ALAMEIN INTERNATIONAL UNIVERSITY
جامعة العلمين الدولية

Arabic News Detection

Aya Tamer Ginidy
Shahenda Adel Abdelrahmen
Abdulrahman Moahmed Eltahan

Abstract:

- 1.problem:

With the rapid increase in digital news, it's important to classify articles accurately to help users find information easily. This is especially challenging for Arabic news due to the language's complexity. The aim is to develop a system that can accurately classify Arabic news articles into specific categories.

- 2.goal:

In this project will use CNN, LSTM, BERT, and AraBERT models to build a high-performing Arabic news classification system. We will use the Akhbarona-News-Classification repository as a starting point and the SANAD dataset for training and evaluation. The project includes several key tasks to ensure thorough development and optimization.

- 1.dataset
- 2.word embedding
- 3.model selections
- 4.extensions models

Baseline

Baseline

The dataset named "sanad" is an Arabic text dataset

Dataset Structure:

- The dataset is divided into a 'train' split.
- The 'train' split contains 141,807 rows.

- **Columns**:
  - **Article**: This column contains the text of the articles in Arabic.
  - **label**: This column contains numerical labels associated with each article, presumably representing different categories or classes.

Labels	Class number	percentage
Finance	1	27.8%
Sports	6	20.8%
politics	2	14.4%
Medical	4	13.7%
Tech	0	13.2%
Religion	3	5.3%
Culture	5	4.8%

Baseline

- Text Processing :

Convert text to lowercase.

Remove punctuations from text:
Example:"مرحبا، بالعالم!" becomes "مرحبا بالعالم"

Remove references and hashtags from text:
Example:" مرحبا@؟كيف حالك ، احمد" becomes "مرحبا ، كيف حالك؟"

Tokenize the words using nltk word tokenizer and remove the stop words using nltk package's Arabic stop words

Model Selections

Naive Bayes: is a simple probabilistic classifier based on Bayes' theorem with strong (naive) independence assumptions between features. Its architecture consists of a probabilistic model that is easy to train and fast to predict. However, its main disadvantage:

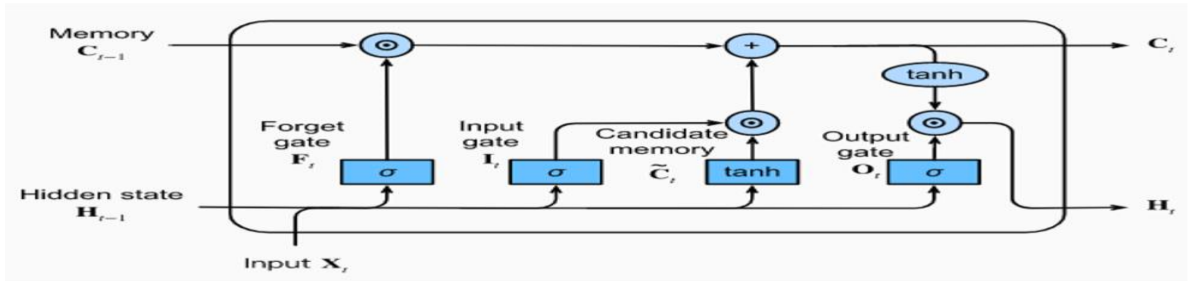
- the assumption of feature independence, which can lead to suboptimal performance when features are highly correlated.

RNN: are a class of neural networks designed for sequential data processing. They work by maintaining a hidden state that is updated at each time step, allowing the network to capture temporal dependencies and patterns in sequences like text or time series. However, RNNs face challenges:

- vanishing and exploding gradients, which can hinder them
- ability to learn long-term dependencies effectively.

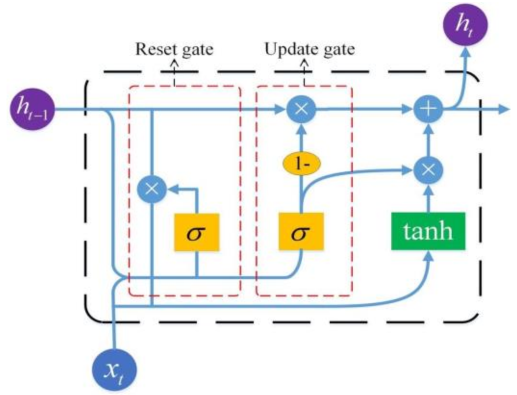
Long Short-Term Memory (LSTM) networks: are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data .LSTMs use memory cells and gating mechanisms (input, forget, and output gates) to control the flow of information, allowing them to maintain context over long sequences and mitigate the vanishing gradient problem, it's Challenge :

- can be slower during inference compared to simpler models
- LSTMs require more memory for storing the network parameters and intermediate states compared to simpler model



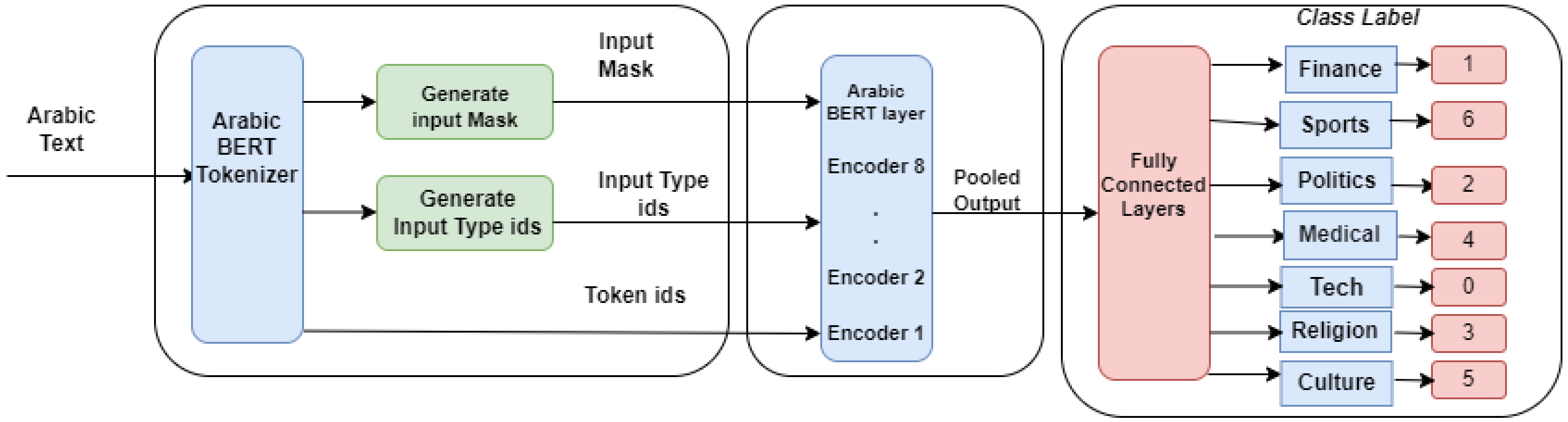
GRU: is similar to LSTM , but with less Nodes and different Gates .It uses gating mechanisms to control the flow of information, combining the input and forget gates into a single update gate and the output gate into a reset gate . GRUs may still struggle :

- with very long sequences and can be less effective than LSTMs in some tasks.
- require careful tuning of hyperparameters to achieve optimal performance.



BERT (Bidirectional Encoder Representations from Transformers): is a state-of-the-art NLP model that reads text bidirectionally, capturing context from both directions to improve understanding. Its architecture includes Transformer encoders that use self-attention mechanisms, enabling natural language comprehension ,it's challenges :

- can be slower in inference compared to simpler models, making it less suitable for real-time applications where speed is crucial.
- its performance is sensitive to fine-tuning settings; small changes in hyperparameters, training duration, or dataset characteristics can lead to significant differences in performance.



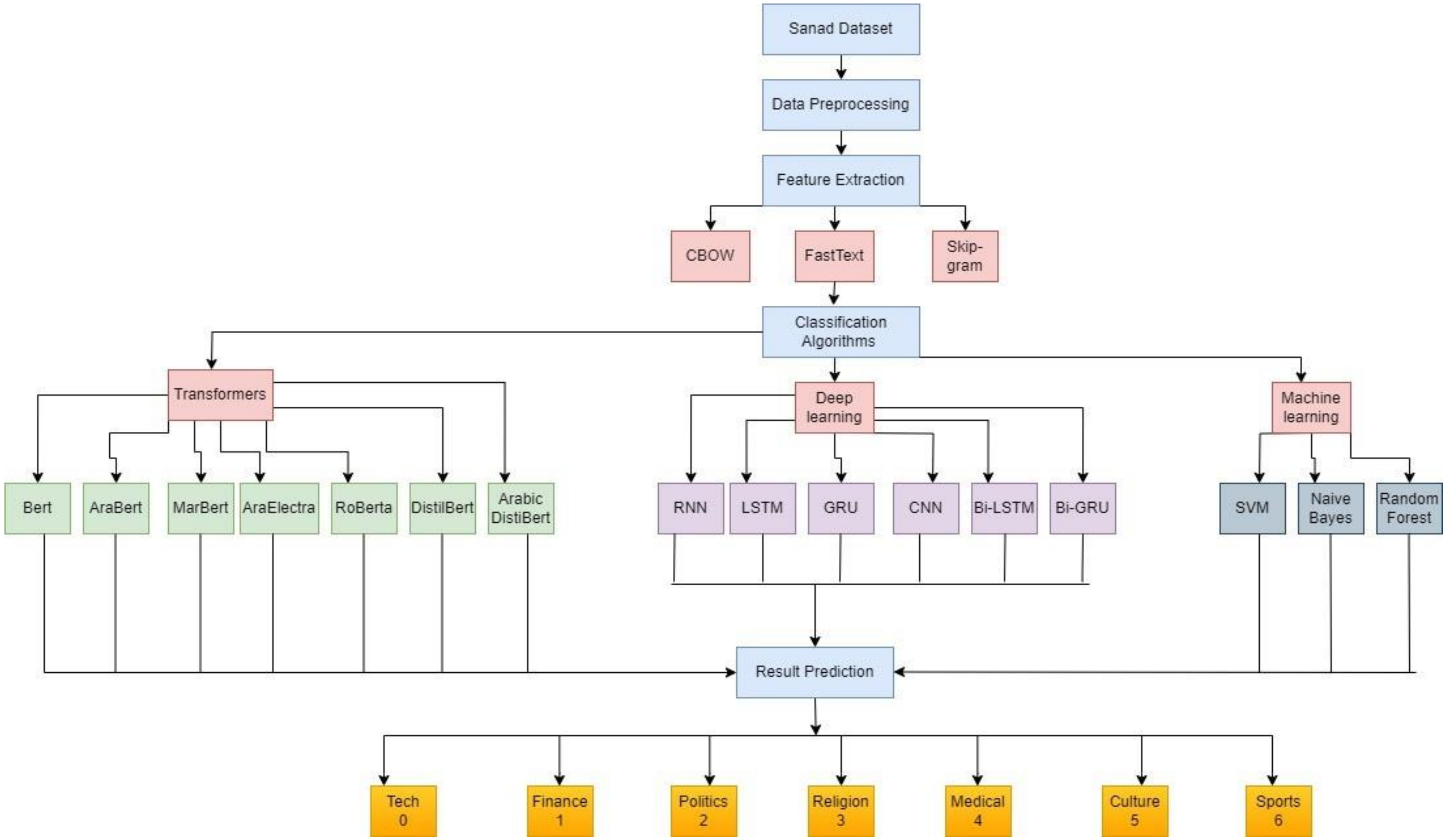
RoBERTa (Robustly optimized BERT approach): is an advanced transformer model based on BERT, designed to improve performance by optimizing the training process. It removes the next sentence prediction objective, trains on larger mini-batches, longer sequences, and more data. The architecture remains similar to BERT Chanellges :

- large model size and increased complexity can result in slower inference speeds
- The improvements come at the cost of increased computational resources, requiring more GPU/TPU resources and longer training times compared to BERT

DistilBERT: is a smaller, faster, and lighter version of BERT, achieving 97% of BERT's performance while being 60% faster and 40% smaller. It works by using knowledge distillation during training,The architecture remains similar to BERT Challenges:

- it may struggle with complex language nuances and domain-specific tasks due to its reduced size.
- its training and fine-tuning still require significant computational resources and it might not capture as rich contextual information as larger models.

Extensions


Sanad Dataset
Data Preprocessing
Feature Extraction
CBOW
FastText
Skip-gram
Classification Algorithms
Transformers
Deep learning
Machine learning
Bert
AraBert
MarBert
AraElectra
RoBerta
DistilBert
Arabic DistiBert
RNN
LSTM
GRU
CNN
Bi-LSTM
Bi-GRU
SVM
Naive Bayes
Random Forest
Result Prediction
Tech 0
Finance 1
Politics 2
Religion 3
Medical 4
Culture 5
Sports 6

Processing :

In our preprocessing, we removed the step to convert text to lowercase. especially since we are working with an Arabic dataset.

Normalize Arabic text:
- Original: "گتاب, لغة, شئ, مؤمن, إأآا بارىء"
- After normalization: "كتاب, لغه, شء, ءمن, اااا باريء"

Remove diacritics
- Original: "اللُّغَة العَرَبِيَّة تُكْتَب بِحُرُوف مِعْجَمِيَّة"
- After removing diacritics: "اللغه العربيه تكتب بحروف معجميه"

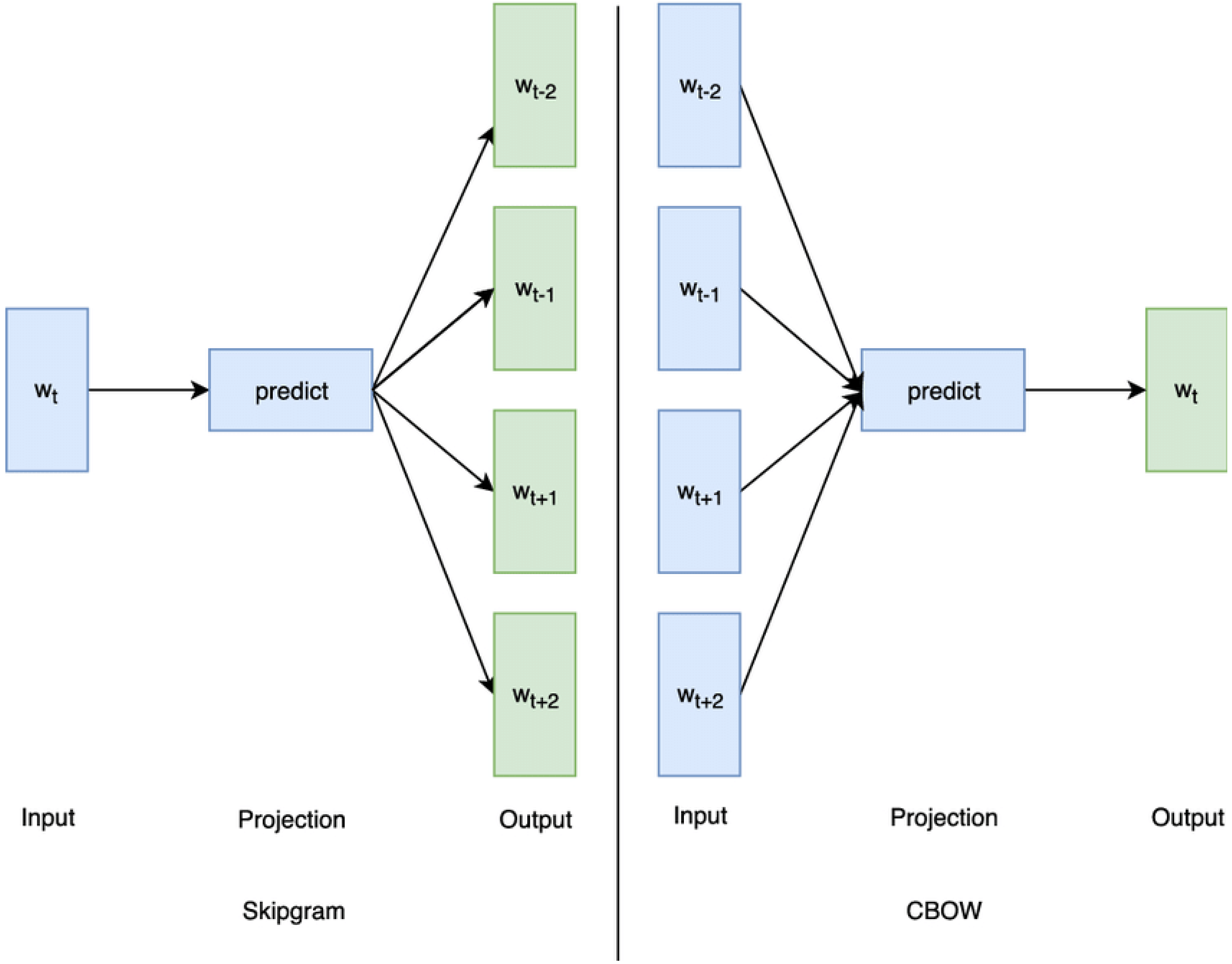
Word Embedding

1. **Skip-Gram:** aims to predict the surrounding words (context) given a target word. This allows the model to learn about the relationships between words based on their context. For example,
   if the target word is "cat," the context might include words like "meow," "pet," or "animal."

2. **CBOW (Continuous Bag of Words):** tries to predict a target word using the words around it within a given window Size, adjusts word vectors so that the context words' vectors combine to predict the target word. For example,
   If the sentence is "The cat sat on the mat," and target word is "sat," it uses surrounding words to predict "sat."

3. **FastText:** learns word embeddings by considering both whole words and subword (character-level) information, making it effective at capturing semantic meanings, especially in morphologically rich languages. For example,
   If the word "unhappiness."it looks at parts of the word like "un," "happy," and "ness."

We tries all of them on LSTM Model


w_t
predict
w_{t-2}
w_{t-1}
w_{t+1}
w_{t+2}
Input
Projection
Output
Skipgram
w_{t-2}
w_{t-1}
w_{t+1}
w_{t+2}
predict
w_t
Input
Projection
Output
CBOW

Models:

Support Vector Machine (SVM): is a powerful classifier that works by finding the optimal hyperplane to separate different classes in a high-dimensional feature space. This makes it particularly effective for text classification, where data is often sparse, it's Challenge :

- can be computationally expensive, especially with large datasets or high-dimensional feature spaces
- can struggle with imbalanced datasets
- Memory Usage

Random Forest: method that builds multiple decision trees and combines their results to improve accuracy and reduce overfitting, it works by randomly selecting features to train multiple decision trees. Each tree makes a class prediction, and the final prediction is based on the majority vote across all the trees, it's Challenge :

- Overfitting with Too Many Trees
- struggle with imbalanced datasets
- Tuning the hyperparameters can be complex and may require extensive experimentation to find the optimal settings

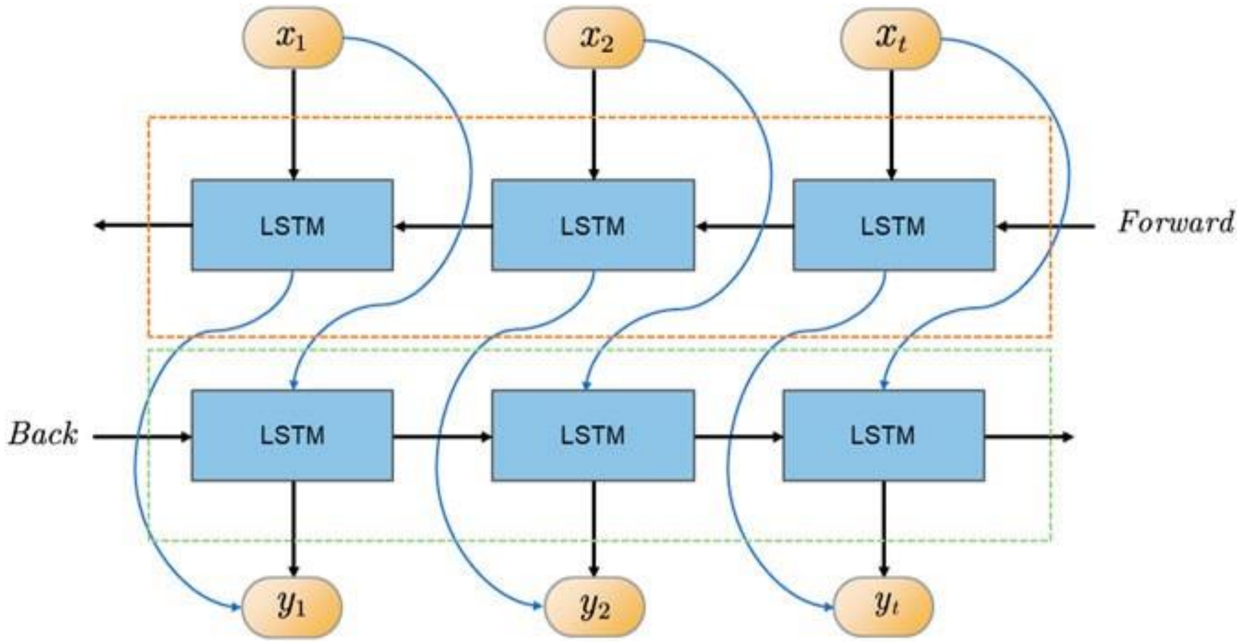
Convolutional Neural Networks (CNNs): are a class of deep learning models designed primarily for image processing. They consist of convolutional layers that automatically and adaptively learn spatial hierarchies of features from input images, followed by pooling layers for down-sampling, and fully connected layers for classification, it's Challenge :

- Require large Dataset can be prone to overfitting if not properly regularized.
- their performance can degrade with variations in image orientation and scale.

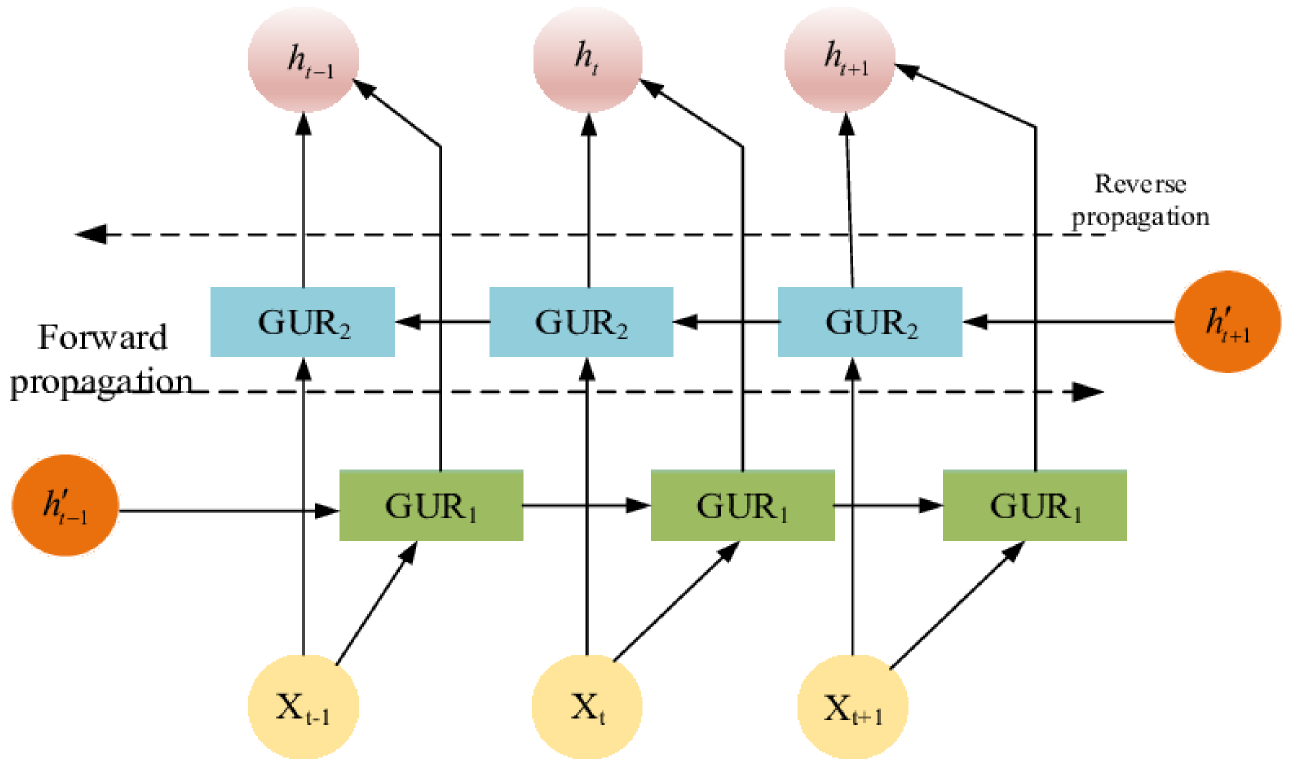
BiLSTM (Bidirectional Long Short-Term Memory) and BiGRU (Bidirectional Gated Recurrent Unit):

are advanced RNN architectures that process input sequences in both forward and backward directions, capturing context from both past and future states. This bidirectional approach enhances understanding of the sequence, especially for languages like Arabic with complex morphology. However, these models are computationally intensive and require substantial memory, making them less efficient for real-time applications or deployment on resource-constrained devices. Additionally, they can be prone to overfitting on smaller datasets, necessitating careful regularization and tuning.

BiLSTM



BiGRU



AraBERT: is a pre-trained BERT model tailored for Arabic text, leveraging the transformer architecture to provide robust contextual understanding. However, it requires significant computational resources for training and fine-tuning, and its performance may be limited by the quality of the training corpus.

AraELECTRA: uses the ELECTRA architecture to improve efficiency by training a discriminator model to distinguish real tokens from generated ones. It is computationally intensive and can overfit on specific domains, limiting its generalizability.

MARBERT: is a BERT variant designed specifically for Arabic dialects, capturing nuances across various regional languages. Despite its strengths, it faces challenges with dialect diversity and resource-intensive training requirements.

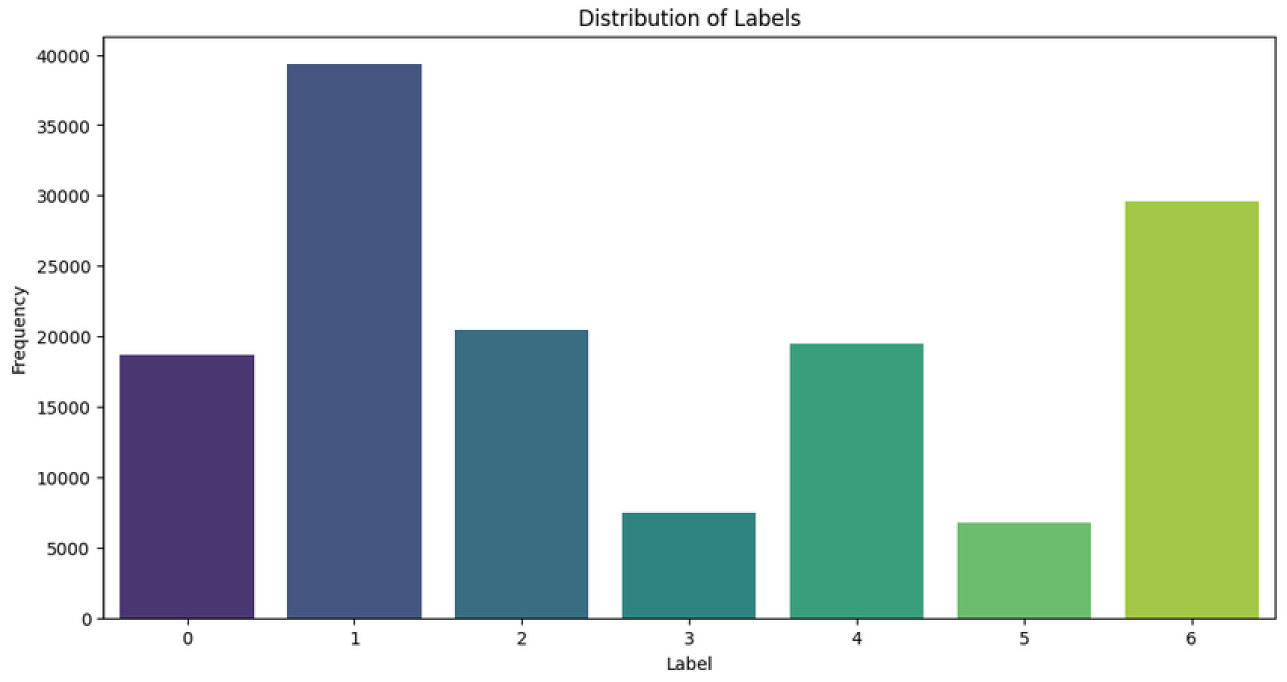
Arabic DistilBERT: version of BERT (Bidirectional Encoder Representations from Transformers) fine-tuned for the Arabic language, it offers a smaller, faster, and more efficient alternative to BERT, while maintaining about 97% of its performance, it's Challenge :

- slight drop in accuracy compared to the full BERT model, particularly in tasks that involve complex or nuanced language
- With fewer layers than BERT, Arabic DistilBERT may not capture deep contextual relationships as effectively

Results

analyzing the data

After analyzing the data, we found that it is unbalanced.



The solution is to perform data augmentation.
After that, we trained the data on a neural model for 3 epochs, and the accuracy is 0.9318.

Hyperparameters for neural network architectures

Model	Epochs	Batch_size	Units	Kernal size	Activation function	Optimizer	Max-pooling	Early Stopping	Dropout
RNN	10	32	128	------	Softmax(output layer) Relu(hidden layers)	Adam	-------	has	0.5
CNN	10	32	128	5	Softmax(output layer) Relu(hidden layers)	Adam	2	has	0.5
LSTM	10	32	128	-----	Softmax(Dense layer) Relu(hidden layers)	Adam	-------	has	0.5
GRU	10	32	128	------	Softmax(Dense layer) Relu(hidden layers)	Adam	-------	has	0.5
BiLSTM	10	32	128	------	Softmax(Dense layer) Relu(hidden layers)	Adam	-------	has	0.5
BiGRU	10	32	128	------	Softmax(Dense layer) Relu(hidden layers)	Adam	-------	has	0.5

Hyperparameters for Transformers

Model	Epochs	Batch_size	Max_ leangh	Weight decay	Learning rate
BERT	3	16	128	0.01	5e-5
DistilBERT	3	16	128	0.01	2e-5
AraBERT	3	16	128	0.01	2e-5
RoBERTa	3	16	128	0.01	5e-5
MaraBERT	3	16	128	0.01	5e-5
AraElectra	3	16	128	0.01	2e-5
Arabic DistilBERT	3	16	128	0.01	2e-5

Model	var_smoo thing	max_feat ures	test_size	kernal
Naïve Bayes	1e-9	None	0.2	None
SVM	None	None	0.2	Linear
Random Forest	None	has	0.2	None

Hyperparameters for different types of machine learning algorithms

Results:

1.Baseline results

Model	Accuracy
RNN	0.578
LSTM	0.938
GRU	0.937
Naïve Bayes	0.83
BERT	0.90
DistilBERT	0.89
RoBERTa	0.78

2.Word Embedding Tries on LSTM

Types	Accuracy
Skip_gram	0.8799422979354858
CBow	0.9116666913032532
FastText	0.9284523725509644

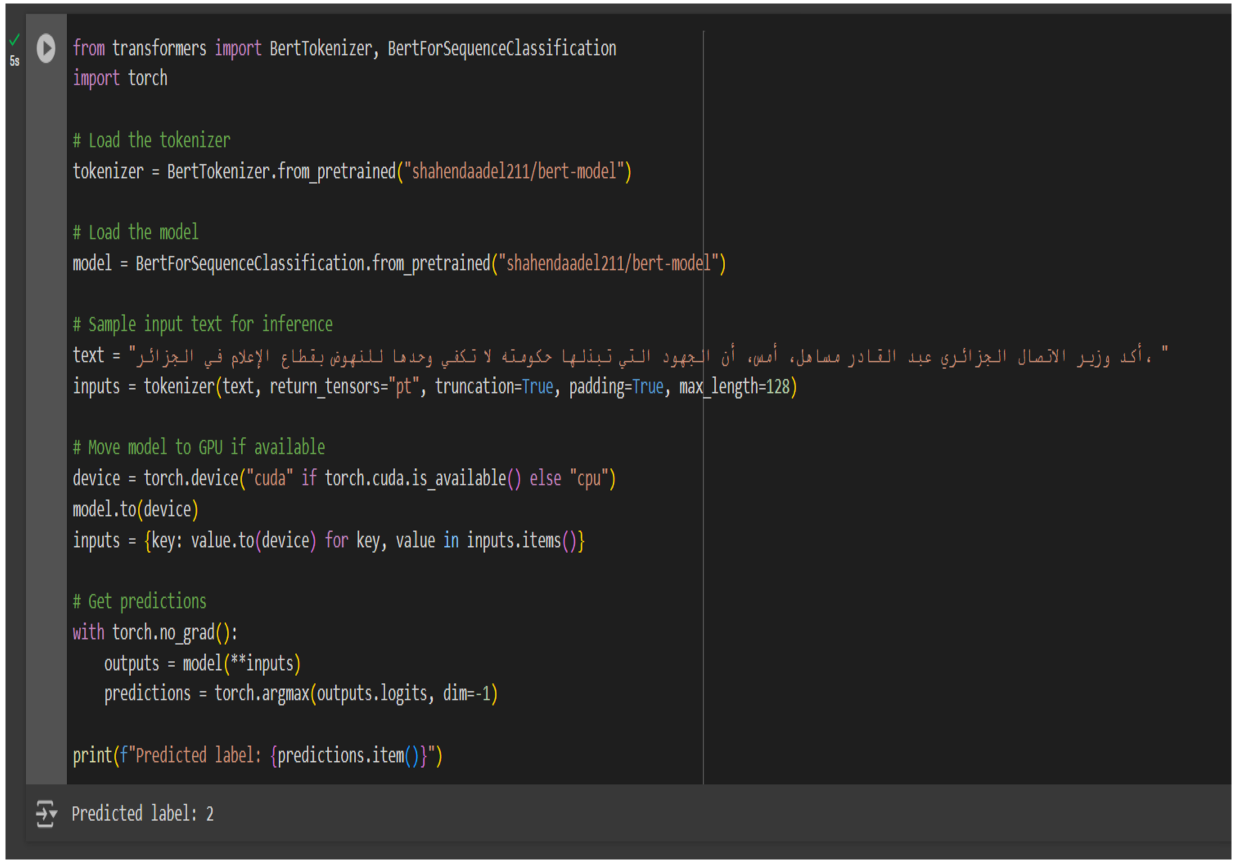
3(a).Our Extensions: with the modified preprocessing function and FastText

Model	Test Loss	Accuracy
RNN	1.0792728662490845	0.6253571510314941
CNN	0.2160913646221161	0.932023823261261
LSTM	0.23052658140659332	0.9284523725509644
GRU	0.19021539390087128	0.9438095092773438
BiLSTM	0.23437868058681488	0.9258333444595337
BiGRU	0.19409875571727753	0.9358333349227905
Naïve bayes	-------------	0.8551190476190477
SVM	-------------	0.934047619047619
Random Forest	-------------	0.9195238095238095

3(b).Our Extensions: with the modified preprocessing function

Model	Training Loss	Validation Loss	Accuracy	F1_score	Precision	Recall
BERT	0.435000	0.574921	0.83	0.814938	0.814551	0.815714
DistilBERT	0.554700	0.779104	0.77	0.729026	0.729483	0.730000
AraElectra	0.142900	0.163163	0.94	0.944977	0.954711	0.955820
RoBERTa	0.848800	1.003361	0.64	0.632415	0.639512	0.641429
AraBERT	0.245900	0.237062	0.95	0.945724	0.946714	0.945714
MaraBERT	0.110300	0.289157	0.94	0.941540	0.942433	0.941429
Arabic DistilBERT	0.125700	0.260642	0.94	0.926858	0.929532	0.927142

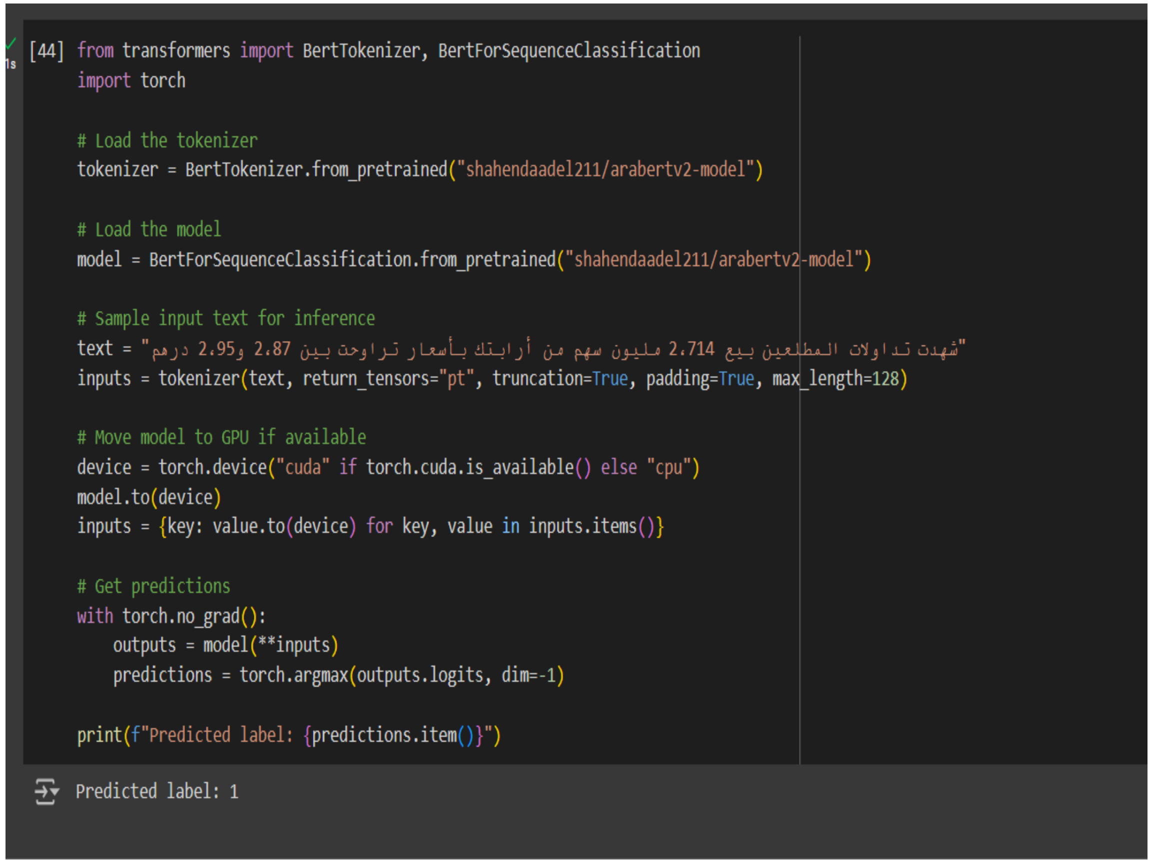
Some Prediction examples:



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

# Load the tokenizer
tokenizer = BertTokenizer.from_pretrained("shahendaadel211/bert-model")

# Load the model
model = BertForSequenceClassification.from_pretrained("shahendaadel211/bert-model")

# Sample input text for inference
text = "أكد وزير الاتصال الجزائري عبد القادر مساهل، أمس، أن الجهود التي تبذلها حكومته لا تكفي وحدها للنهوض بقطاع الإعلام في الجزائر، "
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
inputs = {key: value.to(device) for key, value in inputs.items()}

# Get predictions
with torch.no_grad():
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=-1)

print(f"Predicted label: {predictions.item()}")
```

```
Predicted label: 2
```

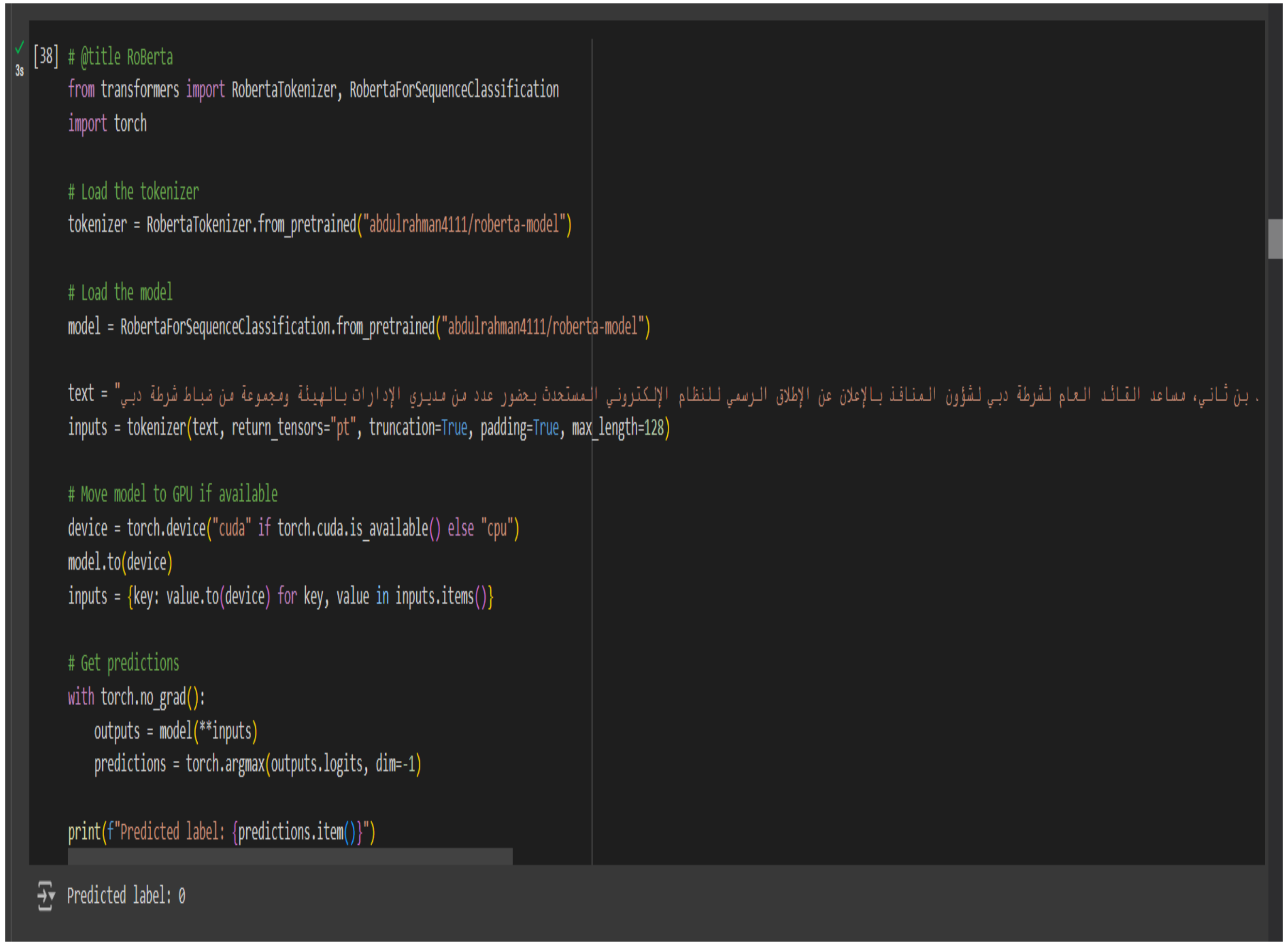
BERT



```python
[44] from transformers import BertTokenizer, BertForSequenceClassification
     import torch

     # Load the tokenizer
     tokenizer = BertTokenizer.from_pretrained("shahendaadel211/arabertv2-model")

     # Load the model
     model = BertForSequenceClassification.from_pretrained("shahendaadel211/arabertv2-model")

     # Sample input text for inference
     text = "شهدت تداولات المطلعين بيع 2،714 مليون سهم من أرابتك بأسعار تراوحت بين 2،87 و2،95 درهم"
     inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)

     # Move model to GPU if available
     device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
     model.to(device)
     inputs = {key: value.to(device) for key, value in inputs.items()}

     # Get predictions
     with torch.no_grad():
         outputs = model(**inputs)
         predictions = torch.argmax(outputs.logits, dim=-1)

     print(f"Predicted label: {predictions.item()}")
```

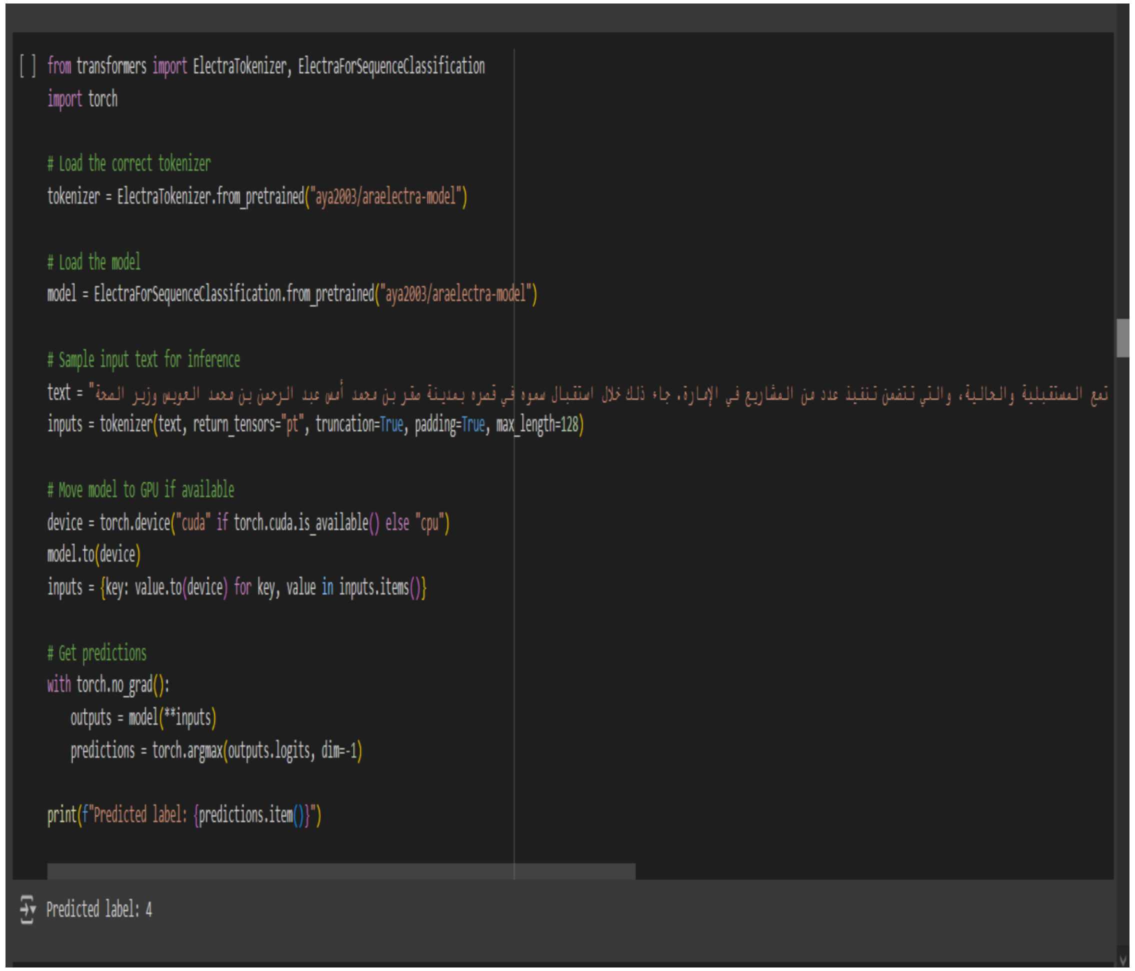
```
Predicted label: 1
```

AraBERT



```
[38] # @title RoBerta
from transformers import RobertaTokenizer, RobertaForSequenceClassification
import torch

# Load the tokenizer
tokenizer = RobertaTokenizer.from_pretrained("abdulrahman4111/roberta-model")

# Load the model
model = RobertaForSequenceClassification.from_pretrained("abdulrahman4111/roberta-model")

text = "بن ثاني، مساعد القائد العام لشرطة دبي لشؤون المنافذ بالإعلان عن الإطلاق الرسمي للنظام الإلكتروني المستحدث بحضور عدد من مديري الإدارات بالهيئة ومجموعة من ضباط شرطة دبي .
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
inputs = {key: value.to(device) for key, value in inputs.items()}

# Get predictions
with torch.no_grad():
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=-1)

print(f"Predicted label: {predictions.item()}")
```

```
Predicted label: 0
```

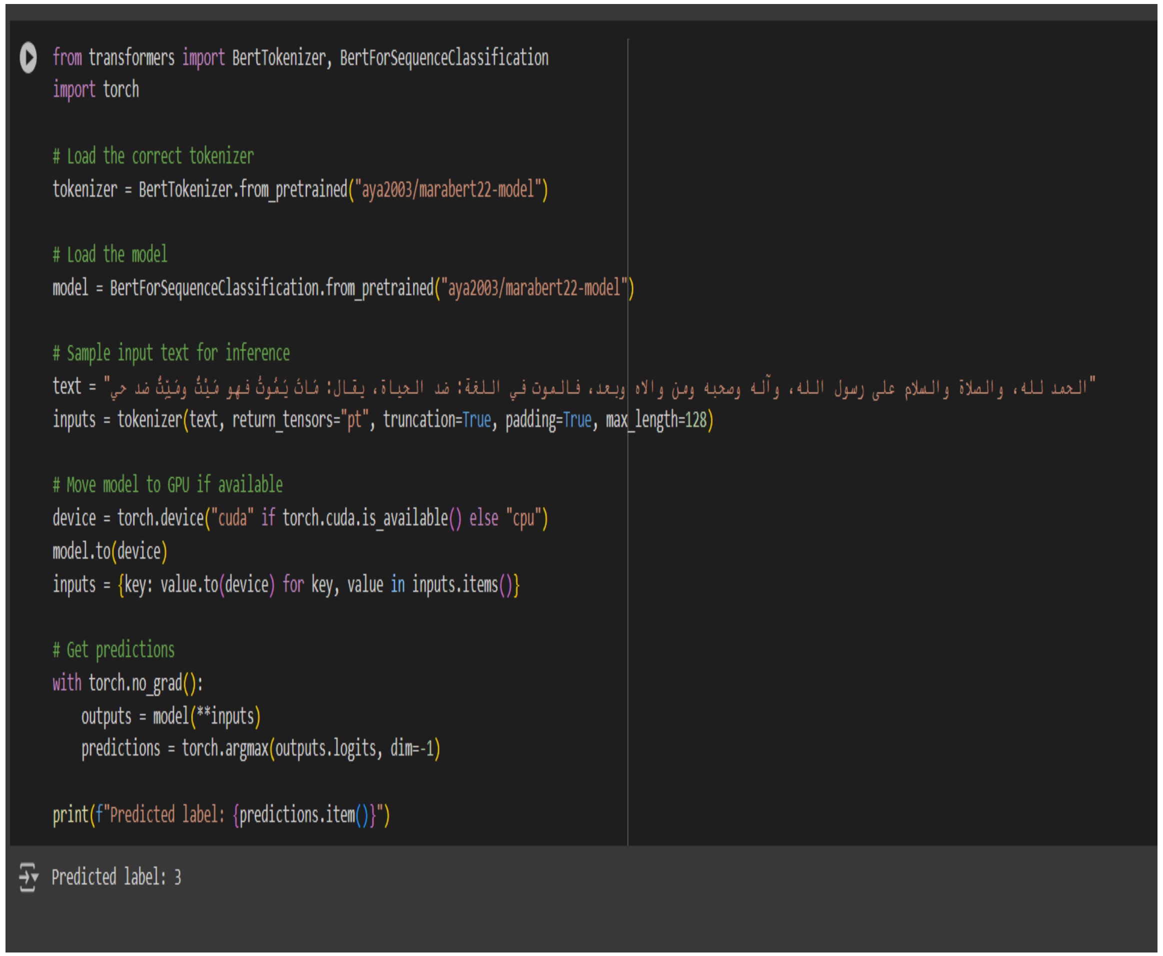


```
[ ] from transformers import ElectraTokenizer, ElectraForSequenceClassification
import torch

# Load the correct tokenizer
tokenizer = ElectraTokenizer.from_pretrained("aya2003/araelectra-model")

# Load the model
model = ElectraForSequenceClassification.from_pretrained("aya2003/araelectra-model")

# Sample input text for inference
text = "نمع المستقبلية والحالية، والتي تتضمن تنفيذ عدد من المشاريع في الإمارة. جاء ذلك خلال استقبال سموه في قصره بمدينة مقر بن محمد أمس عبد الرحمن بن محمد العويس وزير الصحة
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
inputs = {key: value.to(device) for key, value in inputs.items()}

# Get predictions
with torch.no_grad():
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=-1)

print(f"Predicted label: {predictions.item()}")
```

```
Predicted label: 4
```

RoBerta

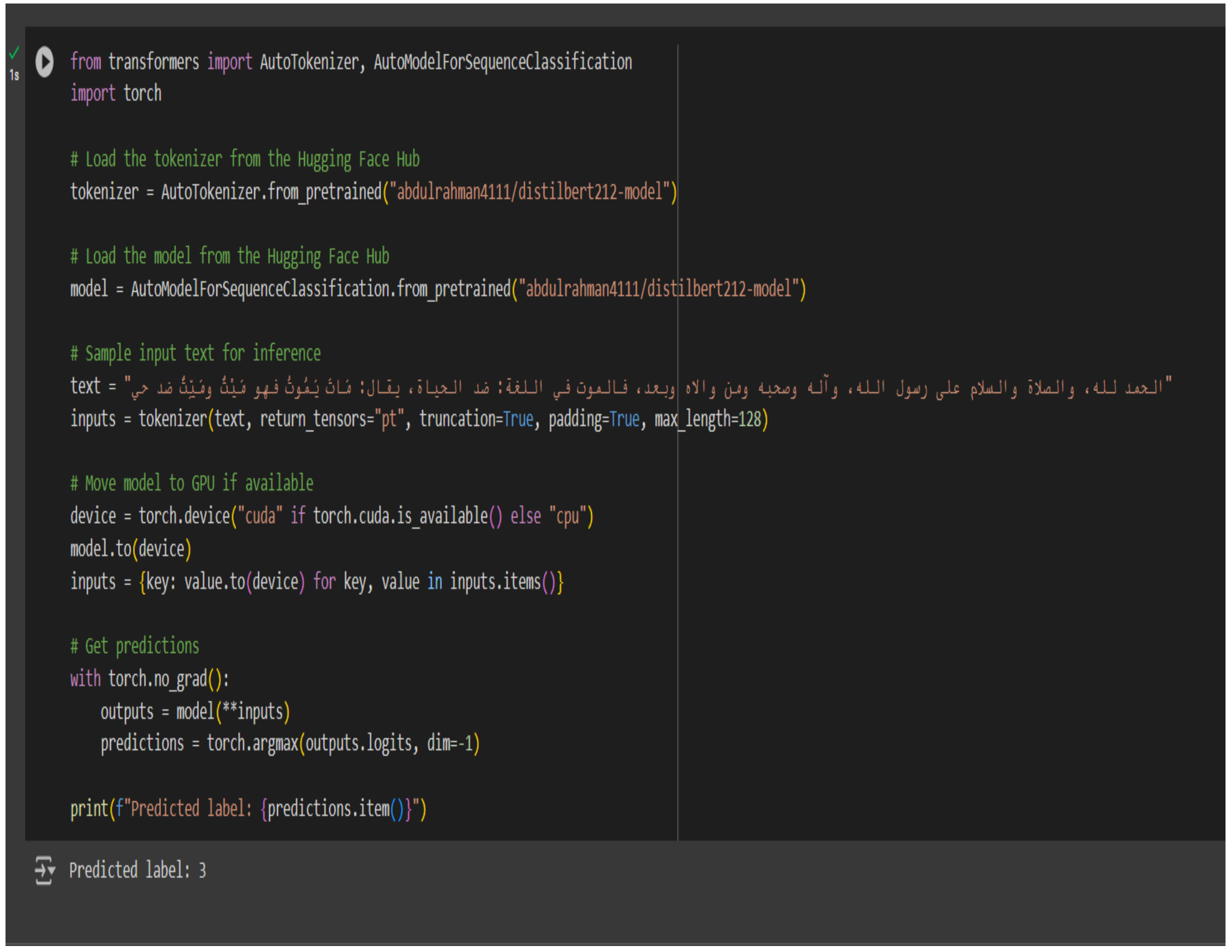
AraElectra



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

# Load the correct tokenizer
tokenizer = BertTokenizer.from_pretrained("aya2003/marabert22-model")

# Load the model
model = BertForSequenceClassification.from_pretrained("aya2003/marabert22-model")

# Sample input text for inference
text = "الحمد لله، والصلاة والسلام على رسول الله، وآله وصحبه ومن والاه وبعد، فالموت في اللغة: ضد الحياة، يقال: مَاتَ يَمُوتُ فهو مَيْتٌ ومَيِّتٌ ضد حي"
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
inputs = {key: value.to(device) for key, value in inputs.items()}

# Get predictions
with torch.no_grad():
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=-1)

print(f"Predicted label: {predictions.item()}")
```

```
Predicted label: 3
```

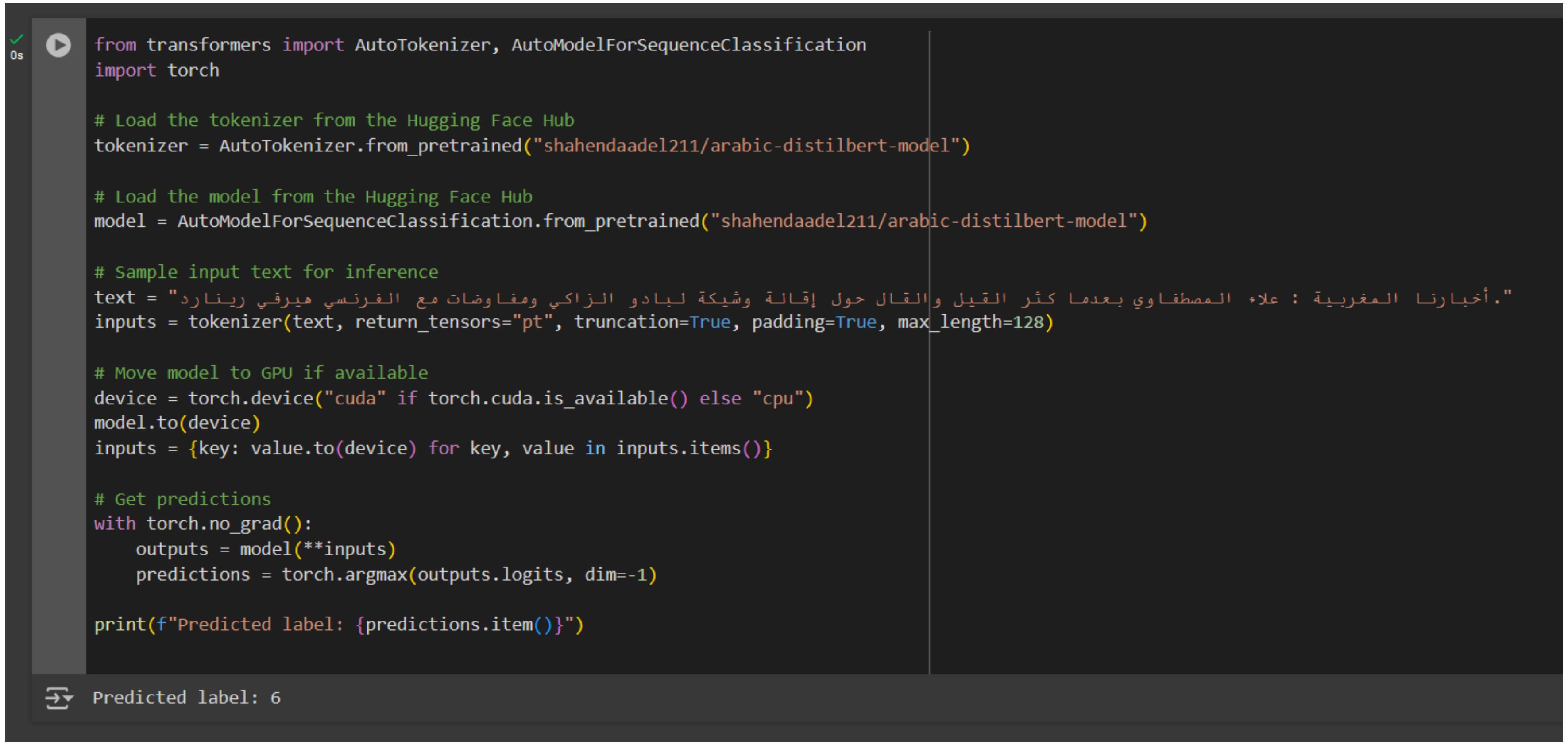
MaraBERT



```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch

# Load the tokenizer from the Hugging Face Hub
tokenizer = AutoTokenizer.from_pretrained("abdulrahman4111/distilbert212-model")

# Load the model from the Hugging Face Hub
model = AutoModelForSequenceClassification.from_pretrained("abdulrahman4111/distilbert212-model")

# Sample input text for inference
text = "الحمد لله، والصلاة والسلام على رسول الله، وآله وصحبه ومن والاه وبعد، فالموت في اللغة: ضد الحياة، يقال: مَاتَ يَمُوتُ فهو مَيْتٌ ومَيِّتٌ ضد حي"
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
inputs = {key: value.to(device) for key, value in inputs.items()}

# Get predictions
with torch.no_grad():
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=-1)

print(f"Predicted label: {predictions.item()}")
```

```
Predicted label: 3
```

DistilBert



```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch

# Load the tokenizer from the Hugging Face Hub
tokenizer = AutoTokenizer.from_pretrained("shahendaadel211/arabic-distilbert-model")

# Load the model from the Hugging Face Hub
model = AutoModelForSequenceClassification.from_pretrained("shahendaadel211/arabic-distilbert-model")

# Sample input text for inference
text = "أخبارنا المغربية : علاء المصطفاوي بعدما كثر القيل و القال حول إقالة وشيكة لبادو الزاكي ومفاوضات مع الفرنسي هيرفي رينارد."
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
inputs = {key: value.to(device) for key, value in inputs.items()}

# Get predictions
with torch.no_grad():
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=-1)

print(f"Predicted label: {predictions.item()}")
```

```
Predicted label: 6
```

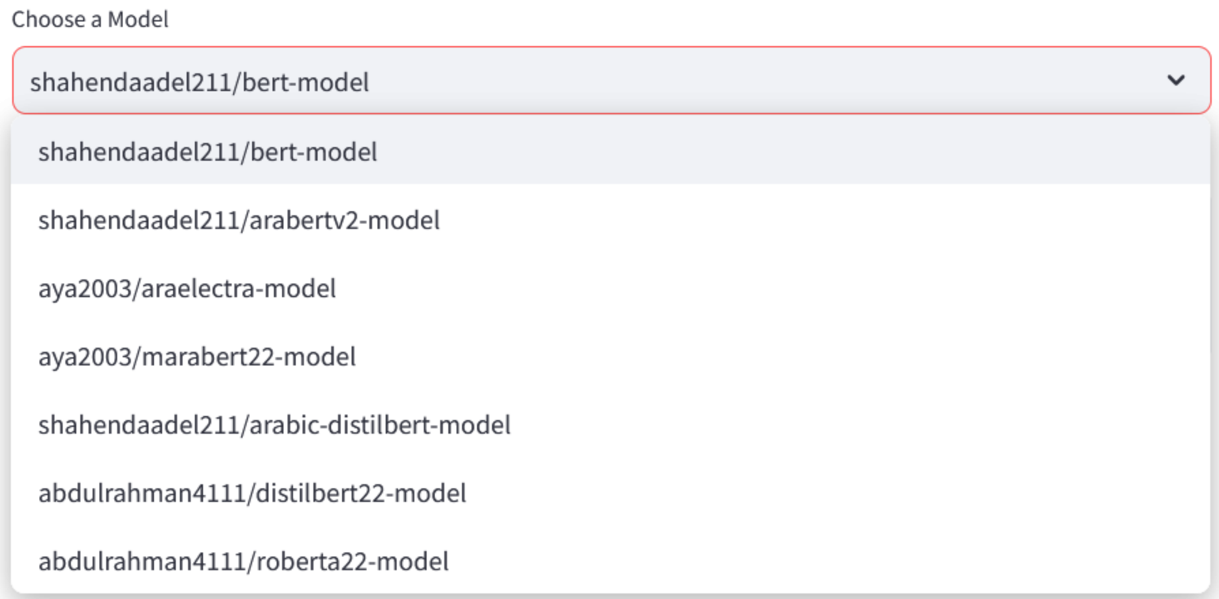
Arabic DistilBERT

Application:

We made our application using Streamlit

The app uses Transformer models to classify Arabic news articles in real-time. Users can input Arabic text and select from different models to categorize news into topics like Finance, Medicine, Religion, Culture, Politics, Sports, and Technology. The app displays model predictions and class distributions, making it easy to analyze and understand news content.

First, user selects a model to try it



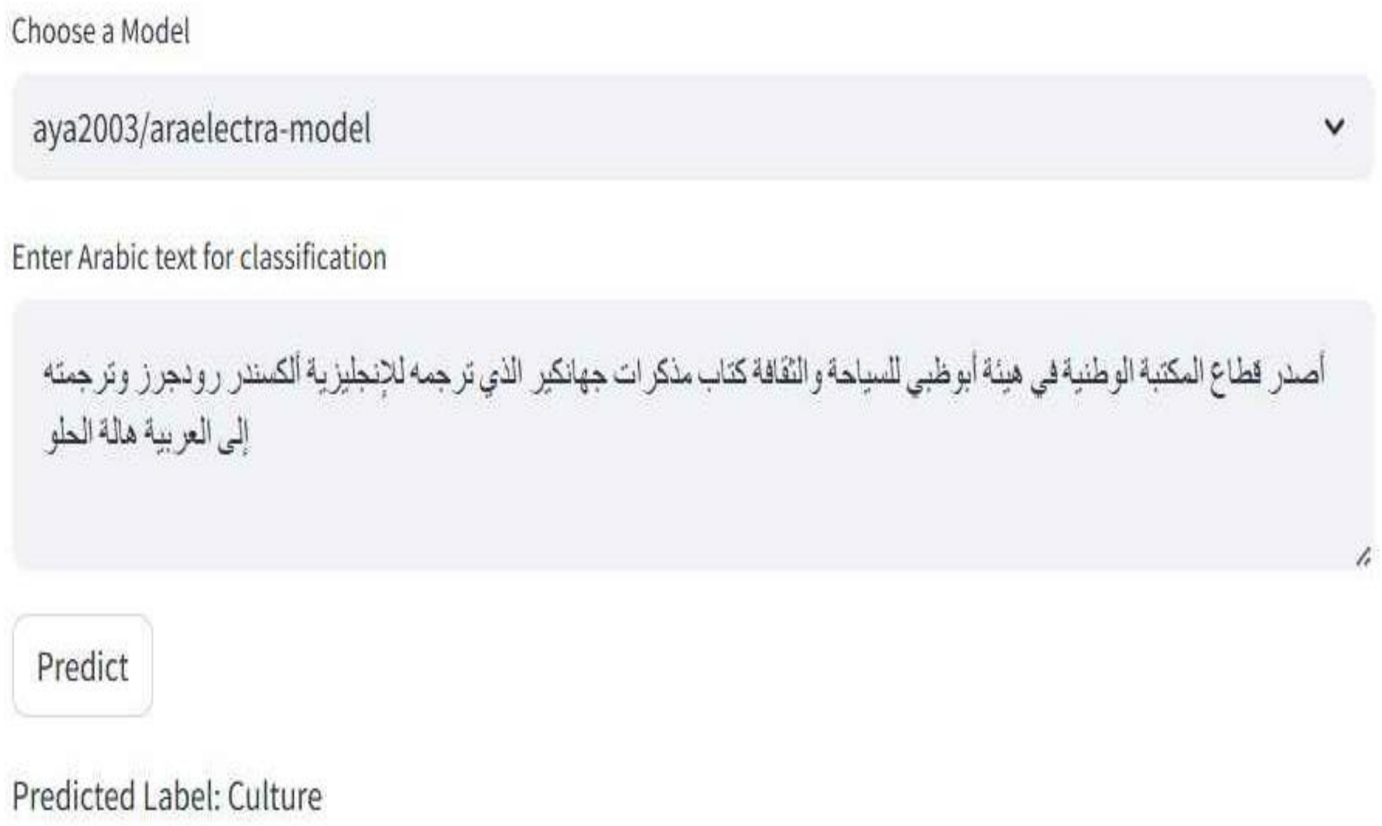
Choose a Model

shahendaadel211/bert-model

- shahendaadel211/bert-model
- shahendaadel211/arabertv2-model
- aya2003/araelectra-model
- aya2003/marabert22-model
- shahendaadel211/arabic-distilbert-model
- abdulrahman4111/distilbert22-model
- abdulrahman4111/roberta22-model

Then, choosing the model the user enters a text and press predict

After, few seconds the predicted category appears



Choose a Model

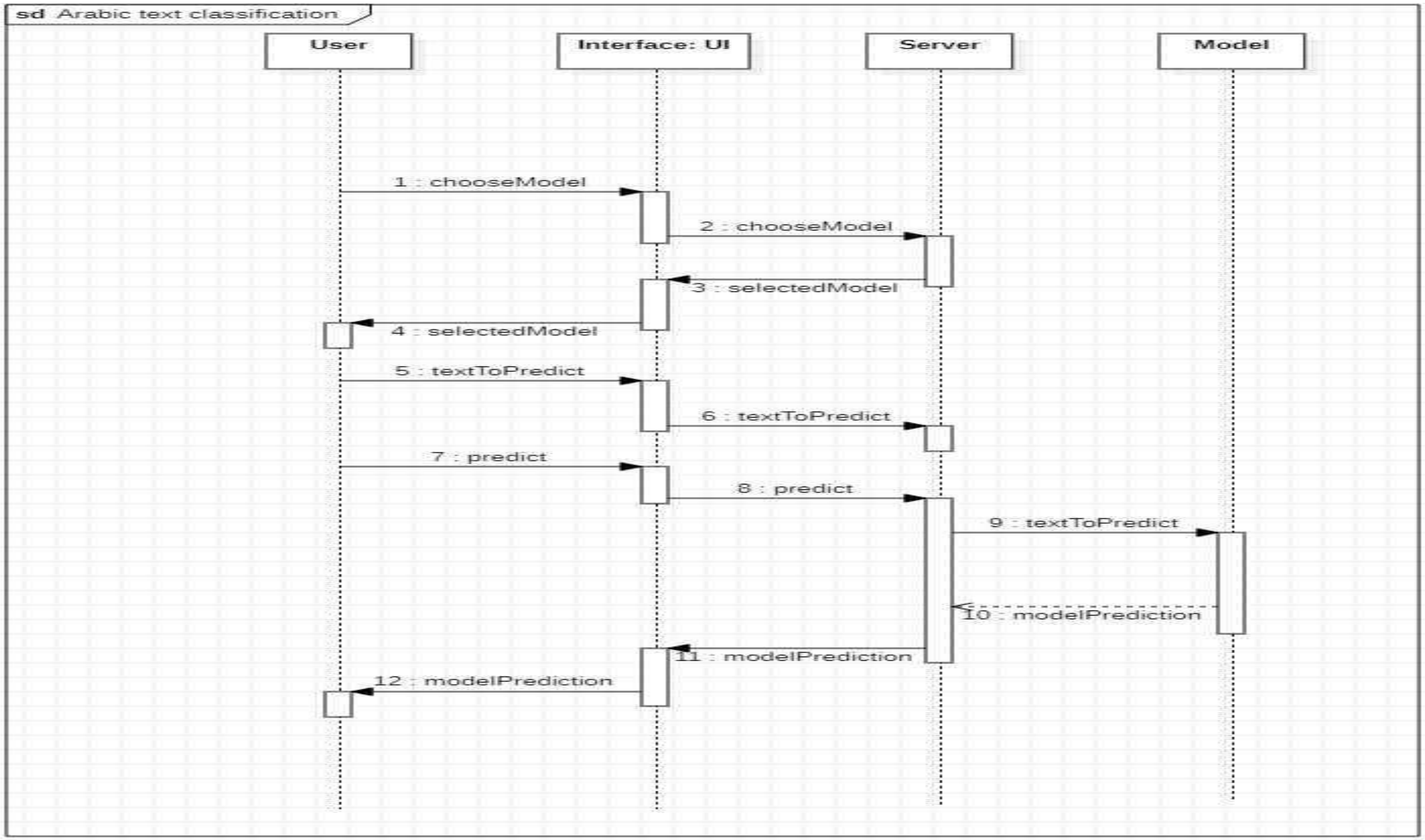
aya2003/araelectra-model

Enter Arabic text for classification

أصدر قطاع المكتبة الوطنية في هيئة أبوظبي للسياحة والثقافة كتاب مذكرات جهانكير الذي ترجمه للإنجليزية ألكسندر رودجرز وترجمته إلى العربية هالة الحلو

Predict

Predicted Label: Culture


sd Arabic text classification
User
Interface: UI
Server
Model
1 : chooseModel
2 : chooseModel
3 : selectedModel
4 : selectedModel
5 : textToPredict
6 : textToPredict
7 : predict
8 : predict
9 : textToPredict
10 : modelPrediction
11 : modelPrediction
12 : modelPrediction

Conclusion

The Arabic News Detection project aims to address the challenges of classifying Arabic news articles accurately using advanced machine learning and NLP techniques. By leveraging datasets like SANAD and employing models ranging from traditional Naive Bayes to modern transformer-based models, the project seeks to improve the classification performance. The comprehensive preprocessing steps and data augmentation techniques contribute to enhancing the quality of the training data, while the evaluation of various models ensures the selection of the best-performing approach.
Despite the challenges posed by the complexity of the Arabic language and the computational demands of advanced models, the project demonstrates the potential of modern NLP techniques in effectively tackling Arabic news detection.

Thank You