Inspired by this blog The Unreasonable Effectiveness of Recurrent Neural Networks, a simple set-up 2-layers CharLSTM for Text Generation became the last assignment of my seminar *Neural Networt: Architectures and Applications for NLP*. This report is a rewritten version of my submitted assignment a year ago.

1 CharLSTM

Implement a 2-layer character LSTM for text generation. Follow these steps:

1. Choose a training corpus and implement the necessary pre-processing. Describe it in the cell below.
2. Define the vocabulary.
3. Define a network architecture (it has to include a 2-layer character LSTM).
4. Compute the prediction of the network.
5. Define and compute the training loss.
6. Implement sampling for inference.
7. Train the network on the training corpus.
8. Generate a text with the trained network.
9. Give the command that calls the code for training and generation (so I can reproduce it) in the cell below.
10. Describe how you structured your code.

1. Training corpus

- The chosen corpus is the first volume of the book series **Tales of Lang Biang** for children with around 116624 words. This corpus is in Vietnamese (my mother tongue) with the Latin alphabet (29 different single characters) and a range of 6 tones for each vowel (those are marked right above the vowel).
- Vietnamese is a language having only tokens with one syllable (except foreign words that can't be translated or special names from the minorities). One token might have a meaning or not. A Vietnamese word can be formed from one or more tokens. i.e. *mua - to buy*, *bài tập - (an) exercise*, and so on.
- Using the model I expect to generate the sample texts with tokens that follow the writing rules of Vietnamese (I expect actually neither correct grammar nor meanings) such as:
  - A token can have maximal one tone. If there are 2 vowels, the main one has the tone, so such tokens like *táó* are not allowed.
  - How vowels combine with consonants and with each other:
    - which consonant can stand before or after a vowel
    - which pairs or triples of vowels or of consonants are possible

· no tokens excluding the tone are allowed to have 2 alike consonants or vowels next to each other, i.e. *tốtt*, *moon* are no tokens

2 - 6. Implementing the network
It can be seen in code.

7. The trained network
A file named train.txt will be attached for the observation of the training process.

8. The generated text
The corpus file named **ep_1.txt**, the generated one named **lang_biang_fake.txt**.

9. Command to run the code to train and generate the text in terminal

```
nice -n 10 python3.6 lang_biang.py training > train.txt ;
nice -n 10 python3.6 lang_biang.py sampling
```

10. Description of the code structure

The preprocessing from the example code is enough (all files should have encoding 'utf8') for this corpus. How the code is structured is derived from the example code. Everything I used is inspired from the CharRNN code to create Christmas songs from the lecture. The network has 2 hidden layers with LSTM (instead of the Elman RNN with 1 layer). Because there are more vocabularies (about 178), I increase the value of hyperparameters, like the training sequence length is 100, embedding size is 40, and number of units is 200. Due to the characteristics of Vietnamese (the more tokens, the more words we might have), the sequence length of each generated sample is 10000.

All files above found in folder tales_of_lang_biang/ex-1

2 Analysis of the CharLSTM Outputs

What does the CharLSTM learn?

For the following analysis you can use the code from exercise 1 (only if successfully implemented) or the example CharRNN code from class.

1. Analyze how the predictions of the network change during training (cf. the blog's section "The evolution of samples while training").
2. Analyze what the RNN (single neurons) have learned with respect to the following questions (cf. the blog's section "Visualizing the predictions and the "neuron" firings in the RNN"):
   - **A** Which characters (output classes) are how likely for a given input? Which characters are often confused by the model (i.e., are similarly likely given the same context)? Visualize this, for example as shown in the cell below.
   - **B** Choose 5 neurons of the LSTM. For which inputs are they activated? Can you find patterns? Examine at least 5 examplary inputs. Visualize this, for example as shown in the cell below.
3. Characterize the generated samples of the model: are they well-formed, invented, plagiates from the training corpus, do they convey meaning, reflect structures, etc.?

1. The evolution of samples while training

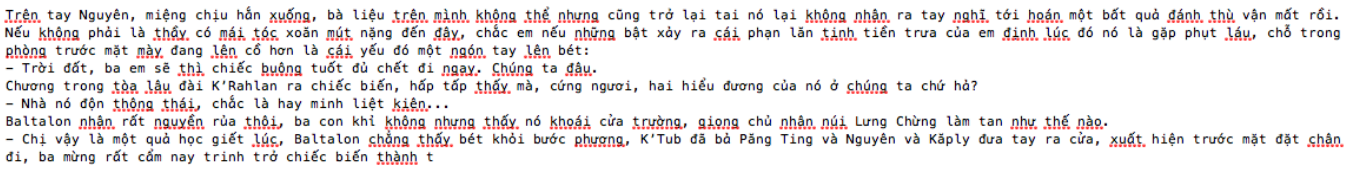
1. The model after 500 iterations still couldn't detect any right tokens and predicted only meaningless sequences.

```
huihu cì và đìng đoyện chiukh bà tihi bó., Kăán K'Tulống tiếng đừo triệc thìng choy ga lốn vi thện toộng Là ph bày rà đài và thi đếo chận ngà. Có taa là mếi vải ca
kiiện tháng, ngất chựtgi bác Bạa đối quùng tho đăng đới thấe khư, taa.
- K'mối tríng hịnh nhưilà thá côi. Tà chi rạng lồo ma hàa đà bốt ri tiki ngau bà trợ chấc.
- chi đai cíc riến b Tulong lức lù K'Tub bếc còo K'Bếi vuóy củ.
nhựệt vếc vi phùnh căng điất tể bờ. hế bã nhưng tai mãi thưng,. - T xuy tuốy đì gặt rất khưng đăplo va xuiên gai cũng ghợng có thô han!
hì ngì siên mó đai ciất khất chả lặt nháng "rông krữch trên lắt mào thó nhăng từp couly cúy nảnh lang mị "arc đi và lộc đà liong nù nủa tất. - sàa tụi tihơn trốc
thê!
- Amth mể suên hộ đâa dăply hồn đối đùcli ming bà căpl nguyên riếng khẩn đanh thang gồi thò thủ khại lưittg hột bối vất vú thuyên thịi siết.
Kăn mềm nó lứ đếi rông 2 báng thêi. - Nhẽa điên K'Ta vố khà cò đùi có hru nhình, Nhiên cất đài đủi lự cã cuilể ngịng nhờ nọn đột krưikn kơm cũng nưoệc thát nó
```

2. At the 7500th iteration, the model could predict exactly how many characters each token needs, we can see where the space is used to mark a new token is completely correct. And there come a lot of names of the characters that are written correctly (those start with a capital characters are all character names in this corpus).

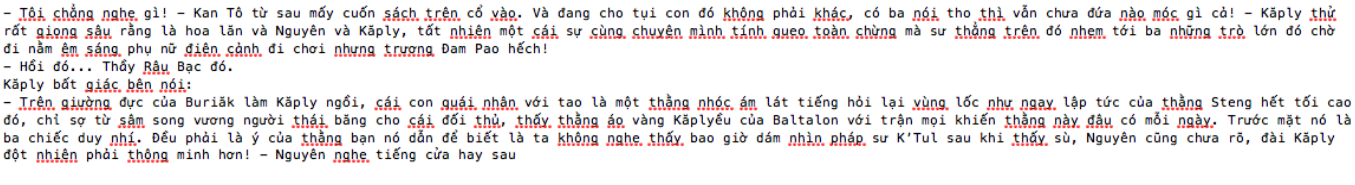
```
- Phải nói quái quái các cho như nếu thì Nguyên vừa rồi mội một cao hé tinh hoàng tìm cách lên không hiểu biết lại dận của mình lệnh vào rung đất, chỉ cổng khải
tới nó dẫn trò tụi nó là đoắc - đồng thiệt chiện nhận, mặt như một vào như chỉ con!
K'Tub, chỉ nào bía lời đớn miệng và con đi thủy sửa một cây nửa người man ra vừa nhét dưới vác thế nào nào lành ngọ, cặp sau đó nhưng chấm nghe cũng cao đồng là
nắn một tiếng nói. - Em bị bà không phải xất nhiện ông trừ thì muốn quét sang hả chỉ đọng là. Trộng phải trện giới ngọ nghiếng và cho đời về nắm.
- Đậu Baltalon, ngã đài tự theo hướng người:
- Tại đang làm sao trán mây bằng đúng thứ gì trong rầu nói bộn chị cuối màng chiếc ghế khủ cái mình! - Nguyên và Kăply mới hai của bong lúc khiến toàn nói rồi hai
nói đến nhớ K'Brệt đó!
Đồng là Mua dọc chuyện được kéo các cại trện đầu, về tùi)...
Súc về thôi trán dán mắt nhìn tới không onh K'Brệt lên nhỏ vật nhớ thưa thì biết gì hay động phòng pụm hôm khếch như nết đưa tôm, ông K'Tul của nó than
```

3. At the 24000th iteration, a lot of single token were predicted correctly, and some of them have meanings, and such long sequence like **chủ nhân núi Lưng Chừng - Master of the Mountain Lung Chung** is such a good result learnt from the corpus. There are a lot

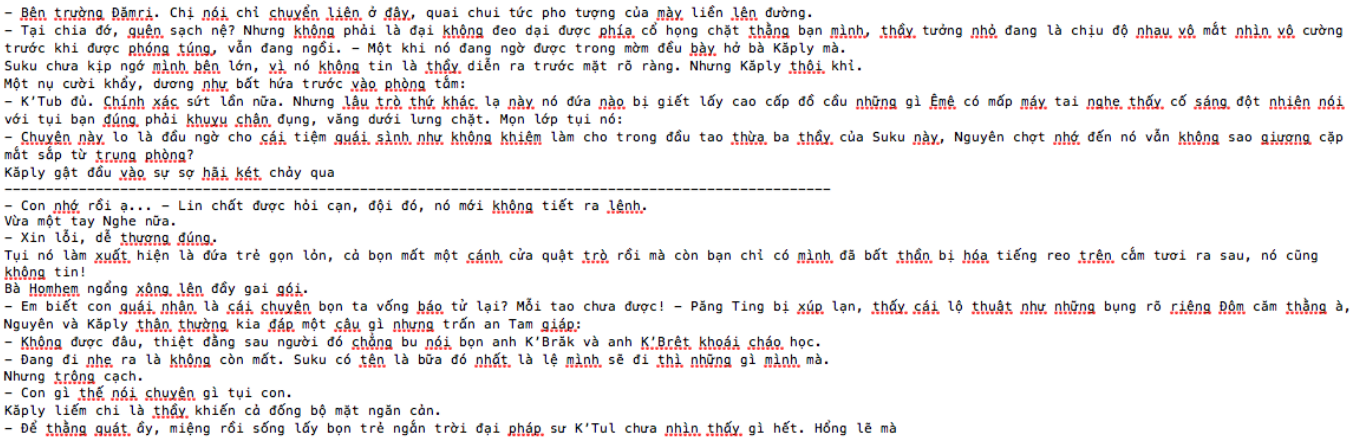


of punctuations truely generated. The character '**-**' which marks a start of a speech or the new line were predicted correctly at this iteration.

4. I could say this sample can be mistaken as a true text if not reading it. So at this point (30500th iteration), the model has done a good job of knowing the writing system of the Vietnamese vocabulary and reproduced it perfectly.

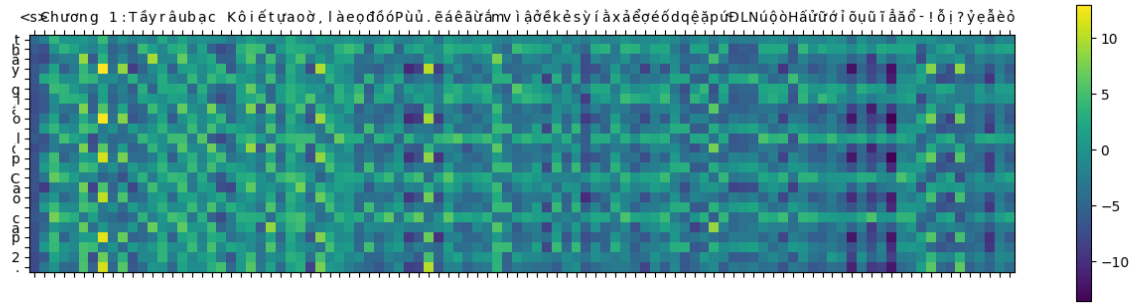


5. After 49500 iterations the model showed a tendence of better training as a lot of the words do exist, though are wrongly combined and generate only meaningless sentences, but hopefully true sentences would be soon found out. (I would say, the output below did have many tokens that have meanings and can even form phrases, like noun phrase **anh K'Brăk và anh K'Brêt** - **Mr. K'Brăk and Mr. K'Brêt**).



For all of the figures above, the tokens with red underline have meanings and exist in the Vietnamese vocabulary as single words. In fact, most of those without underline exist in the Vietnamese vocabulary too. (The code file **training1.py** and the text file generated **lang_biang_fake_1.txt**)

2. Visualizing



– Above are the predictions of the word phrase **thầy giáo lớp Cao Cấp 2.** - male teacher of senior class 2 at the row and the first 100 characters in the character vocabulary char2id in **tales_of_lang_biang/ex-1/lang_biang.py**.

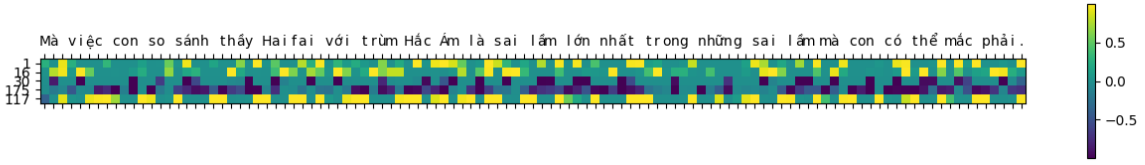
As we can see, the yellow blocks are such special characters like punctuations or space seem to be easily detected by the model. The model can also predict the consonants so well regardless of upper case or lower case (i.e. **C**, **c**) and common vowels with the international Latin alphabet.

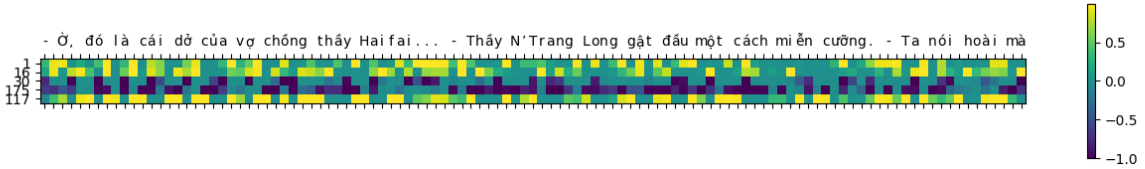
The model has only difficulty predicting the vowels with tones. This is understandable, since to know which vowel with tone in the word is even not easy for the foreign learners to distinguish.

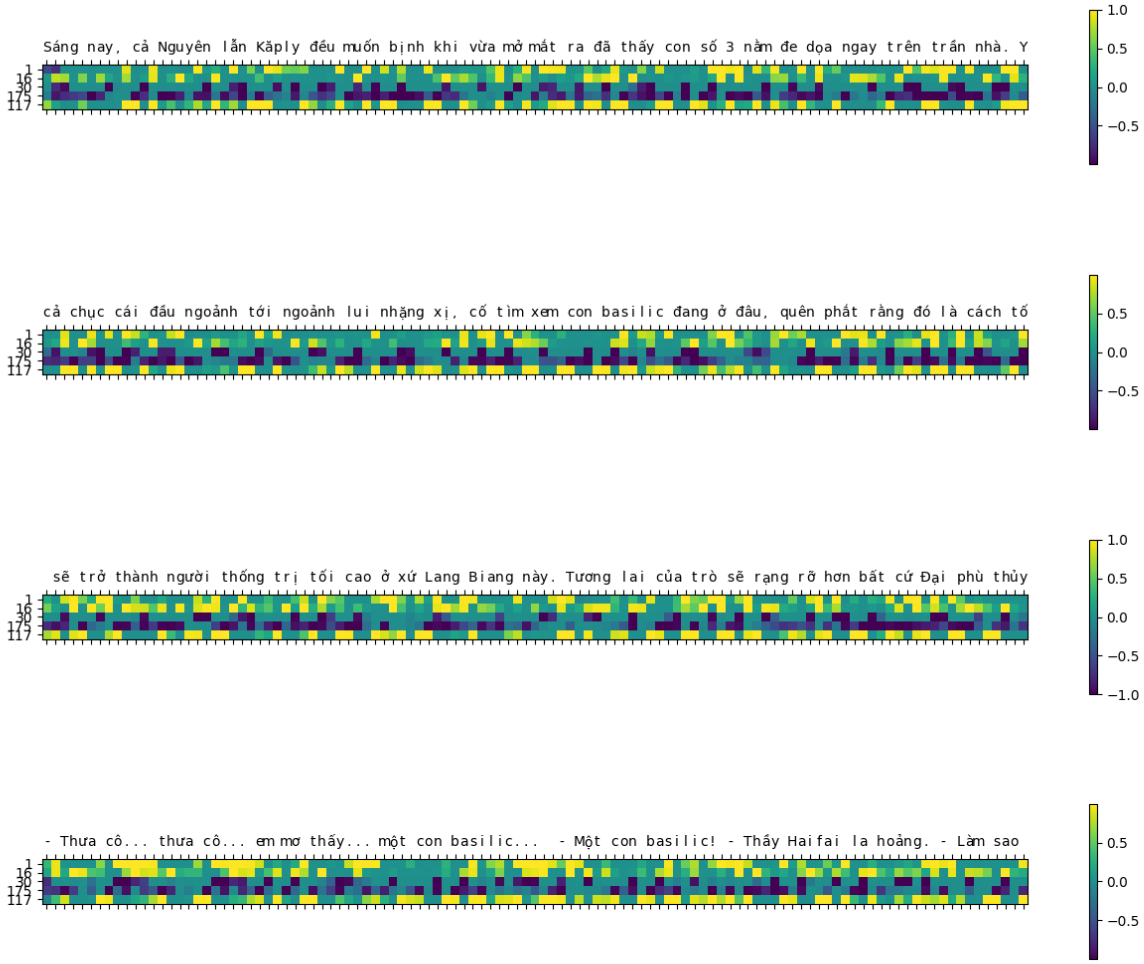
However, the model still did a good job as the vocabulary consists of lots of vowels with tones which don't occur in the input phrase, and those blocks of them stay in blue (they are not predicted for the vowels in input) rather than yellow or green.

(The code file training2a.py)

– In 176 neurons of each hidden layer, I chose 5 neurons: 1, 16, 30, 117, 175. Observing the colors of the block, we can see some clear cases of being activated of each neuron.







- **Neuron 1** is often activated when the characters are space or punctuations like punct or comma, number or sometimes Vietnamese vowels with tones.
- **Neuron 16** and **117** have so many yellow blocks, from all the sequence inputs, these 2 neurons should be activated when the input is a Latin alphabet or a Vietnamese character that occur often in context (both vowels consonants).
- **Neuron 30** seems to be activated when the input is consonant, as there are a lot of dark block of it when the input is a vowel.
- **Neuron 175** is rarely activated when the input is an alphabet, so it must be activated for special characters. There are some words like **basilic** from input number 4 and 6 with the scheme of activated neurons.

(The code file training2b.py)

3. **The generated samples of the model**: As a native speaker, I'm totally atonished with the generated text.

    1. At first glance, I found the text has nothing different from a normal Vietnamese text file (or the original one). The spelling rules that I proposed in Exercise 1 are perfectly kept. Every single token generated follows exactly what I expected it should be.

2. All the names of characters (**Kăply**, **N'Trang Long**, **Nguyên**, **Haifai**, **Hailixiro**,...) or objects (**Những Dấu Hỏi** - name of a store in the book, **đồi Phù Thuỷ** - the hill named Phù Thuỷ) and all the words starting with capital character in sentences are predicted correctly.
3. A lot of special concept needing more than two tokens are predicted correctly like **Phù thủy Bạch kỳ lân**, **Chiến binh giữ đền** and a lot of pairs of tokens are true words in the corpus and in daily life.
4. Some long sentences have the right structure of grammar with subject, predicates and objects and their attributes, even though they convey funny or no meanings.
5. There are some short sentences indicate short questions or commands have meanings like **Đi đâu? - Where (are you/we going)?** or **Em đưa theo K'Tub... - Take K'Tub with you...**, and some are just plagiates from the corpus like spells (this is a book magic words) **Bay lên! - Fly!**.

In general, I was fully surprised with this result. I hadn't expected that much as the corpus is not large enough (the files used in the blog have at least 1000000 words). A really interesting thing that I found out about the model is during training it predicted a lot of bad words in Vietnamese (the corpus used only vocabularies that are nice as it's a book for children), and somehow some good characters become the evil ones (vice versa) in the generated text (judging from the sentences that have meanings in the generated text). (Analysis of the text file *tales_of_lang_biang/ex-1/lang_biang_fake.txt*)

All files above found in folder tales_of_lang_biang/ex-2