Arabic Dialect Identification using Deep Learning Models

AAI 612: Deep Learning and its Applications

Project Report

Mohamad El Kassar

July 28, 2025

Date	Revision	Release Notes
	Rev 01	Initial Release

Contents

1 Abstract

This project presents the results and findings of identifying Arabic dialects from across different regions of the Arab world using different deep learning models. We built different deep learning models and fine-tuned a pre-trained BERT model. The results were not that impressive but built the foundation to use the IADD (Integrated Dataset for Arabic Dialect iDentification) dataset to identify different arabic dialects.

2 Introduction

2.1 Problem Statement

This project aims to develop a deep learning system that can classify Arabic text into regional dialects and specific countries using transformer-based architecture. The dialects that the system will use to classify the text will be Moroccan, Levantine, Egyptian, Iraqi, and Gulf.

The project will aim to use the fine-tuned Arabic pre-trained language model AraBERT, combined with training the model on an Arabic Dialect Identification dataset that should achieve a better accuracy.

2.2 Objectives

The primary goal of this project is to classify Arabic texts into regional dialects. The regions we focused on were based on the IADD dataset which are Moroccan, Levantine, Egyptian, Iraqi, and Gulf.

The second objective was to benchmark the performance of the different models used in this project and to identify which model and architecture is the best to use.

2.3 Limitations

The significant limitations encountered in this project were the limited computational resources. Even with the help of Google Colab, the experiments with the different deep learning models were conducted on hardware with insufficient processing power which impacted the scope of the project. The limited number of training epochs as a result of the extended training times prevented the model from reaching the goal. Smaller batch sizes were used as well because the limited memory constraints affected the performance as well.

I also had to reduce the size of the dataset to only 2,500 samples per region totaling 10,000 samples rather than using the entire 136,317 samples. The reduction of the dataset was made to reduce the training times and prevent memory overload.

The scope was also restricted to simplified deep learning implementations and basic BERT models due to computational limitations without exploring advanced techniques such as bidirectional LSTM layers and AraBERT language models.

3 Dataset Analysis

3.1 Source and Overview

The dataset used in this project is the Identification of Arabic Dialectical Dataset (IADD), an integrated dataset for arabic dialect identification that contains 136,317 text representing 5 regions (Maghrebi (MGH), Levantine (LEV), Egypt (EGY), Iraq (IRQ) and Gulf (GLF)) and 9 countries (Algeria, Morocco, Tunisia, Palestine, Jordan, Syria, Lebanon, Egypt and Iraq). The dataset is created from five Arabic corpora: DART, SHAMI, TSAC, PADIC and AOC. DART contains about 25,000 tweets. SHAMI contains 117,805 sentences that cover the levantine (Palestine, Jordan, Syria, and Lebanon) dialects. TSAC contanins 17,000 comments collected from Tunisian Facebook pages. PADIC contains sentances transcribed from recordings. AOC contains reader commentary from three Arabic newspapers.

The dataset is publicly available through Gihub on https://github.com/JihadZa/IADD.

3.2 Dataset Structure and Format

IADD is stored in a JSON format with the following fields: Sentence, Region (MGH, LEV, EGY, IRQ, GLF or general), Country (MAR, TUN, DZ, EGY, IRQ, SYR, JOR, PSE, LBN), and Datasource (PADIC, DART, AOC, SHAMI or TSAC).

3.3 Dataset Distribution

The official dataset size is 136,317 texts. After analyzing the dataset, it was revealed that 87,573 samples were from the LEV region, 33,996 were from the MGH region, 6,682 were from the GLF region, 4837 from the EGY region, 2,500 were tagged as general, and only 216 samples from the IRQ region.

The dataset shows a clear class imbalance with the Levantine dialect covering over 64% of the dataset while the Iraqi only covering 0.2% of the dataset. Therefore, we had to do some prepossessing on the dataset before diving into the model.

3.4 Data Prepossessing

First I had to do some data balancing and removing the IRQ region since its significantly underrepresented. Then I had to do a sampling strategy to achieve 2,500 per region and then had to remove the General region as well after that.

Prepossessing of the sentences had to be done as well. Removing the null vales, extra white spaces, trailing and leading white spaces, removing non-Arabic characters, and punctuation since they will not affect the classification.

Thr train/validation/test split was as follows: 75% train, 10% validation, and 15% test.

4 Methodology

4.1 Model Architectures

The project implements three different deep learning architectures and compares each one of them. Each deep learning model represents different levels of complexity and provides a comparison between traditional recurrent neural network approaches and transformer-based approaches.

4.1.1 Baseline LSTM Model

The baseline LSTM model was the first model used in this project.

Listing 1: Baseline LSTM Architecture

```
model = Sequential([
    Embedding(input_dim=NUMLWORDS, output_dim=32),
    LSTM(32, dropout=0.2, recurrent_dropout=0.2),
    Dense(16, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

optimizer = Adam(learning_rate=0.001, clipnorm=1.0)

model.compile(loss='categorical_crossentropy',
              optimizer=optimizer, metrics=['accuracy'])
```

Layer Configuration:

- **Embedding Layer:** Maps vocabulary indices to 32-dimensional dense vectors for 1,000-word vocabulary
- **LSTM Layer:** Single recurrent layer with 32 hidden units.
- **Dense Layer:** Fully connected layer with 16 neurons and ReLU activation
- **Dropout Layer:** Dropout layer to prevent overfitting
- **Output Layer:** Softmax classification layer for 4-class dialect prediction

Training Configuration:

- **Optimizer:** Adam with learning rate 0.001 and gradient clipping (clipnorm=1.0)
- **Loss Function:** Categorical crossentropy for multi-class classification

4.1.2 Stacked LSTM Model

The stacked LSTM model represents a better version of the baseline LSTM model.

Listing 2: Stacked LSTM Implementation

```
def create_model(num_words, maxlen):
    model = Sequential([
        Embedding(input_dim=num_words, output_dim=128),
        LSTM(128, dropout=0.2, recurrent_dropout=0.2,
        return_sequences=True),
        LSTM(64, dropout=0.2, recurrent_dropout=0.2),
        Dense(32, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    optimizer = Adam(learning_rate=0.001, clipnorm=1.0)
    model.compile(loss='categorical_crossentropy',
                  optimizer=optimizer,
                  metrics=['accuracy'])
    return model
```

Layer Configuration:

- **Enhanced Embedding:** 128-dimensional embeddings (4× increase from baseline)

- **First LSTM Layer:** 128 hidden units with `return_sequences=True` to pass full sequence information to subsequent layer
- **Second LSTM Layer:** 64 hidden units processing the output from the first LSTM layer
- **Dense Processing:** 32-unit dense layer with ReLU activation
- **Regularization Strategy:** Dropout layer to prevent overfitting
- **Output Classification:** Softmax classification layer for 4-class dialect prediction

Training Parameters:

- **Batch Size:** 64 samples per batch
- **Maximum Epochs:** 10
- **Sequence Length:** 6,400 tokens
- **Vocabulary Size:** 1,000 words (constrained by computational limitations)

4.1.3 Fine-tuned BERT Model

Model Configuration:

Listing 3: BERT Model Setup

```
def create_fine_tuned_bert(num_classes,
        model_name='bert-base-multilingual-cased'):
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = TFAutoModelForSequenceClassification.from_pretrained(
        model_name,
        num_labels=num_classes
    )

    model.compile(
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model, tokenizer
```

Layer Configuration:

- **Base Model:** `bert-base-multilingual-cased` with 177,856,516 parameters (678.47 MB)
- **Multilingual Support:** Pre-trained on 104 languages including Arabic script

Data Preparation Pipeline:

Listing 4: BERT Data Preparation

```
def prepare_bert_data(texts, labels, tokenizer, max_length=512):
    encoded = tokenizer(
        texts,
        truncation=True,
        padding=True,
        max_length=max_length,
        return_tensors='tf'
    )

    return {
        'input_ids': encoded['input_ids'],
        'attention_mask': encoded['attention_mask']
    }, labels
```

Training Configuration:

- **Learning Rate:** 2e-5
- **Batch Size:** 16 samples (constrained by computational limitations)
- **Epochs:** 3

5 Experiments and Results

5.1 Overall Performance Summary

The results across all three models showed low performance, with very low accuracy scores for Arabic dialect classification.

5.1.1 Model Performance Comparison

The accuracy was around 21% when using the baseline LSTM model and only improved slightly when using stacked LSTM model and got up to 24% when using the pre-trained BERT model.

5.2 Performance Analysis and Contributing Factors

The poor performance in all models can be a result of = several factors.

5.2.1 Vocabulary Limitation Impact

The most significant factor contributing to poor performance was the very low vocabulary limitation imposed by computational constraints. This resulted in the loss of regional-specific terms and expressions.

- **Available Vocabulary:** 36,957 unique words discovered in the dataset
- **Utilized Vocabulary:** Only 1,000 words (2.7% of available vocabulary)

5.2.2 Training Duration Limitations

The low number of epochs used across all models resulted in underfitting.

5.2.3 Dataset Size Reduction Impact

Using only a small portion of the available dataset limited the models' learning capacity.

- **Original Dataset:** 135,804 samples across all dialects
- **Utilized Dataset:** 9,918 samples (7.3% of original data)

6 Conclusion

This project investigated Arabic dialect classification using three deep learning approaches: baseline LSTM, stacked LSTM, and fine-tuned BERT. All models performed poorly due to computational constraints that limited vocabulary to 1,000 words, reduced the dataset to 7.3% of original size, and prevented high training epochs.