NATIONAL CHIAO TUNG UNIVERSITY

Institute of Computer Science and Engineering

Deep Learning

Homework 3

Author: Alessandro Saviolo

Student Number: 0845086

23rd December 2019

Summary

1. SimpleRNN

1.1 Dataset

The dataset used for training and testing the Recurrent Neural Network models presented is the Shakespeare Dataset. This dataset has a tranining set consisting in 1450424 samples and a validation set of 73995 samples. The vocabulary of the dataset comprehends 67 unique words.

1.2 Tuning Architecture and Parameters

Since the computational time required for training RNNs is expensive, the training of the following models for the comparison lasts 10 epochs. This is sufficient for comparing the initial behaviour of the different models.

A standard Recurrent Neural Network architecture comprises one input layer, one recursive hidden layer and one output layer (Table 1.1). The simplest choice for the recurrent hidden layer is to use SimpleRNN (see Keras documentation [7]), which is a fully-connected layer where the output is fed back to the input.

After fixing the architecture, we need to tune the hyper parameters. In particular, while the number of units for the dense layer is fixed equal to the size of the input dictionary, the number of units for the SimpleRNN depends by the task. The choice of the number of units for the recursive layer is fundamental for the learning capabilities of the model. This because the length of the hidden state affects the capability of the RNN cell of capturing the structural and semantic features of the input sequence. A standard choice is to pick a value in the range $[32, 256]$. Since the Homework assignment requests a standard architecture and the dataset is neither trivial neither complex, I choose to use SimpleRNN with 128 units as a starting model.

The architecture of SimpleRNN with 128 units is presented in Table 1.1. The hyper parameters used for training are listed in Table 1.2. The learning curves produced during the traning process are illustrated in Figure 1.1.

Layer	Units	Activation	Input Shape	Output Shape
SimpleRNN	128	tanh	(40, 67)	(128,)
Dense	67	softmax	(128,)	(67,)

Table 1.1: Model architecture. Note that the number of units for the dense layer is equal to the size of the input dictionary.

Hyper parameter	Value
Sequence Length	40
Learning Algorithm	RMSprop
Learning Rate	0.001
Step	3
# Epochs	10
Batch Size	128

Table 1.2: Hyper parameters.

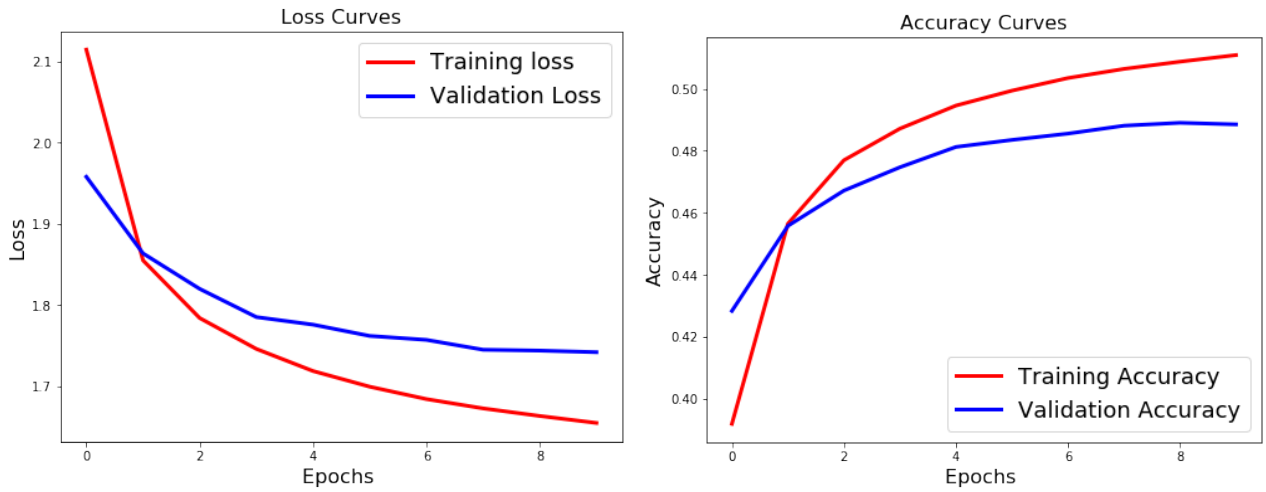

Figure 1.1: Learning curves produced by the model with architecture 1.1 and parameters 1.2. The loss and accuracy produced by the trained model on the traning set are, resp., 1.6543 and 51.09%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.7415 and 48.85%. The model required 265 seconds to train for each epoch on Google Colab GPU [5].

By analyzing the learning curves produced during the training process, we can clearly see that the model is overfitting the dataset after few epochs. In order to prevent overfitting, we can try to halve the number of units in the recursive hidden layer.

The architecture of SimpleRNN with 64 units is presented in Table 1.3. The same hyper parameters used for training SimpleRNN with 128 units are used (Table 1.2). The learning curves produced during the traning process are illustrated in Figure 1.2.

Layer	Units	Activation	Input Shape	Output Shape
SimpleRNN	64	tanh	(40, 67)	(64,)
Dense	67	softmax	(64,)	(67,)

Table 1.3: Model architecture. Note that the number of units for the dense layer is equal to the size of the input dictionary.

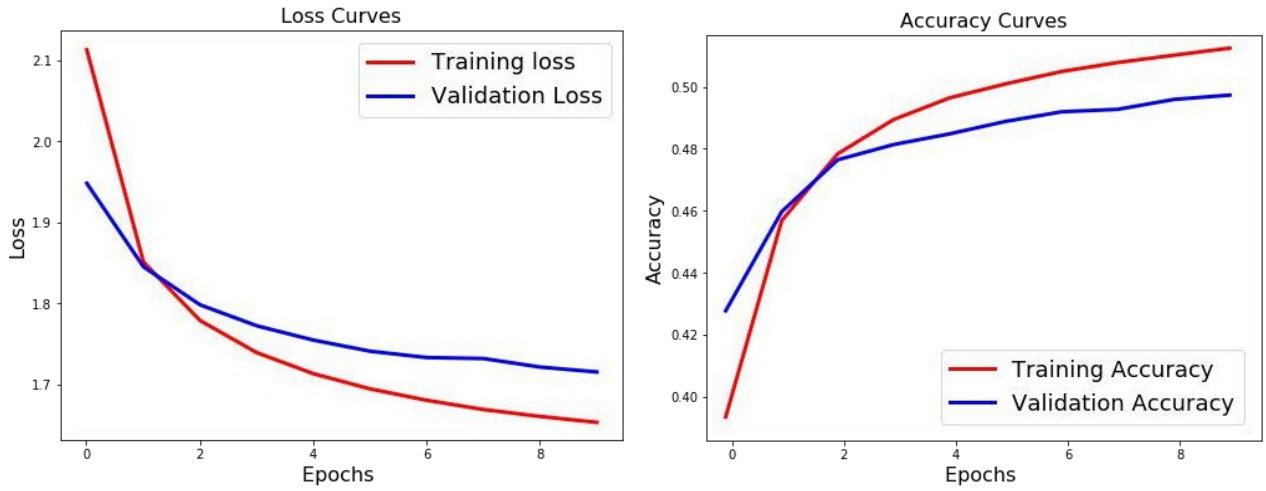

Figure 1.2: Learning curves produced by the model with architecture 1.3 and parameters 1.2. The loss and accuracy produced by the trained model on the traning set are, resp., 1.5833 and 54.30%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.7154 and 49.12%. The model required 233 seconds to train for each epoch on Google Colab GPU.

By analyzing the learning curves produced during the training process, we can clearly see that even if the training and validation losses and accuracies have improved, the model is still overfitting the dataset after few epochs. As a consequence, instead of halving again the number of units (since we still want a sufficient number of units for learning the structural and semantic features of the text), we can reduce the length of the considered sequences.

The same architecture used for training SimpleRNN with 64 units is used (Table 1.3). The hyper parameters used for training are listed in Table 1.4. The learning curves produced during the traning process are illustrated in Figure 1.3.

Hyper parameter	Value
Sequence Length	20
Learning Algorithm	RMSprop
Learning Rate	0.001
Step	3
# Epochs	10
Batch Size	128

Table 1.4: Hyper parameters.

By analyzing the learning curves produced during the training process, we can finally see that our model is capable of generalizing better (i.e., the overfitting problem affects less the performances of the model). But, there is still room for improvement. In particular, we can apply a regularization function such as L2 to the recurrent hidden layer for making the model even more robust.

The same architecture used for training SimpleRNN with 64 units is used (Table 1.3), but this time the recurrent hidden layer is regularized. The hyper parameters used for training are listed in Table 1.5. The learning curves produced during the traning process are illustrated in Figure 1.4.

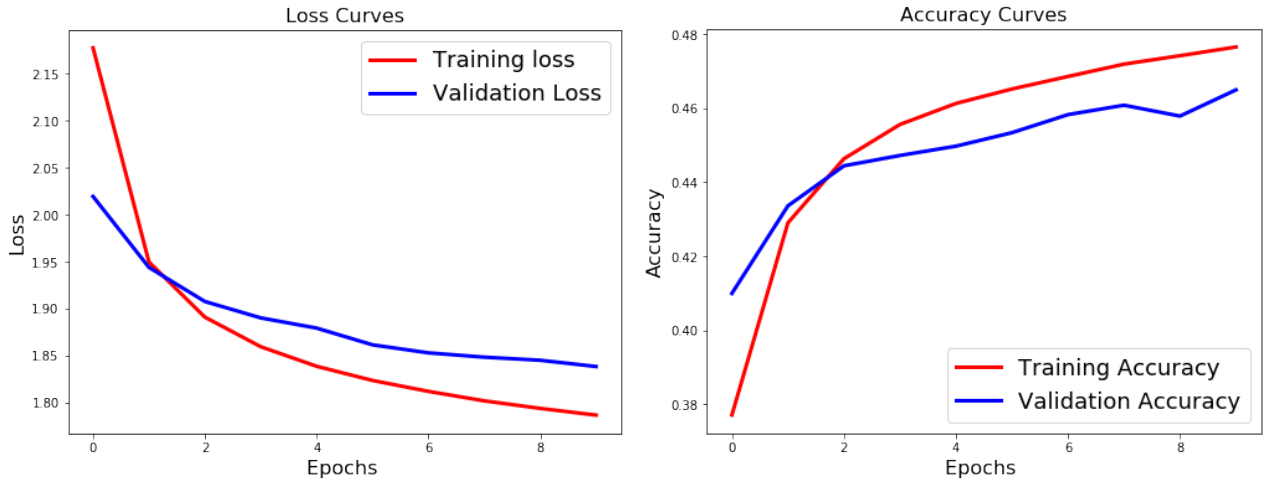

Figure 1.3: Learning curves produced by the model with architecture 1.3 and parameters 1.4. The loss and accuracy produced by the trained model on the traning set are, resp., 1.7233 and 47.82%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.8334 and 47.22%. The model required 174 seconds to train for each epoch on Google Colab GPU.

Hyper parameter	Value
Sequence Length	20
Learning Algorithm	RMSprop
Learning Rate	0.001
Step	3
# Epochs	10
Batch Size	128
Regularization	L2
Regularization Term	0.01

Table 1.5: Hyper parameters.

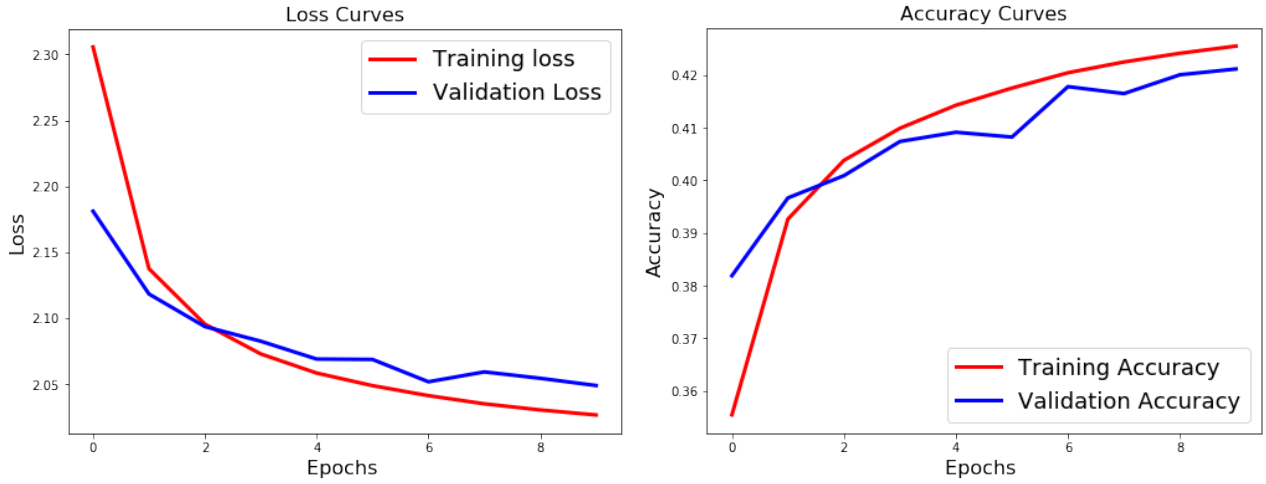

Figure 1.4: Learning curves produced by the model with architecture 1.3 and parameters 1.5. The loss and accuracy produced by the trained model on the traning set are, resp., 2.0312 and 43.98%. The loss and accuracy produced by the trained model on the validation set are, resp., 2.0593 and 41.98%. The model required 155 seconds to train for each epoch on Google Colab GPU.

So far we have considered only standard architectures composed of one single recursive hidden layer. Let's consider the case of a stacked hidden layer, where there are multiple SimpleRNN layers. In particular, the architecture of stacked SimpleRNN is presented in Table 1.6. The same hyper parameters used for training SimpleRNN with 64 units are used (Table 1.5). The learning curves produced during the traning process are illustrated in Figure 1.5.

Layer	Units	Activation	Regularization	Input Shape	Output Shape
SimpleRNN	32	tanh	L2	(40, 67)	(32,)
SimpleRNN	32	tanh	-	(32,)	(32,)
SimpleRNN	32	tanh	-	(32,)	(32,)
Dense	67	softmax	-	(32,)	(67,)

Table 1.6: Model architecture. Note that the number of units for the dense layer is equal to the size of the input dictionary.

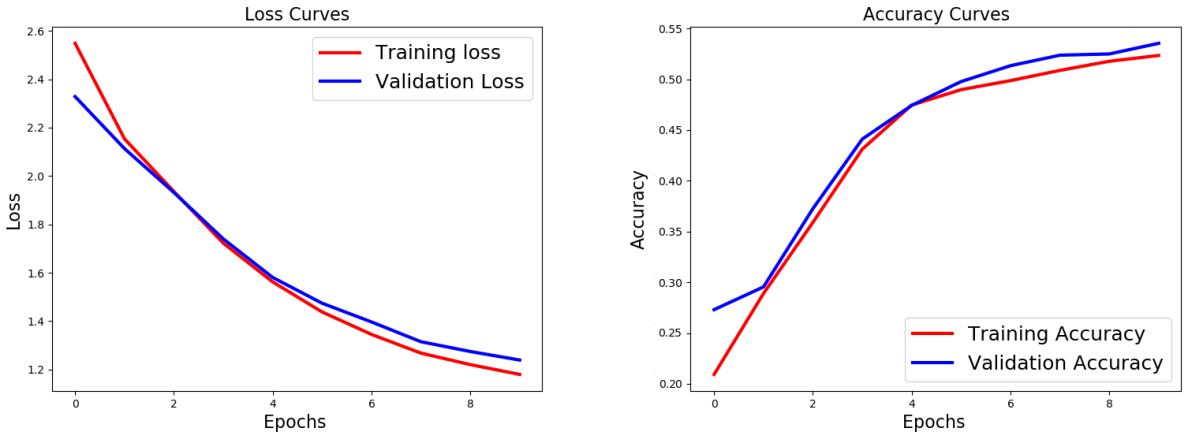

Figure 1.5: Learning curves produced by the model with architecture 1.6 and parameters 1.5. The loss and accuracy produced by the trained model on the traning set are, resp., 1.1932 and 51.21%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.2039 and 50.92%. The model required 421 seconds to train for each epoch on Google Colab GPU.

By starting from a standard architecture, we derived a more deep and complex architecture that produces more encouraging results for the first 10 epochs. In the next section, this model is trained for more epochs and some outputs produced while training are shown.

1.3 Final Model

In Section 1.2 we investigated a robust model for text generation. We started with a standard architecture and then, by trial and error, we derived a more complex model. Let's train again this model but with more epochs and analyze the performances.

The architecture of stacked SimpleRNN is presented in Table 1.6. The hyper parameters used for training are listed in Table 1.7. The learning curves produced during the traning process are illustrated in Figure 1.6.

Hyper parameter	Value
Sequence Length	20
Learning Algorithm	RMSprop
Learning Rate	0.001
Step	3
# Epochs	50
Batch Size	128
Regularization	L2
Regularization Term	0.01

Table 1.7: Hyper parameters.

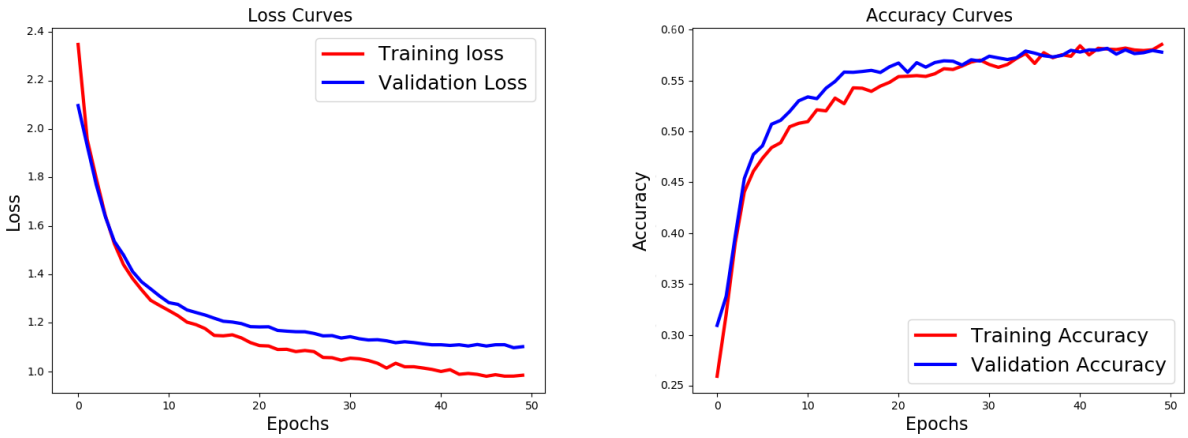

Figure 1.6: Learning curves produced by the model with architecture 1.6 and parameters 1.7. The loss and accuracy produced by the trained model on the traning set are, resp., 0.9230 and 58.12%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.2433 and 57.99%. The model required 422 seconds to train for each epoch on Google Colab GPU.

1.4 Breakpoints

In Section 1.3 we trained the final model for 50 epochs and we analyzed the learning curves produced during the training process. In the following we report 5 breakpoints produced during the training process to show how well the model learns through more epochs. In particular, in each breakpoint we feed some part of the training text into the model and show the text output.

Generating text after Epoch 5 with seed: "t howling. hamlet: what, the fair ophel"
t howling. hamlet: what, the fair ophel ands,–'moyh. must, to mysest tooly, lydin, and. spedus hus: abraver I omnce, come dint-wosilgeramy; answer's mirny: i whey eage no sh-hor poor panty scedy! so hold. who both vaces how'er. i menves, gadu. goden, when the saycwerikl coltanatie my pactinnated futf? beith. in hils aancviochumcbefesh hear; a ven 'er above to: my barn aighiish dedtobresclosion, dufcout: is yofo erpech.

Generating text after Epoch 10 with seed: "iches fineless is as poor as winter to h"
iches fineless is as poor as winter to his mark and by gods that i not not drib. lahot: gold shy, unto our to a wilge is in thy licin and wieched, a chulinoes lawiw, as in prooffol, it are irnabrim king henry ?sfedi: first mustrers: ca rrchstades him, these amles coldeds. sweotsid murder, make; here. batcrm: which, lady take setter and i she were near tamse aid lord hand,–his junis stuck, of how than hemwer, who crutus magk frown brencedio: may, my beach with like with you sinices trather, i wear thou fears and love this, men, for whereir

Generating text after Epoch 15 with seed: “to hear of her great the crown of the”

to hear of her great the crown of the commanded
and of him and made with the hand, that i do
shall strepsed that i shall be forth hand of his hear,
and is him the post of wherein the comes;
and be shall that i thought the auth command,
and and fall the brother concless where be proved the life
to the sot in the contint to the wife.
cardinal:
shall prove the hears that i do worth with
a shame that tho
masor:
what is the hill.
king richard in:
then as priden you?
tamones:
this, comely you my littolicaits,
before and marrins: awake
her, then must thanks.

Generating text after Epoch 20 with seed: “up; stand, and you be a man: for juliet”

up; stand, and you be a man:
for juliet and here that perform, bear and the servant.
daugala:
for the good and says your most noble is the stand;
and when he here so which and some some now.
shallow:
she shall be silth of first, he is she now;
where is the from no more worth him, for to
for propa some mine, and and fine.
margaben:
this perform, i am the great the to see on him,
and then be the forpiant of the fires hath here.
don airinius:
the sent it is the me to the charge of me,
the true of me the fire the served to see,
the strange to her of the proper of me the ments in the sing.

Generating text after Epoch 25 with seed: “t my eldest care, at eighteen years beca”
t my eldest care, at eighteen years becairlow. what in his track the love. brutus: the read and the such of nature of the dume that the word his longer for the speed so die? king lirto: i speak a confore so; be to a come of from chastence, which i am a madam. affolian: i have it in such became to in the lord. cleopatra: and you be the stable dead: and i will be come with my life and come i would thou wilt i do not in the man. brupon: crail, is the action see me upon le fortune, the mamp, but eaf my coese of my chamber to my sessening spurther, i was bold.

2. LSTM

2.1 Tuning Architecture and Parameters

Following the same approach used for SimpleRNN, we start from a standard model and then we derive, by trial and error, a more complex model suitable for the assigned task and dataset.

As starting model, let's consider a RNN architecture with one input layer, one LSTM layer and one output layer, and fix the number of units equal to 128.

The architecture of LSTM with 128 units is presented in Table 2.1. The hyper parameters used for training are listed in Table 2.2. The learning curves produced during the traning process are illustrated in Figure 2.1.

Layer	Units	Activation	Input Shape	Output Shape
LSTM	128	tanh	(40, 67)	(128,)
Dense	67	softmax	(128,)	(67,)

Table 2.1: Model architecture. Note that the number of units for the dense layer is equal to the size of the input dictionary.

Hyper parameter	Value
Sequence Length	40
Learning Algorithm	RMSprop
Learning Rate	0.001
Step	3
# Epochs	10
Batch Size	128

Table 2.2: Hyper parameters.

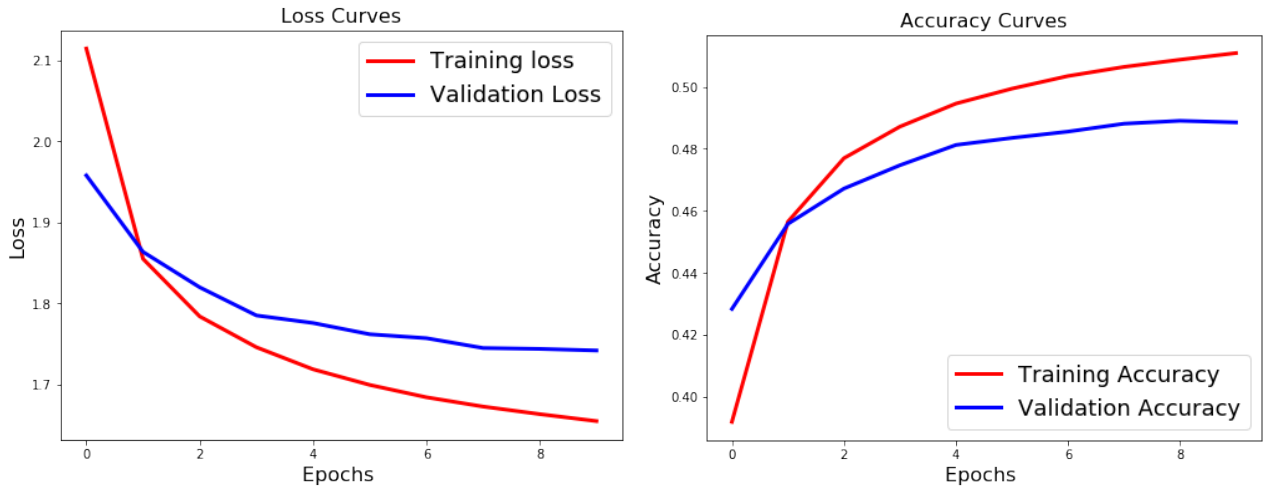

Figure 2.1: Learning curves produced by the model with architecture 2.1 and parameters 2.2. The loss and accuracy produced by the trained model on the traning set are, resp., 1.6642 and 52.23%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.7638 and 48.12%. The model required 735 seconds to train for each epoch on Google Colab GPU.

The LSTM model is overfitting the data after few epochs. Using the same considerations stated for the SimpleRNN model, let's try to halve the number of units.

The architecture of LSTM with 64 units is presented in Table 2.3. The same hyper parameters used for training LSTM with 128 units are used (Table 2.2). The learning curves produced during the traning process are illustrated in Figure 2.2.

Layer	Units	Activation	Input Shape	Output Shape
LSTM	64	tanh	(40, 67)	(64,)
Dense	67	softmax	(64,)	(67,)

Table 2.3: Model architecture. Note that the number of units for the dense layer is equal to the size of the input dictionary.

By analyzing the learning curves in Figure 2.2, we can clearly see that the training and validation losses and accuracies have improved. Following the same approach used with the SimpleRNN model, let's try to halve the sequence length. We expect the model to perform worse, since the strength of the LSTM is derived by combining long and short time dependencies.

The same architecture used for training LSTM with 64 units is used (Table 2.3). The hyper parameters used for training are listed in Table 2.4. The learning curves produced during the traning process are illustrated in Figure 2.3.

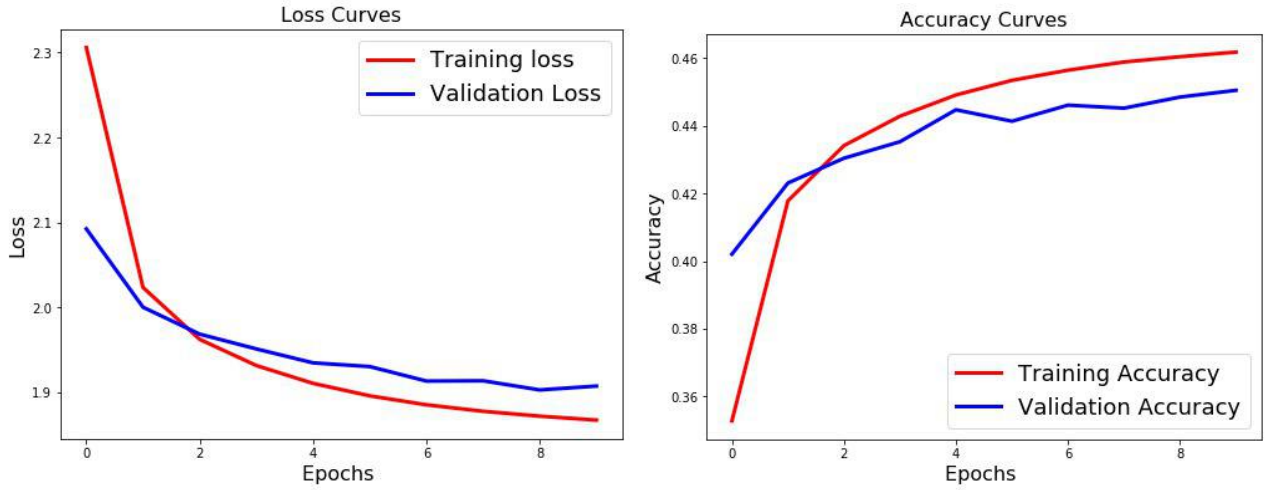

Figure 2.2: Learning curves produced by the model with architecture 2.3 and parameters 2.2. The loss and accuracy produced by the trained model on the traning set are, resp., 1.8832 and 46.02%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.9091 and 44.98%. The model required 622 seconds to train for each epoch on Google Colab GPU.

Hyper parameter	Value
Sequence Length	20
Learning Algorithm	RMSprop
Learning Rate	0.001
Step	3
# Epochs	10
Batch Size	128

Table 2.4: Hyper parameters.

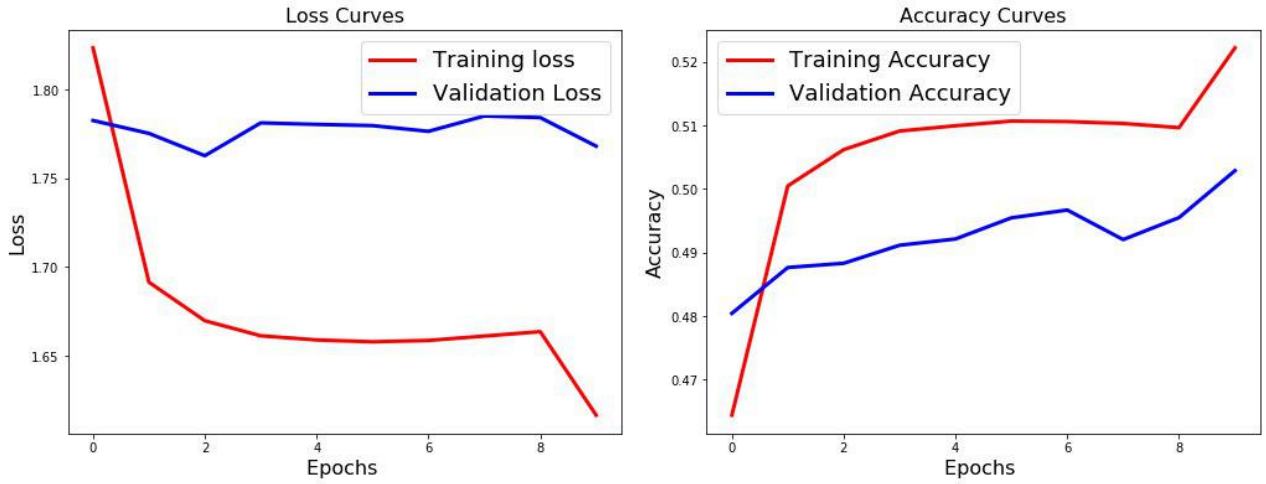

Figure 2.3: Learning curves produced by the model with architecture 2.3 and parameters 2.4. The loss and accuracy produced by the trained model on the traning set are, resp., 1.6192 and 52.02%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.7823 and 50.22%. The model required 584 seconds to train for each epoch on Google Colab GPU.

As expected, the LSTM model performs worse if we shorten the sequence length.

So far we have considered only standard architectures composed of one single recursive hidden layer. Let's consider the case of a stacked hidden layer, where there are multiple LSTM layers. In particular, the architecture of stacked LSTM is presented in Table 2.5. The same hyper parameters used for training LSTM with 64 units are used (Table 2.2). The learning curves produced during the traning process are illustrated in Figure 2.4.

Layer	Units	Activation	Regularization	Input Shape	Output Shape
LSTM	32	tanh	L2	(40, 67)	(32,)
LSTM	32	tanh	-	(32,)	(32,)
Dense	67	softmax	-	(32,)	(67,)

Table 2.5: Model architecture. Note that the number of units for the dense layer is equal to the size of the input dictionary.

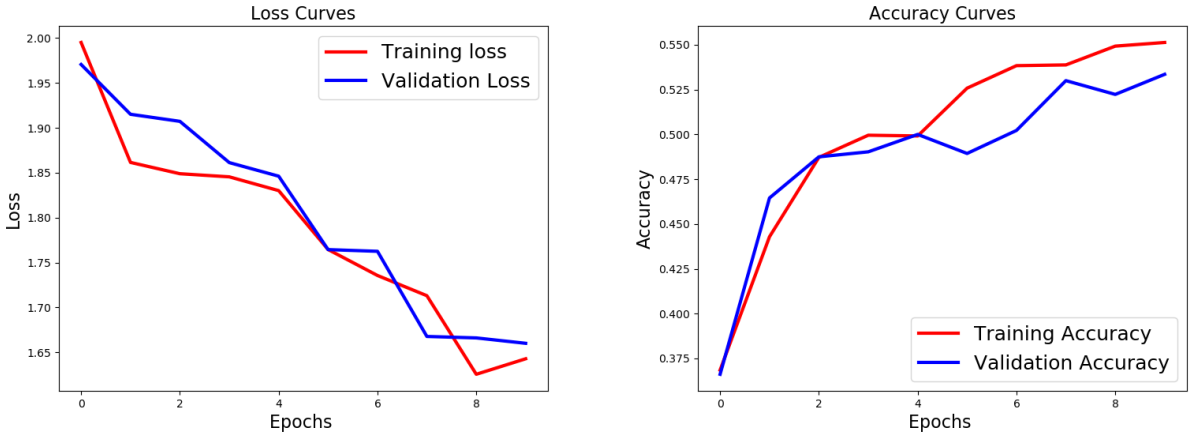

Figure 2.4: Learning curves produced by the model with architecture 2.5 and parameters 2.2. The loss and accuracy produced by the trained model on the traning set are, resp., 1.6428 and 55.13%. The loss and accuracy produced by the trained model on the validation set are, resp., 1.660 and 53.35%. The model required 1264 seconds to train for each epoch on Google Colab GPU.

By starting from a standard architecture, we derived a more deep and complex architecture that produces more encouraging results for the first 10 epochs. In the next section, this model is trained for more epochs and some outputs produced while training are shown.

2.2 Final Model

In Section 2.1 we investigated a robust model for text generation. We started with a standard architecture and then, by trial and error, we derived a more complex model. Let's train again this model but with more epochs and analyze the performances.

The architecture of stacked LSTM is presented in Table 2.5. The hyper parameters used for training are listed in Table 2.6. The learning curves produced during the traning process are illustrated in Figure 2.5.

Hyper parameter	Value
Sequence Length	40
Learning Algorithm	RMSprop
Learning Rate	0.001
Step	3
# Epochs	50
Batch Size	128
Regularization	L2
Regularization Term	0.01

Table 2.6: Hyper parameters.

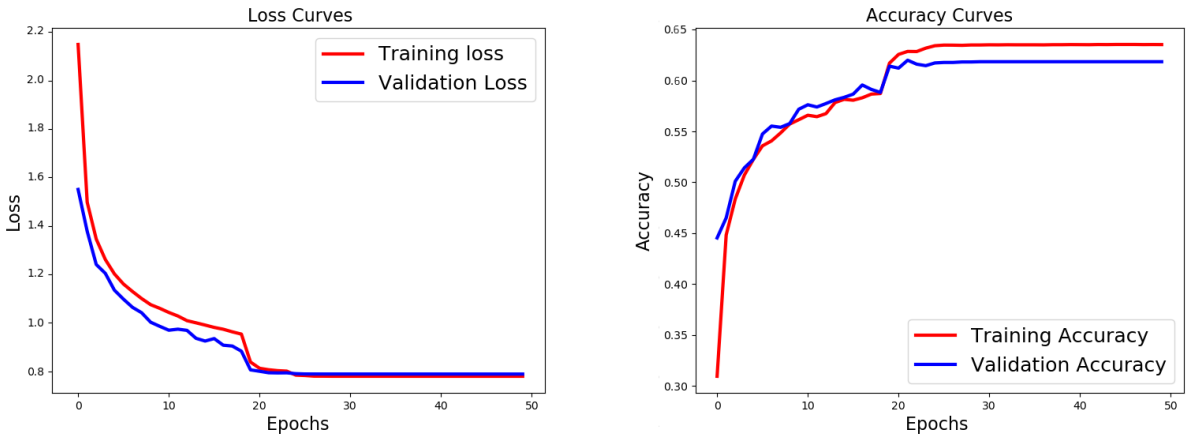

Figure 2.5: Learning curves produced by the model with architecture 2.5 and parameters 2.6. The loss and accuracy produced by the trained model on the traning set are, resp., 0.7938 and 64.12%. The loss and accuracy produced by the trained model on the validation set are, resp., 0.7943 and 62.33%. The model required 1264 seconds to train for each epoch on Google Colab GPU.

2.3 Breakpoints

In Section 2.2 we trained the final model for 50 epochs and we analyzed the learning curves produced during the training process. In the following we report 5 breakpoints produced during the training process to show how well the model learns through more epochs. In particular, in each breakpoint we feed some part of the training text into the model and show the text output.

Generating text after Epoch 5 with seed: "prites: if we obey them not, this will e"
prites: if we obey them not, this will e a nifier want a tobled the a, mateneiciess on culy; and have be libe ismirewer cry, I see well in will fay me ever I is swere him. ircliclio: upon that not shome grower i so werefenchen; i bodce a reads to undess? falpuund: i: ok and romediery know is to you, uperter softery forthrew, that welm'd tor trent the chuncle my a:e spelife a such call did that that so? nalstenia: and of indere

Generating text after Epoch 10 with seed: "long studying at rheims; as cunning in"
long studying at rheims; as cunning in vight! i must humicick, wonarger's thou fair you be gitce. aee matsolm: not a p; wome find you for one dighnead up, amanisish han the as, I hastiour liwel; i fools a hand forder, our still bague embreentdice, you a dock my lord martny, here, anivestre that i virlesh than not od adtion, and face prove a counanty, but'tsed but: why stoss this, if bose them. Ong i shore, and thy shall shord. inmost

Generating text after Epoch 15 with seed: “eat of blood; if frosts and fasts, hard”

eat of blood;
if frosts and fasts, hard my lord.
dolorig:
what now, when make the life of the since
and distrong and see the siming in fights.
seens bearr, sir, i will be with do be would promiled.
siminius:
i was have the some lords a most simess of this
to in the some empering and a country,
there is these shall see this of her some beliain not
and make with my stand stands and the send of this
eye so was the man a sand and feet to
haston:
sore, long unmy madgar. oary cale.
lakel:
but make me, lived meno!
and not as thou that thinks, come, disety alf,
and sicciud,–for chuld cut within, an charded upon and
masiced like pucbus yourthad in will shall truep’s actroitpories is
for your strong’d known, and pock, blessogwifcor:
this ledet, never in wear are.

Generating text after Epoch 20 with seed: “rosaline: fair fall the face it covers!”

rosaline:
fair fall the face it covers!
casenlus:
i thane i am thou art tell me all first here,
you shall know the more wart not my perpure better singing me be bears,
and the to their bear, that the courtes,
by the calls, and her father to the love
let them of the other neceive to the countencan!
menenius:
salficch, he shall affect the brave own hand in this
chifful heart to thy brother of bur the painters,
talard:
he love you now on, gow him which term, strate gloucester:
you will be she is the friends, their which well,
to man with the and beauty you and more
best thought the and for him in head of thee

Generating text after Epoch 25 with seed: “avour vilely. fare thee well. thou art a””
avour vilely. fare thee well. thou art and when so did mercy like i have some with his sir, and so make all his carry of them. king edward ii:tis and, the companion, the stranged of our and that call the stander than the is and the wear? beatricius: such i shall loved the through that a son that shall thou make you with the contlite than is so see; so doth an all the sent than my dear and mouth which to me, for the face, the man of a fare thee well. thou art against you from. katharyto: we have morth, the other to. conspiers: from the rame, and i through heavenly warwick’d now you an’t i worst prouod me that thou hunot fite, of laptus mose avempts and bave, but, for telly, and you should lovers ead af fil a conjuit.

2.4 SimpleRNN vs LSTM

Instead of a SimpleRNN unit that simply applies an element-wise nonlinearity to the affine transformation of inputs and recurrent units, LSTM model has “LSTM cells” that have a self loop in addition to the outer self loop of the SimpleRNN. Each LSTM cell has the same inputs and outputs as an ordinary SimpleRNN, but also it has more parameters and a system of gating units that controls the flow of information. In particular, such gates allow the LSTM model to operate at multiple temporal scales, namely fine-grained time scales (to handle small details) and coarse time scales (to transfer information from the distant past to the present).

Such considerations are supported by the different architectures analyzed in Section 1.2 and Section 2.1. In fact, the LSTM model achieves best results when using a longer sequence length. This because it can capture the long term structures and semantics of the text. This translates in a more robust model and better performances.

3. Models Priming

3.1 SimpleRNN Priming

Seed: “juliet”
juliet: sir, i could have told you the hand, his sons were that still. first citizen: when brought, that at her long to the show of, it that i stard it of little for his honourdance. juliet: what hath he was behind you at the son stand that and the women, and i should be live of the with the truth. corncample: i come my fellow. mark antony: thou did you must be her are that and suck to, that say his tenting was that come. heldian: i will sut the hundred that for the signior down, flown that i have which are are that ha falstaff: why, what?

3.2 LSTM Priming

Seed: “juliet”
juliet: feelingly of him, he is the cardress blood. cortain: the lail to him, that the play in man, he are not so man of the same be we are the chark in the way. macbeth: sell villains in the remains me to a both and the fair that with the shall with them, who worthy back of the court. king henry viii: oh your rome? there are thrust and ther words that in the contrrance to the words, and the buth to the countency. about lord: let the duke to his own of the state: the monsom and can but for the heart; and to a father in some endus comes you, then, the neck! why long found to him. you do! ragip? diana: i ammars fit digns! price the promisi.

APPENDIX

A.1 GitHub Repository

The code of the project is available on GitHub, in the repository [4].

Bibliography

[1] Bottou L., and Bousquet O. *The Tradeoffs of Large Scale Learning*. Optimization for Machine Learning. MIT Press, pp. 351–368, Cambridge, 2012.

[2] Nair V., and Hinton G. *Rectified Linear Units Improve Restricted Boltzmann Machines*. ICML. 2010.

[3] Taddy S. *Stochastic Gradient Descent*. Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions. McGraw-Hill, pp. 303–307, New York, 2019.

Sitography

[4] Github Repository, last access 23rd Dec. 2019, [Online].
https://github.com/AlessandroSaviolo/DeepLearning_0845086_HW3

[5] Google Colab, last access 23rd Dec. 2019, [Online].
https://colab.research.google.com

[6] Keras, last access 23rd Dec. 2019, [Online].
https://keras.io/

[7] Keras Recurrent Layers, last access 23rd Dec. 2019, [Online].
https://keras.io/layers/recurrent/