Custom-trained Named Entity Recognition (NER) in Lorimer's *Gazetteer* with Spacy

I. Project motivation.

This article is second in the series exploring automated Named Entity Recognition (NER) systems for the historical texts from the Lorimer's Gazetteer. Although NLTK, as mentioned in the first blogpost, provides one of the most well-recognized and used NER systems, there are several important reasons to explore other approaches due to NLTK's limitations when applied to the Lorimer's Gazetteer dataset.

Firstly, most state-of-the-art systems such as NLTK were trained on English language texts : although such systems can accurately identify English Named Entities (NEs), they perform worse when applied to non-English, transliterated words. The Lorimer's Gazetteer primarily contains geographical names that were transliterated from Arabic to English and NLTK tends to either incorrectly tag these words (a city tagged as a person) or not recognize them. Moreover, many NEs from the Lorimer's Gazetteer mostly appear in the historical context and do not have a large web footprint, which might make these words (i) to be underrepresented in the training data for most modern systems and, consequently, (ii) not properly recognized by these systems.

Moreover, in NLTK, types of recognizable NEs (e.g. location, organization) are predetermined and NLTK does not officially support custom training of the NER classifier to solve the above mentioned issues. In other words, it is not possible to train the system to recognize unfamiliar words or correct the NE labels (e.g. train to tag Oman not as a person, but a location). Also, even more importantly, it is not possible to introduce new Named Entity types such as tribe names or some environmental/cultural features such as religion. In order to address these limitations and create a more robust classifier, we introduced a custom-trained Spacy NER model for transliterated entities in the Lorimer's Gazetteer.

This project would be valuable to a larger research community in two ways. Firstly, recognition of transliterated entities, which describe foreign people, places etc., is useful for many applications such as cross language text retrieval and translation. This pipeline can be applied to other transliterated texts with changes to the training data.

Secondly, the Lorimer's Gazetteer is one of the most influential and well-documented sources on the Gulf States, Saudi Arabia and Persia: each text file in the Gazetteer corresponds to a certain location. Given this, extracting custom-trained entities (tribes, languages, environment features) for each of the locations from the Gazetteer would contribute to establishing a structure and connection across locations in the Middle East based on various environment, political, cultural, geographic features, which previously would be challenging due to time, labor constraints.

II. Current NER solutions for Non-English texts and our approach.

II A. Theoretical background: other approaches towards NER of transliterations.

Before exploring the technical implementation of the Spacy based system, it would be valuable to provide theoretical background into existing NER approaches for transliterated texts. We also provide an introduction to the Spacy library and compare features of NLTK and Spacy.

Notably, the NER models for Arabic and transliterated texts (from Arabic to English and vice versa) are still underdeveloped. As most scientific studies are conducted in English in almost all Arabic-speaking countries, there is no urgency to investigate NER for Arabic and transliterated texts for many areas such as bioinformatics, drug or chemicals (Shaalan,2013). Numerous state sponsored initiatives have been created to encourage greater participation from NLP researchers.

Currently, the task of recognizing transliterated entities (including English transliterations of Arabic words) has been dominated by NER models that devise an algorithm to tag foreign language entities (in this case, Arabic) using either (i) metadata from Wikipedia such as inter-wiki links, article categories, and cross language links or (ii) other parallel English-foreign language data. In the first approach, the model is trained on Wikipedia texts that are available in several languages and the NER of non-English and/or transliterated words are predicted based on the trained model. This approach has several limitations as Wikipedia information can be distributed unequally as few Wikipedia contributors tend to work on the same language. Moreover, articles can have different content depending on the language: some countries do not allow articles on certain topics or an article on a divisive topic such as Crimea contains different content depending on the language (e.g. Russian or English). Given these disparities, using Wikipedia articles available in several languages for training might not yield accurate results. Automatic translation seems as a possible solution for this approach, but also might yield inaccuracies.

The second approach that involves training NER models on annotated data seems to be more reliable, however, there is a noticeable unavailability of tagged corpora for transliterated entities from Arabic to English. Moreover, the majority of the tagged corpora for Arabic, in general, features modern web content, where historical names of locations, cities (that frequently appear in the Lorimer's Gazetteer) might be underrepresented. To address these limitations, in this project, we annotate the Lorimer's Gazetteer and produce rich training data that tags historical, transliterated NEs from the Lorimer's Gazetteer. This annotated dataset can further benefit other projects working on the historical or nonhistorical texts containing transliterated words.

II B. Technical implementation: Custom NER with Spacy.

What is Spacy?

To implement the second approach (discussed above), particularly, to build a custom NER system trained on the annotated data, we decided to use Spacy, an open-source library for advanced Natural Language Processing in Python designed for production use. Spacy, similarly to NLTK, provides a default model which can recognize named or numerical entities. However, in contrast to NLTK, spaCy also allows addition of arbitrary Named Entity categories to the NER model (can be same as default or new)and training and updating the model. Spacy's named entity recognition has been trained on the OntoNotes 5 corpus and it supports the following entity types:

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.

Figure 1. This table shows examples of named entities and their types supported by Spacy.

Differences between NLTK and Spacy

The table below illustrates main differences between two libraries.

NLTK	Spacy
NLTK is a string processing library. It takes strings as input and returns strings or lists of strings as output.	Spacy uses an object-oriented approach. When we parse a text, spaCy returns a document object whose words and sentences are objects themselves.
NLTK doesn't have support for word vector	Spacy has support for word vector
NLTK attempts to split the text into sentences In sentence tokenization, NLTK outperforms spaCy.	As spaCy uses the latest and best algorithms, its performance is usually good as compared to NLTK. spaCy constructs a syntactic tree for each sentence, a more robust method that yields much more information about the text in word tokenization and POS-tagging spaCy performs better,

III. System description

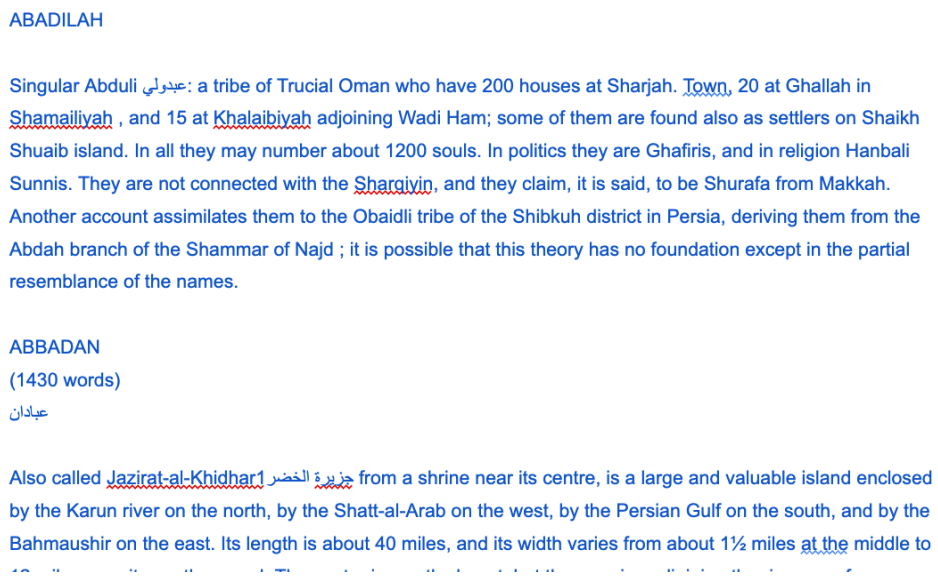
Compared to the NLTK model, since we manually train our classifier, we firstly create training data based on the Lorimer's Gazetteer using the NER Annotator and labelling the Named Entities and their tags. Then, we train the Spacy model on the created training data. Finally, we evaluate our model using a sample test file.

1. Create training data.

Creation of the training data has two stages: ii) create or select an input file that contains desired Named Entities that we want our model to recognize and ii) annotate the input file by tagging the target entities and converting it into a suitable training format.

1 A. Create a training input file (txt) that contains target Named Entities.

For this project, we annotate not all of the dataset (800+ text file) but a sample (4940 words) that contains all target NEs that we want our model to recognize.
This sample was created by merging paragraphs from some of the text files in the Lorimer's dataset. In figure 2, you can view a screenshot of the input file.



ABADILAH

Singular Abduli عبدولي: a tribe of Trucial Oman who have 200 houses at Sharjah. Town, 20 at Ghallah in Shamailiyah , and 15 at Khalaibiyah adjoining Wadi Ham; some of them are found also as settlers on Shaikh Shuaib island. In all they may number about 1200 souls. In politics they are Ghafiris, and in religion Hanbali Sunnis. They are not connected with the Sharqiyin, and they claim, it is said, to be Shurafa from Makkah. Another account assimilates them to the Obaidli tribe of the Shibkuh district in Persia, deriving them from the Abdah branch of the Shammar of Najd ; it is possible that this theory has no foundation except in the partial resemblance of the names.

ABBADAN
(1430 words)
عبادان

Also called Jazirat-al-Khidhar1 جزيرة الخضر from a shrine near its centre, is a large and valuable island enclosed by the Karun river on the north, by the Shatt-al-Arab on the west, by the Persian Gulf on the south, and by the Bahmaushir on the east. Its length is about 40 miles, and its width varies from about 1½ miles at the middle to

Figure 2. A screenshot from the txt input file used for training the Spacy model.

It can be noticed that paragraphs from both Abadilah.txt and Abbadan.txt files were included in the training input file as these paragraphs contain target geographic Named Entities such as Najd, Persian Gulf, which were not tagged correctly by NLTK and that we aim Spacy model to correctly recognize. The list of target NEs was provided by other student researchers and can be viewed here -it includes the names of main geographical locations - cities, countries and regions.

Returning to the input file and figure 2, note that these paragraphs also contain information about tribes - an additional Named Entity type that we would like our model to recognize. The final version of the input file prepared for annotation can be viewed here.

1 B. Annotate the training input file and convert it to a json format.

Note that our input file for training is in a txt format - we have to annotate this input file and transform it into a suitable json format for Spacy to process.

Spacy requires a training file to be a list of tuples or lists. Each tuple should contain the text and a dictionary; a dictionary has start, end indices of the Named Entity and the category or label of the Named Entity. In the figure 3, Elon Musk is a Person entity and London, Berlin are Loc entities.

```
TRAIN_DATA_EXAMPLE = [
    ('Who is Elon Musk?', {
        'entities': [(7, 15, 'PERSON')]
    }),
    ('I like London and Berlin.', {
        'entities': [(7, 13, 'LOC'), (18, 24, 'LOC')]
    })
]
```

Figure 3. An example of training data in the json format required by Spacy.

As we work with a large amount of data, manual annotation is not an efficient solution. We use the NER Annotator, an open source platform to annotate and save results to a Spacy's json format. NER Annotator requires the following steps:

1. Upload the input file - in our case we upload the input txt file from the earlier section.
2. Create Named Entity tags. As mentioned earlier, for the Spacy model, compared to the NLTK default tag list, we can create our own Named Entity tags such as Tribe or Religion. We can annotate a subset of the data to tag these custom entities, train our model based on the annotations and retrieve these custom entities across the whole dataset.
3. Annotate the text - we can annotate by line (can skip text) or the whole text as a string.
4. Export annotations as a json file.

See figure 4 to view a screenshot of the NER annotator interface.

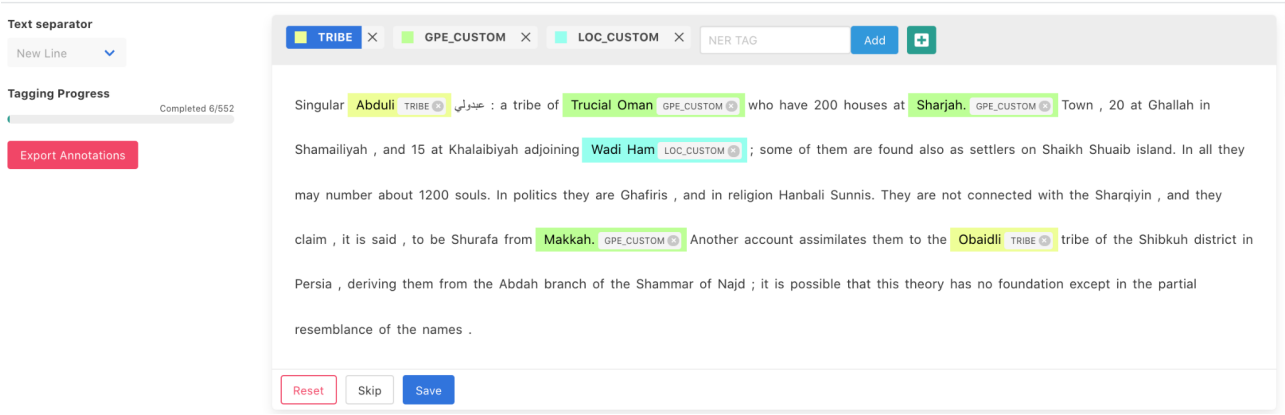


Figure 4. The NER Annotator interface. A user can tag the Named Entities with custom tags.

At this stage in the project, we introduce GPE_CUSTOM, LOC_CUSTOM, TRIBE Named Entities. 'GPE_CUSTOM' Named Entity tag refers to geo-political entities (e.g. cities, towns, countries) and is defined in the same way as the default GPE tag in Spacy; we add 'custom' suffix to differentiate two tags for potential finetuning possibilities in the future. Similarly, LOC_CUSTOM, refers to non-geo-political locations (e.g. plains, mountains, etc.). We also

introduce the custom TRIBE tag, which is not defined in Spacy or NLTK. In the future, we can also introduce other tags referring to religion, environment variables, animal types, etc.

As we finish annotation of the input file, we export our results in a json format and the resulting training file can be viewed in figure 6. You can view the full file here.

Import libraries.

After creating the training data, we can work on creating the Spacy NER model. Firstly, we import required libraries: we import the 'spacy' library for the NER recognition, 'json' library to handle reading and writing, 'random' library for randomizing the training data classes. We also load the 'en_core_web_sm' Spacy pipeline, which has a pre-trained NER model; the pipeline also has tagger, tokenizer, lemmatizer and other components. Finally, we import the 'displacy' library to visualize the Named Entity Recognition results.

```
#Download packages
!python3 -m spacy download en
!python3 -m spacy download en_core_web_sm

#Import libraries
import json
import spacy
import random
import en_core_web_sm
from spacy import displacy
```

Figure 5. Import libraries

Load training data.

As mentioned in the earlier section 'Creating training data', we have created a json training file that follows the Spacy format for training. The resulting file is a dictionary of lists with each list corresponding to a text segment (in our example, text is divided into segments by lines); each text segment has a dictionary of entities and their locations (start, end index) in the text.

```
# Load the training data
example_path='/content/drive/MyDrive/training_data1.json'
with open(example_path) as fp:
     training_data = json.load(fp)
```

```
training_data
```

```
{'annotations': [['Villages on the east shore of Abbadan island from the bifurcation of the Karun and Bahmanshir rivers to the mouth of the
   {'entities': [[30, 37, 'GPE_CUSTOM'],
     [73, 78, 'LOC_CUSTOM'],
     [83, 93, 'LOC_CUSTOM'],
     [135, 147, 'LOC_CUSTOM']]}],
  ['', {'entities': []}],
  ['150 mud houses; opposite Muhammareh Town. There are about 15 shops; Ballams and sailing craft are built, and water-pots, jars and Abas a
   {'entities': [[25, 41, 'GPE_CUSTOM'], [219, 231, 'TRIBE']]}],
  ['1 mile below Shakhat Haji Is-haq.',
   {'entities': [[13, 32, 'LOC_CUSTOM']]}],
  ['A term sometimes applied by mariners to the entire coastal region between the towns of Kuwait and Qatif in Eastern Arabia. Bedouins, how
   {'entities': [[87, 93, 'GPE_CUSTOM'],
     [98, 103, 'GPE_CUSTOM'],
     [107, 122, 'LOC_CUSTOM'],
     [261, 280, 'GPE_CUSTOM'],
     [357, 368, 'GPE_CUSTOM']]}],
  ['A branch of the Al Subaih section of the BaniKhalid tribe, found in Qatar and Bahrain: in the former they have 400 houses at Wakrah, and
   {'entities': [[68, 85, 'GPE_CUSTOM'],
     [125, 131, 'GPE_CUSTOM'],
     [181, 195, 'GPE_CUSTOM'],
     [425, 430, 'GPE_CUSTOM']]}],
  ['An important town on the Persian Coast at the entrance of the Persian Gulf; it constitutes in itself an administrative district and is s
   {'entities': [[25, 38, 'LOC_CUSTOM'],
     [62, 74, 'LOC_CUSTOM'],
     [220, 232, 'GPE_CUSTOM'],
     [280, 286, 'GPE_CUSTOM'],
     [319, 330, 'GPE_CUSTOM']]}],
  ['1 mile below Sha aibiyeh.', {'entities': [[13, 24, 'LOC_CUSTOM']]}],
  ['About 4 miles below Bakhakh and 15 miles from the mouth of the Bahmanshir.',
   {'entities': [[20, 27, 'GPE_CUSTOM'], [63, 73, 'GPE_CUSTOM']]}],
  ['3 miles below Al Bu Hamaid.', {'entities': [[14, 26, 'LOC_CUSTOM']]}],
  ['Villages on the north shore of Abbadan island from the bifurcation of the Karun and Bahmanshir rivers to the confluence of the Karun and
   {'entities': [[31, 38, 'GPE_CUSTOM'],
     [74, 79, 'LOC_CUSTOM'],
```

Figure 6. Output from the NER Annotator: training data in a json format specified by Spacy.

Using Spacy's default Named Entity Recognition.

In the next blogpost, we outline in detail the steps for setting up blank and pretrained Custom NER models with Spacy - full pipeline can be viewed here. To view a glimpse into the system, in this article, we show how Spacy's default NER pipeline, 'en_core' recognizes Named Entities. You can view the code in figure 7; we use the default model to recognize Named Entities from a file called 'abbas_bandar.txt' from the Lorimer's Gazetteer dataset. A detailed overview of the functions and steps will be provided in the next blogpost. We also print the list of default Named Entities recognized by the model.

```
# Load English tokenizer, tagger, parser and NER
nlp = en_core_web_sm.load()
#Load the en_core_web library and create the nlp pipeline
ner=nlp.get_pipe('ner')
#Read the test text
test_txt = open("/content/drive/MyDrive/abbas_bandar.txt")
text=test_txt.read()
text=text.replace("\n", " ")
#Process the text and recognize Named Entities
doc = nlp(text)
```

```
[Default Entities] =  ('CARDINAL', 'DATE', 'EVENT', 'FAC', 'GPE', 'LANGUAGE', 'LAW', 'LOC',
```

Figure 7. Load the default en-core pipeline and detect the Named Entities.

We then use the 'displacy' library to visualize the NER outputs - view the results in figure 8.

```
#Visualize the NER
displacy.render(doc, style="ent", jupyter=True)
```

Abbas PERSON are:— Class. Number. Total tonnage. Total hands employed. Baghlahs ORG 3 CARDINAL 1250 DATE 114 CARDINAL Ghunchahs ORG
CARDINAL 100 CARDINAL Mashuwahs ORG 15 CARDINAL 80 CARDINAL 90 CARDINAL Jolly-boats 10 CARDINAL 50 CARDINAL 50 CARDINAL
CARDINAL 2025 502 CARDINAL Apart from this small mercantile marine, and from native boats of other ports which call here, Bandar Abbas PERSON is dependent for
vessels to the number of 158 CARDINAL with a tonnage of 241000 CARDINAL entered the port. All but 8 CARDINAL were under the British NORP flag. Transport
10 CARDINAL horses and 350 donkeys QUANTITY ; there are no camels or mules. The town is dependent for pack-carriage on the surrounding district of Shamil GPE
DATE there is a large influx of Afghans NORP with camels and at that season the number of camels in the neighbourhood sometimes rises to 2000 DATE or more. Th
article on the Shamil District LOC Bandar Abbas PERSON is now connected with Hanjam island LOC by telegraph. The shore end of the cable is landed about ¾ o
PERSON is one CARDINAL of the Gulf Ports LOC and as such is subject to the Governor who has his head-quarters at Bushehr Town FAC . The local representat
entitled in theory to exercise official influence over the Kalantar of the Shamil District ORG , whose seat is at Ziyarat GPE ; but in practice the Kalantar ORG general
The Deputy-Governor is an unsalaried official, who pays a premium for his post and recoups himself as best he can by the collection of dues and taxes; the predecessor of the
DATE , and the highest annual DATE sum ever contracted for is said to have been 14000 CARDINAL Tumans LANGUAGE From the Deputy-Governors sources o
ORG , are of course excepted. His chief items of revenue are a shop tax called Asnafiyeh NORP اصناقیه at the rate of 7½ CARDINAL Tumans LANGUAGE per annu

Figure 8. Named Entity Recognition by default en-core pipeline and visualization of the results.

Notably, the default, non-customized pipeline for Spacy is also trained on English corpus, thus, similarly to NLTK, some transliterated Named Entities such as Bandar Abbas are mislabelled (labelled as a person instead of location). Fortunately, compared to NLTK, Spacy allows custom training of the model, therefore, we are able to improve the results. As mentioned earlier, custom NER model setup will be explained in the next article