Arabic and Arabic Transliterated Named Entity Recognition with Transform-Based Approach

Omar Ahmed
Computer Engineering
Ain Shams University
Cairo, Egypt
omar.ahmed.abdullah@gmail.com

Omar Hatem
Computer Engineering
Ain Shams University
Cairo, Egypt
omarhatem221@gmail.com

Abdulrahman Atef
Computer Engineering
Ain Shams University
Cairo, Egypt
boda998@yahoo.com

Shimaa Mohamed
Computer Engineering
Ain Shams University
Cairo, Egypt
shimaamohamed18286@gmail.com

Doaa Hesham
Computer Engineering
Ain Shams University
Cairo, Egypt
doaahesham1399@gmail.com

Abstract—Named Entity Recognition (NER) is a serious subtask of Natural Language Processing (NLP) tasks which is used in many applications such as data retrieval, machine translation, question answering, and text classification by locating and classifying the named entities in input text into multiple categories differ from a model to another. Arabic named entity recognition (ANER) is more difficult than other languages, because deep learning models require large amount of training data, which is extremely insufficient in Arabic language. Also, there is no capitalization in the Arabic script, so Arabic language can be ambiguous. In addition, the Arabic language is known for its rich morphology due to the concatenation of affixes and clitics on a stem. Furthermore, the insufficiency of lexical resources inhibits Arabic Natural Language Processing (NLP) research including NER. In this paper, we are going to discuss the difficulties in ANER, and the latest papers and approaches that try to deal with the problem and solve it, the results of our application that contains from pre-trained Arabic Bidirectional Encoder Representations from Transformers (ArabERT) and classification layer based on dataset contains 52 named entities for fine-tuning. The model achieves an F1 score of 78% testing on ANER Corp, outperforming Arabic current solutions which we would discuss, then the model is deployed into website application to be user-interface support Arabic Transliterated (Franco), Arabic sentences or both.

Keywords—Named Entity Recognition, Arabic named entity recognition, Bidirectional Encoder Representations from Transformers, Franco, Arabic Bidirectional Encoder Representations from Transformers, Natural Language Processing.

I. INTRODUCTION

Deep learning (DL) is an Artificial Intelligence (AI) function that behaves like the methodology of the human brain uses for taking decisions by creating patterns and relationships between data [4], DL is a subfield of Machine Learning (ML), commonly used for recognizing speech, making decisions, machine translating languages. Natural Language Processing (NLP) is an AI specialization area that deals with human language, and working on understanding and generate the human language, by making the computer understands the human language. NLP is used in Named Entity Recognition by understanding the language and the text given to the model then identify the entities in the given input like (names, location, organizations, etc.). [9] NER is a subtask of Natural Language Processing (NLP) that used for locating and classifying named entities mentioned in input text into certain named entities, we categorize the entities into 52 labels such as (Scientist, Engineer, Nation, athlete, Educational, Food, Drug, etc.). English named entity recognition recently have various solutions which supply powerful entity recognition such as Stanford CoreNLP, spaCy, and NLT, but Arabic named entity recognition (ANER) is more difficult than other English and other languages, because large amount of data is used while training of deep learning models, which is extremely insufficient in the Arabic language. In addition to the ambiguity in Arabic results from different meaning of the same word, in case changing “التشكيل” to “تشكيل” Of the word. We made an Arabic Named Entity Recognition model which takes sentences from the user, whether it is Arabic, Franco which is a new format of the Arabic language; this format uses English letters to express Arabic words [10], or mix between them and outputs the entities, which are 52 entities. This model is deployed to a website, this website is built using flask, html, CSS, some php, JavaScript. It is deployed on google cloud service which offers superb services for the price.

II. MOTIVATION

Arabic is a complex language that poses many challenges to Natural Language Processing (NLP). This is due to three important factors. First: Arabic has a rich inflectional and cliticization morphology system, for example, the Modern Standard Arabic (MSA) word، وسیکرونها (wsytbwnA (wa+sa+y+ktbwna+hA) ‘and they will write it’ has two proclitic (a conjunction and a tense marker), one circumfix (showing agreement in person, gender and number) and one enclitic (a direct object). Second: Arabic has a high degree of ambiguity resulting from its diacritic optional writing system and common deviation from spelling standards. The Standard Arabic Morphological Analyzer (SAMA) [16] produces 12 analyses per MSA word on average. Third: Arabic has several modern dialects that significantly diverge from MSA, which is the language of the news and of formal education. Using NLP tools built for MSA to process dialectal Arabic (DA) is possible but is plagued with very low accuracy: for example, a state-of-the-art MSA morphological analyzer only has 60%
coverage of Levantine Arabic verb forms. In the presence of these challenges, there is a need for tools that address fundamental NLP tasks for MSA and the dialects. Arabic named entity recognition (ANER) as mentioned before has major issue that hinders Arabic NLP research including NER has lack of sufficient lexical resources. There aren’t many tools (mobile apps, websites...etc.) in ANER that have good accuracy and good results.

III. RELATED WORK

There are several applications and libraries that support Natural Language Processing (NLP) tasks such as (text tokenization, dependency parsing, lemmatization, text classification, named entity recognition). When searching for solutions deal with named entity recognition (NER), several solutions are found with different features, accuracy, speed, usability, supportability for Arabic. As we are keen on Arabic NER, so we could categorize the solutions into two main parts: applications did not support Arabic NER - applications support Arabic NER.

StanfordNLP, NLTK, OpenNLP, SpaCy, Gate and MonkeyLearn are from examples of applications that don’t support Arabic. Madamira, Farasa, Camel Tools are from the most effective solutions that support NLP tasks for Arabic language. We tested all these apps to find all their Pros and Cons, so more details about Arabic solution would be indicated.

Farasa tool [1] is a full package language processing supports Web API and user-interface supporting Arabic language processing such as segmentation, Spell checker, Part of speech tagging, Lemmatization, Discretization, and Arabic Named Entity Recognition, main problem of using this model in ANER that the user-interface highlight the entities only without giving classification of every token, with paid Web API.

CAMEL tools [11] supports multiple language processing tasks such as segment analysis, and Arabic Named entity Recognition, CAMEL uses Pre-trained AraBert modelIV0.1 making fine-tuning with fully connected layer have 4 output layers, categorizing the tokens into four categories (Person, Location, Organization, Miscellaneous), also uses Farasa tokenizer for tokenizing the words of the text, providing a model with four main problems, does not support user-interface, to use NER model all the models with size 1.8 GB must be downloaded, support 4 entities only, the length of the sentence should be maximum 256 words.

Madamira [14] is considered as user-interface of CAMEL tools model, as the website model depends on the fin-tuned model of CAMEL, supporting extra features such as English named entity recognition, tokenization, and Lemmatization, so it does not provide more progress in the domain of Arabic named entity recognition.

Automatically developing Fine-grained ANER Copus [7], the paper provides an approach for developing large-fine grained Named Entity (NE) corpus and gazetteer automatically using the high potential of Wikipedia articles, finally uses learning NE with the automatically developed corpus to overcome the limitation and traditional Arabic tasks.

Arabic named entity recognition using deep learning approach [6], deep learning model based on the architecture of the bidirectional Long Short-Term Memory (LSTM), and Conditional Random Fields (CRF) and experimented with various commonly used hyperparameters, getting two sources of information about input words: pre-trained word embeddings and character-based representations and eliminated the need for any task-specific knowledge or feature engineering, achieving F1 Score 90.6% but in ANER Corp after training the model with train dataset portion.

The current solutions weren't enough to fulfill the stakeholder needs that can be represented by these main points:

- Ready software solutions (mobile apps, website, desktop apps...etc.)
- Support for many entities like (scientist, athlete...etc.)
- Model which has high enough accuracy.
- Providing more information about each entity, for example, highlighting the entity and gives him a link to Wikipedia.
- Support for Franco (transliterated Arabic).
- Support for slang Arabic.
- Support for Modern Standard Arabic (MSA).
- Support for text that has mix of slang Arabic and Franco and MSA.
- Speed and Reliable tool.

As we saw before, the solutions may not provide ANER already, and the ones that support it do not support all the needs of the stakeholders. Our model that is deployed as a website supports all the needs of the stakeholders.

IV. METHODOLOGY

There are main success indicators, which we seek to achieve, as this success indicators are not supported in current solutions such as supporting Franco text, fifty-two named entity recognition, and user-interface connected with the Wikipedia API to search by the extracted entities in Wikipedia, so a certain methodology is driven to achieve these targets.

A. DATASETS

We used Fine-grained Arabic Named Entity [2] Corpora. Which we got from “king Abd El-Aziz University”, the dataset contains 500k tokens, which is gathered from Wikipedia based articles and manually annotated. To test our we used NewsFANE_Gold [2] dataset which we got also from “king Abd El-Aziz University” which contains 170k manually tokens collected from newswire-based corpus.

B. PREPROCESSING DATA

1) General data preprocessing

We start with removing the Unicode characters, newline characters, tabs characters from the dataset. Then, we divide it into sentences, after preprocessing we get dataset of 2D list, every row has one column for words or sentence, and the other for tags.

2) BERT Specific Preprocessing

In order to make the dataset on the acceptable format for BERT, we create Pytorch dataset class, by taking every sentence and tokenize the words for it, then we add the Special Bert token ([CLS], [SEP], etc..), then add some padding tokens, as Bert wants all sentences to have the same number of tokens.

So, we analyze our dataset to use the most suitable maximum sentence length. After analyzing the dataset, we found that
most of the sentences in our dataset have length smaller than 256 tokens, so we chose that.

3) Tokenization
For tokenization, we used AraBERT [3] tokenizer (Version, 0.2) which is one of the most outperforming tokenizers available in Arabic language. The tokenizer consists of 64k tokens including 4k used tokens to allow further pre-training of the model, the tokenizer checks every word in the input sentence, if exists in vocabulary, the tokenizer substitute the word with the corresponding unique identifier in the vocabulary, else if the word does not exist in the vocabulary of the tokenizer, it divides the word into parts and tokenize each part individually.

C. ARCHITECTURE
The supported input text to the application is Arabic, Franco, or mixed from them, then process the input to be form of the accepted text of google input tools, then the language detector classifies the text into Arabic word or Franco word, then google input is used to translate Franco words into Arabic to be preprocessed by the next module by dividing the large sentences to be classified by our Transform-based model as presented in figure IV.1.

D. IMPLEMENTATION
1) Preprocessing text block
Preprocess the text to remove new lines, spaces, and emojis from the input text, to convert the text into pure text could be translated be Google Input API

2) Language Detector
Classify every word of the sentence, if the word is Franco, so it is requested to Google Input API to be translated into Arabic, else if the word is Arabic so it is given to the preprocessing script before fine-tuned model.

3) Google Input API
Google Input API is totally free API provided by google serviced, which is used for translating Franco words into Arabic words, by requesting the API ‘inputtools.google.com’ by Franco word, then receive the Arabic translated word, which is used as input for preprocessing script of fine-tuned model.

4) Preprocessing script
The selected maximum length of input text to the fine-
tuned model is 256 words, so if the input text is larger than this threshold, we divide the text into sentences with maximum selected length. If the sentence is smaller than 256 words, then padding tokens is used to unify the sentence length.

5) AraBERT Pre-trained Model
The AraBERT model is BERT [5] based model, is trained on manually scrapped articles from Arabic news websites, in addition to, they used to large Arabic Corpora 1.5 billion words Arabic Corpus [12] and OSIAN: the Open-Source International Arabic News Corpus [13] giving a total of 70 million sentences corresponding to 24GB of text.

The pre-training process of AraBERT is made by employing masked language modeling, by randomly masking 15% of the input tokens, and trains the model trying to predict these tokens. AraBERT also employed a task that helps the model to understand the relationships between two sentences, called Next Sentence Prediction which is used for tasks which need understanding of the language.

6) Fine-Tuned Model
We used AraBeT-V02 base pre-trained model, then we added a Fully Connected layer with 102 output nodes, to classify the tokens, with every node representing a specific class. Then to get the predicted class, we choose the class with the highest probability.

7) Deployment
The website provides an easy way for users to interact with and can easily insert sentences and get the result of the detected entities from our model in a user-friendly interface providing quick access to our service. The website is accessed anywhere across the world providing that internet access is present. The deployment process is mainly divided into three main modules, 1) Interface: implemented using HTML, CSS, JavaScript, jQuery, and some PHP. The user can visit the service page on the website or just hit get started to start using and testing our service, 2) Backend: For the backend of our website, we used flask [8] to make our http requests through. As Flask is a web application framework written in Python. It has multiple modules that helps developers for writing applications without worry about thread or protocols management details and other task, giving various choices for the developing process of the web application using tools and libraries. 3) Hosting: hosted in google cloud service, presenting flexible, fast, stable web application.

V. RESULTS AND DISCUSSION
In current Arabic solutions mainly supported entities is four entities (person, location, organization, miscellaneous), achieving accuracy of F1 score with range of (65% - 75%) when the model is tested against a dataset different of trained dataset, as our target to make an application outperforms current available solutions.
Table V.1 results of testing through ten different epochs

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1</td>
<td>81%</td>
<td>87%</td>
<td>84%</td>
</tr>
<tr>
<td>Run 2</td>
<td>85%</td>
<td>86%</td>
<td>86%</td>
</tr>
<tr>
<td>Run 3</td>
<td>85%</td>
<td>89%</td>
<td>87%</td>
</tr>
<tr>
<td>Run 4</td>
<td>85%</td>
<td>89%</td>
<td>87%</td>
</tr>
<tr>
<td>Run 5</td>
<td>85%</td>
<td>88%</td>
<td>86%</td>
</tr>
<tr>
<td>Run 6</td>
<td>86%</td>
<td>89%</td>
<td>87%</td>
</tr>
<tr>
<td>Run 7</td>
<td>87%</td>
<td>89%</td>
<td>88%</td>
</tr>
<tr>
<td>Run 8</td>
<td>86%</td>
<td>89%</td>
<td>88%</td>
</tr>
<tr>
<td>Run 9</td>
<td>86%</td>
<td>90%</td>
<td>88%</td>
</tr>
<tr>
<td>Run 10</td>
<td>87%</td>
<td>90%</td>
<td>88%</td>
</tr>
</tbody>
</table>

We trained the model with wiki-Fine dataset with 500k tokens achieving fifty-two different entities, dividing the tokens into 80% training, 10% evaluation, 10% testing.

The testing process is made using the portion of testing in wiki-Fine dataset, and the total ANER Corp dataset, which contains 170k tokens used for testing.

A. Evaluation Metrics

- Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.
  \[ \text{Precision} = \frac{TP}{TP + FP} \]
- Recall is the ratio of correctly predicted positive observations to all observations in actual class - yes.
  \[ \text{Recall} = \frac{TP}{TP + FN} \]
- F1 Score is the weighted average of Precision and Recall.
  \[ \text{F1} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \]
  Where TP = True Positives, FP = False Positives, FN = False Negatives

B. Results

1) Testing results at validation set after every epoch

Table V.1 presents the values of evaluation metrics, through training the model with wiki-Fine dataset, making ten epochs in the fin-tuning of the model with selected value of learning rate = 5 x 10^-4. The training process is accomplished through Google Colaboratory service, using GPU of 15GB.

2) Testing with test portion of wiki-Fine dataset

Results in Table V.2 show how the model became powerful in classifications of tokens, as it reaches Recall of 97%, Precision of 98% and F1 score of 97%.

Table V.2 Results after testing model for wiki-Fine dataset

<table>
<thead>
<tr>
<th>Metric for our model</th>
<th>Wiki-Fine dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>97%</td>
</tr>
<tr>
<td>Precision</td>
<td>98%</td>
</tr>
<tr>
<td>F1</td>
<td>97%</td>
</tr>
</tbody>
</table>

Table V.3 Our model and CAMEL model with ANER Corp dataset

<table>
<thead>
<tr>
<th>ANER Corp Dataset</th>
<th>Our model</th>
<th>CAMEL Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>78%</td>
<td>81%</td>
</tr>
<tr>
<td>Precision</td>
<td>77%</td>
<td>84%</td>
</tr>
<tr>
<td>F1</td>
<td>78%</td>
<td>83%</td>
</tr>
</tbody>
</table>

3) Testing with ANER corp dataset

Despite that the scores of CAMEL tools are seemed to be larger than our score, but in fact ANER Corp is the dataset which CAMEL tools trained their model with, so it is reasonable to achieve metrics larger than our model in this dataset, but when comparing our model tested with wiki-fine achieving previous metrics, 97%, 98%, 97% of recall, precision, and F1 score against CAMEL tools with these scores, our model obviously outperforming CAMEL Tools accuracy.

The scores of Table V.3 not high as results in Table V.2 but that came from the previous dataset was a dataset which we fine-tuned the model with train portion and consists of Wikipedia, magazine articles, newspapers articles, in another hand the results in Table V.3 achieved by completely different dataset consists of mainly retweet datasets and articles, so the results are also amazing.

C. Discussion

We aimed to improve upon the existing NER tools and try to add some mandatory features for better use for our model and services in general. Our best contribution is that we support 50+ Entities in different fields representing five main categories (Person, Location, Organization, Weapon, Miscellaneous), as for the current tools for NER, most of them only support four Entities (Person, Location, Organization, Miscellaneous). Secondly, we supported sentences from any sizes. As most of the current solutions only support sentences with maximum size of 256 words. Lastly, our biggest concern, was to package all that in a user-friendly application, so that anyone can use it and make the most of it easily, without any difficulties or waste of time; As for our competitors, most of them provide only command line based (CLI) libraries, which are difficult to setup and use.

VI. CONCLUSION AND FUTURE WORK

We presented Our Arabic NER solution, a Fine-tuned Arabert model, a NER model trained to relatively large dataset (500k Tokens) compared to our competitors (170k tokens). An addition we support more than 50 entities, compared to only 4 entities supported by the other solutions. We also introduced a valid solution for the lack of NER solutions for Transliterated Arabic (Franco), by transliterating from Franco to Arabic. We packaged all that in a user-friendly UI and hosted our entire project on the internet, so that anyone anywhere can utilize our services easily.

Some Future work we aim to work on include:

- A model trained on dataset with more general data that is annotated with more classes (We currently support 50 class).
- Make a more specific models in different fields, to achieve the state or the art results in those fields (Medical, history, etc.…).
• Create a dataset of Arabic slang (for exp. Egyptian dialect) and train the model on these data to achieve better results for Slang Arabic data.
• Create a dataset of Transliterated Arabic (Franco), and train the model on these data to achieve better results for Franco data.

ACKNOWLEDGMENT
First and foremost, we are extremely grateful to our supervisors Prof. Dr. Ahmed Hassan and Eng. Sara Abd El-Aziz for their invaluable advice, continuous support, and patience during our graduation project. We want also to thank the AraBERT team for their effort in enriching the Arabic NLP community by providing BERT model for Arabic language.

REFERENCES