ARABIC FAKE NEWS DETECTION MODELS

Presented and made by :

ABDELATIF MEKRI
NAHLA YASMINE MIHOUBI
HALIMA NFIDSA
BOUKADER IMADEDDIN
SALAHEDDINE TEFFAHI
ABDELMALEK SLAMANI

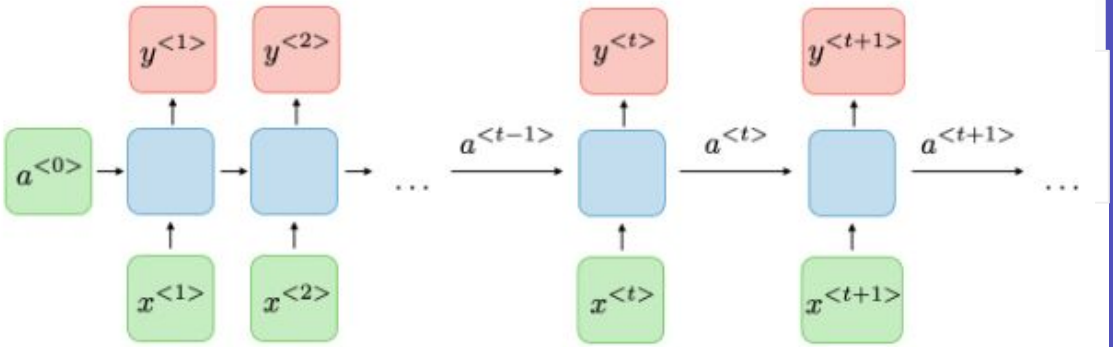
#LSTM

Long-Short-Term-Memory

Recurrent neural network

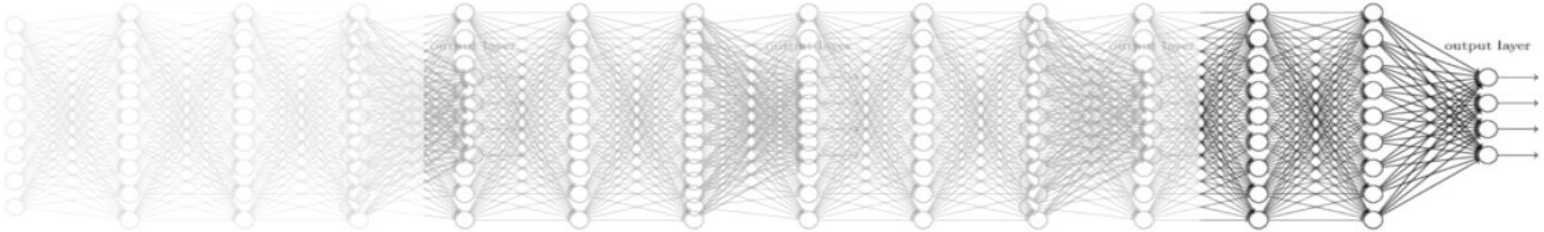
$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

où :



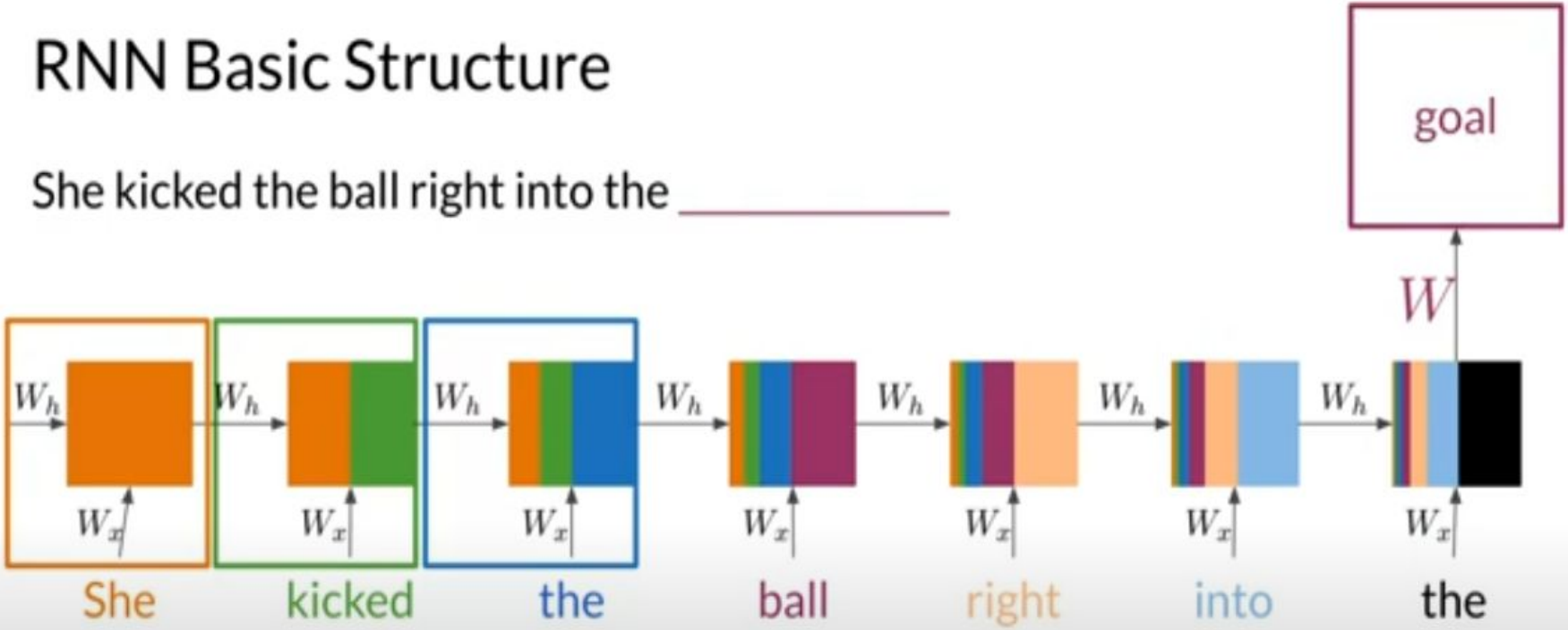
- x_t est le vecteur d'entrée à l'instant t
- h_t est le vecteur de la couche cachée à l'instant t
- y_t est le vecteur de sortie à l'instant t
- W_h est la matrice entre la couche d'entrée et la couche cachée, U_h est la matrice entre la couche cachée et elle-même, W_y est la matrice entre la couche cachée et la couche de sortie, et b est le vecteur de biais (W_h, U_h, W_y, et b sont les paramètres)
- σ_h et σ_y sont les fonctions d'activation respectivement au niveau de la couche cachée et de la couche de sortie.

Vanishing gradient (NN winter2: 1986-2006)



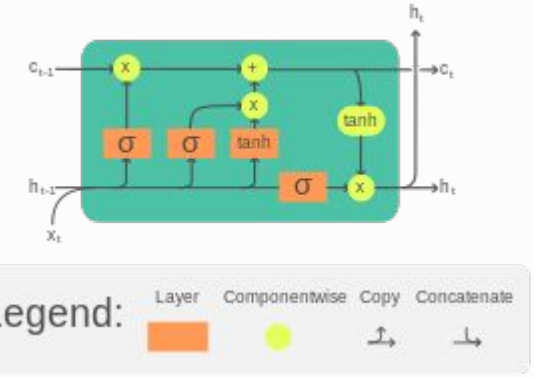
RNN Basic Structure

She kicked the ball right into the _____



What is #LSTM?

Long short-term memory (**LSTM**)[1] is a type of recurrent neural network (RNN) aimed at dealing with the vanishing gradient problem[2] present in traditional RNNs. Its relative insensitivity to gap length is its advantage over other RNNs, hidden Markov models and other sequence learning methods. It aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "**long** short-term memory".[1]



The compact forms of the equations for the forward pass of an LSTM cell with a forget gate are:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

- $x_t \in \mathbb{R}^d$: input vector to the LSTM unit
- $f_t \in (0,1)^h$: forget gate's activation vector
- $i_t \in (0,1)^h$: input/update gate's activation vector
- $o_t \in (0,1)^h$: output gate's activation vector
- $h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit
- $\tilde{c}_t \in (-1,1)^h$: cell input activation vector
- $c_t \in \mathbb{R}^h$: cell state vector
- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training

What is # Fake News?

It refers to false or misleading information presented as news. It can be created and spread intentionally to deceive readers, manipulate opinions, or generate clicks and revenue.

Fake News can has significant #influence

Impact and Cons

- Fake news can have significant social, political, and economic consequences.

Importance of detection

- Protecting the public from misinformation and manipulation.
- Combating the spread of rumors, conspiracy theories, and propaganda.

20 3766 real

16 2141 fake

3 variables

	title	text	label
0	فيديو, هل لديك حساسية طعام؟المدة, 25,18	...يعاني الكثير الشباب منطقة الشرق الأوسط وشمال أ	real
1	...اخر الاخبار اليوم محافظ المنيا ورئيس الجامعة ي	...الدكتور مصطفي عبد النبي رئيس جامعة المنيا والل	fake
2	...مدبولي يتابع الموقف التنفيذي لمشروع تطوير وتنم	...وأكد رئيس الوزراء المشروع القومي الكبير سيتم أ	real
3	...تسرب بسببها فصل بالكامل.. فاطمة رشدى ضربت الطا	...شكرا لقرائتكم خبر تسرب بسببها فصل بالكامل فاطم	fake
4	...سقوط تشكيل عصابي للاتجار بالمخدرات وحيازة الأس	...سقوط تشكيل عصابي للاتجار بالمخدرات وحيازة الأس	real

Preprocessing

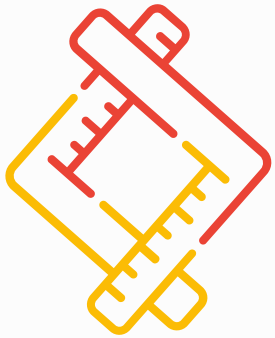
In this section:

- We identify and handle null values.
- Check for and remove duplicate entries in the dataset.
- Limit the length of text data to 200 as a maximum length and 20 as a minimum length.
- Encode categorical labels (e.g., 'fake' as 1, 'real' as 0) into numerical values.

	title	text	label	text_length
0	فيديو, هل لديك حساسية طعام؟المدة, 25,18	يعاني الكثير الشباب منطقة الشرق الأوسط وشمال أ...	0	200
1	اخر الاخبار اليوم محافظ المنيا ورئيس الجامعة ي...	الدكتور مصطفي عبد النبي رئيس جامعة المنيا والل...	1	200
2	مدبولي يتابع الموقف التنفيذي لمشروع تطوير وتنم...	وأكد رئيس الوزراء المشروع القومي الكبير سيتم إ...	0	200
3	تسرب بسببها فصل بالكامل.. فاطمة رشدى ضربت الطا...	شكرا لقرائتكم خبر تسرب بسببها فصل بالكامل فاطم...	1	200
4	سقوط تشكيل عصابي للاتجار بالمخدرات وحيازة الأس...	سقوط تشكيل عصابي للاتجار بالمخدرات وحيازة الأس...	0	200

Code now

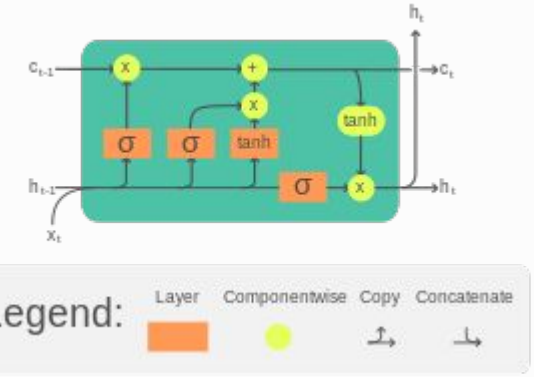
#BI-LSTM



Bidirectional Long-Short-Term-Memory

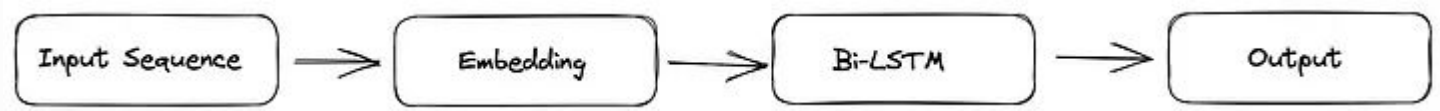
What is #LSTM?

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) aimed at dealing with the vanishing gradient problem present in traditional RNNs.
Its relative insensitivity to gap length is its advantage over other RNNs, hidden Markov models and other sequence learning methods. It aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory".



What is #BiLSTM?

A bidirectional LSTM (BiLSTM) layer is an RNN layer that learns bidirectional long-term dependencies between time steps of time-series or sequence data. These dependencies can be useful when you want the RNN to learn from the complete time series at each time step.



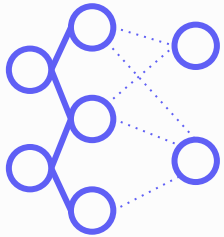
Why #BiLSTM ?

Sequential problem

Learning problem is sequential like
Predicting next sequences

Solving Short-term Memory

carrying information from earlier time steps to later ones.

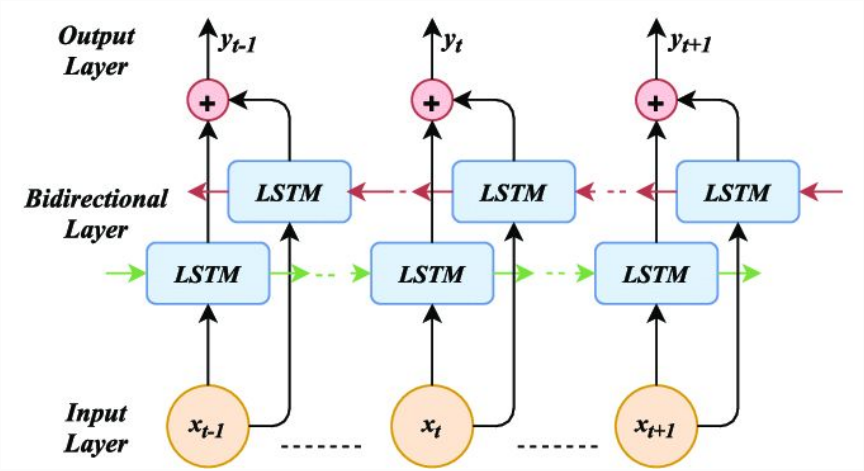


RNN Architecture

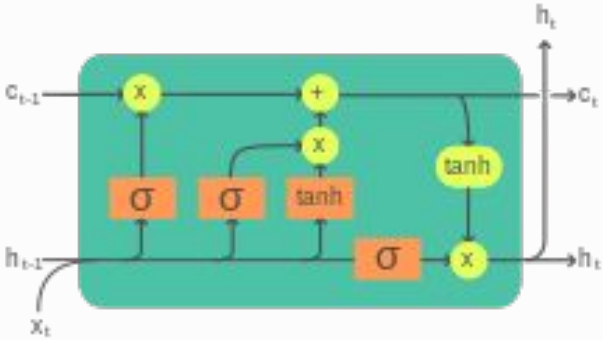
Learning problem is sequential like
Predicting next sequences

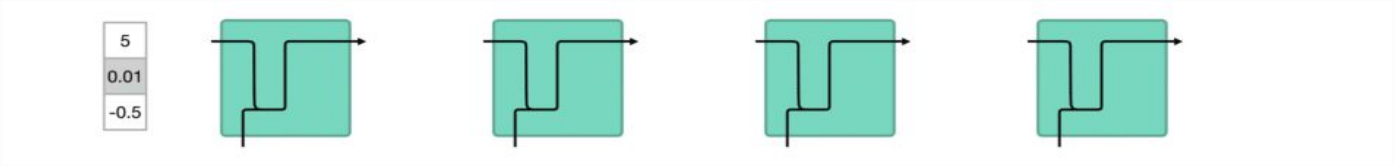
BiLSTM PROCESSING

Unlike traditional RNNs that process input sequences in only one direction (either forward or backward), Bi-LSTM processes the sequence in both directions simultaneously. It consists of two LSTM layers: one processing the sequence in the forward direction and the other in the backward direction. Each layer maintains its own hidden states and memory cells.

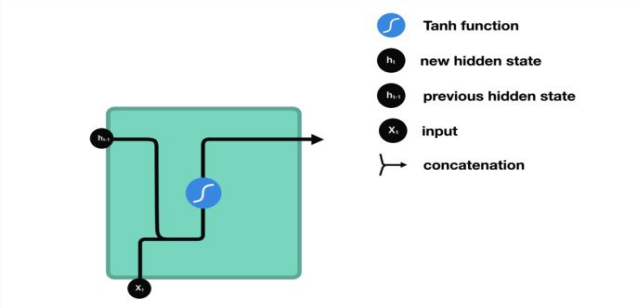


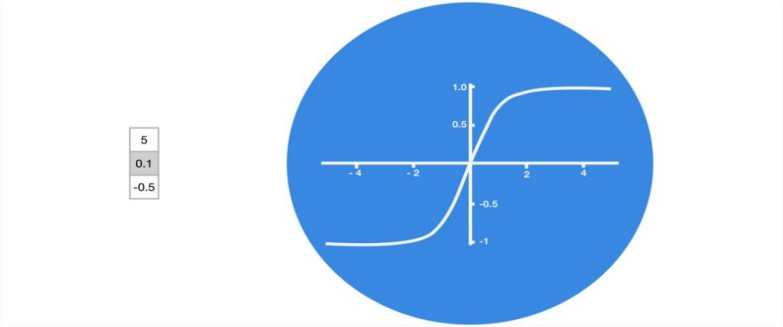
HOW DOES LSTM UNIT WORK ?

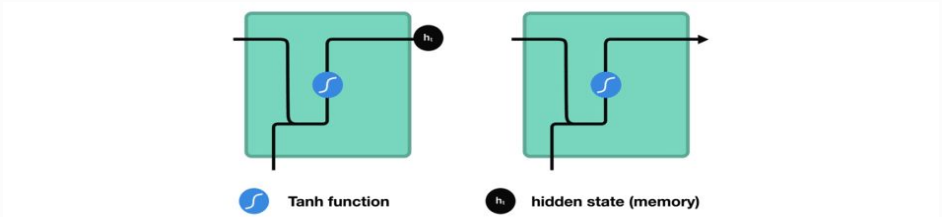


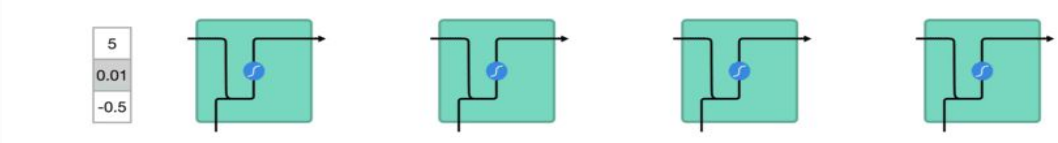


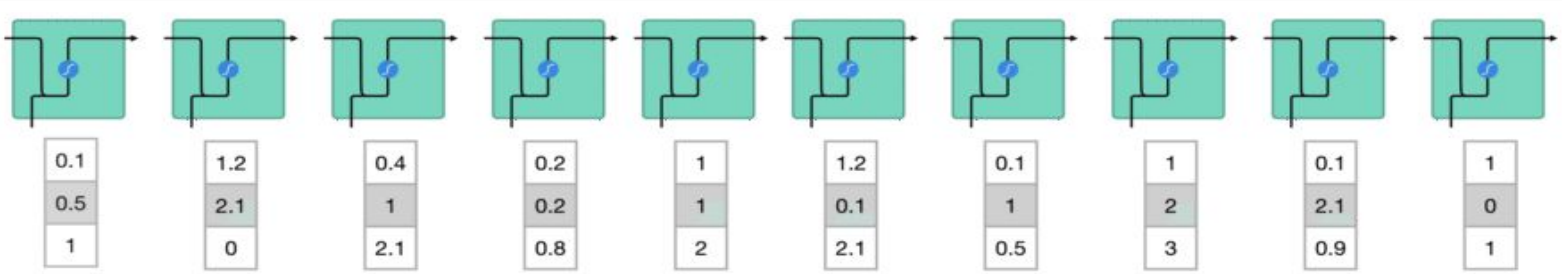
WITH TANH FUNCTION

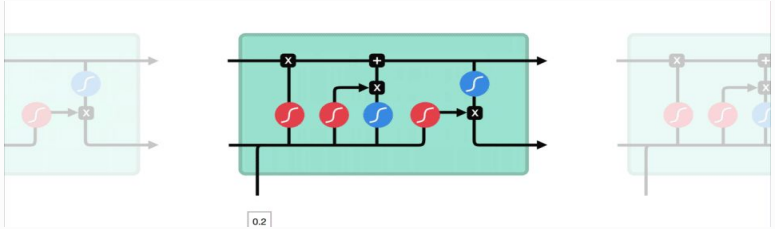


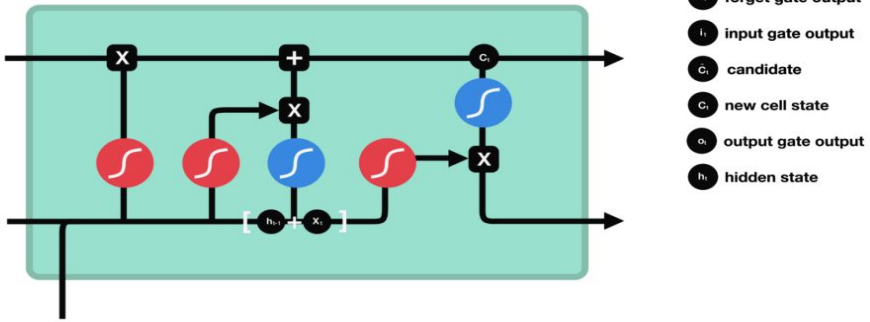







0.1
0.5
1
1.2
2.1
0
0.4
1
2.1
0.2
0.2
0.8
1
1
2
1.2
0.1
2.1
0.1
1
0.5
1
2
3
0.1
2.1
0.9
1
0
1


0.2


previous cell state
forget gate output
input gate output
candidate
new cell state
output gate output
hidden state

Data Preparation

- Convert labels to numerical format
- Split into training and testing sets
- Tokenization and Padding

Set up Hyper-parameters & Model initialisation

- Calling the model from Keras
- Adding Bidirectional LSTM layers
- Adding Dense layers

Running the Model

Results & Evaluation

Code explanation

We run the code on the Kaggle platform for GPU availability and sharing ability.

- Open code on VSC

Intro Hook Explore Explain Apply Share Evaluate Expand

#BERT

Bidirectional Encoder Representations from Transformers

#Our Process

01 Preprocessing

02 BERT

03 Tokenization

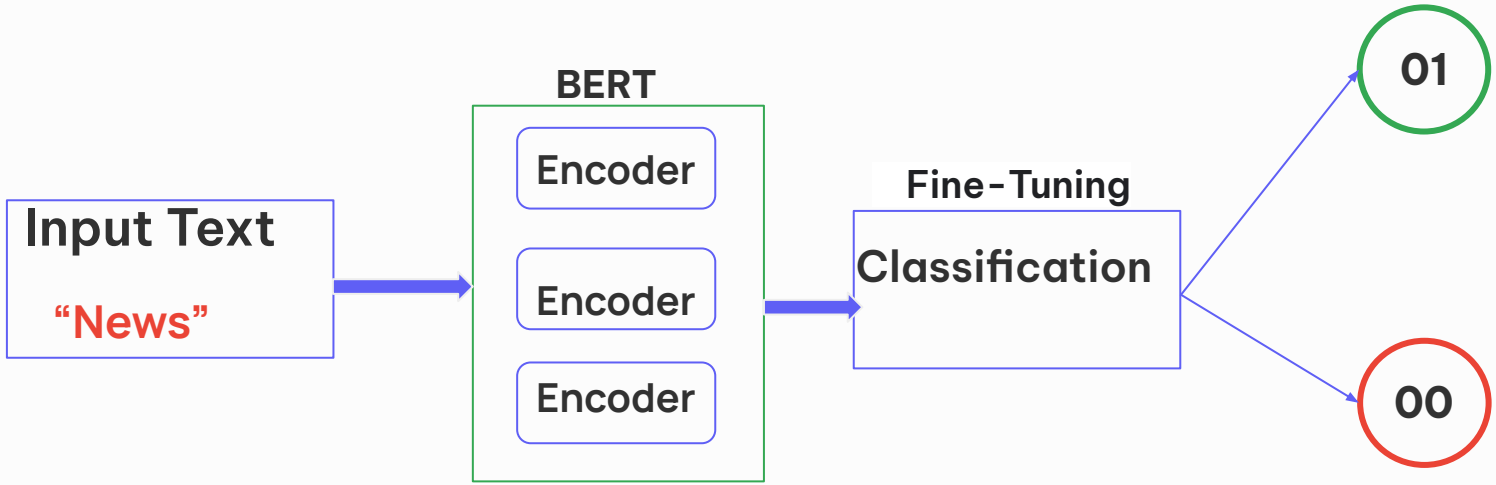
04 Splits & loaders

05 Optimizer

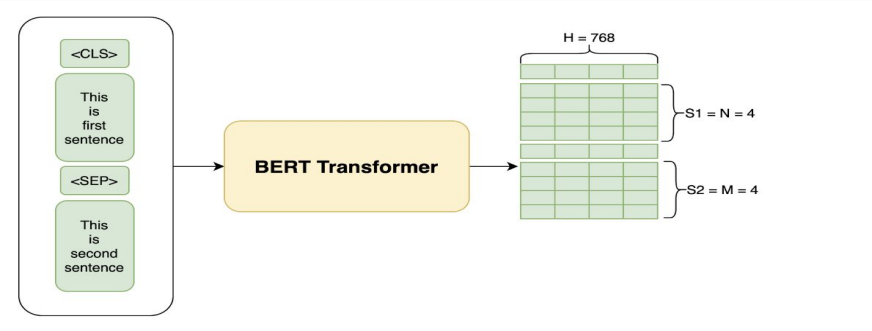
06 Evaluate

07 Conclusion

#Our Process


Input Text
"News"
BERT
Encoder
Encoder
Encoder
Fine-Tuning
Classification
01
00

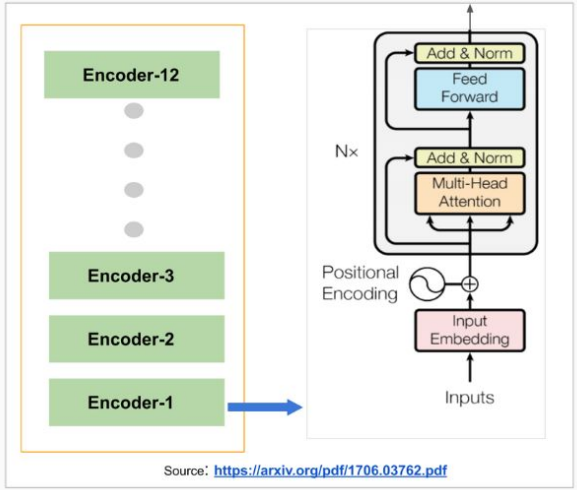
What is #BERT?



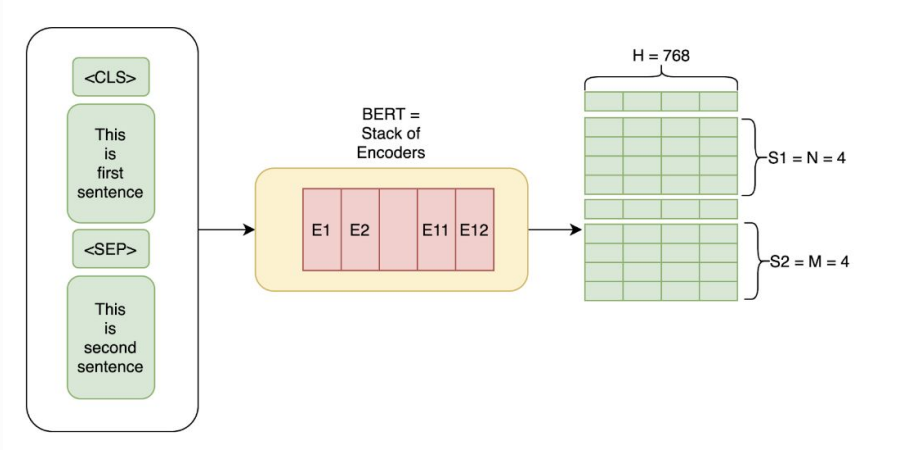
BERT stands for **Bidirectional Encoder Representation of Transformers.** It is a deep learning based unsupervised language representation model developed by **researchers at Google AI** Language. It is the first deeply-bidirectional unsupervised language model. The language models, until BERT, learnt from text sequences in either left-to-right or combined left-to-right and right-to-left contexts. Thus they were either not bidirectional or not bidirectional in all layers.

BERT Architecture

The diagram shows a 12 layered BERT model(BERT-Base version). Note that each Transformer is based on the Attention Model.



BERT Base: Number of Layers L=12, Size of the hidden layer, H=768, and Self-attention heads, A=12 with Total Parameters=110M



BERT Tokenization

```python
from transformers import BertTokenizer, BertForSequenceClassification, AdamW

# Load the Arabic BERT tokenizer
tokenizer = BertTokenizer.from_pretrained("aubmindlab/bert-base-arabertv2")

# Tokenize training texts
train_encodings = tokenizer(train_texts.tolist(), truncation=True, padding=True, max_length=200, return_tensors="pt")

# Tokenize test texts
test_encodings = tokenizer(test_texts.tolist(), truncation=True, padding=True, max_length=200, return_tensors="pt")
```

Data Loader

```
# DataLoader
train_loader = DataLoader(train_dataset, sampler=RandomSampler(train_dataset), batch_size=16)
test_loader = DataLoader(test_dataset, sampler=SequentialSampler(test_dataset), batch_size=16)
```

Model

```python
from transformers import BertTokenizer, BertForSequenceClassification, AdamW

# Load the Arabic BERT tokenizer
tokenizer = BertTokenizer.from_pretrained("aubmindlab/bert-base-arabertv2")

# Load the Arabic BERT model for sequence classification
model = BertForSequenceClassification.from_pretrained("aubmindlab/bert-base-arabertv2")

# Set up the optimizer
optimizer = AdamW(model.parameters(), lr=2e-5)
```

Training

```
for epoch in range(epochs):
    model.train()
    total_loss = 0
    for batch in train_loader:
        batch = tuple(t.to(device) for t in batch)
        b_input_ids, b_attention_mask, b_labels = batch
        optimizer.zero_grad()

        outputs = model(b_input_ids, attention_mask=b_attention_mask, labels=b_labels)

        loss = outputs.loss
        total_loss += loss.item()
        loss.backward()
        optimizer.step()
        scheduler.step()
    print(f"Epoch {epoch+1}/{epochs} Loss: {total_loss/len(train_loader)}")

model.eval()
predictions, true_labels = [], []
```

Evaluation

Model “ **Bert-base-uncased**”:

Accurancy: 0.7275

```
Precision: [0.7690957  0.67247387]
Recall: [0.75647668 0.68764846]
F1 Score: [0.762734   0.67997651]
Support: [1158  842]
```

```
Confusion Matrix:
[[876 282]
 [263 579]]
```

Evaluate

Model “ Bert-base-arabertv2”:

Accurancy: 0.81

	precision	recall	f1-score	support
0	0.82	0.84	0.83	2193
1	0.80	0.77	0.78	1807

```
Confusion Matrix:
[[1836  357]
 [ 410 1397]]
```

Awards

ON ACCURACY LEVEL