DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
HIGHER NATIONAL SCHOOL OF COMPUTER SCIENCE

2CS SIL1 2022-2023

TP BDM

Deep Learning
Supervised Classification using RNN

Realized by :

- Ines ABDELAZIZ
- Meriem Afaf HADDOU

Supervised by :

- Leila HAMDAD

Contents

Chapter 1

What is deep learning ?

1.1 Definition

Deep learning is a branch of machine learning that is inspired by the way our brain functions to acquire knowledge. It is sometimes referred to as hierarchical learning or deep structured learning. The concept of deep learning is based on artificial neural networks that are designed to process vast amounts of data by increasing the depth of the network in a particular way. By using this technique, a deep learning model can extract features from raw data and gradually learn about these features in each layer, leading to a higher-level understanding of the data. This process is known as hierarchical feature learning and requires minimal human intervention. Pioneers in the field have provided various definitions of deep learning:

> *A sub-field within machine learning that is based on algorithms for learning multiple levels of representation to model complex relationships among data. Higher-level features and concepts are thus defined in terms of lower-level ones, and such a hierarchy of features is called a deep architecture [1]*

> *The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI, deep learning [2]*

Deep learning has become popular and widely used because of its ability to scale effectively. In contrast to older machine learning algorithms, which have a plateau in performance and cannot handle large amounts of data, deep learning models can theoretically process an unlimited amount of data and even surpass human comprehension. This is demonstrated in modern image processing systems that are based on deep learning, which have been shown to outperform human performance. Therefore, one of the key advantages of deep learning is its scalability, which enables it to perform better with more data.

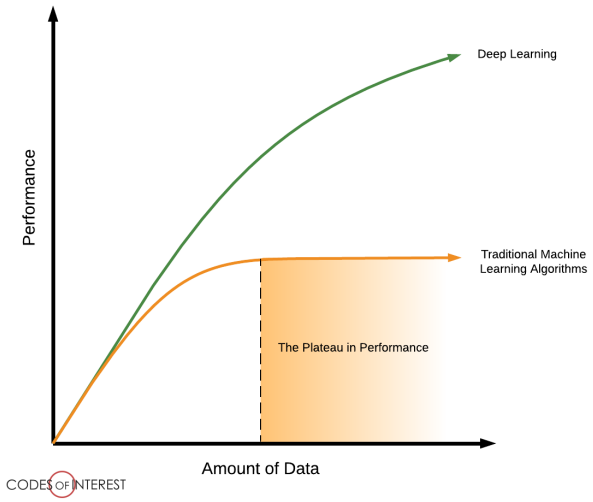

Figure 1.1: The lack of plateau in performance in deep learning

Chapter 2

The objective of deep learning

> *The key objective of deep learning is to enable computers to automatically learn and improve from experience, without being explicitly programmed. [3]*

The quote by Schmidhuber illustrates the primary objective of deep learning, which is to enable computers to learn and improve on their own through experience, without requiring explicit programming by humans. The technique of using neural networks to achieve this goal involves training these models on vast amounts of data, allowing them to learn complex patterns and make predictions or decisions based on that data.

Deep learning has the potential to revolutionize numerous industries, including healthcare, finance, transportation, and entertainment, by enabling machines to learn and improve in a way that is similar to human learning. By allowing computers to learn from experience and improve autonomously, deep learning has the potential to solve complex problems that are otherwise difficult to solve using traditional machine learning methods.

The ability to learn and improve through experience is what distinguishes deep learning from other machine learning techniques, which rely on explicit programming and handcrafted features. By learning to recognize patterns and extract features from data on their own, deep learning models can achieve state-of-the-art performance on a wide range of tasks, including image recognition, natural language processing, and recommendation systems.

Overall, deep learning represents a powerful tool for solving complex and challenging problems, and its potential impact on various industries and domains is only beginning to be realized.

Chapter 3

Sentiment140 dataset with 1.6 million tweets

The Sentiment140 dataset is a collection of 1.6 million tweets that have been labeled with the sentiment expressed in the tweet.
The dataset is balanced, which means it contains an equal number of negative and positive tweets. The tweets were collected in February 2009 using the Twitter API, and each tweet has been labeled as positive, negative, or neutral based on the presence of positive or negative emoticons. It has been widely used for training and testing machine learning models for sentiment analysis, and has also been used for research in natural language processing and social media analysis.
The dataset contains the following columns:

- target : The sentiment label of the tweet, where 0 = negative, 2 = neutral, and 4 = positive.
- ids: The unique ID of the tweet.
- date : The date and time at which the tweet was created.
- flag : A query parameter used to filter out spam tweets.
- user : The user who posted the tweet.
- text: The text of the tweet.

The summary statistics of this dataset are as follows:

- The dataset contains a total of 1,600,000 tweets.
- The tweets were posted between April 6, 2009 and June 25, 2009.
- There are no missing values in any of the columns.
- The most common sentiment label is neutral, with 800,000 tweets labeled as such.
- The positive and negative labels are each represented by 400,000 tweets.

Chapter 4

Supervised classification using RNN

Sentiment analysis has become an increasingly important tool for understanding people's emotions, opinions, and sentiments towards a given topic. It involves the process of extracting and analyzing opinions from various sources, such as social media, reviews, and surveys

> *Sentiment analysis is the combination of emotions, opinion and sentiments towards the given topic. It is the procedure of extracting opinion from particular task. Sentiment analysis is the powerful tool for specific event such as movies reviews and general elections [4]*

In this report, we aim to perform a supervised classification of the Sentiment140 dataset using a recurrent neural network (RNN). RNNs are a type of neural network that are well-suited for processing sequential data, making them an ideal choice for sentiment analysis

4.1 Foundational Concepts

4.1.1 Recurrent Neural Network

A recurrent Neural Network (RNN) is a class of neural networks whose connections between neurons form a directed cycle. Unlike feedforward neural networks, RNN can use its internal "memory" to process a sequence of inputs, which makes it popular for processing sequential information. The"memory" means that RNN performs the same task for every element of a sequence with each output being dependent on all previous computations, which is like "remembering" information about what has been processed so far.[5]

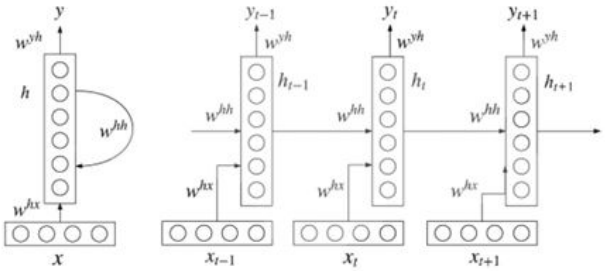

Figure 4.1: Recurrent Neural Network (RNN)

figure 4.2 shows an example of RNN.The left graph is an unfolded network with cycles, while the right graph is a folded sequence network with three time steps. The length of time steps is determined by the length of input. For example, if the word sequence to be processed is a sentence of six words, the RNN would be unfolded into a neural network with six time steps or layers. One layer corresponds to a word.

4.1.2 LSTM

LSTM (Long Short Term Memory) is the part of RNN which is used to learn long-range dependencies for text sequences. LSTM contains memory blocks which also known as gates to control the text flow. The memory blocks contain three gates named as input gate; forget gate and output gate to control the flow of information [6]

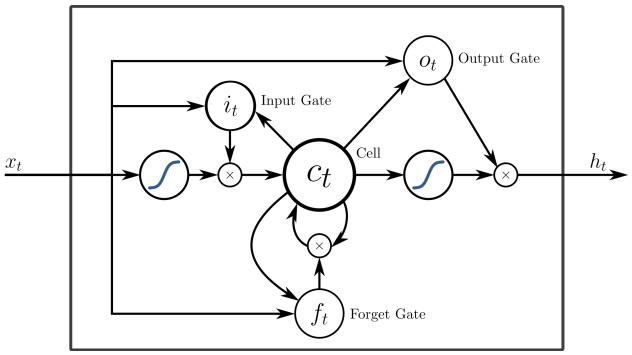

Figure 4.2: Long Short-Term Memory (LSTM)

4.2 Loading the Dataset

We start with importing the necessary packages for text manipulation and model building.

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Sequential
```

We load the dataset into a pandas dataframe. We only use the first (sentiment) and sixth(text) columns of the dataset and rename them as label and text

```
df = pd.read_csv('Sentiment140.csv', encoding='ISO-8859-1', header=None)
df = df.iloc[:, [0, 5]]
df.columns = ['label', 'text']
```

The data looks like this :

	label	text
0	0	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	is upset that he can't update his Facebook by ...
2	0	@Kenichan I dived many times for the ball. Man...
3	0	my whole body feels itchy and like its on fire
4	0	@nationwideclass no, it's not behaving at all....
...	...	...
1599995	4	Just woke up. Having no school is the best fee...
1599996	4	TheWDB.com - Very cool to hear old Walt interv...
1599997	4	Are you ready for your MoJo Makeover? Ask me f...
1599998	4	Happy 38th Birthday to my boo of alll time!!! ...
1599999	4	happy #charitytuesday @theNSPCC @SparksCharity...

1600000 rows × 2 columns

Figure 4.3: Sentiment140 dataset

4.3 Data Preprocessing

We convert all the tweet texts to lowercase and remove all non-alphanumeric characters and numbers. Finally, we remove any leading and trailing white spaces. This preprocessing step ensures that the text data is clean and standardized, which is essential for training the RNN model.

```
df['text'] = df['text'].apply(lambda x: x.lower())
df['text'] = df['text'].str.replace('[^\w\s]', '')
df['text'] = df['text'].str.replace('\d+', '')
df['text'] = df['text'].str.strip()
```

4.4 Labeling and Making Train-Test Splits

We replace the numeric labels with the corresponding string labels 4's and 0's into ('positive', 'negative') respectively.

```
df['label'] = df['label'].replace({0: 'negative', 4: 'positive'})
```

Finally, we split the dataset into train and test parts. We use 80% of the dataset for training and 20% for testing.

```
train_size = int(len(df) * 0.8)
train_data = df[:train_size]
val_data = df[train_size:]
```

Before being fed into the LSTM model, the data needs to be padded and tokenized:

- **Tokenizing:** Keras' inbuilt tokenizer API has fit the dataset, which splits the sentences into words and creates a dictionary of all unique words found and their uniquely assigned integers. Each sentence is converted into an array of integers representing all the individual words present in it.
- **Sequence Padding:** The array representing each sentence in the dataset is filled with zeroes to the left to make the size of the array ten and bring all collections to the same length.

```
# Tokenization
tokenizer = Tokenizer(num_words=10000, oov_token='<OOV>')
tokenizer.fit_on_texts(train_data['text'])
# Padding
train_sequences = tokenizer.texts_to_sequences(train_data['text'])
train_padded = pad_sequences(train_sequences, maxlen=50, padding='post',
↪  truncating='post')
val_sequences = tokenizer.texts_to_sequences(val_data['text'])
val_padded = pad_sequences(val_sequences, maxlen=50, padding='post',
↪  truncating='post')
```

In this step, we use the Tokenizer class from Keras to tokenize the tweet texts in the training data. We create an instance of the Tokenizer class with the num_words parameter set to 10000, which means that we only consider the 10000 most frequent words in the training data.
We then fit the tokenizer on the training text data. This updates the internal vocabulary of the tokenizer based on the training data.
After fitting the tokenizer, we convert the tweet texts in the training data to sequences of tokens. This replaces each word in the text with its corresponding index in the tokenizer's vocabulary.
Finally, we pad the tokenized sequences to a fixed length of 50. Padding ensures that all input sequences have the same length.
This step is important because it converts the text data into numerical data that can be used as input to the RNN model.

4.5 Building the Model

A Keras sequential model is built. It is a linear stack of the following layers :

- **Embedding Layer:** Converts each word in the input sentence into a dense vector of size 32. The input dimension is the vocabulary size (10,000 words), and the output dimension is 32.
- **LSTM Layer:** Processes the sequence of dense vectors and captures long-term dependencies, with 64 LSTM units and dropout of 0.2.
- **Dense Layer:** Produces a probability distribution over the two possible sentiment classes (positive or negative) using a softmax activation function, with 2 output units.

```
#model architecture
model = Sequential([
    Embedding(input_dim=10000, output_dim=32, input_length=50),
    LSTM(64, dropout=0.2, recurrent_dropout=0.2),
    Dense(2, activation='softmax')
])

#Compiling the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
↪  metrics=['accuracy'])
#Summary
model.summary()
```

The model is compiled with binary cross-entropy loss and adam optimizer. Since we have a binary classification problem, binary cross-entropy loss is used. The Adam optimizer uses stochastic gradient descent to train deep learning models. Accuracy is used as the primary performance metric. The model summary can be seen below:

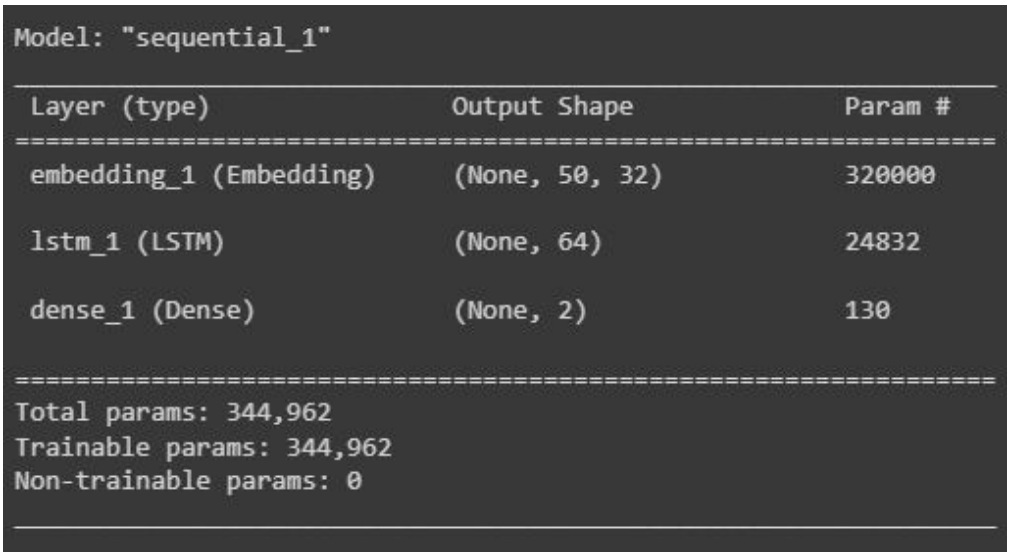

Figure 4.4: Model Summary

4.6 Model Training and Evaluation

The model was trained for 10 epochs on the training dataset consisting of 10,000 samples. During each epoch, the model calculated the loss and accuracy metrics for both training and validation datasets. The training process took around 12 minutes per epoch on average.

The model achieved a peak training accuracy of 85.26% and a peak validation accuracy of 73.56%. The training loss was gradually decreasing during the epochs, indicating that the model was learning and improving its performance. However, the validation loss was increasing from epoch 2 to epoch 10, indicating that the model was overfitting to the training dataset.

The final evaluation of the model was performed on the validation dataset using the trained model, which resulted in a validation loss of 0.5835 and a validation accuracy of 73.56%. These metrics suggest that the model performed reasonably well in classifying the sentiment of the movie reviews. However, further optimization may be required to improve the validation accuracy and to prevent overfitting.

```
Epoch 1/10
10000/10000 [==============================] - 740s 74ms/step - loss: 0.4214 - accuracy: 0.8068 - val_loss: 0.5789 - val_accuracy: 0.7277
Epoch 2/10
10000/10000 [==============================] - 733s 73ms/step - loss: 0.3849 - accuracy: 0.8260 - val_loss: 0.6659 - val_accuracy: 0.6517
Epoch 3/10
10000/10000 [==============================] - 742s 74ms/step - loss: 0.3728 - accuracy: 0.8322 - val_loss: 0.5718 - val_accuracy: 0.7334
Epoch 4/10
10000/10000 [==============================] - 735s 73ms/step - loss: 0.3636 - accuracy: 0.8372 - val_loss: 0.5663 - val_accuracy: 0.7389
Epoch 5/10
10000/10000 [==============================] - 733s 73ms/step - loss: 0.3565 - accuracy: 0.8412 - val_loss: 0.5801 - val_accuracy: 0.7238
Epoch 6/10
10000/10000 [==============================] - 747s 75ms/step - loss: 0.3510 - accuracy: 0.8439 - val_loss: 0.5802 - val_accuracy: 0.7236
Epoch 7/10
10000/10000 [==============================] - 738s 74ms/step - loss: 0.3460 - accuracy: 0.8469 - val_loss: 0.6440 - val_accuracy: 0.7045
Epoch 8/10
10000/10000 [==============================] - 752s 75ms/step - loss: 0.3419 - accuracy: 0.8489 - val_loss: 0.6373 - val_accuracy: 0.6998
Epoch 9/10
10000/10000 [==============================] - 752s 75ms/step - loss: 0.3382 - accuracy: 0.8508 - val_loss: 0.5581 - val_accuracy: 0.7418
Epoch 10/10
10000/10000 [==============================] - 757s 76ms/step - loss: 0.3347 - accuracy: 0.8526 - val_loss: 0.5835 - val_accuracy: 0.7356
10000/10000 [==============================] - 78s 8ms/step - loss: 0.5835 - accuracy: 0.7356
Validation loss: 0.58350670337677
Validation accuracy: 0.7355999946594238
```

Figure 4.5: Model Evaluation

Bibliography

1. Deng, L., Yu, D., *et al.* Deep learning: methods and applications. *Foundations and trends® in signal processing* **7,** 197–387 (2014).
2. Goodfellow, I., Bengio, Y. & Courville, A. *Deep learning* (MIT press, 2016).
3. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks* **61,** 85–117 (2015).
4. Heredia, B., Khoshgoftaar, T. M., Prusa, J. & Crawford, M. *Cross-domain sentiment analysis: An empirical investigation* in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)* (2016), 160–165.
5. Zhang, L., Wang, S. & Liu, B. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **8,** e1253 (2018).
6. Sak, H., Senior, A. & Beaufays, F. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128* (2014).