Lyric Generation using LSTM

Francesco Pappalardo PAPPALARDOFG@OUTLOOK.IT
Vanessa Perticone[1] VANESSAPERTICONE12@GMAIL.COM

1. Model Description

We describe here **LiverNet** a LSTM based model which, given an input sequence of words, predicts the next word. To implement this mechanism we want to convert all the words into an embedding and pass it to the LSTM network. Then we use a dropout layer in order to avoid overfitting and, finally, a linear layer predicts the next word. At the end, the model should provide as output an ensemble of words that should shape a text likely written by Kanye West.

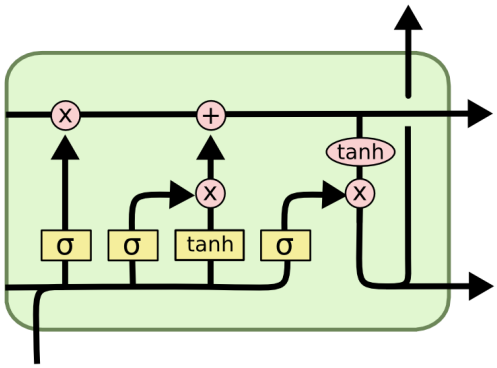

Figure 1: A simple representation of the proposed model.

LSTM (Long Short Term Memory) modules contain computation blocks that control information flow. They work into different steps: first of all they forget irrelevant parts of the previous state, then, they store relevant new information into the cell state, selectively update cell state values, and finally the output gate controls what information is sent to the next time step.

LSTMs seems to be promising for text generation, in our case for lyric generation. Our network's first layer is an embedding layer. After the embedding layer, we use an LSTM layer with a dropout, a regularization method in which input and recurrent connections to LSTM units are excluded from activation and weight updates to reduce overfitting. The dropout is set to 0.3. Then we apply a ReLU as activation function in order to solve the dependencies problem and to prevent vanishing gradients. Finally, we use a linear layer as decoder, and it will be the output layer.

2. Dataset

The dataset was provided by Kaggle in Public Domain License. It consists of 364 rap verses from 243 songs by Kanye West, collected by the user who uploaded the dataset by using Rap Genius library. All the verses are taken from Kanye West discography, and other artists' parts are excluded by it, leaving only parts written by the chosen artist. The data is stored in a .txt file where all verses are separated by empty lines. The data has been cleaned in order to remove any unnecessary words or characters not part of the actual verses.

3. Training procedure

To compute the training loss we apply as criterion the Cross Entropy Loss:

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y_i})) \tag{1}$$

where y_i is the true output and $haty_i$ the predicted one.

For the baseline model we use standard binary cross-entropy loss.
The model is trained from scratch for 20 epochs on a Nvidia Tesla K80 with 12GB of VRAM. Batch size is set to 16 while sequence lenght is set to 32. RNN size is 2048 and embed size is 1024. Adam is chosen as optimizer algorithm using a learning rate of 0.01.

4. Experimental Results

Several experiments were conducted to test the effectiveness of the proposed network. More precisely,the obtained network was trained and tested in two versions, one with ReLU activation function, and one without it. Models were trained as described in the previous section. Table 1 shows results for the proposed architectures as well as of ablation studies (i.e., different variants of the final architecture when adding or removing layers).

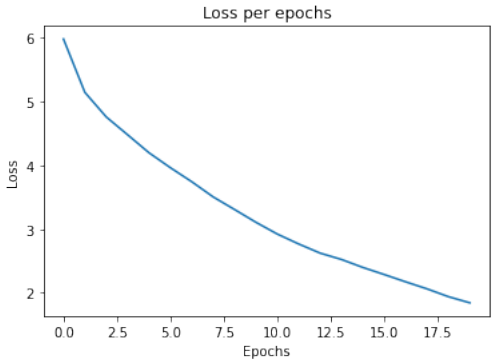

Figure 2: Training process in model with ReLU.

Despite the lower loss of the model without ReLU, we are not interested in minimizing it, because of the fact we are trying to make a model that is able to generate new lyrics, with unused sentences, and not to have a model who is good in predict how an existing

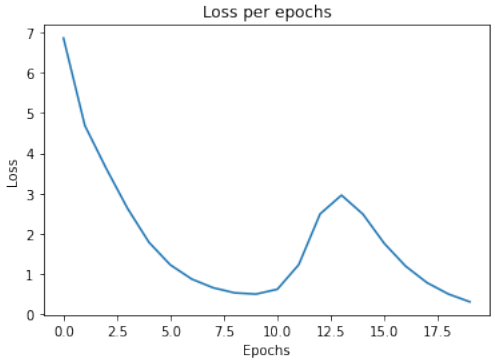

Figure 3: Training process in model without ReLU.

Model	Loss
Baseline Net	0.31
– with ReLU	1.84
Your final model	1.84

Table 1: Performance of the tried models.

song continues. In fact, trying to generate text from the two models, we obtain existing songs from the model with lower loss, and generated songs in the one with highest loss. So we choose it, the model with ReLU.