مُعَرَّب

Under supervisor of:

Dr.Mona Nagy Elbedwehy

By: Mariam Talaat Ahmed Ghazy

Introduction

- Problem statement.:
- Most intelligent systems lack the ability to accurately classify Arabic sentences according to their grammatical type, especially given the wide variety in sentence structure and grammatical styles. Furthermore, traditional solutions often rely on explicit linguistic rules, making them inflexible in the face of linguistic diversity .

Introduction

- <u>Objectives.:</u>
- Developing an intelligent model capable of accurately classifying Arabic sentences into nominal and verbal types using deep learning techniques (AraBERTv2).
- Facilitating the process of learning the Arabic language for primary school students through an interactive tool that helps them distinguish between sentence types and understand their grammatical components.

Introduction

- Objectives
- Designing a model that can later be integrated into educational applications to help students practice linguistic analysis skills in an interactive and simple way.
- Achieving high classification accuracy to provide a reliable model that can be used in educational environments, whether inside the classroom or in smart educational applications..

Data

Data source

Since there was no ready database for this task in Arabic, I created the data myself.
In order to be able to train the model to classify Arabic sentences into nominal and verbal, it was necessary to build a dataset that was accurately labeled.

Reason for creating the dataset manually:

Because my goal was to teach children to practice between the two types of sentences, I made sure the sentences were:

Short

Simple in vocabulary

Properly formatted

Free of complexity or cumbersomeness

Educational benefit:

This is ideal for later training the dataset model in an interactive educational application that helps elementary students understand and analyze it in detail.

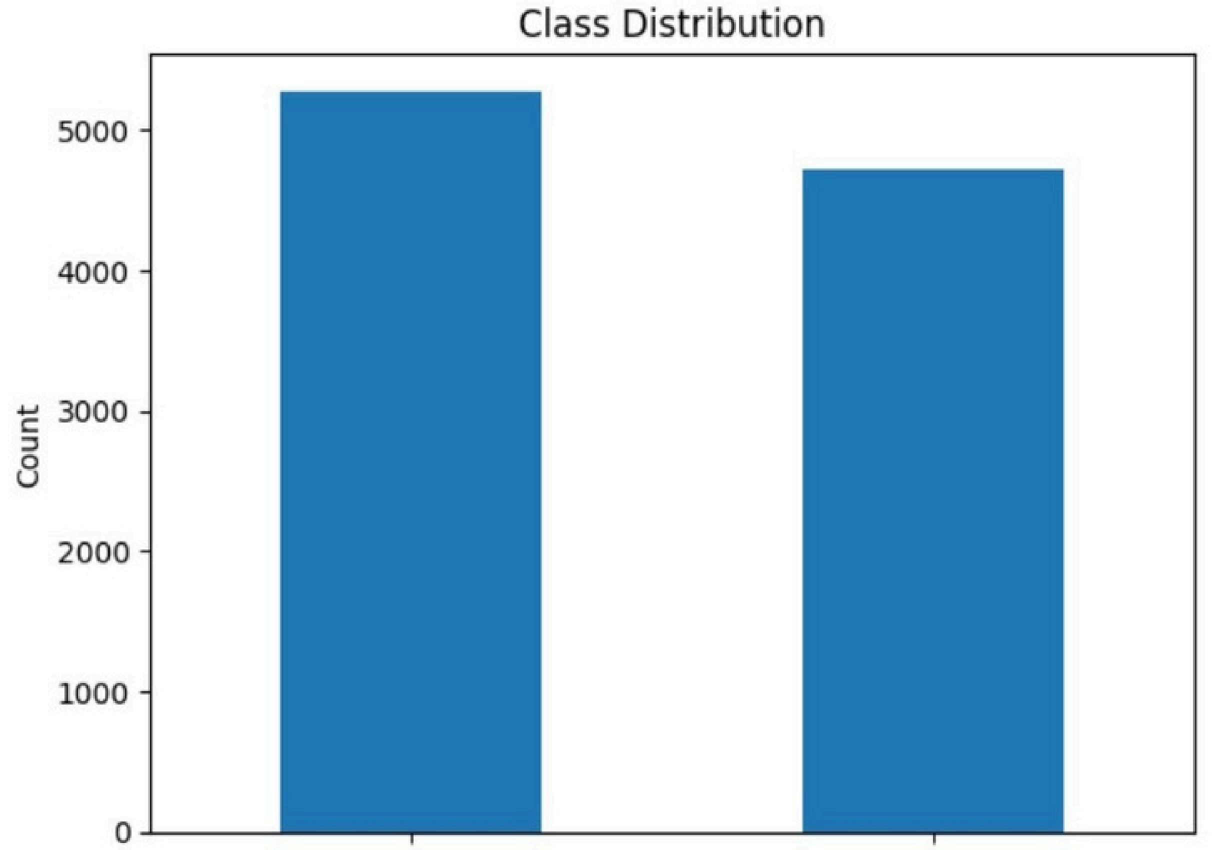
Data

الجملة	التصنيف
السماء صافية.	اسمية
الحديقة جميلة.	اسمية
الشمس ساطعة.	اسمية
الوردة متفتحة.	اسمية
البحر هادئ.	اسمية
الكتاب مفيد.	اسمية
الولد ذكي.	اسمية
البنت مجتهدة.	اسمية
الهواء نقي.	اسمية
القمر مضيء.	اسمية
الغرفة واسعة.	اسمية
السيارة سريعة.	اسمية
المنزل كبير	اسمية

البحر مائج.	اسمية
السماء زرقاء.	اسمية
كتب الطالب الدرس.	فعلية
قرأ الطفل القصة.	فعلية
أكل الولد التفاحة.	فعلية
شرب الرجل الماء.	فعلية
رسم التلميذ اللوحة.	فعلية
كسر الطفل الزجاج.	فعلية
فتح العامل الباب.	فعلية

Data

```
التصنيف
اسمية      5275
فعلية      4725
Name: count, dtype: int64
التصنيف
اسمية      52.75
فعلية      47.25
Name: proportion, dtype: float64
```



Data

```
import pandas as pd
import json
from sklearn.model_selection import train_test_split

# تحميل ملف CSV
df = pd.read_csv('/content/arabic_sentences_final_with_labels.csv')

# تحويل التصنيف إلى أرقام
label_map = {'اسمية': 0, 'فعلية': 1}
df['label'] = df['التصنيف'].map(label_map)

df = df.rename(columns={'الجملة': 'text'})


df = df[['text', 'label']]

# تقسيم البيانات
train_df, temp_df = train_test_split(df, test_size=0.2, random_state=42, stratify=df['label'])
val_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42, stratify=temp_df['label'])

# حفظ الملفات بصيغة JSON
train_df.to_json('train.json', orient='records', force_ascii=False, indent=2)
val_df.to_json('validation.json', orient='records', force_ascii=False, indent=2)
```

Arabertv2

It's specifically designed for the Arabic language, which means it's better at understanding the nuances of Arabic text.

It's been pre-trained on a large dataset of Arabic text, which includes a wide range of sources like news articles, books, and websites.

It's available in different sizes, including a smaller version that's easier to use on smaller devices.

Arabertv2

How does AraBERT work?

It uses a technique called "pre-segmentation" to break down Arabic text into smaller pieces that are easier to understand. It uses a special kind of neural network called a "transformer" to analyze the text and understand its meaning.

Arabertv2

1. Converting Words into Numerical Representations (Embeddings):

Each word in the sentence is converted into a numerical representation using an embedding layer. In AraBERTv2, this embedding is enriched with morphological information specific to the Arabic language, allowing the model to understand word properties more deeply.

Arabertv2

2. Contextual Analysis Using Transformer Layers:

The model consists of 12 Transformer layers, and each layer includes:

Multi-head Self-Attention: Allows the model to understand the relationship between each word and all other words in the sentence.

Feed-forward Layers: Non-linear processing layers that learn the complexities of the language.

Layer Normalization + Residual Connections: Help improve training stability and efficiency.

Arabertv2

3. Using the [CLS] Representation:

At the end of the Transformer layers, the special token [CLS] (the first token in the sequence) is used as a comprehensive representation of the entire sentence.

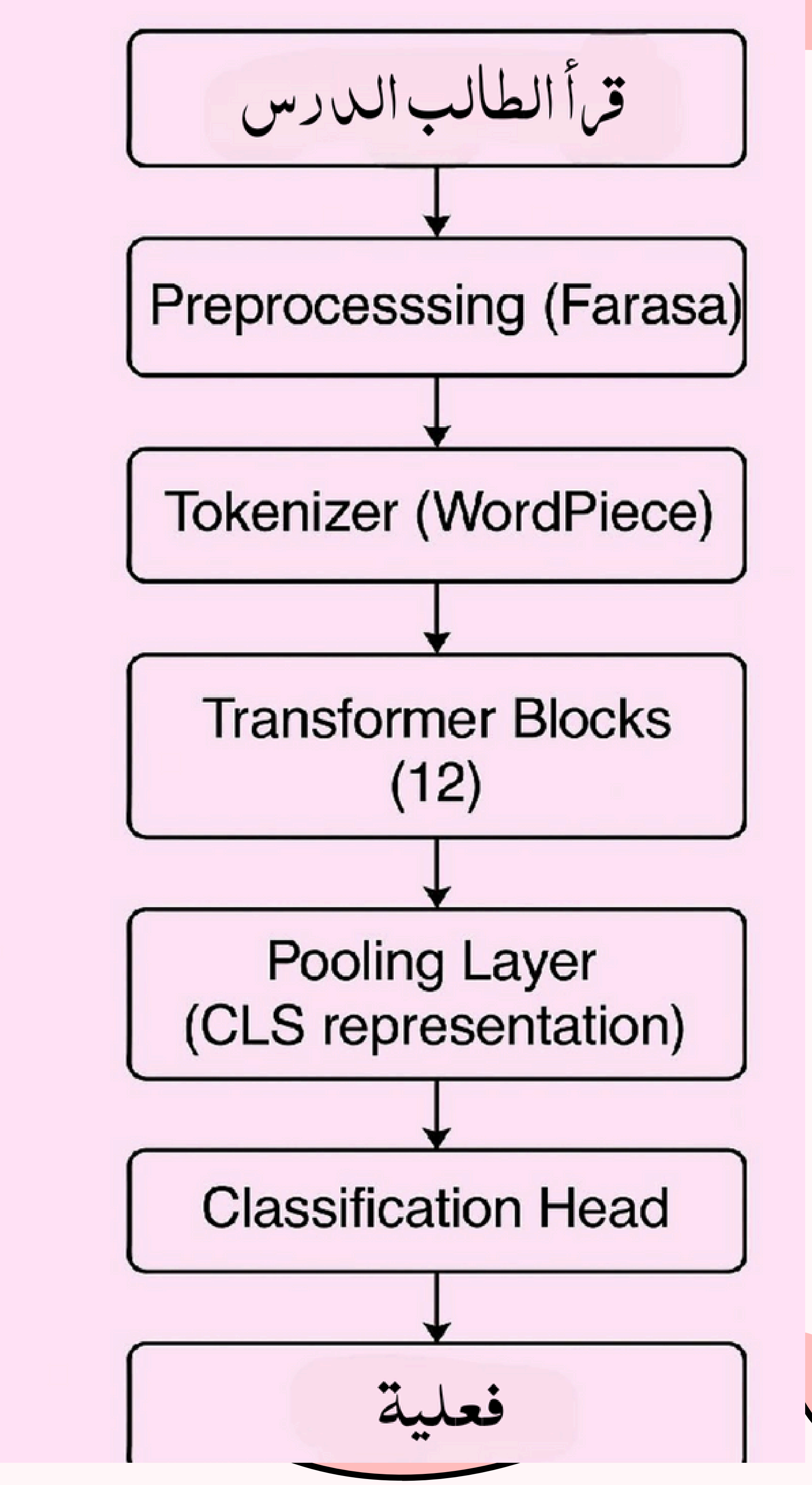
4. Passing [CLS] to the Classification Head:

The final layer is a linear layer with an output size equal to the number of classes The [CLS] representation is passed to this layer, and the model computes logits (raw scores before activation).

Arabertv2

5. Decision Making Using Softmax:

Softmax is applied to the logits to convert them into probabilities.
The class with the highest probability is selected as the prediction.



Algorithm of Arabertv2

1. Word Embeddings:

Each word is converted into a numerical representation using:
Word embeddings,Segment embedding,Position embeddings
The final representation of a word:

$$x_i = E_{word}(w_i) + E_{position}(i) + E_{segment}(s_i)$$

2. Transformer Layers (12 layers in AraBERTv2):

Each layer consists of Multi-Head Self-Attention and a Feed Forward Neural Network. In each layer:

a. Attention Calculation:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

Algorithm of Arabertv2

b. Multi-Head Attention:

$$\mathrm{MultiHead}(X) = \mathrm{Concat}(head_1, \ldots, head_h)W^O$$

c. Feed Forward Layer:

$$\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$$

$$\mathrm{ReLU}(x) = \max(0, x)$$

d. Residual Connection and Layer Normalization:

$$\mathrm{Output} = \mathrm{LayerNorm}(x + \mathrm{SubLayer}(x))$$

Algorithm of Arabertv2

3. Final Sentence Representation (from [CLS]):

$$h_{[CLS]} \in \mathbb{R}^d$$

4. Classification Layer:

Where is the probability distribution over classes (., nominal or verbal sentence).

$$\hat{y} = \text{softmax}(h_{[CLS]} W + b)$$

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Code of Arabertv2 Model

```
import os
os.environ["WANDB_DISABLED"] = "true"
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments , EarlyStoppingCal]
import numpy as np
import evaluate
from sklearn.metrics import accuracy_score, precision_recall_fscore_support

data_files = {
    "train": "train.json",
    "validation": "validation.json",
    "test": "test.json"
}
dataset = load_dataset("json", data_files=data_files)

# تحميل Tokenizer الخاص بـ AraBERTv2
model_checkpoint = "aubmindlab/bert-base-arabertv2"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)

# Tokenizer funcation
def tokenize_function(example):
    return tokenizer(example["text"], padding="max_length", truncation=True)

# apply Tokenizer
tokenized_datasets = dataset.map(tokenize_function, batched=True)

# تحميل موديل التصنيف مع تفعيل gradient checkpointing لتقليل الذاكرة
model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labels=2,
                                                           use_cache=False)
model.gradient_checkpointing_enable()  #  Gradient Checkpointing
```

Code of Arabertv2 Model

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(64000, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
```

```
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

Code of Arabertv2 Model

```python
# evaluation
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, predictions, average='weighted')
    acc = accuracy_score(labels, predictions)
    return {"accuracy": acc, "f1": f1, "precision": precision, "recall": recall}

# إعدادات التدريب
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    num_train_epochs=10,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    logging_dir="./logs",
    logging_steps=10,
    save_total_limit=2,
    metric_for_best_model="accuracy",
    greater_is_better=True,

)
```

Code of Arabertv2 Model

```
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    logging_dir="./logs",
    logging_steps=10,
    save_total_limit=2,
    metric_for_best_model="accuracy",
    greater_is_better=True,


# (Trainer)
rainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    compute_metrics=compute_metrics,
allbacks=[EarlyStoppingCallback(early_stopping_patience=2)]
)

rainer.train()

# حفظ النهائي
odel.save_pretrained("./arabertv2_sentence_classifier")
okenizer.save_pretrained("./arabertv2_sentence_classifier")
```

Evaluation

Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
0.000000	0.000025	1.000000	1.000000	1.000000	1.000000

```
abertv2_sentence_classifier/tokenizer_config.json',
abertv2_sentence_classifier/special_tokens_map.json',
abertv2_sentence_classifier/vocab.txt',
abertv2_sentence_classifier/added_tokens.json',
abertv2_sentence_classifier/tokenizer.json')
```

Final Results

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

# دالة للتنبؤ
def predict(text):
    # تحويل النص إلى tokens
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)

    # إرسال النص عبر النموذج للحصول على التنبؤ
    with torch.no_grad():
        logits = model(**inputs).logits

    # تحويل النتائج إلى فئة متوقعة (0 أو 1)
    prediction = torch.argmax(logits, dim=-1).item()

    # تخصيص الفئات: 0 = "اسم" ، 1 = "فعل"
    if prediction == 1:
        return "فعلية(فعل-فاعل-مفعول به)"
    else:
        return "اسمية(مبتدأ-خبر)"

# تجربة النصوص الجديدة
new_text = " دخل الطالب المدرسة "
prediction = predict(new_text)

print(f"    الجملة: {prediction}")
```

```
الجملة:  فعلية(فعل-فاعل-مفعول به)
```

Final Results

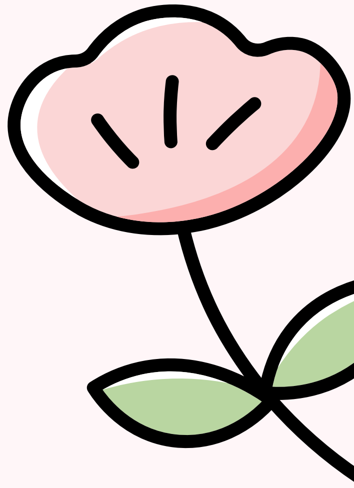
```python
# دالة للتنبؤ
def predict(text):
    # تحويل النص إلى tokens
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)

    # إرسال النص عبر النموذج للحصول على التنبؤ
    with torch.no_grad():
        logits = model(**inputs).logits

    # تحويل النتائج إلى فئة متوقعة (0 أو 1)
    prediction = torch.argmax(logits, dim=-1).item()

    # "تخصيص الفئات: 0 = "اسم" ، 1 = "فعل
    if prediction == 1:
        return "فعلية(فعل-فاعل-مفعول به)"
    else:
        return "اسمية(مبتدأ-خبر)"

# تجربة النصوص الجديدة
new_text = "الطقس مستقر"
prediction = predict(new_text)

print(f"    الجملة: {prediction}")
```

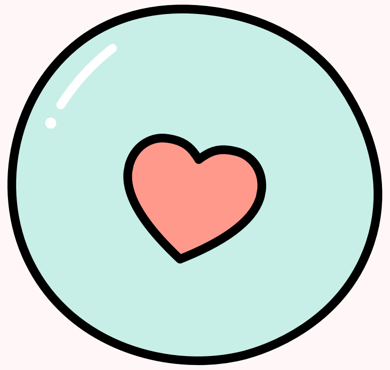
الجملة: (مبتدأ-خبر)اسمية

Comparison

Comparison Table: BiLSTM vs BERT vs AraBERT vs AraBERTv2

Feature	BiLSTM	BERT	AraBERT	AraBERTv2
Architecture	Recurrent (LSTM)	Transformer	Transformer	Transformer
Pretraining	Not pretrained	Pretrained on English	Pretrained on Arabic (Farasa preprocessing)	Improved pretraining on Arabic
Tokenizer	Manual/Custom	WordPiece	Farasa + WordPiece	Farasa + WordPiece
Language Support	Any (manual)	Primarily English	Arabic (specific)	Arabic (enhanced coverage)
Contextual Understanding	Sequential	Bidirectional	Bidirectional	Bidirectional
Speed	Slower	Fast	Fast	Fast
Embedding Layer	Word embeddings + LSTM	Positional + Token embeddings	Positional + Token embeddings	Positional + Token embeddings
Performance on Arabic NLP	Low	Medium	High	Very High
Number of Transformer Layers	None	12	12	12 (with improvements)





Does Anyone Have A Question?

Thank You