Homework #3

Assignment For this third laboratory, the assignment is to train a neural network to produce a sequence of *n* characters in order to generate a text as if it would have been written by an author. To do so, the following tasks have been implemented :

- after selecting the dataset, reduced it's alphabet by changing and eliminating some characters;
- tuned the neural network (NN) parameters with a grid search and k-fold;
- with a *softmax* and a *temperature* tuned the final generated text in order for it to "make sense".

I decided to use as dataset **"Richard III" by William Shakespeare**, and everything about this paper is referred to it, but at the end I will also say few words about the same network trained with "Le baruffe chiozzotte" by Carlo Goldoni, "La divina commedia" by Dante and "The Importance of Being Earnest" by Oscar Wild.

1) Preprocessing In this first phase, I make the following changes to the dataset:

- eliminate the number, brackets and symbols;
- convert all the accented vowels in the correspondent vowels;
- convert the upper case letters in lower case letters.

The remaining characters are the English alphabet plus the signs '\n', ' ', '!', "'", ',', '.', ':', ';', '?' for a total alphabet length of 34. The changes have been made to speed up the learning process. Before being fed to the NN, the dataset has been encoded, so that to every char is assigned a number between 0 and 33, and then again a one-hot-encoding transformation is applied as in Table 1. It is then applied a random cropping and finally the vector is converted to tensor and the network fed with it.

Original letter	First encoding	One-hot-encoding
"\n" (new line)	0	[1 , 0 ... , 0, 0]
" " (space)	1	[0 , 1 ... , 0, 0]
...	...	...
z	33	[0, 0, ... , 0, 1]

Table 1: *From the left, the original character, the numeric encoding and the one-hot-encoding.*

2) Model The proposed neural network is the same as the one seen in the laboratory: I tried to change it by adding layers or using a Gated Recurrent Unit (GRU), but the network was either very slow during the training or the results were very bad, so I decided to focus on finding the best parameters for the network.
The NN is composed by *2* layers of *Long short-term memory* (LSTM) and a final *Linear* layer. The input and output size are both 34 and it corresponds to the alphabet length; the size of the hidden units is to be determined.
The network optimizer is *RMSprop* with a learning rate to be found; the loss function I use for the parameter search and final model is the *cross entropy loss*.

3) Training To select the best parameters for the training, I have implemented a *3-fold cross validation* and a *grid search* for *the hidden units*, *the number of input letters* and *learning rate*. The many networks used to find the parameters and the final network are trained in batches. The *validation* set used in the k-fold tests how the current network is performing, it does not help the network in any way.
The fixed parameters are: *the drop-out probability* as **0.3**; *the training batch* size as **5 000**. Both these parameters have been decide by manually tuning the networks many times until an acceptable loss was produced.
During the *grid search*, each network is trained for 5 000 epochs. To speed up this process (there are 27 networks to try), I decided to discard every network producing even just a single batch loss above 4.5 on the validation set. This "magic number" has been found manually, and it gets rid of more than half of the networks during the training. I decided that the best parameters for the network are the ones providing the smallest average loss in the *validation* set and I used them to train the main model; in Table 2 the hyper-parameters of the network.

Fixed Parameters		Grid search	
Parameter	Value	Parameter	Values
Dropout probability	**0.3**	Hidden units	[64 , 128, 256]
Batch size	**5 000**	Number of input letters	[20, 40, 80]
Alphabet length	**34**	Learning rate	[0.1, 0.01, 0.001]

Table 2: *Table with the fixed parameters (left) and the ones to find with a grid search (right).*

The best loss is obtained by using **256** hidden units, **40** input letters and **0.001** as learning rate.

To train the final model, I decided to use at least 10 000 epochs and then let the algorithm stop only if a new minimum loss isn't produced for 200 epochs.
Originally I put 100 000 epochs for the training, but I realised it was not necessarily and the loss was getting very bad after 90 000 epochs, so I left 80 000 epochs as possible upper bond.
The network stops the learning process after **10 673** epochs.

4) Post-processing The final output of the network is transformed using the inverse process described in the Preprocessing paragraph, so that it works with letters again. The output can't be used as it is to produce interesting results, it needs a *softmax* function to predict which character is the most correct, not only probable.
In particular I used a *softmax* with a *temperature* which is a parameter used to boost or inhibit some choices: a high temperature decrease confidence in the top choices.

5) Results In Table 3 I tried to complete some words with the proposed system (0.15 *softmax temperature*). Most of the times the model is able to finish a single word correctly, but it fails with words it has never seen before.

Input		Output	
bloo swor rich	->->->	blood sword / swords richard	With very common words to complete, it behaves as expected.
giraf telev	->->	giraft televntwore	When it has to finish unknown words, the results are random.
yyy *any new word* *it doesn't know*	->	yyy. yyy: yyyy, *followed by a new line*	With complete new words, it seems like it wants to move on with a punctuation mark and then it starts a new line with a new random topic.

Table 3: *Behaviour or the network with an incomplete word or a word it has never seen before. Some observation on the right related to the different cases.*

In Table 4 a full text generated by the network (0.15 *softmax temperature*); the English is ancient, but out of 100 generated words only 8 do not exist. I analysed another generated text and I found 7 errors out of 99 words. As in many other tries, the punctuation marks are usually at the end of a line, before starting a new one.
To check if the model is over-fitting, I took some phrases and searched them on the dataset and on Google. I never found a generated text copied entirely from the dataset, but some 2/3 consecutive words as common expressions and a maximum of 8 consecutive words in a single case.

blood of my enemies,
the kindnesse the death of all the world of care on my heart,
that will i makes be connedprans, great vs, and buckingham be brought to salsbury:
richmond in ongre to the death with all the morning.
in command liue, to greet the tender princes knees would haue it sode let to his ∗
thrie dead:
the king mine vnckle is too blame for her lords at pomfret, my lord of buckingham,
that appe outharies both vnkles that the moueare me forme,
and should be so dulling to brother of thy faire and falthing dispaire and warre ∗
mind heart,
that came

Table 4: *Final test produced by the network, input seed "blood of my enemies". The underlined words are wrong. The " ∗" denotes when I add a new line for this report, just for a better formatting.*

Appendix - other datasets With the same network, I tried to use other datasets without changing anything but the final softmax temperatures.

In this section I present some considerations:

- Le baruffe chiozzotte by Carlo Goldoni.
  The network perfectly mimic the structure of the original text and it produces a dialogue: every new line starts with the name of the person speaking. It is also interesting to see that not only many recurrent words of the book are produced, but also full expressions like "via, via" or "va in malora".

la mia locanda non ha mai camere in ozio, e paron toni canestro? ors. sior si. isi. a beppo va via e serra la porta ors. m o via, che te voi dar el mio credeu? tof. a mi, andemo, mando a caron a trovare el mio compare de diana, $*$ va in malora, lustrissimo. isi. via, via, mando a mi, mare de diana! anca i anni si, che no veggio. tof. m i so, da sara tittan ane, e nu ghe navera per mi! tof. sangue de diana, che no te tarta da parlare, $*$ se no la vorta sentire de dire, se no la ghastu a far dare,

Table 5: *Generated text from "Le baruffe chiozzotte", 0.4 softmax temperature, seed "la mia locanda non ha mai camere in ozio". The "* $*$ *" denotes when I add a new line for this report.*

- La divina commedia by Dante Alighieri.
  As for the Goldoni dataset, the networks still recognises Dante's structure and maintain the shape of a poem. It is interesting to see how it puts the punctuation marks always at the end. Most of the times it recognises and complete the structure '... disse: " .... " ', but many times it fails and only opens a ' " ' without ever closing it.

virgilio a me disse: "o me si pria che 'l viso e con le spare tanto, per che non pur a l'altra per tutta e suo seguito bianci vedere a l'altra per lo spente onde stella disciallo, e non saran per la mia che tutta seguita la fianta; e gia man che vedere a la sua man provanta, per che 'l santo mi fa che le ciglia che tu ti scellenti spenta, che per lo schermo discender le corte e contenti". come son la mia che la rove scarallaro, che vale in che si fa tolto si discostrava la vero suo parlare in su la balfagiante

Table 6: *Generated text from "La divina commedia", 0.45 softmax temperature, seed "virgilio a me disse".*

- The Importance of Being Earnest by Oscar Wilde.
  In this case the results are worse than the others: the model does not recognise the structure of the dataset and does not reproduce the dialogues, it's just a normal text.
  The model needs different hidden units and learning rate to work in a presentable way.