UNIVERSITE HASSIBA BENBOUALI DE CHLEF

FACULTE DES SCIENCES EXACTES ET INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

RECURRENT NEURAL NETWORKS

BELBACHIR Moundir-Oussama

ALLOUACHE Imadeddine

REDAOUIA abdelhakim

CONTENTS

1. WHAT IS RNN?

A recurrent neural network is a type of artificial neural network commonly used in speech recognition and natural language processing. Recurrent neural networks recognize data's sequential characteristics and use patterns to predict the next likely scenario.

1. WHY RNN?

Issues in the feed forward neural network:

- Can’t handle sequential data.
- Consider only current input.
- Can’t memorize the previous input.

2. APPLICATIONS OF RECURRENT NEURAL NETWORK

Recurrent Neural Networks (RNNs) are widely used for data with some kind of sequential structure. here is a list of common applications:

- Speech Recognition
- Sentiment Classification
- Machine Translation (i.e. Chinese to English)
- Video Activity Recognition
- Name Entity Recognition — (i.e. Identifying names in a sentence)

3. THE ARCHITECTURE OF AN RNN

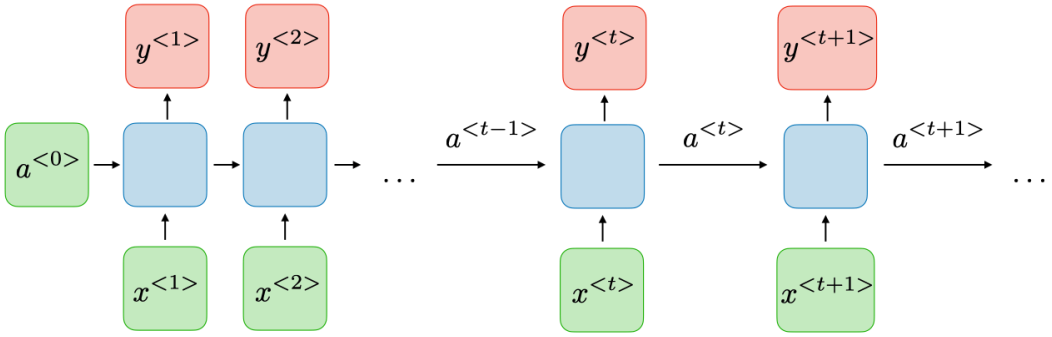


FIGURE 1 RNN ARCHITECTURE

For each timestep '***t***', the activation '***a<t>***' and the output '***y<t>***' are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad \text{and} \quad y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

where '***W_ax***', '***W_aa***', '***W_ya***', '***b_a***', and '***b_y***' are coefficients shared across time steps. '***g1***' and '***g2***' represent activation functions.

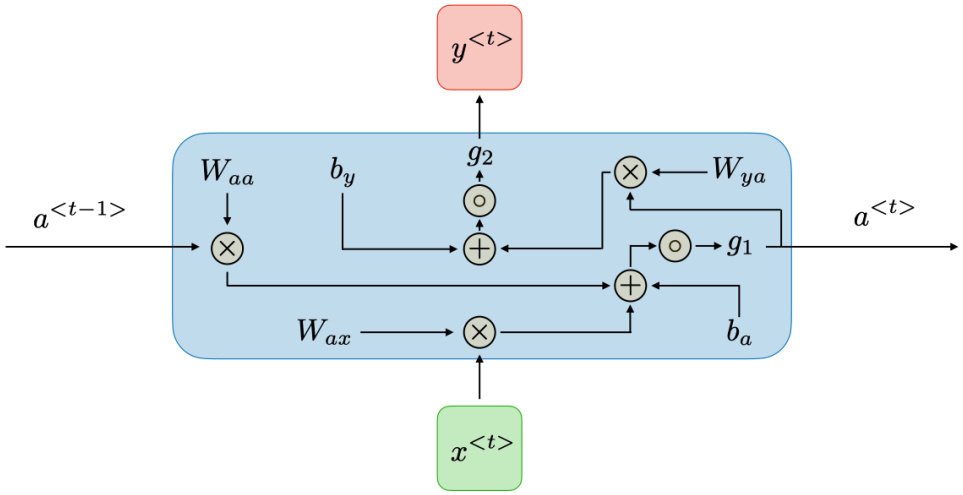
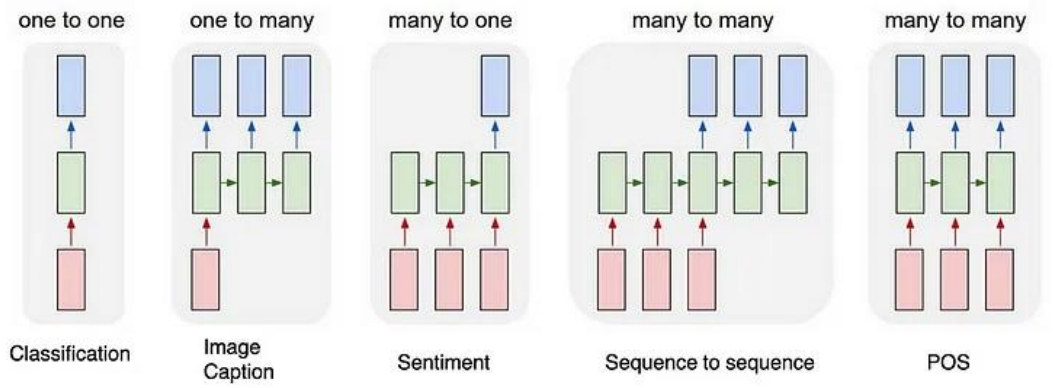


FIGURE 2 RNN HIDDEN STATE

4. TYPES OF RNN



One to One

This type of RNN behaves the same as any simple Neural network it is also known as Vanilla Neural Network. In this Neural network, there is only one input and one output.

One To Many

In this type of RNN, there is one input and many outputs associated with it. One of the most used examples of this network is Image captioning where given an image we predict a sentence having Multiple words.

Many to One

In this type of network, many inputs are fed to the network at several states of the network generating only one output. This type of network is used in the problems like sentimental analysis. Where we give multiple words as input and predict only the sentiment of the sentence as output.

Many to Many

In this type of neural network, there are multiple inputs and multiple outputs corresponding to a problem. One Example of this Problem will be language translation. In language translation, we provide multiple words from one language as input and predict multiple words from the second language as output.

5. LOSS FUNCTION

In a recurrent neural network, the overall loss function ***L*** across all time steps is defined based on the loss at each individual time step**:**

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$$

6. BACKPROPAGATION THROUGH TIME

At each time step, backpropagation is performed. At timestep ***T***, the derivative of the loss ***L*** with respect to the weight matrix ***W*** is expressed as follows:

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial W}\Big|_{(t)}$$

2. ADVANTAGES AND DISADVANTAGES

Advantages

1. Possibility of processing input of any length
2. Model size not increasing with size of input
3. Computation takes into account historical information
4. Weights are shared across time.

Disadvantages

1. Computation being slow
2. Difficulty of accessing information from a long time ago
3. Cannot consider any future input for the current state

3. IMPLEMENTATION :

```
url = "https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz"

dataset = tf.keras.utils.get_file("aclImdb_v1.tar.gz", url,
                                  untar=True, cache_dir='.',
                                  cache_subdir='')

dataset_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')
os.listdir(dataset_dir)
```

This code downloads the IMDb movie review dataset and extracts it to the current directory, providing access to the dataset for further analysis or training.

```
batch_size = 1024
seed = 123
train_ds = tf.keras.utils.text_dataset_from_directory(
    'aclImdb/train', batch_size=batch_size, validation_split=0.2,
    subset='training', seed=seed)
val_ds = tf.keras.utils.text_dataset_from_directory(
    'aclImdb/train', batch_size=batch_size, validation_split=0.2,
    subset='validation', seed=seed)
```

This code creates training and validation datasets from the 'aclImdb/train' directory with a batch size of 1024, using 80% for training and 20% for validation, ensuring reproducibility with a seed of 123.

```
embedding_dim=16

model = Sequential([
    vectorize_layer,
    Embedding(vocab_size, embedding_dim, name="embedding"),
    SimpleRNN(8),
    Dense(1, activation='sigmoid')
])
```

This code constructs a simple Sequential model with a text vectorization layer, followed by an embedding layer with a dimensionality of 16, a SimpleRNN layer with 8 units, and a dense layer with a sigmoid activation function for binary classification.

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

This code compiles the model using the Adam optimizer, binary cross-entropy loss function with logits, and accuracy as the evaluation metric.

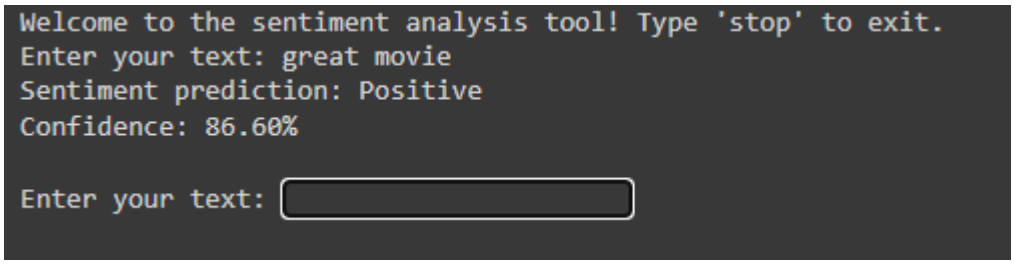
```
model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=15
)
```

This code trains the model using the training dataset train_ds, validating on val_ds, for 15 epochs.

```
while True:
    curr_input = input("Enter your text: ")

    if curr_input.lower() == 'stop':
        print("Exiting the sentiment analysis tool. Goodbye!")
        break

    if not curr_input.strip():
        print("Please enter some text.")
        continue

    pred = model.predict([[curr_input]], verbose=False)[0][0]

    print("Sentiment prediction:", "Positive" if pred >= 0.5 else "Negative")
    print("Confidence:", "{:.2f}%".format(pred * 100))
    print()
```

This code continuously prompts the user to enter text for sentiment analysis. Entering "stop" exits the tool. It predicts the sentiment of the input text using the trained model and displays the prediction along with the confidence score as a percentage.

OUTPUT EXAMPLE:



REFERENCES:

- TECHTARGET - RECURRENT NEURAL NETWORKS
- MEDIUM - UNDERSTANDING RECURRENT NEURAL NETWORKS (RNNS)
- STANFORD CS230 CHEATSHEET - RECURRENT NEURAL NETWORKS
- TENSORFLOW - WORD EMBEDDINGS