seed

# Named Entity Recognition for Hebrew using BERT

Lee Fingerhut, Peleg Zborovsky
GitHub: https://github.com/LeeFB/AlephBert-NER.git

## Problem Statement and Background

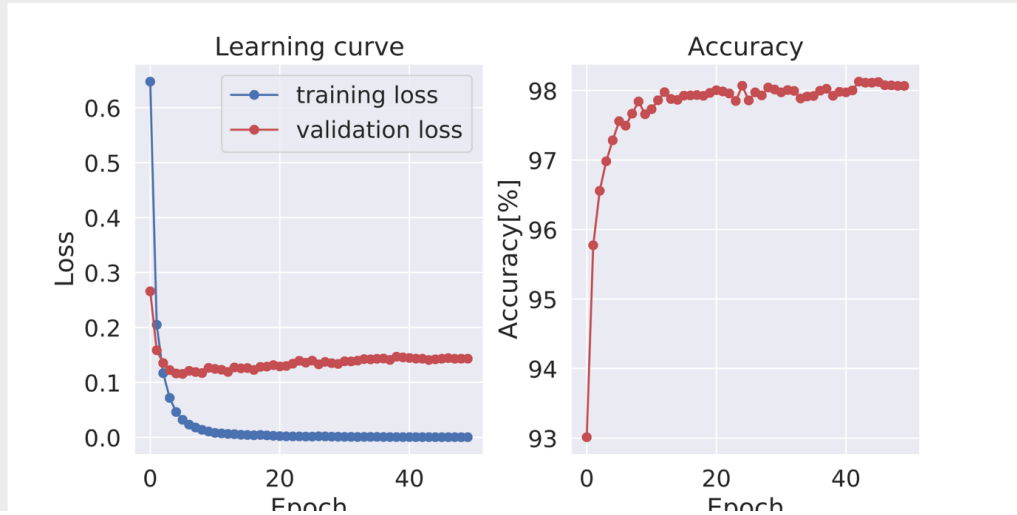
One of the most researched topics in Natural Language Processing in recent years is Named Entity Recognition. While the common models are trained on broadly spoken languages, such as English or Chinese, only a few research dealt with more rare languages such as Hebrew.
Named Entity Recognition is the task of identifying and classifying entities in text. This opens the door to other sentiment analysis tasks, such as multi-hop question answering, etc…
A proven working model for the task of NER (and many others) is Bert by Google. Bert is a transformer-based model, that its key novelty is applying bi-directional training over the transformer. While previous methods trained a transformer from either left-to-right or right-to-left, BERT trains it in both directions. This allows a deeper understanding of the text and pair-wise correlation of the context.

## Approach

One of the strengths of transformers is enabling transfer learning. In the time past from transformers raised, it was shown how powerful they are in understanding text, to its most complex relations. Moreover, it was demonstrated how the learned parameters are language-free, i.e: the learned context can be transferred to languages different from the originally trained language.
To apply NER over text in Hebrew, we took a pre-trained transformer (on NER in English), and finetuned it for Hebrew, hoping the model was able to learn semantics in any text, and fine-tune will make it suitable for Hebrew.

## Results

We used only the NEMO dataset for training our model. the dataset contains 73740 sentences.
We trained on 90 percent of the data, 10 percent for the test.
All annotations are in BIOSE format (B=Begin, I=Inside, O=Outside, S=Singleton, E=End).
Widely-used OntoNotes entity category set: GPE (geo-political entity), PER (person), LOC (location), ORG (organization), FAC (facility), EVE (event), WOA (work-of-art), ANG (language), DUC (product).




## User Interface

**Part A:**

**Installations Guide**

1. Install an environment manager. Recommeneded: Miniconda3. Here is a Getting Started guide.
2. Clone the repo:
3. git clone https://github.com/LeeFB/AlephBert-NER.git
   cd AlephBert-NER
4. Create a new environment from environment.yml (you can change the environment name in the file)
5. conda env update -f environment.yml
   conda activate ner-bert

**Training**

usage: 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
BERT model is pretrained.
You can enable all its parameters for training.
Example:

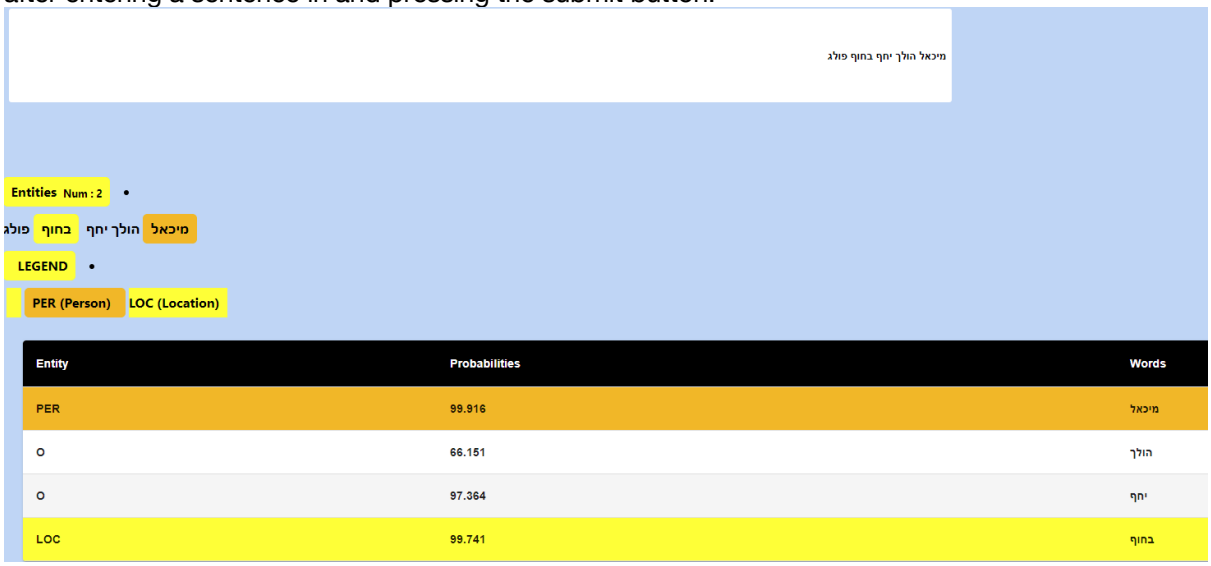
python ner_training.py --train-file dataset/dataset.csv --name sprml-train

**FineTuning**

BERT model is pretrained.
you can freeze the encoder and finetune the classifier solely, by simply adding --finetune to training command.
Example:
python ner_training.py --train-file dataset/dataset.csv --name sprml-finetune --finetune

**Part B:**
in order to make the ui running you should open a terminal on the folder “APP” located inside AlephBert-NER,
run the installations of the different packages found in the “APP” folder inside the “setup_requirments.txt”, after that change to the “Client” folder and run the command : npm install after that run the command : npm start.
in the folder of “APP/server” paste the model.pth and “tokenizer_0_tags_1.pkl” files, open the terminal on the “server” folder and run the command : python server.py. and that’s it the ui is ready!
the UI looks simple and very easy to use with few simple steps and colorful design it is easy and convenient to type in a sentence in hebrew and extract entities from it with the extraction comes the probabilities for each entity extracted from the sentence and the entity will be marked with unique color with the option to hover over any entity with color to see the probabilities the model predicted also there is legend to get the full entity pronounce.
the ui looks like this :


after entering a sentence in and pressing the submit button:




and an example to the hover effect:


## Tools

We used a combination of Jupyter Notebooks and Google Colab (free GPU usage) to train our models. We wrote our neural models in PyTorch because we were familiar with it and PyTorch had all the libraries we needed to create our models.

## Team Contributions

Lee - data preprocessing, BERT model, report(except UI.Part B)
Peleg - BERT model, UI

## References


- https://github.com/OnlpLab/NEMO?fbclid=IwAR1-e3h7WiNh0Ao-gXfF2H1Yufs9OrAFN0x5BIiDDmUjL0hXSUc5slul0ek
- https://www.depends-on-the-definition.com/named-entity-recognition-with-bert/
- https://medium.com/cogitotech/how-does-named-entity-recognition-work-ner-methods-f23201a69648
- https://github.com/AvrahamRaviv/Deep-Learning-in-Hebrew