Generating Baby Names Using Character Level Language Modeling in RNN's

By
Harvinder Singh
Karan Bhatia

Natural Language Processing

- Why is it difficult to learn natural languages?
  - Ambiguity
  - Contextual
- Is a particular word valid?

Statistical Language Modeling

- Main goal of any language model is to learn the joint probability distribution of sequence of characters/words

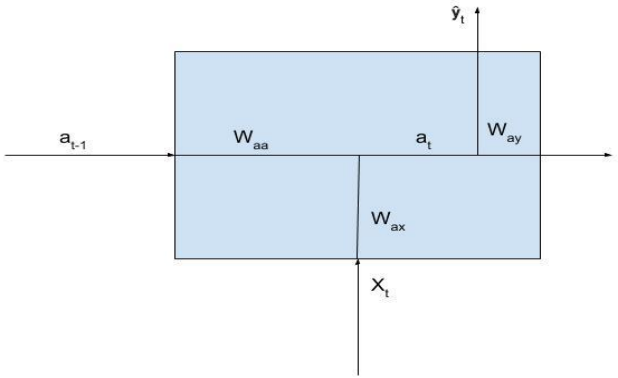
$$P(w_1, w_2, \ldots, w_T) = \prod_t P(w_t | w_{t-1})$$

$$= P(w1) * P(w2|w1) * P(w3|w1,w2) * ..$$

Simple Recurrent Unit

This figure represents a simple recurrent unit cell that gets an input X^t and current time step ‘t’ and activation from previous recurrent unit.

The set of weights W_{aa}, W_{ay} and W_{ax} are shared by all the units in a layer.



Activations of a Recurrent Unit

The activations of a recurrent units are governed by the following equations:

$$a^t = \tanh(w_a.a^{t-1} + w_x.x^t + b_a)$$

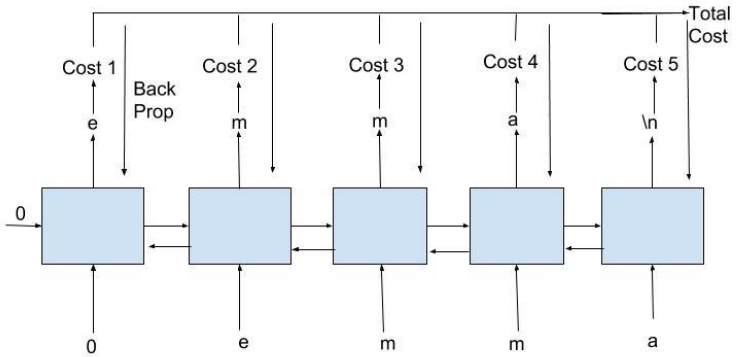
$$\hat{y}^t = \text{softmax}(w_y .a^t + b_y)$$

Loss function at a time step is cross-entropy loss given as:

Loss = $-y^t * \log(\hat{y}^t)$, where y^t is the one-hot encoding of original character and $\hat{y}^t$ is the softmax probability vector.

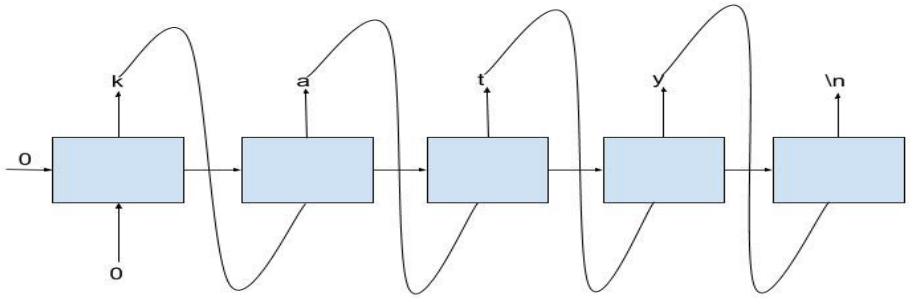
Training Process

This figure depicts the training process for a single example. Here, we take the word ‘emma’ as the training word are use it to make 1 gradient descent update using back-propagation.



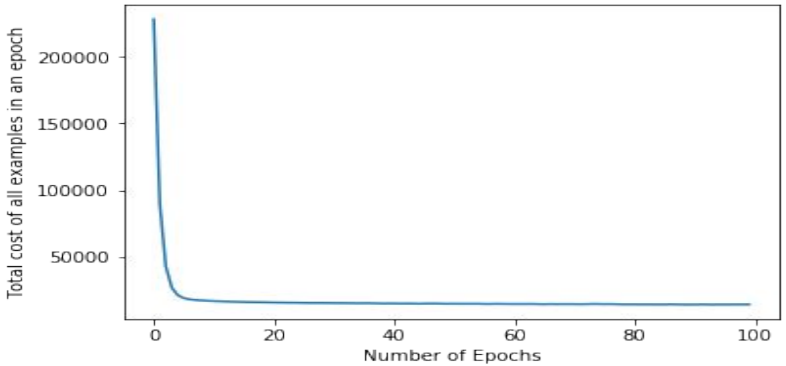
Sampling Process

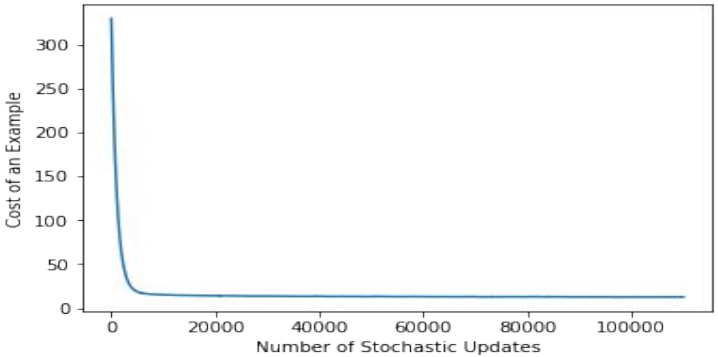
This figure depicts the process of sampling a new name from the probability distribution at random. We initialize the sampling process with a vector of zeros and then the system picks the sequence of characters from the distribution at random.



Results And Conclusion

The figures below represent the learning curves for the training process. The network is trained using 1100 unique baby girl names for 100 epochs. So, total 110,000 gradient updates are done.





Performance of the Sampler

The below table depicts the performance of the sampler during the training time. As we can see, during the initial phase of learning, sampler produces some random strings that are unrealistic but as the training proceeds, the sampler starts picking the names from the distribution.

Epoch 1	Epoch 3	Epoch 7	Epoch 10
Lxejeleyniha	Erah	Gaary	Eliz
Ea	Arelyn	Iszen	Keiga
Aeaenaya	Bran	Bainktily	Gissjena
Mafitresrann	Brina	Mastie	Alissa
le	Miane	Klecyan	Vivie
E	Kaitrprilye	Elian	Recie
Argatana	Keita	Jociy	Audlivla
Gci	Cokikicha	Miavka	Kesley

Table 1: Progress of the Sampler with time

Nearest Matching Names

The table given below shows some examples of the names generated by the sampler and compares it with the nearest matching names in the training set.

Generated Name	Nearest Name in Training Set	Common Substring ratio
Alica	Alice	(alic)
Malie	Malia	(mali)
Mina	Amina	(mina)
Keira	Keiga	(kei)
Araya	Zara	(ara)
Mira	Amira	(mira)
Mada	Madlyn/Madlynn	(mad)
Ardana	Arden	(ard)
Jena	Jenny	(jen)

Table 2: Nearest Matching Names

Some Realistic And Similar Names Generated

Avelyn

Eliva

Audyn

Alica

Soubia

Riana

Reneva

And many others.

Names Generated on basis of Initial Characters

We can generate names on the basis of some initial characters by manually forward propagating through the network and then generate names randomly from the distribution.

System is well trained to ensure that if we forward propagate a complete name manual and then ask the system to generate from the distribution, system will most probably generate end of line ‘\n’ character.

Eg: Out of 20 names generated starting with ‘claire ’, system generates ‘claire’ 11 times and other 9 names are like clairia, clairen etc.

Future Scope

Even though we focus on traditional recurrent units like tanh, our results show that these simple recurrent units can generate new , realistic and similar names used to train recurrent units.

Two of the more sophisticated Recurrent units which are used to handle long range dependencies in a sequence, which are LSTM(Long Short Term Memory) and GRU(Gated Recurrent Units) can be used to generate names with long range dependencies between the characters.

References

[1] Junyoung Chung, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling 2014.

[2] Alex Graves, Supervised Sequence Labeling with Recurrent Neural Networks. 2012.

[3] B.Hammer, On the Approximation Capability of Recurrent Neural Networks. Neurocomputing, 2000.

[4] Ji-Sung Kim, Deep learning driven jazz generation using Keras and Theano.

[5] Andrej Karpathy, Multi-layer Recurrent Neural Networks (LSTM, GRU, RNN) for character-level language models in Torch.