Deep Learning Course – Final project

Native Language Identification using RNN

Gilad Eini 034744920
Liat Nativ 031709082

1 Introduction

Native language identification is the task of determining the native language of the author, given only a text in a foreign language. The task that was introduced by Koppel et al. (2005) has gained much popularity, typically aimed at the language of learners. However, this task is relevant in the much more challenging context of advanced non-native speakers, especially since while English language dominates the internet, there are far more non-native English speakers than native speakers.
This Project focuses on the task of native language identification of highly fluent speakers based on a corpus of Reddit posts in released by Rabinovich et al. (2018).
Our task is to identify the native language of authors from 5 different counties of origin. The baseline accuracy for this task – as our data is balanced – is 20%.
We obtained accuracy of 57.23% on the test set (completely unseen by the model at the training phase).

2 Data

2.1 Dataset:

We used the Reddit dataset released by Rabinovich et al. (2018).
Reddit is a popular online community consisting of thousands of forums in a wide range of topics. The dataset includes Reddit posts whose content is generated by users specifying their country as a flair (metadata attribute). Rabinovich et al. (2018) showed that these annotations are accurate and that the English of reddit non-native authors is highly advanced, almost at the level of native speakers, making the NLI task particularly demanding.
Additionally, we consulted Prof. Shuly Wintner regrading this project, and he pointed out that our task becomes even more challenging as our classification samples are annotated sentences whereas it is more common and less challenging to use chunks of sentences.
We selected 5 native languages with a large amount of data for our dataset: USA, Germany, Turkey, France and Russia. We sampled an equal portion of sentences for each class (100K sentences) and shuffled the samples (we reshuffled the data on each run). We then divided our data to (90%) training set and 10% test set. We further divided the training set to (90%) training set and 10% dev (validation) set. Our resulting dataset is 81000, 10000 and 9000 sentences for the training test and dev sets respectively.

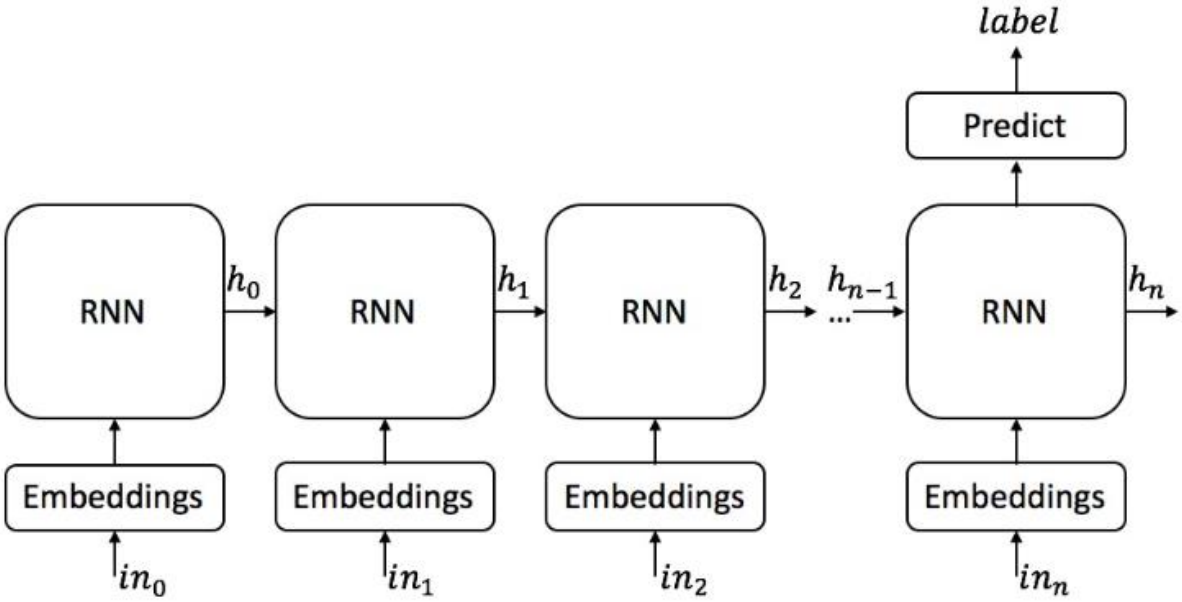
2.2 Preprocessing & data representation

We performed some clean-ups and pre-processing aimed at getting better classification results. We removed any non-alphanumeric characters from the text, except for punctuation (dots and commas) that might be meaningful for determining the author's native language.

We also restricted the length of the sentences, getting best results with sentences from 25 to 150 tokens length. Also, to avoid sentences like: “d d d d d d d d d … d” (a real example from our corpus), we ignored sentences that contain the same token 20 time or more.
To create a suitable machine-learning representation we converted each word to a vector of real numbers using word embeddings. We used GloVe pre-trained word embeddings, based on Wikipedia (2014) containing 6B tokens, 400k vocabulary[1] . We got best results when using 300-dimension word vectors.
The resulted dataset for our task was a derivative of many experiments and modifications to the data size, the selected countries, the sentences length embedding dimension etc. and are described below in the experiments section.

3 Model

we implemented bi-directional RNN. Each direction consists of a multi Rnn Cell that contain 2 GRU cells. Each of the multi RNN cells is wrapped with a dropout layer.
Our selected loss function was the mean of softmax cross entropy, and we used Adam optimizer with default parameters as well (learning rate = 0.001).
The model total number of trainable parameters is 362,205



- *For simplicity, this diagram represents only the forward direction. Our model is bi-directional.*
- *Each RNN cell contains 2 GRU cells(meaning we have h0 and h0’) creating a multi-RNN cellwhich is wrapped by a dropout cell .*
- *In each GRU cell we have 300 hidden units.*

4 Evaluation methodology

As explained above, we divided our training data to train and dev (validation) set, aiming to avoid overfitting, which was one of the most significant challenges in this project. We updated our model based on the results obtained from train data but evaluated the model

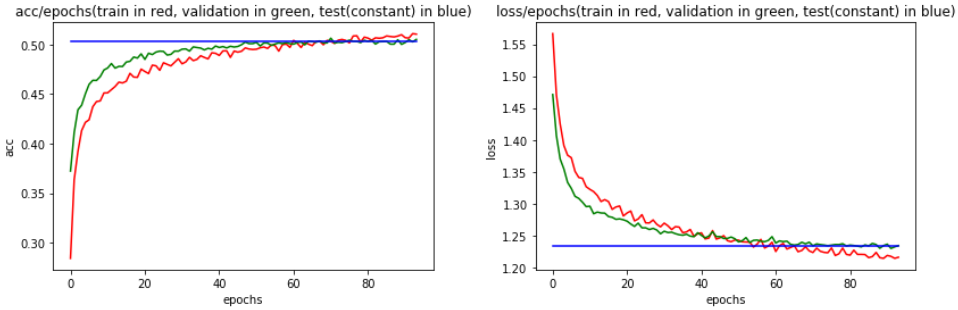
[1] https://nlp.stanford.edu/projects/glove/

accuracy on the dev set only (we did not update the model during evaluation). We performed this evaluation on the dev set on each epoch twice, once in the middle of the epoch and another with its termination.
With the completion of the training phase, we used our model on the unseen test data. As can be seen in the results section, our "final" model generalizes well, as test and dev accuracy results are similar.

5 Work Process, Experiments and improvements

5.1 Initial attempt – 1 direction RNN:

We started by seeking online an existing RNN network for text classification. We found a project that classifies crime description (sentence) to a category (link).
The model used CNN followed by TensorFlow static one directional RNN.
We adjusted the code to suit our task and our data. Once we had a running model on our data, we tested it by modifying some of the parameters.
we achieved 53% accuracy and none of our experiments improved the results, hence we decided to build a new bi-directional model from scratch. The model we built is described above at the Model section. Towards the end of our experiments on our bi-directional model, we decided to attempt the initial 1-dierction model again with the set of parameters that yielded best results on the bi-directional model.
Accuracy and loss across epochs for this attempt is shown in the figures below



As the graphs show this model does not suffer from overfitting as often happens. This is presumably due to the convolutional layer and l2-norm loss regularization. Unfortunately, the accuracy did not rise above 53%.

5.2 Attempts and experiments on the Bi-directional RNN Model:

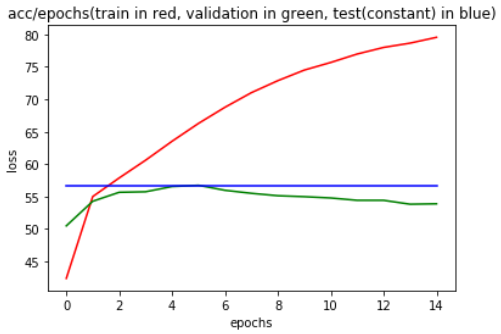
We now describe the list of improvements and experiments we performed on the submitted bi-directional model. We used this tutorial as a reference for building the model from scratch:
https://medium.com/syncedreview/tensorflow-programs-and-tutorials-1b7c397e911a
Those are the initial settings we used. If not stated otherwise the experiments were conducted under these settings.

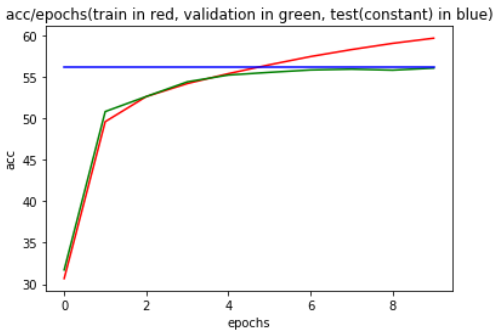
- Number of Epochs: 10
- Batch size: 100
- Number of hidden units in a cell: 64
- embedding dimension: 50
- dataset size: 250K sentences (50K from each class)

- LSTM vs GRU:
  We built the model with basic LSTM cells. We got accuracy values as high as base line and could not rise above it. We replaced the LSTM cells with GRU cells and we could see that the model started to learn and got a substantially better accuracy values.
- Learning Rate (initial value 0.001):
  We tried to modify the learning rate, we saw that a static increase to 0.1 and 0.01 was too volatile, and a static decrease resulted in a very slow learning process. We took a dynamic approach where we manually change the learning rate with the epochs advancement. We also tried to use TensorFlow exponential decay mechanism. However, our attempts did not yield better classification results.
- Hidden unites (initial value 64)
  We doubled the hidden units number to 128 and achieved 1% accuracy improvement. Increasing the hidden units number to 300 did show a slight improvement in the accuracy results and was also the largest number of hidden units that did not exhaust the machine resources.
- Word Embedding dimension (initial value 50)
  We had 4 optional embedding dimension files: 50,100,200,300.
  We decided to increase the dimension to 100 and observed 2% improvement. We further increased the dimension to 300 and reached a milestone of ~56% accuracy.
- Optimizer:
  We tried changing our initial choice of Adam optimizer with AdaGrad and RMS-Prop. With AdaGrad we got substantially inferior results. With RNSProp we obtained similar results to those of Adam's, but still a bit behind – which led us to keep Adam optimizer.
- Multi-layer cells
  We started with single layered cells. We added another layer to each direction, meaning each direction had a multi cell that was compiled from 2 GRU cells with 300 hidden units. Accuracy improved, so we added layers to the cells – up to 7 layers in each direction. The accuracy did not improve with the additional increase of cells, but the runtime extended much – and so we stayed with 2 layers for each cell.
- Sentence Length (initial boundaries 25-150 tokens in a sentence)
  We tested our model on various sentences length. For example, 5-100 boundaries, but accuracy dropped to 47%, as one might expect (short sentences does not reveal much). We tried additional modifications to the sentences length but got our best results with the 25-150 boundaries.
- Regularization
  The accuracy on train set reached much higher values than the validation (dev) set which is an indication of overfitting. (Note that our "final" model is the model with best accuracy results on the dev rather than the train). We took 4 main actions to avoid the overfitting.
  Dataset extension: we extended our training data-set, sampling 100,000 sentences from each class. The extended data set included 500,000 sentences, stored in a ~100 MB file.
  Increasing batch size: we modified our batch sizes a few times, with size ranging between 100 and 600 samples per batch. Best results obtained with batch size of 400 sentences.
  Controlling dropout: we started with drop probability value of 0.5 and increased it up to 0.85.

Adding l2 norm of the weight matrix to the calculated loss: this was the most effective method – restrain the train set accuracy and loss values.

Applying these regularization methods did help in preventing overfitting, but accuracy values were slightly lower, as can be seen in the figures below.





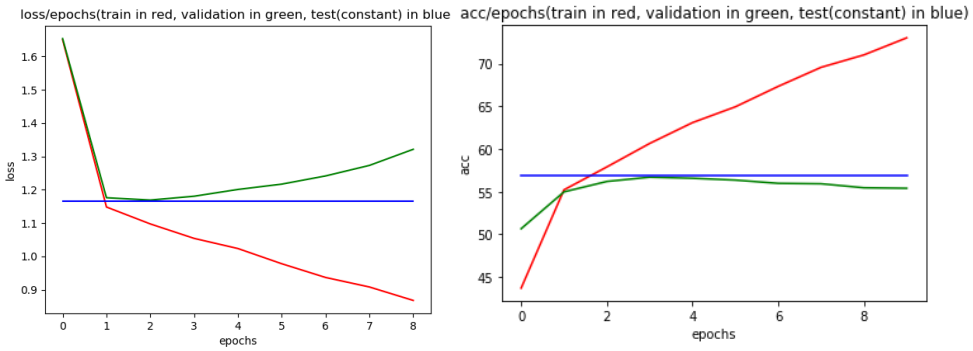
- Attention:
  We added attention cell wrappers to each cell. This resulted in a much larger number of parameters which forced us to reduce the number of hidden units - eventually yielding inferior accuracy results. This led us to leave attention out of our model.

6 Summary & Results

In this project we approached the challenging task of native language identification (NLI) of highly advanced non-native speakers from 5 different countries of origin. As we balanced our dataset the accuracy baseline for this task is 20%.
We obtained 53% from our attempts with the initial one directional model described at section 5.1 above. Aiming for better accuracy scores, we built a new dynamic bi-directional model and obtained accuracy of ~ 57. Best accuracy results were obtained in a very early stage of the training phase - explaining the low number of epochs (we did attempt a much larger number of epochs, but it did not matter).
We tried to improve the results with many attempts and configurations, particularly trying to fight overfitting. We were able to produce a non-overfitted model, but accuracy results dropped. Our conclusion is that this is indeed a challenging task.
Our best model and the model which is less over-fitted graphs are shown below, as well as the accuracy print-outs from the actual run.
Our Github repository with code, experiments documentation in (over 50) Word documents, and a central excel shit that specifies each experiment content and details can be found here.

- Best accuracy model:

```
Test loss average is 1.1650
Accuracy on test set - (28444/50000) -> accuracy: 56.8880%
    Class turkey : (6653/9982) -> accuracy: 66.6500%
    Class germany: (5197/9919) -> accuracy: 52.3944%
    Class russia : (6849/10119) -> accuracy: 67.6846%
    Class us     : (4741/9959) -> accuracy: 47.6052%
    Class france : (5004/10021) -> accuracy: 49.9351%
```



- Lowest overfit model:

```
Test loss average is 1.2495
Accuracy on test set - (27858/50000) -> accuracy: 55.7160%
    Class turkey : (6709/10169) -> accuracy: 65.9750%
    Class germany: (4948/9933) -> accuracy: 49.8138%
    Class russia : (6416/9956) -> accuracy: 64.4436%
    Class us     : (5127/10019) -> accuracy: 51.1728%
    Class france : (4658/9923) -> accuracy: 46.9414%
```