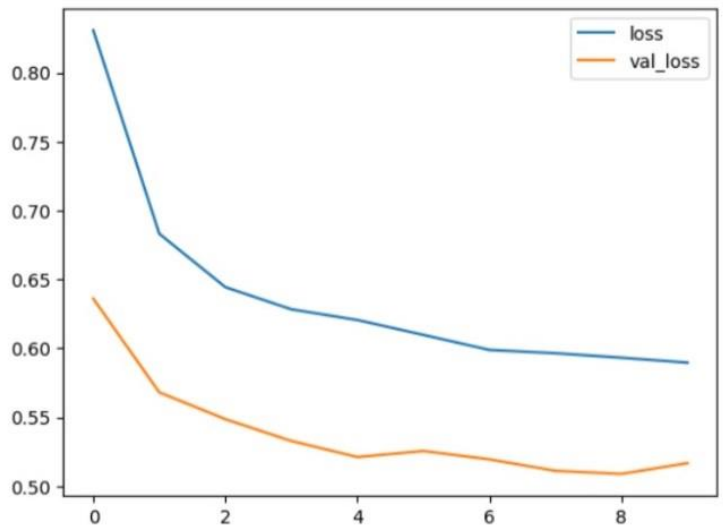
<table>
<tr>
<td>

accuracy without lstm layer

Highest accuracy: 82.04

hyperparameters:

- *Neurons of bidirectional: 10*
- *Dense1: 5*
- *Dense1_activation: tanh*
- *Dense1_dropout: 0.3*
- *Dense2: 2*
- *Dense2_activation: sigmoid*
- *Dense2_dropout: 0.6*
- Learning rate: 0.00178449986

</td>
<td> 
</td>
</tr>
<tr>
<td>

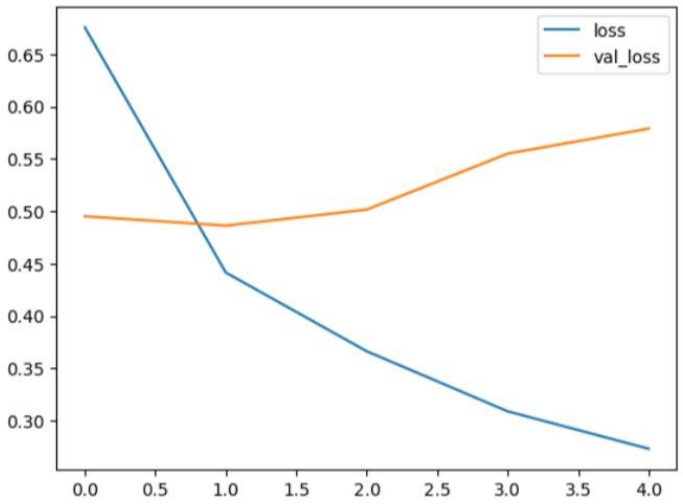
Highest accuracy: 84.11

Hyperparameters:

- *Units “neurons”: 64*
- *Dropout: 0.5*
- *Neurons of bidirectional:64*
- *Dense1: 64*
- *Dense1_activation: relu*
- *Dense1_dropout: 0.1*
- *Dense2: 32*
- *Dense2_activation: sigmoid*
- *Dense2_dropout: 0.2*

</td>
<td>

Possible overfitting



</td>
</tr>
</table>

-

transformers model evaluation

Best accuracy "submitted."	**Best hyperparameters:**
Accuracy: 84.1604	- 'd_model': 96, - 'Num_heads': 8, - 'Ff_dim': 128 - Dropout rate': 0.1 - Learning rate': 0.00022491452958964034.
highest accuracy: 84.53	**Hyperparameters:** - 'Dropout rate': 0.2 - 'Num_transformer_blocks': 2 - 'd_model': 256 - 'Num_heads': 6 - 'Ff_dim': 1024, - 'Learning rate': 0.00046695477734523. - 'optimizer': 'rmsprop'
With position encoding: Accuracy: 83.957	**Hyperparameters:** - 'Dropout rate': 0.4, - 'Num_transformer_blocks': 5 - 'd_model': 256 - 'Num_heads': 4 - 'Ff_dim': 512 - 'Learning rate': 0.00030615367588649 57. - 'optimizer': 'Adam'

Summary:

transformers model gives better results than LSTM as its architecture covers mostly the complex relationships and sequential flow of information and this is good approach for sentiment analysis task