Predict The Dialectal Arabic

Natural language processing

By:

Reema Alzahrani 2110156

Bushra Dajam 2110054

Yara Alshehri 2113140

Prepared for:
Dr.Latifah Al-Jafari

01

Data Collection

Domain Description:

The dataset is in Arabic Dialect Identification, a subset of NLP. It aims to understand and process various Arabic dialects using machine and deep learning. The main goal is to identify dialects accurately from text, crucial for improving communication tech, creating region-specific content, and supporting linguistic diversity.

The dataset consists of the following:

1. Number of tweets: 540,000
2. Number of users: 2,525
3. Number of countries covered: 18

Statistical Overview

Preprocessing

02

Preprocessing is a critical step in the Natural Language Processing (NLP) pipeline. It involves preparing raw text data for further analysis and processing. Effective preprocessing can significantly enhance the performance of NLP models by reducing noise and focusing on the essential features of the text. Here's a detailed look at common preprocessing steps for NLP:

```python
classes = {
'EG': 'EG',
'DZ': 'AF',
'TN': 'AF',
'LY': 'AF',
'MA': 'AF',
'JO': 'SHAM',
'LB': 'SHAM',
'PL': 'SHAM',
'SY': 'SHAM',
'IQ': 'IQ',
'KW': 'KJ',
'SA': 'KJ',
'AE': 'KJ'
'OM': 'KJ',
'QA': 'KJ',
'YE': 'YE',
'SD': 'AF',
'BH': 'KJ'
}
df.loc[:, 'dialect']=df['dialect'].replace(classes)
```

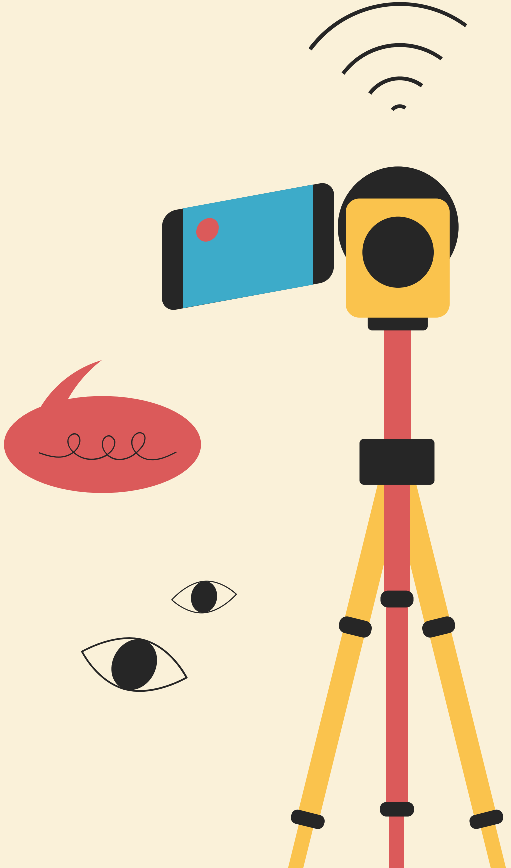
```
Os  [ ]  df=df.drop_duplicates()

Os  [15] df.duplicated().sum()

         0

Os  [16] df.isnull().sum()

         dialect    0
         tweets     0
         dtype: int64
```



used to remove duplicate rows from the DataFrame.

!pip install tashaphyne

```
[17] !pip install tashaphyne

     Requirement already satisfied: tashaphyne in /usr/local/lib/python3.10/dist-packages (0.3.6)
     Requirement already satisfied: pyarabic in /usr/local/lib/python3.10/dist-packages (from tashaphyne) (0.6.15)
     Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from pyarabic->tashaphyne) (1.16.0)
```

The command !pip install tashaphyne installs the Tashaphyne library, a Python tool designed for processing Arabic text, including tasks like light stemming and segmentation. This is essential for preparing Arabic text for natural language processing applications.

The preprocess_text function cleans Arabic text for NLP tasks. It removes non-Arabic characters, tokenizes, removes stopwords, and applies light stemming using Tashaphyne.

```
[ ]  import re
     from nltk.tokenize import word_tokenize
     from nltk.corpus import stopwords
     from nltk.stem.isri import ISRIStemmer
     from tashaphyne.stemming import ArabicLightStemmer

     import nltk
     nltk.download('punkt')
     nltk.download('stopwords')

     # Define preprocessing function
     def preprocess_text(text):
         # Remove non-Arabic characters
         text = re.sub(r'[^\u0600-\u06FF\s]', '', text)

         # Tokenize text
         tokens = nltk.word_tokenize(text)

         # Remove stopwords
         stop_words = set(stopwords.words('arabic'))
         tokens = [token for token in tokens if token not in stop_words]

         # Stem tokens

         stemmer = ArabicLightStemmer()
         tokens = [stemmer.light_stem(token) for token in tokens]

         # Join tokens back into a single string
         processed_text = ' '.join(tokens)

         return processed_text
```

The line df.tweets = df.tweets.apply(lambda x: preprocess_text(x)) processes each tweet in the DataFrame df by applying a preprocessing function.

```
df.tweets = df.tweets.apply(lambda x: preprocess_text(x))
```

```
[ ]  from sklearn.preprocessing import LabelEncoder
     label_encoder = LabelEncoder()
     y = label_encoder.fit_transform(df['dialect'])

     X=df['tweets']
```

The code encodes the 'dialect' column using LabelEncoder from scikit-learn, assigning a numerical label to each unique dialect. Then, it assigns the preprocessed tweets to X, preparing them as input data for a machine learning model.

Train and Test split

```
[ ] from sklearn.model_selection import train_test_split
    X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.2,random_state=2, stratify=y)
```

The code snippet divides the dataset into training and testing subsets for model training and evaluation. It utilizes the train_test_split function from scikit-learn, designating 20% of the data for testing (test_size=0.2). Additionally, it ensures that the class distribution is maintained in both subsets (stratify=y). The random_state=2 parameter ensures reproducibility of the split.

LSTM (Long Short-Term Memory)

In the context of Natural Language Processing (NLP), LSTM (Long Short-Term Memory) networks are a type of recurrent neural network (RNN) architecture designed to process and understand sequential data, such as text. LSTMs address the challenge of learning long-range dependencies in text data by using memory cells with gates that control the flow of information over time. This allows LSTMs to capture complex patterns and relationships in language, making them well-suited for tasks like language modeling, sentiment analysis, machine translation, and named entity recognition.

Naive Bayes classifier

Naive Bayes is a simple yet effective probabilistic classifier based on Bayes' theorem with an assumption of independence among predictors. Here's a concise overview:

Comparative Analysis of LSTM and Naive Bayes Classification Models

Based on these results, it is clear that the deep learning model is better than the machine learning model based on the accuracy, which is 0.71 in the Lstm algorithm, higher than the Naive Byas algorithm, which is 0.70. And f-score, recall in lstm is higher than naive byas but precision is lower than naive byas algorithm

Results

ML:Naive Bayes classifier

The Classification Report:

	precision	recall	f1-score	support
0	0.78	0.54	0.64	17580
1	0.82	0.63	0.71	11527
2	0.92	0.14	0.24	3100
3	0.66	0.91	0.77	34343
4	0.71	0.68	0.69	23105
5	0.79	0.02	0.04	1985
accuracy			0.70	91640
macro avg	0.78	0.49	0.51	91640
weighted avg	0.73	0.70	0.68	91640

Accuracy test: 0.7031754692274116
Accuracy train: 0.801785261228131

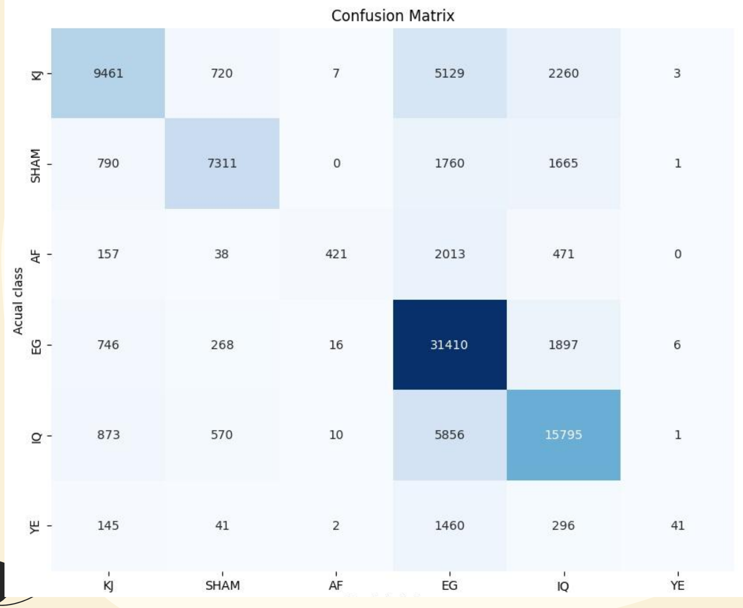
DL:LSTM

Classification Report:

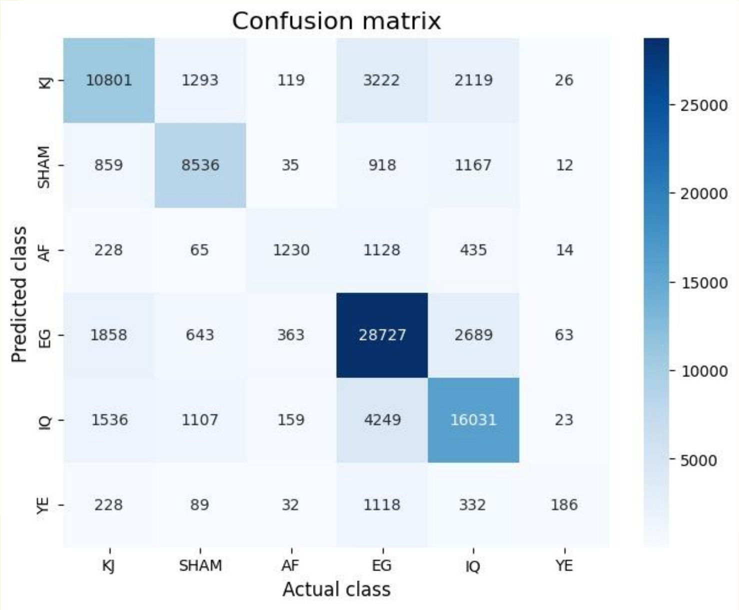
	precision	recall	f1-score	support
KJ	0.70	0.61	0.65	17580
SHAM	0.73	0.74	0.73	11527
AF	0.63	0.40	0.49	3100
EG	0.73	0.84	0.78	34343
IQ	0.70	0.69	0.70	23105
YE	0.57	0.09	0.16	1985
accuracy			0.71	91640
macro avg	0.68	0.56	0.59	91640
weighted avg	0.71	0.71	0.71	91640

confusion matrix

ML



DL



Classification Metrics