# Group 12:An approach to Digaai

*Yueh Ying Lee, James Craver, Jeffrey Wu, Bowen Zhang*
*{yyleetw, jcraver, jeffwu,bwzhang}@bu.edu*

| Id | Firstname | Lastname | IsBrazilian |
|---|---|---|---|
| 0 | Ithalohfonseca | Chaverinho | 1 |
| 1 | Gustavo | Gon\xc3\xa7alves | 1 |
| 2 | Rafael | Geraldo | 1 |
| 3 | Cristyan | Victor | 1 |

Table 1. A list of entries from our training set

## 1. Project Task

The goal of this project is to identify inputs, which are first and last names, as Brazilian or not, as shown in Table 1. As many Romance languages share similar features, it can sometimes be difficult to identify the particular nationality to which a name belongs. Furthermore, Brazil is not the only nation which primarily speaks Portuguese. Finally, the spelling of immigrant names is frequently influenced by the destination country, so we may encounter Anglicization of Brazilian names. Our aim therefore is to construct a classifier that is capable of discerning Brazilian names despite these difficulties.

## 2. Related Work

Two candidate methodologies were found for feature extraction on alphabetical data. On one hand, [1] proposed a word-level feature extraction using a skip-gram model. On the other hand, [2] proposed a character-level feature extraction using a LSTM to capture temporal dependency. Both methods convert words into numerical vector space rather than using a direct one-hot encoding.

## 3. Approach

Figure 1 illustrates our system. Here $x_i$ is i-th training record, comprising first name and last name. We preprocess our data by first decoding (UTF-8), converting to lower case and then concatenating first name and last name using $ and + as delimiters. This delegates the job of learning the relationship between first name and last name to LSTM. For example, if $x_i = (Gustavo,\ Gon\backslash xc3\backslash xa7alves)$, preprocessing transforms this record into $\hat{x}_i = \$gustavo\$ + gonçalves +$

For feature extraction, we first generate 1-gram, 2-gram, and 3-gram sequences for each $x_i$ and then embed them using the pre-trained skip-gram model proposed in [2]. Figure 2 shows how we trained skip-gram the model. Suppose there are X different words. Given a sequence of training words $w_1, w_2, ..., w_T$ , the objective of this model is to maximize average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} logp(w_{t+j}|w_t)$$

range [-c,c], called its window. The probability is defined as

$$p(w_a|w_b) = \frac{exp(\langle v'_a, v_b\rangle)}{\sum_{w=1}^{X} exp(\langle v'_w, v_b\rangle)}$$

, where $\langle a, b\rangle$ is the inner product of a and b, and $v_x$, $v_{x'}$ are the input / output representation of word x, respectively. Figure 3 shows an example how to use the pre-trained skip-model to obtain an embedded result. We train the embedding model using an efficient implementation Word2Vec [3]. Note that skip-gram models are different for 1-gram, 2-gram, and 3-gram.

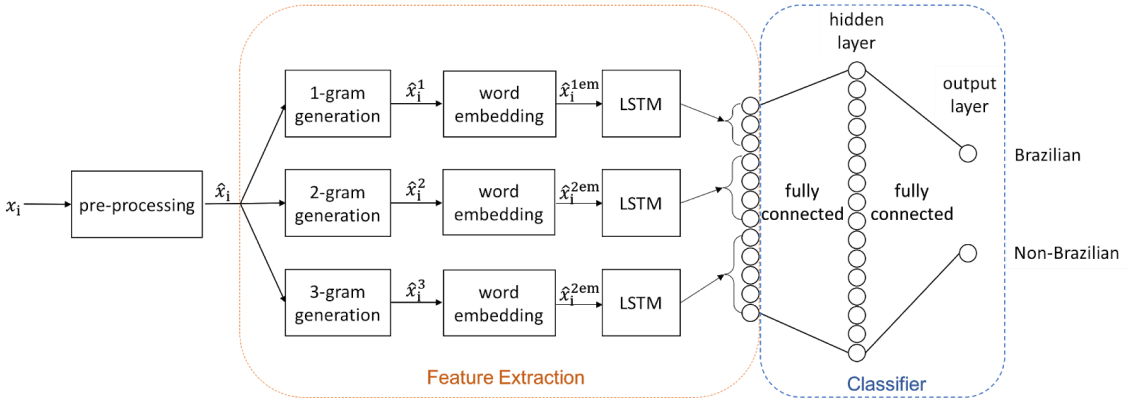


Figure 1: System overview

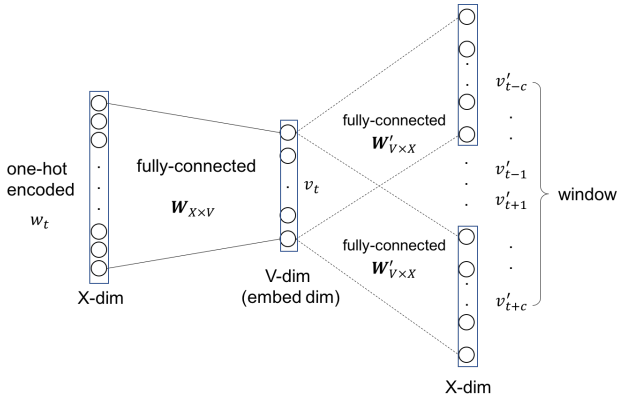


Figure 2: Skip-gram model. Note the shared output weight matrix $W'_{V\times X}$

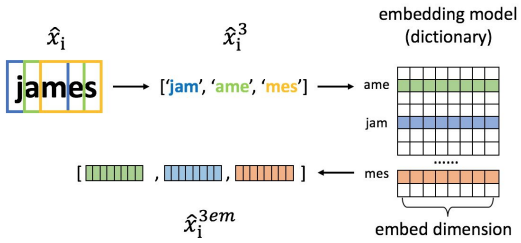


Figure 3: A 3-gram embedding example

After obtaining the embedded N-grams, we use 3 many-to-one (last output) LSTMs for each N-gram respectively to learn naming conventions seen in Brazilian names, then concatenate the 3 LSTM outputs as our final feature.

Next is the classification layer. We classify using a fully-connected layer. The training stage uses softmax loss as its loss function. To avoid overfitting, dropout is applied to the hidden layer with probability 0.5 during the training stage. Since this is a binary classification problem, the output layer consists simply of 2 logits. The output layer returns a pair of scores (non-Brazilian score, Brazilian score); we predict the label using the following scheme:

$$\hat{y} = \begin{cases} 1 & Brazilian \geq non\ Brazilian \\ 0 & otherwise \end{cases}$$

## 4. Dataset and Metric

As shown in Table 1, each training entry contains 4 columns: Id (the index of the entry, not of use for our classifier), Firstname, Lastname, and IsBrazilian (which is an element in the set {0, 1}, where 1 indicates that the entry represents a Brazilian name). The test entries are similar, except IsBrazilian is set to -1 for all entries, as we have not been given the true value these names.

| Training (pos/ neg/ total) | Testing (pos/ neg/ total) | Validation (pos/neg/total) |
|---|---|---|
| 23965/24048/48013 | NA/ NA/ 11941 | NA/ NA/ NA |

Table 2, Dataset summary

We use direct classification error to measure our success; recall the definition thereof as $\frac{FN+FP}{TP+TN+FN+FP}$, where the symbols in the formula match those in the confusion matrix. Our baseline method classifies the data using a Random Forest on the embedded N-grams $(\hat{x}_i^{1em}, \hat{x}_i^{2em}, \hat{x}_i^{3em})$ directly. We chose to use this classifier as a baseline because the skip-gram model can be view as low-level feature, and this can give us a sense of the improvement our method brings over the baseline.

## 5. Results

a. Baseline Method

As described in section 4, we directly take $(\hat{x}_i^{1em}, \hat{x}_i^{2em}, \hat{x}_i^{3em})$ as features and apply a Random Forest classifier. Since there is no validation set in the dataset, we use 10-fold cross validation to choose our hyper-parameters in the baseline method; the hyper-parameters here include maximum tree depth, and number of trees. We fixed the embedding parameters as shown in Table 3. Parameters are chosen based on experimentation; we provide the result in Figure 4 and summary in Table 4.

| | |
|---|---|
| 1-gram embed dim | 10 |
| 2-gram embed dim | 100 |
| 3-gram embed dim | 200 |
| window size | 5 |
| # of training iteration | 10 |

Table 3: Skip-model parameters

| Optimal tree depth | Optimal number of trees | Average 10-fold cross-valid error |
|---|---|---|
| 10 | 50 | 0.8324 |

Table 4: Baseline hyper-parameters and performance

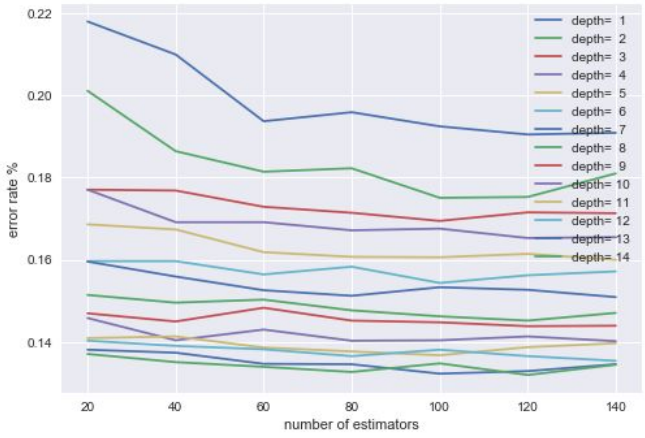


Figure 4: Baseline parameter configuration experiment. As in our main approach, the metric is classification error. This diagram illustrates testing on 1 validation set.

b. Main Approach and Tuning

Here we describe our training process. Before training, we first split the dataset into 1 validation set and 1 training set, the ratio between which is 1:9, again using 10-fold cross validation to choose our hyper-parameters. We then shuffle the training set and split it into batches. We use each batch in order, and upon exhaustion of all batches, we reshuffle and regenerate the batches. This randomness is necessary because the given dataset is ordered by classification group and we wish to remove any dependency on the order of presentation to the classifier. Furthermore, since we use batch training, we must deal with variate time-steps issues for training entries before using LSTM. We overcome this by padding zero vectors to the end of shorter training entries. For example, in 2-gram, the largest time-step size is 44. Supposing we have an entry of length 40, we pad the entry as $\left[\hat{x}_i^{2em}, 0_{2gram\ dim}, 0_{2gram\ dim}, 0_{2gram\ dim}, 0_{2gram\ dim}\right]$, where stands for embedding dimension for 2-grams.

Table 5 shows a summary of various hyper-parameters and performance. The optimal value is chosen based on average error of 10-fold cross validation. Figure 5 shows a diagram of batch size vs number of epochs, while Figure 6 shows a diagram of learning rate vs number of epochs.

The testing error of almost all parameter combinations converged to approximately 10% after 1500 epochs, so we deemed it reasonable to choose the number of training epochs as 2000. We furthermore observed that while training error decreased with more epochs, testing error did not increase. We therefore conclude that we have not succumbed to overfitting to the training data.

After the hyper-parameters were determined, we tried different skip-gram model parameters and found little effect, as shown in Figure 7. We therefore used the same parameters as in Table 3.

| Hyper-parameters | optimal value |
|---|---|
| Batch size (128, 256, 512, 1024, 2048) | 512 |
| Dropout ratio (0.25, 0.5, 0.75) | 0.5 |
| Learning rate (1e-2~1e-5) | 3.5e-3 |
| Hidden layer dimension (50~300) | 200 |
| Average cross validation error | 0.9128 |

Table 5: Hyper-parameter summary and performance

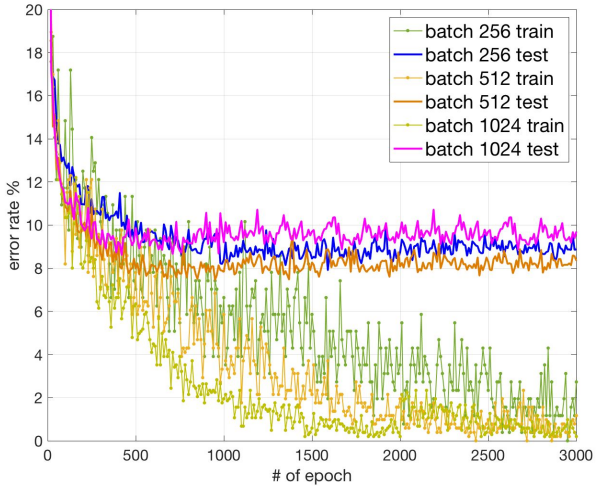


Figure 5: Error rate for chosen batch sizes, testing on one validation set. Conducted with learning rate is 1e-3, dropout ratio 0.5, hidden layer dimension 200.

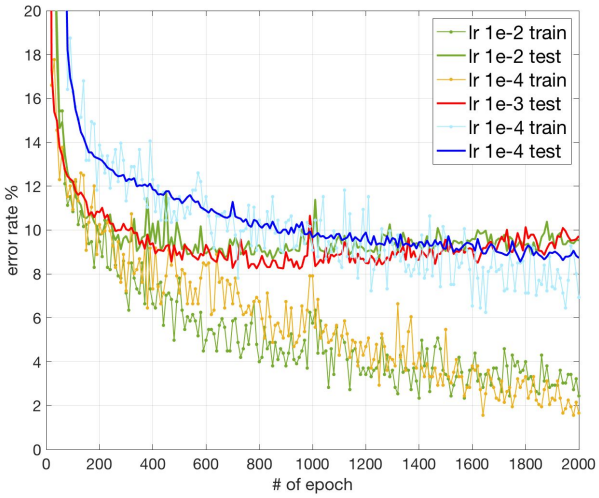


Figure 6: Error rate for chosen learning rates, testing on one validation set. Conducted with batch size 1024, dropout ratio 0.5, hidden layer dimension 200.

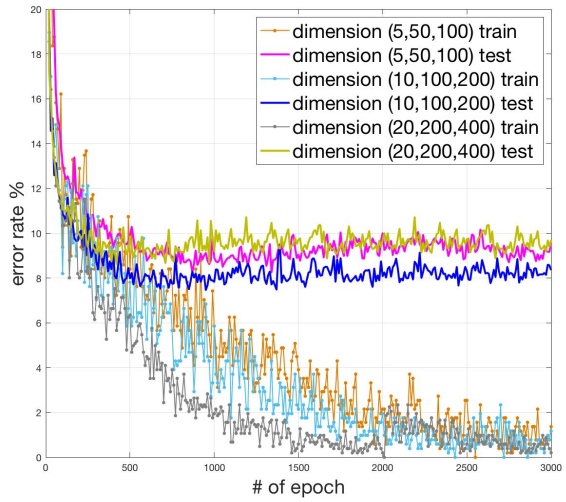


Figure 7: Experiment on different embedding dimensions in skip-gram model. Hyper-parameters were fixed according to Table 5. The triplet in legend means (1-gram embed dimension, 2-gram embed dimension, 3-gram embed dimension).

c. Kaggle Submission Results (as of Dec 6)

| rank | Team | accuracy |
|---|---|---|
| 1 | **gonna_GG (our approach)** | 0.9083 |
| 2 | Yuxi Jiang | 0.9026 |
| 3 | Federico Siano | 0.8944 |
| 4 | **RF_GG (our baseline)** | 0.8380 |

Table 6: Kaggle submission rank(the bold ones are our submission)

d. Typical Mistakes and Future Plans

To inspect why our validation error does not decrease, we analyzed some incorrect predicted validation entries. The false positive ratio and false negative ratio had similar amounts, both around 5%. We found out our false positives consisted of 2 categories. One kind is rarely used names, to which we can only find similar names using a very loose regular expression query in the dataset. The other kind is one-sided labels; for example, there are 107 last names found using the regular expression query victor*, and only 2 of these are not Brazilian names. Nevertheless, our model still predicted these 2 as Brazilian names. Similarly, our false negatives consisted of 2 categories of error: rare names and names corresponding to immigrants (e.g. Asian names such as Linh Hoang or Len Da).

To reduce the error of rare names, more training entries would be required, but for second kind error in both the FP and FN cases, more entries would not necessarily help, as these correspond to outliers in their respective group. Further features not available to us would be required to reduce this kind of error, as names have proven to be insufficient.

**References**

[1] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean Distributed representations of words and phrases and their compositionality, NIPS 2013

[2] Jinhyuk Lee, Hyunjae Kim, Miyoung Ko, Donghee Choi, Jaehoon Choi, Jaewoo Kang, Name Nationality Classification with Recurrent Neural Networks, IJCAI 2017

[3] Word2Vec library, https://radimrehurek.com/gensim/models/word2vec.html