Deep Learning Course

Native Language Identification using RNN

Gilad Eini and Liat Nativ

# **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 56.17% on the test set (completely unseen by the model at the training phase).

# **Data**

## 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 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.

## 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”, we ignored sentences that contain the same token 20 time or more.  
We shuffled the sentences in all 3 groups to prevent any bias.  
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]](#footnote-1). We got best results when using 300-dimension word vectors.

**Model**   
The model we used to achieve the results above is described as follows:  
We implemented bi-directional RNN. Each direction consists of a multi Rnn Cell that contain 3 GRU cells. Each of the multi rnn cells is wrapped with a dropout layer.   
Those 2 multi cell, togher with the embbeding of the data, were given as arguments to the bi-directional RNN.   
Our selected loss function was the mean of softmax cross entropy, and we used Adam optimizer with default parameters as well.

**Evaluation methodology**

As explained above, we divided our training data to train and dev (validation) set, aiming to avoid overfitting. We updated our model based on the results obtained from train data but evaluated the model accuracy on the dev set. 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 untouched test data. As can be seen in the results section, our model generalizes well and we will show very close results for best model saved and the testing accuracy.   
**Work Process, Experiments and improvements**

Initial attempts:  
We started by seeking online an existing RNN network for text classification. We found a project that classified crime description (sentence) to a category ([link](https://github.com/jiegzhan/multi-class-text-classification-cnn-rnn)).   
The model used CNN followed by static one direction RNN.   
We adjusted the code to suit out task and our data. Once we had a running model, we tested its model with different 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. We now describe the list of improvements and experiments we performed:

* Basic settings:  
  Epocs  
  batch size 100,   
  hidden cells in a GRU cell 64,   
  embedding dimension 50  
  data set size
* LSTM vs GRU:  
  The new model built was with LSTM cells. We didn’t budge from the baseline no matter what we tried until we replaced the cells with GRU cell type. The change was dramatic.
* Learning Rate:  
  To start with, we fiddeled the learning rate(we will call it ‘lr’) of the adam optimizer. We tried the dynamic approach. We initialize the lr to 0.01 and decreaced it by epochs or by achievements. We didn’t made any progress. We changed the initial lr to be 0.001 from 0.01. That change improved our results to ~53.50 on train and test. Later on we figured out there is a mechanism that changes the learning rate. It’s called tf train exponential decay. <insert here the results of using it>
* Batch Size:  
  We modified our batch sizes a few times. (100, 200, ….) best results obtained with batch size of XXXXX
* Hidden unites   
  We doubled the hidden units to 128 and achieved 1% improvement. Increasing more to 200 and 300 units, didn’t neccerily improved our results.
* Word Embedding dimension   
  We had 4 optional embedding dims: 50,100,200,300. We decided to test the 100 dim and a big 2% improvement achieved. We reached a milestone of ~56%. So we took it a step further and used the biggest embbeding dim we had which was 300. This did improve our results.
* Optimizer:  
  We tried changing adam optimizer with adaGrad and rmsProp. adaGrad failed completely and rmsprop got clost to adam’s results but still came short.
* Initial state initialization:  
  Then we added An initializer for our weight matrix . we used Xavier initialize ([link](https://www.tensorflow.org/api_docs/python/tf/contrib/layers/xavier_initializer)). Results were improved a bit.
* Multi-layer 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. We achieved 57.67% on train and 57.2% on test. Since the improvement was great, we added a third layer to each direction and achieved very close results. We test as much as 7 layers in each direction and didn’t reach better results under this setting.
* Dataset size  
  To create less overfitted more generalizable model we enlarged our training data-set, sampling 100,000 sentences from each country of origin. The enlarged data set included 500,000 sentences, with a XXX MB file.
* Sentence Length  
  We then want to remove some of the restriction on the data and dropped the boundaries to 5-100 (from 25-150) tokens. The accuracy dropped significantly.
* Regularization  
  We tried to regulization see results in excel #33-35
* Loss function  
  We tried 1 more loss functions. We changed our mean cross entropy with minimum squared loss. The results were very close
* After discussing with Alaa, he advised us to change the dropout cell wrapper from being applied on the multi cell to being applied on each cell separately. The results slightly improved/
* We added attention cell wrappers to each cell. <insert here the results>. We also tried to apply the attention with the dropout. <insert here the results>.

**Summary**

We started with a baseline of 20%. After experimenting with the static model, we reached 53% which became our new standard. Then we built a new dynamic bi-directional model and reach as high as 57.457% in training and 57.252% in testing. The parameters for that model were <insert here>  
Our Github repository with code, experiments documentation in (over 50) Word documents, and central excel shit that specifies each experiment content and details can be found [here](https://github.com/2easy4wizzi/DL.git)

**Results**Accuracy & other measures   
Learning Graph

1. https://nlp.stanford.edu/projects/glove/ [↑](#footnote-ref-1)