ARIEL UNIVERSITY

Project Book

Named Entity Recognition Using Bert Model

By Dor Getter & Aiman Younis & Liad Ben Moshe

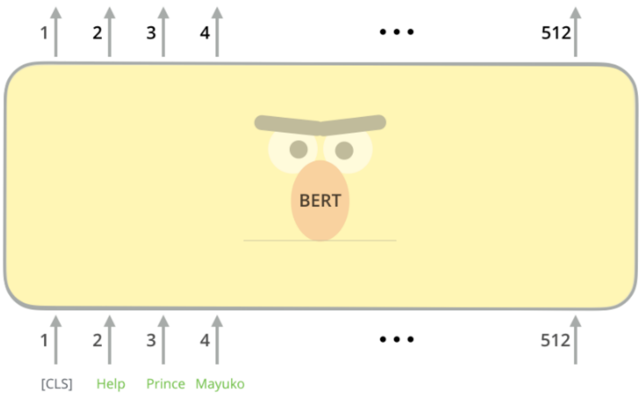


Table Of Contents

Background:

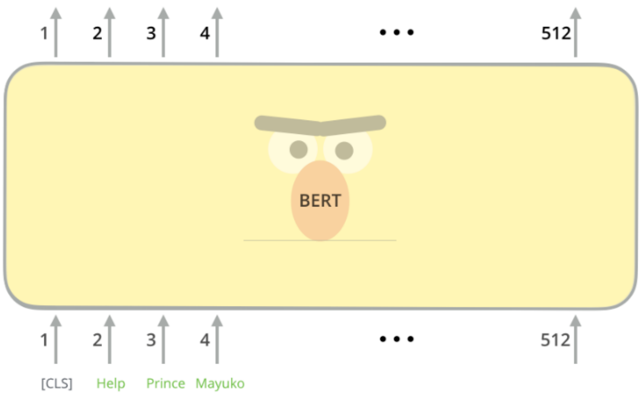
NER is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

How NER BERT model works:

BERT (Bidirectional Encoder Representations from Transformers).

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This contrasts with previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.



About the Project:

In our project we fine-tuned BERT models to perform named-entity recognition (NER) in three languages (English, Hebrew and Arabic).

The project provides web-based interface (written in REACT) that allows the user to enter text in one of the languages and get an output of the text labeled into entities, and a table that shows for each word in the text what is the entity, and the confidence score.

Our models can extract the following entities from the text:

For the English language:

'ang': Language
'geo': (Geography)
'per': (Person)
'gpe': (Geo-Political Entity)
'org': (Organization)
'tim': (Thermal Interface Materials)
'nat': (Nationality)
'art': (Artistic)
'eve': (Event)
'O' , 'PAD': other

For the Hebrew language:

'GPE' (geo-political entity)
PER (person)
LOC (location)
ORG (organization)
FAC (facility)
EVE (event)
WOA (work-of-art)
ANG (language)
DUC (product).

For the Arabic language:

'LOC': (location)
'PERS': (person)
'ORG': (organization)
'MISC': (miscellaneous)

Project goal:

We will show how to finetune the Bert model in many languages to do state-of-the art named entity recognition in English, Hebrew, and Arabic While making the use of models accessible with the help of a friendly user interface.

The Project Architecture:

Our project is made up of two main parts:

1) Back-end:

This section is entrusted with –

- Setting up the servers
- Running the information on the models
- Analyzing the results returned after running the model
- Arranging the information for use in the user interface.

This section mostly written in Python.

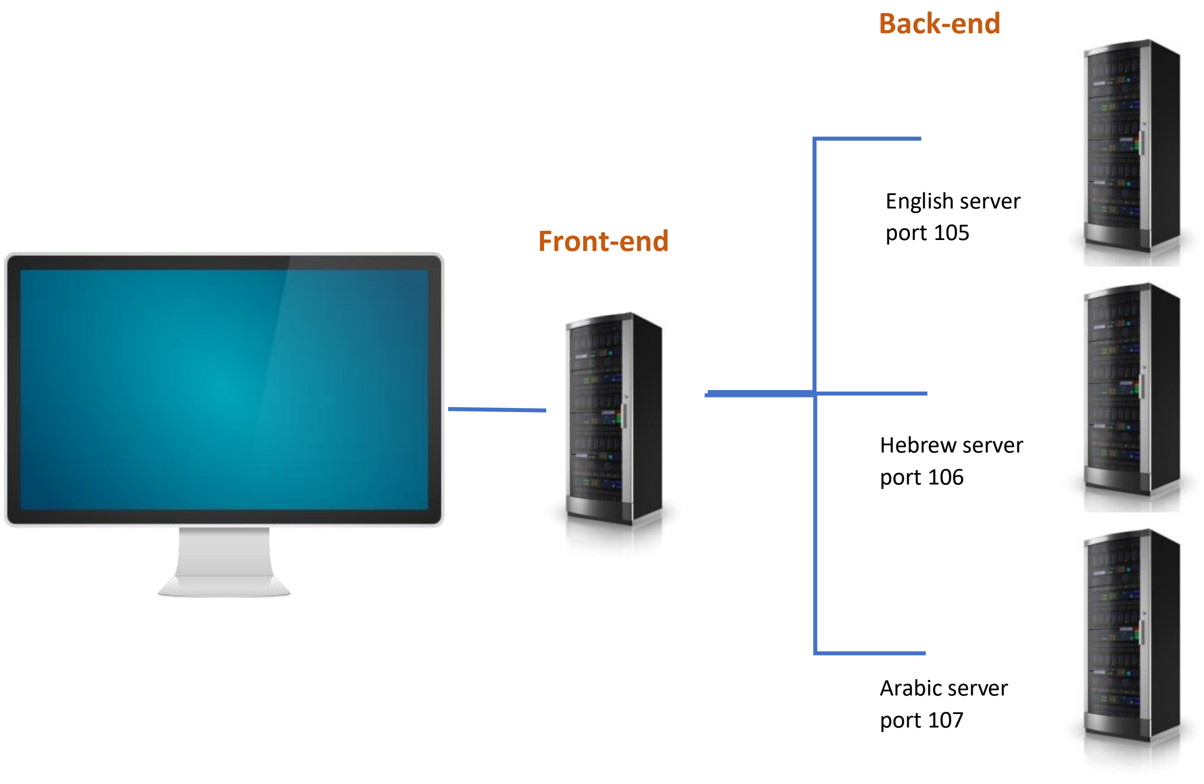
2) Front-end:

This section is entrusted with –

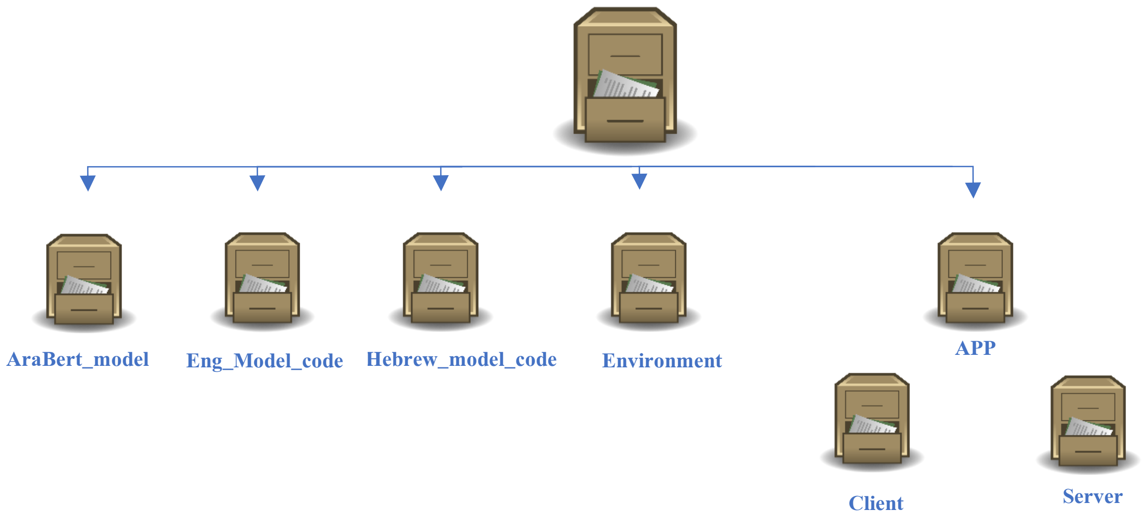
- Getting information from the user.
- Communicating with the servers sending and receiving data.
- Making the information accessible to the user.

This section mostly written in JavaScript using REACT.

To implement and combine the two parts the project is built with the help of four processes running in parallel.
Three for the three back-end servers (three to support our three languages) and one for the front-end.



Files Hierarchy:



APP folder -
contains the front & beck -end files.
Including the models and tokenizers files.

AraBert & Eng_Model & Hebrew_model folders -
containing the scripts needed for training the different models.

Environment Folder -
contains the .yml 's files needed for building the conda's environments.

Installation & Setup:

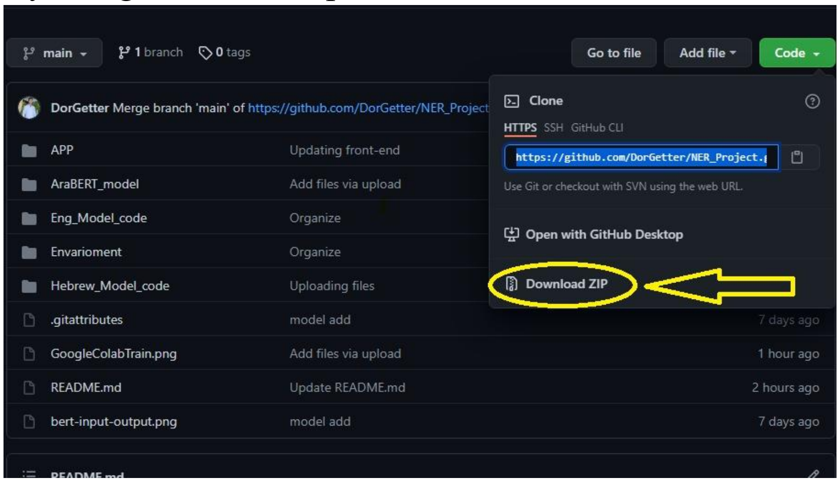
Step 1: Open the operation system

Make sure that you are running on Ubuntu 20.04 version.

- VM – please refer to https://ubuntu.com/download/desktop for downloading the image.
- After installation, please make sure you are up-to-date by using:
  - $ Sudo apt get update.
  - $ Sudo apt get upgrade.

Step 2: Cloning the repository

1. Navigate to the desired location to clone the project.
2. Clone the repository by:
   a. Using git clone command:
   ```
   $ git clone https://github.com/DorGetter/NER_Project.git
   ```
   b. By using download zip in the GitHub website:



DorGetter Merge branch 'main' of https://github.com/DorGetter/NER_Project

APP	Updating front-end	
AraBERT_model	Add files via upload	
Eng_Model_code	Organize	
Envarioment	Organize	
Hebrew_Model_code	Uploading files	
.gitattributes	model add	7 days ago
GoogleColabTrain.png	Add files via upload	1 hour ago
README.md	Update README.md	2 hours ago
bert-input-output.png	model add	7 days ago

Step 3: Creating Anaconda Environments:

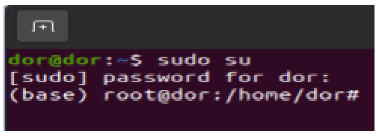
1. Download anaconda please refer to this guide:

#importent#

Since the environments we will create will use for opening ports you will have to get 'root' permissions.

So before proceeding into the guide below make sure you using your command line as 'root' by first executing the command:

```
$ sudo su
```



```
dor@dor:~$ sudo su
[sudo] password for dor:
(base) root@dor:/home/dor#
```

https://phoenixnap.com/kb/how-to-install-anaconda-ubuntu-18-04-or-20-04

2. Creating the environments using the .Yml files:

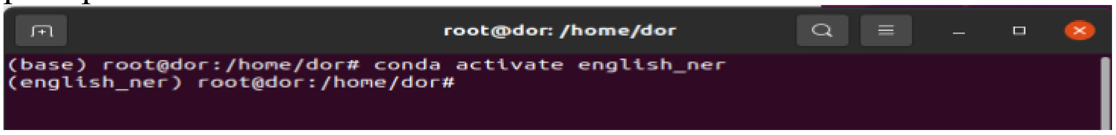
2.1. Open two terminals with root permission to the ‘Environment’ folder.
There you will find two .yml files:

‘aleph_ner.yml’ and ‘english_ner.yml’.

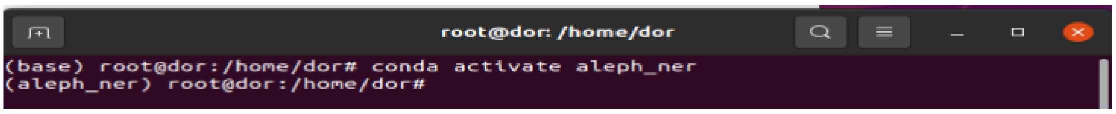
2.2. execute:
terminal 1: `$ conda env create --file aleph_ner.yml`
terminal 2: `$ conda env create --file english_ner.yml`

2.3. activate both environments:
terminal 1: `$ conda activate aleph_ner.yml`
terminal 2: `$ conda activate aleph_ner.yml`

Make sure that in the end of 2.3. you will get in the LHS of the command prompt the environment name.



```
root@dor: /home/dor
(base) root@dor:/home/dor# conda activate english_ner
(english_ner) root@dor:/home/dor#
```



```
root@dor: /home/dor
(base) root@dor:/home/dor# conda activate aleph_ner
(aleph_ner) root@dor:/home/dor#
```

Cheap Sheet for using Anaconda - https://docs.conda.io/projects/conda/en/latest/_downloads/843d9e0198f2a193a3484886fa28163c/conda-cheatsheet.pdf

Step 4: Downloading React Node.js & NPM:

1. Installing NPM in Ubuntu:

To install npm on Ubuntu Linux, login into your terminal as a **sudo** user and invoke the command below:
`$ sudo apt install npm.`

To be on the safe side when the installation is complete check:
`$ npm -- version`   # check that indeed you have version installed.
`$ node -- version`   # check that indeed you have version installed.

2. Install the npm dependencies for REACT:

2.1 Navigate to the ‘Client’ folder NER_PROJECT/APP/Client

2.2 Execute:

```
$ sudo npm install
$ sudo npm install styled-components
$ npm i react-tooltip
 $ npm install semantic-ui-react semantic-ui-css
```

Training & Fine tuning the models:

For each model we did training and finetuning process, we declared the hyper-parameters for each model, and we finetuned each model with specific dataset, and labels.

Fine tuning process is:

1) FINE-TUNNING means making small adjustments to achieve the desired performance.

2) Using the weights of a deep learning algorithm before getting a new model (the pretrained model) i.e.: bert-case-model, arbert-base-model.

3) The weights of all the other layers remain the same, we only want the weights in our new or modified layers to be updating.

4) After we do this, all that's left is just to train the model on our new data.

5) The weights from all the layers we kept from our original model will stay the same, and only the weights in our new layers will be updating.

6) At the end we will get a new model with our adjustments.

For Eng-BERT model:

The scripts for training and fine-tuning the English model is in NER_Project/Eng_Model_code directory:

https://github.com/DorGetter/NER_Project/tree/main/Eng_Model_code

The dataset for training and fine-tuning the model is in the 'dataset' directory.

- For training using the Google Collab use the Train_english_model_colab.ipynb file.
  make sure to upload the dataset to the Google Collab.
- For Training locally use the 'main_code.py' file.

For Arab-BERT model:

For more information of training and fine-tuning and Dataset you will find This link useful:

https://github.com/DorGetter/NER_Project/tree/main/AraBERT_model

We saved the model as object, to run it please check this link:

https://github.com/DorGetter/NER_Project/blob/283d448acb3ddea2d944a915d78ae9dc40cf83cf/APP/server/run_arab.py

For Aleph-ERT model:

The scripts for training and fine-tuning the English model is in NER_Project/Hebrew_Model_code directory:

```
$ python3 trainer.py –train-file dataset/dataset.csv –name
tokenizer_0_1_he.pkl
```

Fine tuning:

```
$ python3 trainer.py –train-file dataset/dataset.csv –name
tokenizer_0_1_he.pkl -- finetune
```

TRAINING PARAMETERS:

```
ner_training.py [-h] [--seed SEED] [--name NAME] --train-file TRAIN_FILE [--max-seq-len MAX_SEQ_LEN] [--finetune]
                [--num-epochs NUM_EPOCHS] [--batch-size BATCH_SIZE] [--learning-rate LEARNING_RATE] [--optimizer-eps
OPTIMIZER_EPS]
                [--weight-decay-rate WEIGHT_DECAY_RATE] [--max-grad-norm MAX_GRAD_NORM] [--num-warmup-steps
NUM_WARMUP_STEPS]

optional arguments:
  -h, --help          show this help message and exit

general:
  --seed SEED         seed for reproducibility
  --name NAME         name of directory for product

dataset:
  --train-file TRAIN_FILE
                path to train file
  --max-seq-len MAX_SEQ_LEN
                maximal sequence length

training:
  --num-epochs NUM_EPOCHS
                number of epochs to train
  --batch-size BATCH_SIZE
                batch size

optimizer:
  --learning-rate LEARNING_RATE
                learning rate
  --optimizer-eps OPTIMIZER_EPS
                optimizer tolerance
  --weight-decay-rate WEIGHT_DECAY_RATE
                optimizer weight decay rate
  --max-grad-norm MAX_GRAD_NORM
                maximal gradients norm

scheduler:
  --num-warmup-steps NUM_WARMUP_STEPS
                scheduler warmup steps
```

Important

After training each model a .pth / .bin files and .pkl files will be save in your local machine (or in case of training in Google Collab they will saved in your Google Drive folder).

For making use of the Project please copy those files to accordingly folders in the NER_Project/APP/server.

Eng_Model_code -

saving the model under the name:	'model.pth'
saving the .pkl file under the name:	'tokenizer_0_tags_1.pkl'

AraBERT_model -

saving the model under the name:	'pytorch_model.bin'
saving the .pkl file under the name:	'tokenizer_0_tags_1.pkl'

Hebrew_Model_code -

saving the model under the name:	'model_aleph.pth'
saving the .pkl file under the name:	'tokenizer_0_tags_1_aleph.pkl'

Running the Project:

Step 1: Running the beck-end servers:

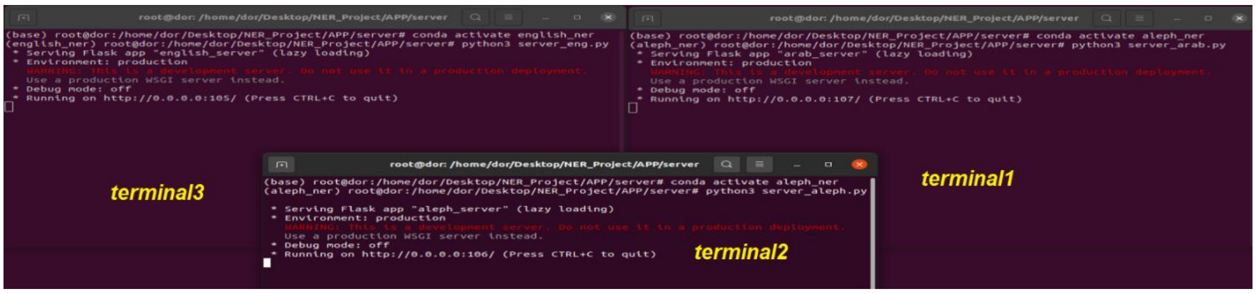
1. Open three terminals as 'root'
   (execute: $ sudo su)

2. Navigate with all terminals to the 'server' folder.
   ($ cd /APP/server)

3. Open the conda environments:
terminal 1 : $ conda activate aleph_ner
terminal 2 : $ conda activate aleph_ner
terminal 3 : $ conda activate english_ner

4. Open the servers:
terminal 1 : $ python3 server_aleph.py
terminal 2 : $ python3 server_arab.py
terminal 3 : $ python3 server_eng.py

Make sure all servers are running:



```
root@dor: /home/dor/Desktop/NER_Project/APP/server
(base) root@dor:/home/dor/Desktop/NER_Project/APP/server# conda activate english_ner
(english_ner) root@dor:/home/dor/Desktop/NER_Project/APP/server# python3 server_eng.py
 * Serving Flask app "english_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:105/ (Press CTRL+C to quit)
```

```
root@dor: /home/dor/Desktop/NER_Project/APP/server
(base) root@dor:/home/dor/Desktop/NER_Project/APP/server# conda activate aleph_ner
(aleph_ner) root@dor:/home/dor/Desktop/NER_Project/APP/server# python3 server_arab.py
 * Serving Flask app "arab_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:107/ (Press CTRL+C to quit)
```

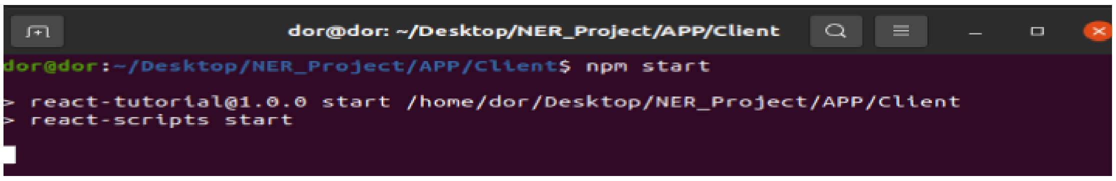
```
root@dor: /home/dor/Desktop/NER_Project/APP/server
(base) root@dor:/home/dor/Desktop/NER_Project/APP/server# conda activate aleph_ner
(aleph_ner) root@dor:/home/dor/Desktop/NER_Project/APP/server# python3 server_aleph.py

 * Serving Flask app "aleph_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:106/ (Press CTRL+C to quit)
```

Now all servers are running and waiting for data.

Step 2: Running the front-end server:

1. Open a terminal and navigate to 'Client' folder
   $ cd /APP/Client

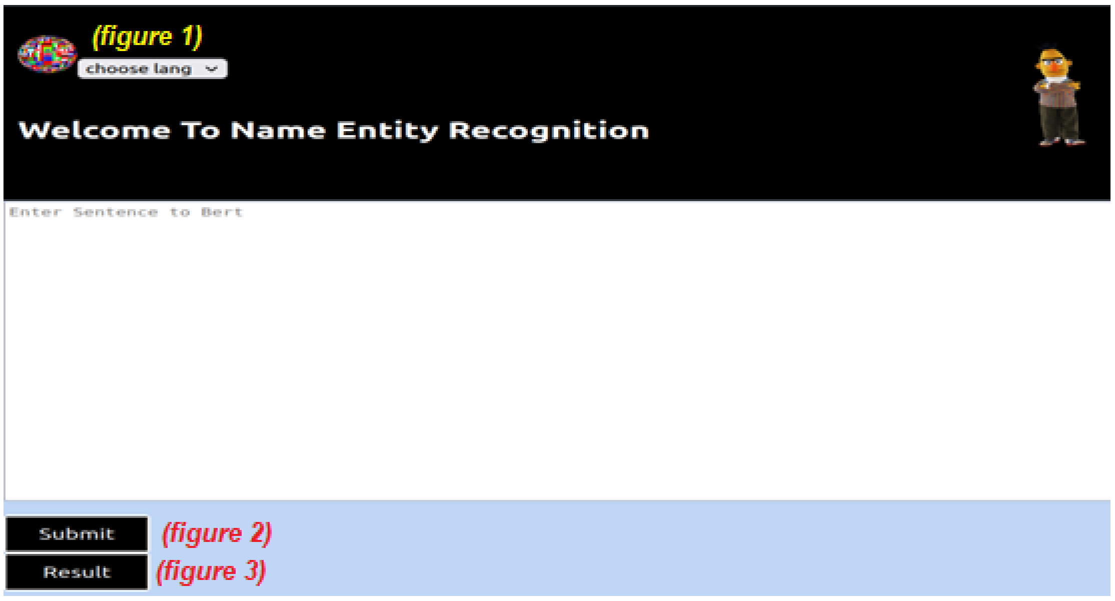
2. execute:
   $ npm start
this command will start the Client server and will open a local host in your web browser.



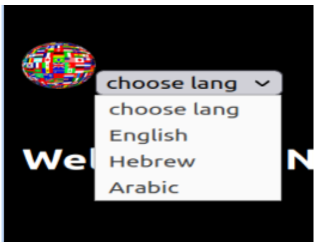
```
dor@dor: ~/Desktop/NER_Project/APP/Client
dor@dor:~/Desktop/NER_Project/APP/Client$ npm start

> react-tutorial@1.0.0 start /home/dor/Desktop/NER_Project/APP/Client
> react-scripts start
```

😉 Now the application is all set! all the servers are running. 😉

Using the web API:

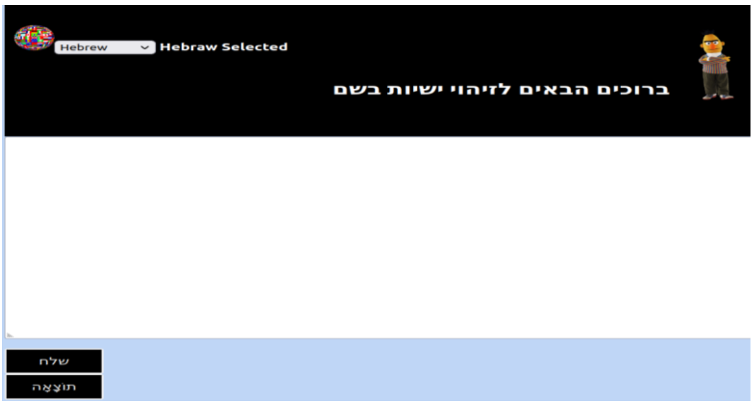


Buttons & Fields

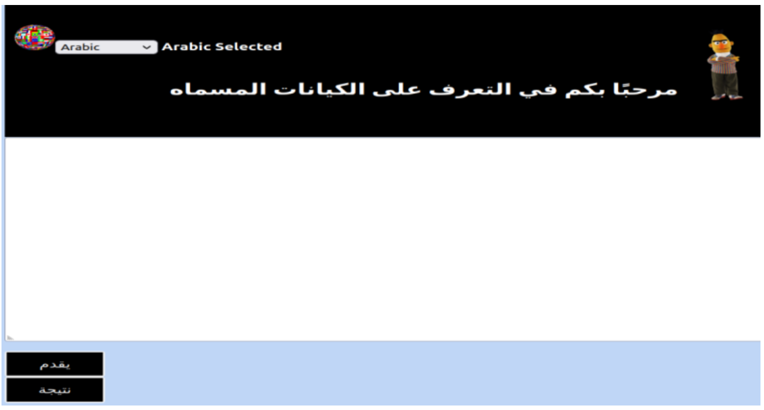


Language selection: (figure 1)
click on the drop-down for choosing the language you wish.
Clicking on the language will cause the language to change throughout the whole interface.

Clicking on Hebrew:

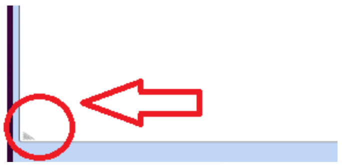


Clicking on Arabic:



Text Field: (White Box)
This is the place where you can enter the text that you wish to extract entities from.

The Text Box is auto-incrementing, but also you can use the sign in the LHS (English) or RHS (Hebrew / Arabic) to control the text box size.



Submit / שלח / يقدم
After choosing your desired language and entering the text click on the 'submit' button to run the text on the model.

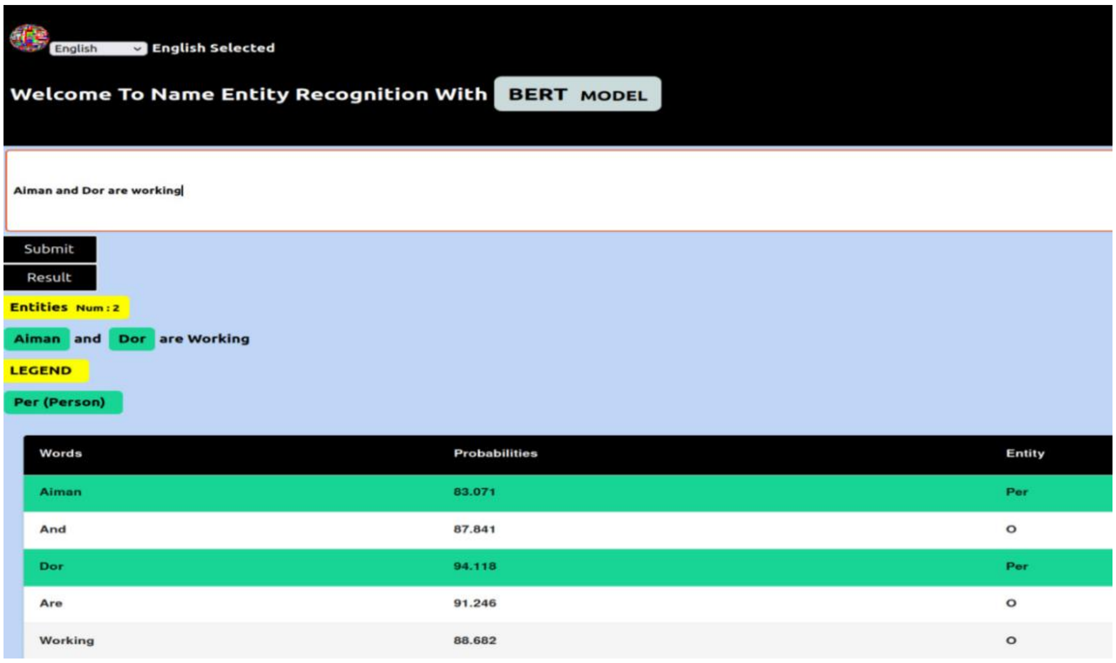
Result / תוצאה / نتيجة
After submitting the text to the model clicking on this button will show you the following information:

1) How many Entities found in the text.

2) The result text with marked words that have a special entity.

3) The Legend for the marked words.

4) Table which shows for each word the level of confidence (in percentage representation) and the entity attached to it.



Welcome To Name Entity Recognition With BERT MODEL

Aiman and Dor are working

Entities Num : 2

Aiman and Dor are Working

LEGEND

Per (Person)

Words	Probabilities	Entity
Aiman	83.071	Per
And	87.841	O
Dor	94.118	Per
Are	91.246	O
Working	88.682	O