Arabic Named Entity Recognition

Arabic NER with ML, DL Models, and a User-Friendly Interface

1st QOSSAY RIDA
1211553
Ramallah, Palestine
1211553@student.birzeit.edu

2nd Mohammad Zidan
1212387
Ramallah, Palestine
1212387@student.birzeit.edu

3rd Tariq Zaghal
1210158
Ramallah, Palestine
1210158@student.birzeit.edu

***Abstract*—Named Entity Recognition (NER) plays a crucial role in understanding and processing Arabic text, particularly in identifying entities organization, time, location, money, person, event numerical and language. This project aims to implement an Arabic NER system using Python since the vast majority of the literature related to the subject is implemented in Python and its libraries. The datasets utilized include Wojood, which provide annotated examples of various named entities. The objective is to develop three models that accurately identifies and classifies these entities from raw Arabic text into organization, time, location, money, person, event numerical and language.**

I. Introduction

In this project, the objective is to develop an Arabic Named Entity Recognition (NER) system utilizing Python and the Wojood dataset, which contains annotated instances of named entities in the Arabic language. The dataset will undergo processing to consolidate 21 entity categories into 8 primary categories: organization, time, location, money, person, event, numerical, and language. I will implement and assess three models: a deep learning model (Recurrent Neural Networks, RNN) alongside three machine learning models (Conditional Random Fields, Decision Tree, and Naive Bayes). The evaluation of these models will be conducted based on precision, recall, F1-score, and accuracy to identify the most effective approach for Arabic NER.

II. Data Analysis

From Wojood dataset we have a merged the 21 categories into 8 categories (organization, time, location, money, person, event numerical and language) by merging currency (CURR) and money into MON, TIME and DATE into TIME, PERS and (NORP) into PER, LOC and GPE into LOC, and PERCENT, QUANTITY, and CARDINAL into NUM the rest stayed the same and have removed seven categories (website, occupation, product, facility, law, unit, and ordinal).

III. Models

We have applied our cleaned dataset (train, test and validation) to three model, one deep learning models Recurrent Neural Networks (RNN) and three machine learning model Conditional Random Fields (CRF),Descion Tree and Naive Bayes, we will compare their results to determine the most effective approach for Arabic NER tasks, focusing on metrics like precision, recall, F1-score, and overall accuracy, this comparative analysis aims to highlight the strengths and limitations of each approach.

A. *Conditional Random Field* (*CRF*)

A Conditional Random Field (CRF) is a probabilistic graphical model commonly used in sequence labeling tasks like Part-of-Speech Tagging and Named Entity Recognition. It considers contextual features and neighboring examples to predict a sequence of labels for a sequence of input samples based on conditional probabilities.

Main Steps:

1) **Feature Extraction for Training Data:** Extract features for each word in the sentence, which include:
   - The word itself.
   - Whether the word is a digit.
   - Prefix and suffix of the word.
   - Whether the word is in Arabic.
   - The previous word.
   - The next word.
2) **Label Extraction:** Assign labels for each word in the sentence, e.g., `B-PER`.
3) **Model Initialization and Training:** Initialize a CRF model using `sklearn` and train it using the extracted features and labels.
4) **Feature Extraction for Testing and Validation Data:** Extract features for words in the testing and validation datasets.
5) **Prediction:** Use the trained CRF model to predict the labels for the testing and validation datasets.
6) **Evaluation:** Compute evaluation metrics for the testing and validation data, including:
   - Precision
   - Recall
   - F1-score
   - Accuracy

Evaluation Results: The evaluation metrics for testing and validation data can be summarized in a tabular format:

Identify applicable funding agency here. If none, delete this.

Test data:

	precision	recall	f1-score	support
B-EVE	0.97	0.93	0.95	2400
B-LAN	0.99	0.77	0.87	183
B-LOC	0.94	0.93	0.93	10562
B-MON	0.98	0.85	0.91	228
B-NUM	0.94	0.90	0.92	1809
B-ORG	0.97	0.95	0.96	14331
B-PER	0.97	0.90	0.93	10531
B-TIME	0.98	0.97	0.97	14126
I-EVE	0.94	0.90	0.92	5313
I-LAN	1.00	0.80	0.89	5
I-LOC	0.97	0.97	0.97	7979
I-MON	0.99	0.96	0.97	383
I-NUM	0.94	0.97	0.95	688
I-ORG	0.97	0.97	0.97	20922
I-PER	0.97	0.93	0.95	9911
I-TIME	0.97	0.99	0.98	50680
O	0.99	0.99	0.99	354672
accuracy			0.98	504723
macro avg	0.97	0.92	0.94	504723
weighted avg	0.98	0.98	0.98	504723

Fig. 1: CRF Test Evaluations

Validation data:

	precision	recall	f1-score	support
B-EVE	0.97	0.91	0.94	2682
B-LAN	0.99	0.75	0.85	199
B-LOC	0.93	0.92	0.93	11687
B-MON	0.98	0.82	0.89	252
B-NUM	0.92	0.88	0.90	1994
B-ORG	0.96	0.95	0.96	15908
B-PER	0.96	0.88	0.92	11650
B-TIME	0.98	0.96	0.97	15751
I-EVE	0.93	0.88	0.90	5902
I-LAN	1.00	0.57	0.73	7
I-LOC	0.96	0.96	0.96	8832
I-MON	0.99	0.93	0.96	423
I-NUM	0.92	0.96	0.94	737
I-ORG	0.96	0.96	0.96	23223
I-PER	0.96	0.92	0.94	10920
I-TIME	0.97	0.99	0.98	56485
O	0.99	0.99	0.99	395626
accuracy			0.98	562278
macro avg	0.96	0.90	0.92	562278
weighted avg	0.98	0.98	0.98	562278

Fig. 2: CRF Valid Evaluations

Example Sentence: Consider the following sentence in Arabic:

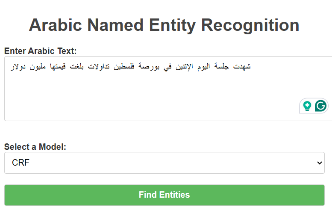

Fig. 3: CRF Example

Results:

Entity	Value
O	شهدت
O	جلسة
B-TIME	اليوم
O	الإثنين
O	في
B-ORG	بورصة
I-ORG	فلسطين
O	تداولات
O	بلغت
O	قيمتها
B-MON	مليون
I-MON	دولار

Fig. 4: CRF Result

B. Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a type of deep learning architecture designed to handle and transform sequential data inputs into corresponding sequential data outputs. Sequential data encompasses various forms, including words, sentences, and time-series information, where the elements are interconnected through intricate semantic and syntactic relationships. An RNN functions as a computational framework comprising numerous interlinked components, emulating the human ability to convert sequential data, such as translating text between different languages[3].

Main Steps:

1) **Create Vocabulary and Label Mappings:** Unique words and tags are assigned a number (index), with special numbers for padding
2) **Convert Data to Numbers:** Words and labels in the sentences are converted to their respective numbers using the created mappings.
3) **Pad Sequences to Equal Length:** All sentences and labels are padded or truncated to a fixed length (100), ensuring uniformity for model input.
4) **Model Initialization and Training:** Initialize a RNN model using `keras` and train it.
5) **Prediction:** Use the trained model to predict the labels for the testing and validation datasets.
6) **Evaluation:** Compute evaluation metrics for the testing and validation data, including:
   - Precision
   - Recall
   - F1-score
   - Accuracy

Test Metrics:

- **Test Loss:** 0.04014710709452629
- **Test Accuracy:** 0.962664635181427

Validation Metrics:

- **Validation Loss:** 0.04141910746693611
- **Validation Accuracy:** 0.9625218820571899

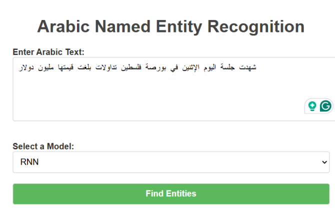

Fig. 5: RNN Example

Results:

Entity	Value
O	شهدت
B-EVE	جلسة
I-TIME	اليوم
I-TIME	الإثنين
O	في
B-ORG	بورصة
I-ORG	فلسطين
O	تداولات
O	بلغت
O	قيمتها
B-MON	مليون
I-MON	دولار

Fig. 6: RNN Results

C. Decision Tree

A decision tree is a form of supervised learning algorithm utilized in machine learning to model and forecast outcomes based on input data. It is structured like a tree, where each internal node evaluates an attribute, each branch represents a value of that attribute, and each leaf node signifies the ultimate decision or prediction. This algorithm is classified as a supervised learning technique and can be applied to address both regression and classification challenges[3].

Main Steps:

1) **Feature Extraction for Training Data:** Extract features for each word in the sentence, which include:
   - The word itself.
   - Whether the word is the first word in the sentence.
   - Whether the word is the last word in the sentence.
   - The first, second, and third prefixes of the word.
   - The first, second, and third suffixes of the word.
   - Whether the word is numeric.
   - Whether the word contains any digits.
   - The ”shape” of the word, where Arabic characters are replaced with ”X” and digits with ”d”.
2) **Label Extraction:** Assign labels for each word in the sentence, such as `B-PER`.
3) **Convert Features to a Format Suitable for scikit-learn:** The features extracted for each word in the sentence are converted into a format that can be used by scikit-learn for machine learning tasks. This is achieved using the
4) **Model Initialization and Training:** Initialize DecisionTreeClassifier using `sklearn` and train it using the extracted features and labels.
5) **Feature Extraction for Testing and Validation Data:** Extract features for words in the testing and validation datasets.
6) **Prediction:** Use the trained model to predict the labels for the testing and validation datasets.
7) **Evaluation:** Compute evaluation metrics for the testing and validation data, including:
   - Precision
   - Recall
   - F1-score
   - Accuracy

Test Metrics:

- **Test Accuracy:** 0.872209276764124
- **Test Recall:** 0.5432229600633042
- **Test Precision:** 0.63736973285719
- **Test F1:** 0.5710763374096245

Validation Metrics:

- **Validation Accuracy:** 0.8765702371644514
- **Validation Recall:** 0.5462055286842342
- **Validation Precision:** 0.6380716900907505
- **Validation F1:** 0.5723042541505168

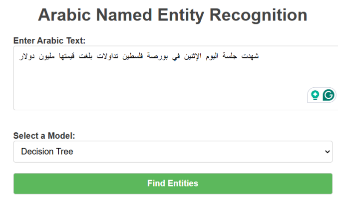

Fig. 7: Decision Tree Example

Results:

Entity	Value
O	شهدت
O	جلسة
B-TIME	اليوم
I-TIME	الإثنين
O	في
B-ORG	بورصة
I-ORG	فلسطين
O	تداولات
O	بلغت
O	قيمتها
I-MON	مليون
I-MON	دولار

Fig. 8: Decision Tree Result

D. Naive Bayes

The Naïve Bayes algorithm is a supervised learning technique grounded in Bayes’ theorem, primarily employed for addressing classification challenges. It is particularly effective in text classification tasks that involve high-dimensional training datasets. The Naïve Bayes Classifier is recognized as one of the simplest yet most efficient classification algorithms, facilitating the development of rapid machine learning models capable of making swift predictions. As a probabilistic classifier, it operates by predicting outcomes based on the likelihood of an object[4].

Main Steps:

1) **Feature Extraction for Training Data:** Extract features for each word in the sentence, which include:
   - The word itself.
   - Whether the word is the first word in the sentence.
   - Whether the word is the last word in the sentence.
   - The first, second, and third prefixes of the word.
   - The first, second, and third suffixes of the word.
   - Whether the word is numeric.
   - Whether the word contains any digits.
   - The ”shape” of the word, where Arabic characters are replaced with ”X” and digits with ”d”.
2) **Label Extraction:** Assign labels for each word in the sentence, such as `B-PER`.
3) **Convert Features to a Format Suitable for scikit-learn:** The features extracted for each word in the sentence are converted into a format that can be used by scikit-learn for machine learning tasks. This is achieved using the
4) **Model Initialization and Training:** Initialize MultinomialNB using `sklearn` and train it using the extracted features and labels.

5) **Feature Extraction for Testing and Validation Data:** Extract features for words in the testing and validation datasets.
6) **Prediction:** Use the trained model to predict the labels for the testing and validation datasets.
7) **Evaluation:** Compute evaluation metrics for the testing and validation data, including:
   - Precision
   - Recall
   - F1-score
   - Accuracy

Test Metrics:

- **Test Accuracy:** 0.8102440096724555
- **Test Recall:** 0.4808387415465718
- **Test Precision:** 0.47164107148267054
- **Test F1:** 0.4567870870568938

Validation Metrics:

- **Validation Accuracy:** 0.8141777430284076
- **Validation Recall:** 0.48862108141621924
- **Validation Precision:** 0.4930946666029553
- **Validation F1:** 0.46817488860237034

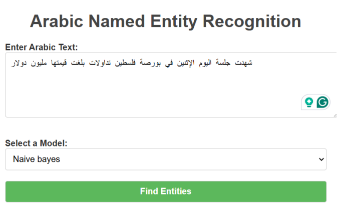

Fig. 9: Naive Bayes Example

Results:

Entity	Value
O	شهدت
O	جلسة
B-TIME	اليوم
I-ORG	الإثنين
O	في
B-ORG	بورصة
I-ORG	فلسطين
O	تداولات
O	بلغت
O	قيمتها
O	مليون
I-MON	دولار

Fig. 10: Naive Bayes Result

References

[1] A Survey on Arabic Named Entity Recognition. *arXiv*, https://arxiv.org/abs/2302.03512.
[2] Guellil, Arabic Natural Language Processing: An Overview. *ScienceDirect*, https://www.sciencedirect.com/science/article/pii/S1319157818310553.
[3] RNN *Amazon*, https://aws.amazon.com/what-is/recurrent-neural-network/#:~:text=A%20recurrent%20neural%20network%20(RNN,complex%20semantics%20and%20syntax%20rules..
[4] Decision Tree in Machine Learning *GeeksForGeeks*, https://www.geeksforgeeks.org/decision-tree-introduction-example/.
[5] Naïve Bayes *JavAtPoint*, https://www.javatpoint.com/machine-learning-naive-bayes-classifier.