Deep Learning Project

Victor Mongeleca

Plan of attack

1. Initialize an workspace on Google colab AND gradient
   - ↳ Will only be used once we are comfortable with COLAB
2. Creation of the dataloader

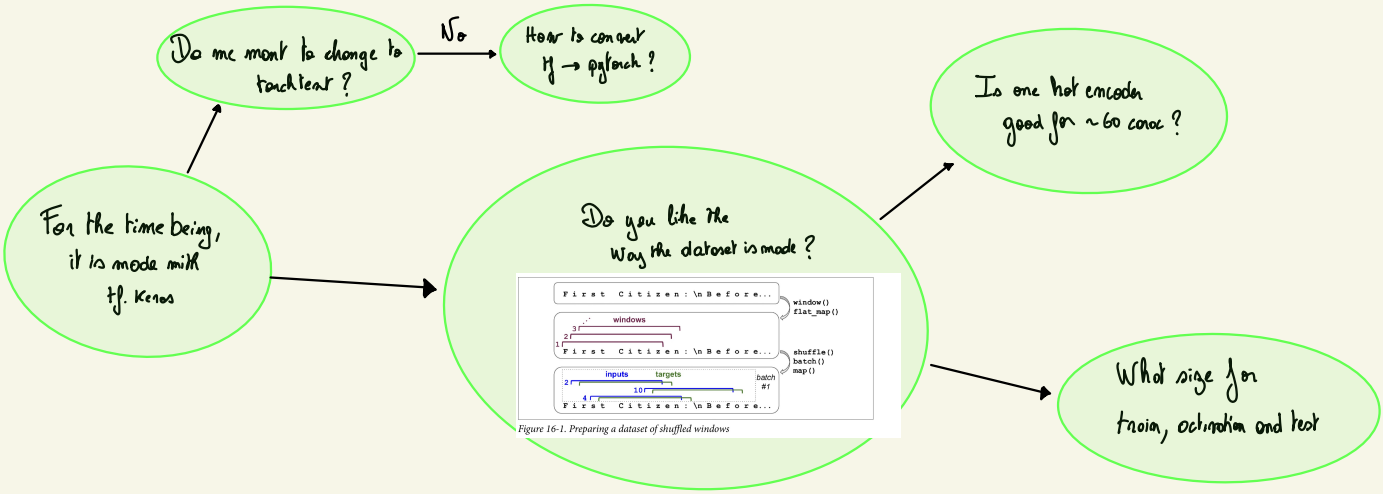
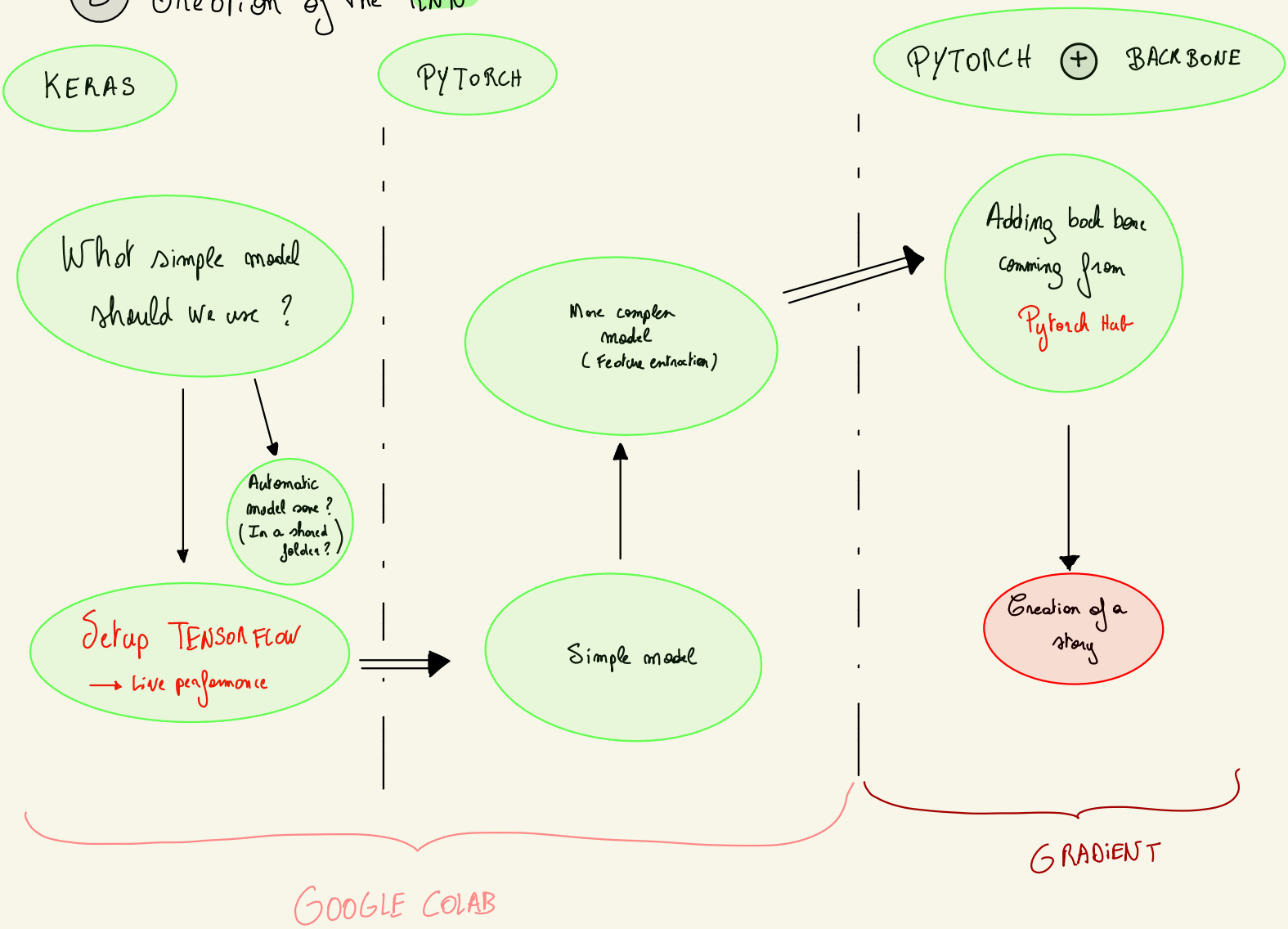


Figure 16-1. Preparing a dataset of shuffled windows

3. Creation of the RNN



SBOT V2 – Feat GASPAR

" Harry come into the room "
(X_1 X_2 X_3 X_4 : Input ; Y_1 : output)

page 534 → 540

Note: it can be done using Keras!

Dataset's info:

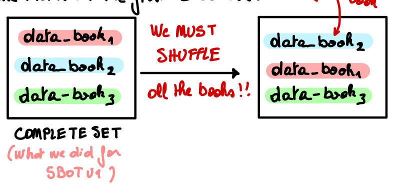
* Words to word:

① X words → Y words (5, 20 / 15, 50)

② - m_{max} = max length of a sentence
- m_{curr} = length of current sentence
- m_{curr} – X' words → X' words ; ⊕ (m_{max} – m_{curr}) padding ($\uparrow$ $m_{curr}/2$)

In this case, it will learn punctuation

* If we train on the first 3 books: in the (also a shuffle book)



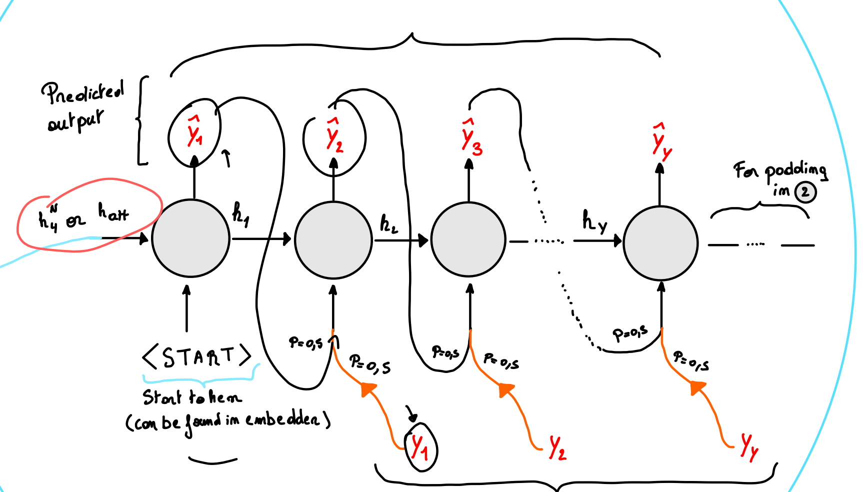
Decoder's information:

① First, we only use GRUs

② We introduce force teaching (it improves training speed)

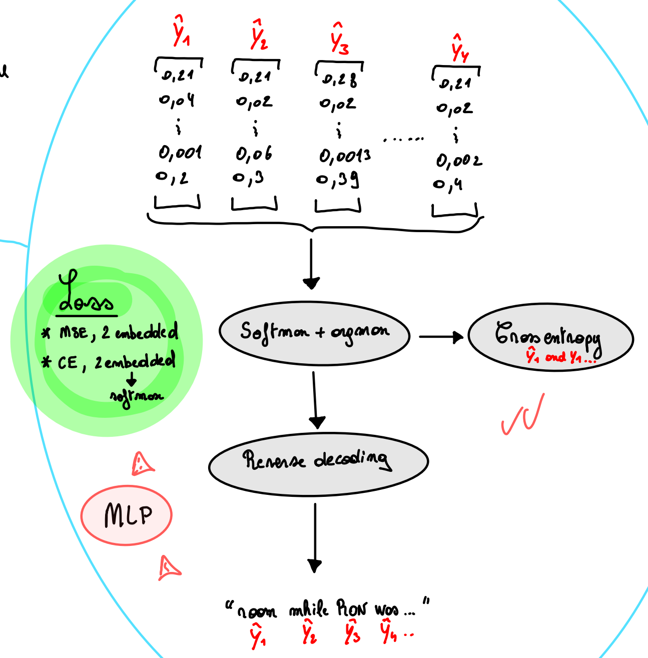
③ We introduce padding GRUs

→ MLP → n_vocabulary



Force Teaching with probability $P=0,5$ of giving the exact output instead of predicted

Reverse Embedding



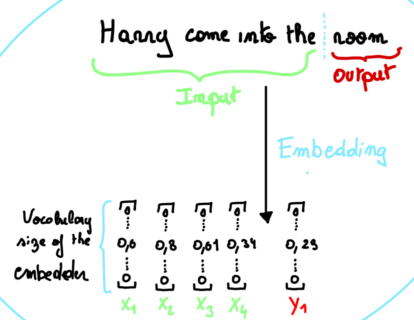
Embedder's information

① We use the pretrained embedder named GLOVE.

Training speed improvement → The vocabulary size of GLOVE is >> compared to the one of Harry Potter! Thus, we reduce the vocabulary of GLOVE to the one of Harry potter

Note: – Tokenizer gives a lot of info!
– Check missing words inside GLOVE (Ex: if "Harry" is missing we are fucked!)

② If the vocabulary of HP is small or GLOVE does not work well, we make our own embedder!

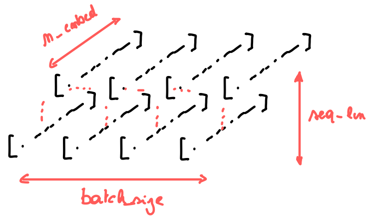


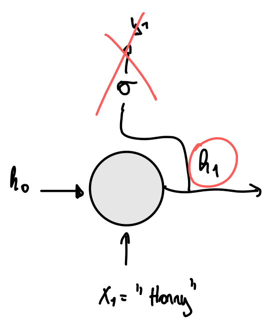
https://medium.com/@martinpella/how-to-use-pre-trained-word-embeddings-in-pytorch-71ca59249f76

Data cleaning

input : X

[X] = [seq len, batch size, n_embd]





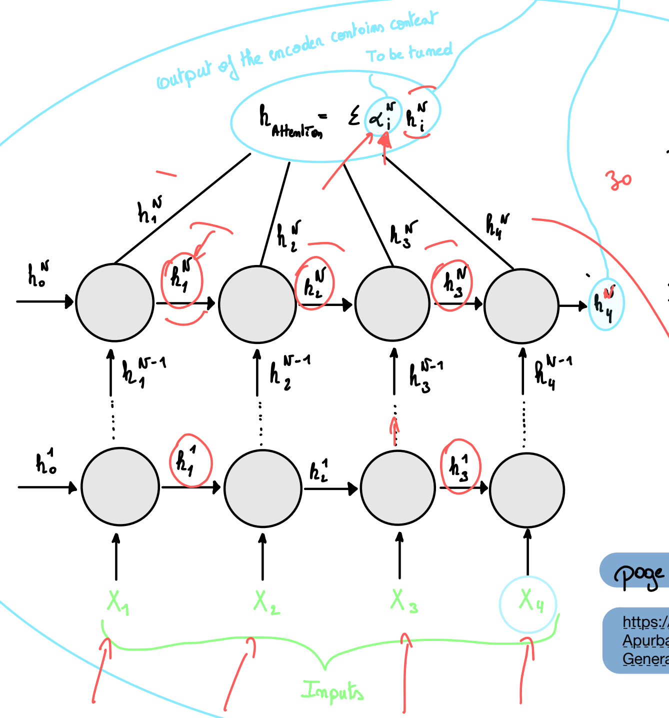
Implements from SBOT V1

* Output only words!
* Context is taken into account by — Encoder decoder / Attention mechanism (Fucking AMAZING)
* Longer output sequences (>30 due to attention)

Encoder's information

① First, we only use GRUs (CASE 1)

② We add an attention layer (CASE 2)



Attention mechanism (page 543 – 554)

X or (m_{max} – X') – Stacked GRU (must be TRAINED)

Note: GRU are as good as lem ongoing (Gaspar)

page 542 – 548

https://github.com/ApurbaSengupta/Text-Generation

Other stuff to do

* Inscription on GRADIENT
* Load saved model ?
* nn. dropout ?
* more info about embedders

SBOT V2 – Feat GASPAR

Plan of attack

To-Do List

* Functions : Cleaning and documentation
* Dictionary : Unique version
* Parametrization and dropout

TRAINING SCRIPT

* Training function
* Monitoring the training (What tool to use ?)
* Saving the model and KILL-SWITCH
* Load former models to test !

METRICS

* Crossentropy
* Bleu
* Rouge
* MSE

Attention !!

1- We must SAVE during each epoch the evolution of the crossentropy or/and MSE

2- When do we make BLEU or/and ROUGE measurements ?

THE REAL FUN BEGINS !!

IT DOES NOT WORK !!

VALIDATION PHASE

Testing our SBOT-V2.1

DATASET IMPROVEMENT

① Book → List of SENTENCES

② [<SOS> Input , OUTPUT <EOS> <NTOS>]
size l (OUTPUT <EOS>), size N (<NTOS>)
N = max sentence length – l

③ Adaptation of our functions to use tokens !

Note :
1- Glove has tokens ?
2- What value should we use ?

One man job

Inscription on GRADIENT

ATTENTION IS ALL YOU NEED

STARTING EXPERIMENTING

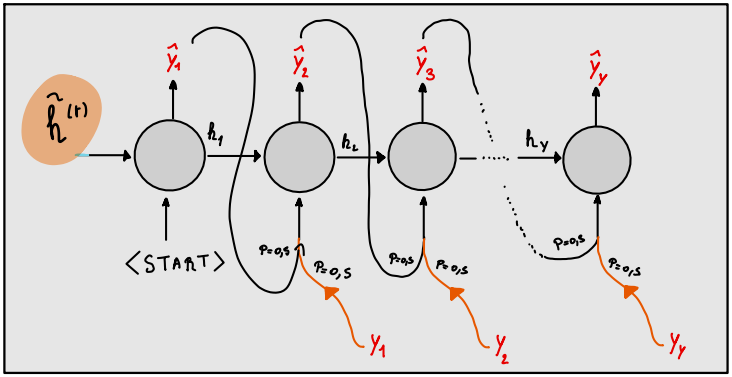
Unidirectional | Bidirectional | U+A | B+A

* For each architecture, 15~20 epochs on 7 books
  Increasing SBOT dimensions improve results (Better results or convergence ?)
* Comparing best model of each architecture:
  - nb epochs
  - Validation } (CE, BLEU, ROUGE) } Quantitative study
  - Test
  - OWN FEELING } Qualitative study
* With or without force teaching

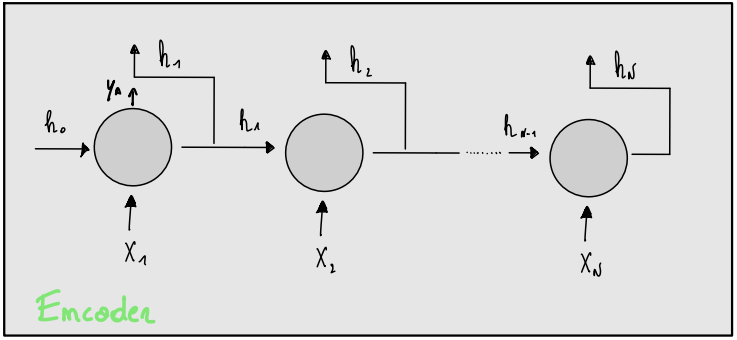
Note :

- We do not need any activation layers !
  ① Between ENCODER - DECODER
  → Induce a lost of info on h
  ② After the MLP use a SM
  → No because CE does it for us !
  ③ How to use CE:
  $\tilde{y} = [\ldots 0{,}2 \ldots]$ (proba), $y = (30)$ (index of $\tilde{y}$)

Attention is all we need



Attention

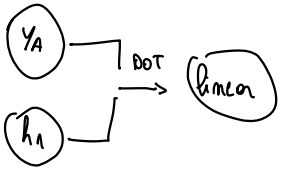


Encoder

SHAPE GUIDE

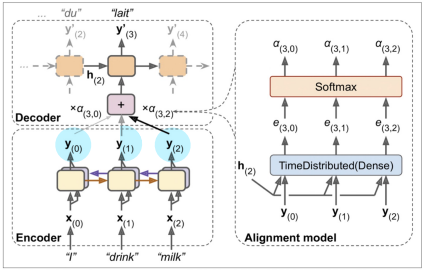
- X = [batch_size, seq_len, embed_size]
- h = [batch_size, n_encs · n_direc, hidden_dim]

COMPUTE ATTENTION ⟹ DOT product (others if I have time)



sec code !

Recapitulating



Equation 16-1. Attention mechanisms

$$\tilde{\mathbf{h}}_{(t)} = \sum_i \alpha_{(t,i)} \mathbf{y}_{(i)} \quad \text{(weights)}$$

$$\text{with } \alpha_{(t,i)} = \frac{\exp\left(e_{(t,i)}\right)}{\sum_{i'} \exp\left(e_{(t,i')}\right)} \quad \text{(softmax)}$$

$$\text{and } e_{(t,i)} = \begin{cases} \mathbf{h}_{(t)}^\top \mathbf{y}_{(i)} & \text{dot} \\ \mathbf{h}_{(t)}^\top \mathbf{W}\, \mathbf{y}_{(i)} & \text{general} \\ \mathbf{v}^\top \tanh\left(\mathbf{W}\left[\mathbf{h}_{(t)}; \mathbf{y}_{(i)}\right]\right) & \text{concat} \end{cases}$$

(scoring values)