Neural Network and deep learning: homework 3
of Tommaso Tabarelli

Abstract

In this homework we are asked to modify a given python script in order to train some Recurrent Neural Networks (RNN). The goal is to learn main features of RNN and use them to try to learn a text and make the RNN write a new text using a simple seed.

Code understanding

First of all we had to undestand how the code works and how the Recurrent Neuran Network (RNN) model is implemented using pytorch. Moreover, we had to understand how the two examples of datasets (sonnets and a text) are loaded and pre-processed to have clearer and simpler data to handle (for example, removing useless *new line* characters, removing rare puntutation signs, etc.). We were also supposed to understand how the train process is implemented and how to retrieve its results to generate a new text.

Code development

My choice was to work with jupyter notebook since I find it more familiar and cleaner; anyway a python script was finally developed as homework requests.

After having understood the given examples the choice was to modify training text (in this case the texts on which train the network) and try to understand better parameters influece and to tune them.

The parameter changes were focusing on the number of hidden units, hidden layers, number of epochs, the batch size, the sampling function to generate new characters/words:

- for what concerns hidden units, the expextation is that the more they are the larger is the number of states the network can learn, thus the higher the number of different *contexts* the RNN can handle. In this way, incrementing the number of hidden units one expects that the network improves and can guess better the following character because its internal representation of the state is more complex. A disadvantage of larger numbers of hidden units is the train time increment (which is not a big deal, but it can become quite important) and the fact that to be trained it needs a larger training set, which is hard to achieve considering the following argument: if we want the net to mimic a style of ah author (especially for what concerns sonnets) the training would be performed on the poetries we want it to mimic and in this sense we can not enlarge the dataset adding different pieces of the same author, otherwise the style would not be the expected one (of course this is only a *theoretical* argument; indeed my networks can not learn to write some common words, so they are far from learning an author style...)

- for what concerns hidden layers, the considerations are pretty much the same of those of hidden neurons: increasing their number one expects that the net can learn more complex patterns, but it would require more training time and more data to train (without increasing much the performances in my case);

- number of epochs: the main goal of our network is to learn how to write starting from examples of text: in a certain sense, it would be better for the network to *overfit*; in this case indeed the

more the net trains the more it can achieve an acceptable result and mimic the text it learned from. Using this argument, the number of epochs was incremented to be between 500 and 2000 (depending on the task: the net can spend much time in training so the number of epochs was adjusted to have a reasonable amount of time for each training procedure);

- batch size: this parameter really depends on the dataset: for example, in *Songs of innocence* the elements were chosen to be the sonnets, in *Picture of Dorian Gray* the elements were the chapters, while in the *word2vec* version the elements were chosen to be the sentences. From this point of view, this parameter was considered not so important. In general, it was chosen to be small in order to have more than one update per epoch.

- sample function: to try to avoid the generative process getting stucked in some *cyclic* iteration, for example, repeating the same pattern of characters (usually quite "small", involving 5-10 repeated characters) the function to sample the new characters was changed from *argmax* to a procedure that samples from a softmax distribution created using the network output. In this way, the network can be more *creative* and it can avoid repeating the same *characters patterns* (*words paterns* in the case of *word2vec* architecture).

Some result examples are given in the appendix.

The choice was to implement different networks:

- one of the kind *seq2seq* to be trained on sonnets using Blacke's *Songs of Innocence*

- another *seq2seq* network, to be trained on Wilde's *The picture of Dorian Gray*

- third one was built using *skip-gram word2vec* architecture setting embedding dimension to 5 (arbitrary choice) to be trained on Wilde's *The picture of Dorian Gray*
(link: https://towardsdatascience.com/implementing-word2vec-in-pytorch-skip-gram-model-e6bae040d2fb).

Seq2seq for sonnets

In this implementation the RNN was feeded using Blake's *Songs of Innocence* sonnets. The dataset was split following the example shown in class; after some initial trials, a *good alphabet* was chosen (hardcoded in *dataset* class) in order to keep all more common punctuations while the less common were tranformed in commas during encoding. After some slight changes (fixing batch size and fixing the use of Dataloader) the RNN was trained on the sonnets.

To try to generate new text the code was modified not to use the *argmax* function on the network output, but to sample from a softmax distribution applied to network output.

Seq2seq for text

In this implementation the RNN was feeded using Wilde's *The picture of Dorian Gray* text. The dataset was split following the example shown in class; it was also preprocessed to eliminate *bad punctuation* to make the analysis simpler from code viewpoint. After some other slight changes (fixing batch size and fixing the use of Dataloader) the RNN was trained on the text.

To try to generate new text the code was modified not to use the *argmax* function on the network output, but to sample from a softmax distribution applied to network output.

Word2vec for text

This kind of implementation was more demanding; *The picture of Dorian Gray* was used as train set, and it was preprocessed as in the previous case.

The general idea of this approach is to represent words using vectors (of arbitrary dimension; it is explained shortafter; my choice for the words embedding dimension was 5) and use those vectors (instead of *one hot vectors*) to try to generate new words in the generative process. Of course the whole RNN structure has to be reshaped to handle this kind of changes.

To start, the text was split to take single words and build a vocabulary. To achive this, all spaces, all *new line* characters and all punctuation characters were eliminated and whole text is transformed to lower-case. After this, from the vocabulary the *weird words* are erased (for example numbers and null character “”).

At this point, a vector representation of words is needed. To build it, sentences are needed in order to have a context to learn words links. To get sentences the text is loaded again: after removing all white spaces, the text itself was split according to punctuations and a list of sentences was generated; only sentences with more than five words were kept, the others were discarded (splitting using punctuation gave some unwanted sentences, such as single names or single words which maybe were between two commas, so the decision was to discard this kind of sentences). To go on, words also needed a context window that represents how much they *link* with other nearer words (the larger this window, the larger the number of words "near" the considered word is taken into account): this was chosen to be equal to 2. In this way, all possible pairs of words involving a *center word* and four context word (two previous words and two following words) were created. At this point, a feedforward network was trained on the pairs:

- its input size corresponded to the vocabulary dimension; practically, it was feeded with a *one hot representation* of the word with respect to the vocabulary.
- first layer had dimension corresponding to the *word embedding dimension*;
- output layer size corresponded to vocabulary dimension.

This network was trained so that the representation of the words corresponded to a vector which size corresponds to the embedding. At the end the result were represented by two matrices (the weights matrices), first having dimension (embedding dimension, vocabulary size) while the second being (vocabulary size, embedding dimension). They contained the representations of the words in two different contexts: the first contains the representation for "the center word", thus when they apperar at the center of the *window context* (as explained before) while the other represents the words when they appear as a context for another "center word". This last representation was chosen to represent the words (here the choice is only matter of convention: one can also take as representing matrix an average of the two matrices (properly transposing one of them), see link: https://towardsdatascience.com/implementing-word2vec-in-pytorch-skip-gram-model-e6bae040d2fb).

The next step was to modify the given RNN to be trained using these five-dimension chosen vectors. To do it, the network was modified to be able to take as input 3-dimensional matrices having the first dimension characterizing the batch size, the second one describing the crop len and the third one being the embedding dimension (thus substituting the one hot version).

The training procedure was similar to that adopted in the given examples: a crop was randomly selected between the sentences and its last "word" (here a 5-dimensional vector) was selected as label. The network was then trained using the rest of the crop as input. The loss was chosen to be MSE loss: it is evaluated between the network output and the label, which are two five-dimensional vectors in this case.

In this way the trained network was able to output a five-dimensional vector. Of course it almost surely was not present in the word representation list, so to estimate the new word the following procedure was adopted:

- all distances between the network output and the word representations was evaluated;
- their inverse were evaluated (i.e. $1/d$ were evaluated $\forall d \in$ *word representation distances*);
- the inverse of the distances were used to create a probability distribution via a *softmax* procedure;
- the distribution was sampled and the corresponding word was chosen to be the next.

Another important thing to notice for this network is that it is conditioned by the vocabulary and its construction: for example, if a word is not in the vocabulary it will never be written (while in the *seq2seq* architecture there are no "prohibited" words); moreover, the seed must contain only vocabulary words divided by spaces, otherwise the network will fail to interpret it.

Results

The results of the network executions are very poor. The three network can not build meaningful sentences (the *seq2seq* architectures usually can neither achieve printing some meaningful words). Results are reported in the appendix.

For the python script it was chosen to use the *word2vec* architecture since it at least writes comprehensible words (althought sentences are usually meaningless).

Comments

This exercise was useful to undestand how to deal with RNN, how to use them to try to predict simple features, for example characters or words in a text, and the main approaches that can be implemented in these scenarios. Moreover, this exercise helped me learning how to preprocess a text using *regular expressions* and string handling functions.

Appedix

Here some results are reported.

seq2seq network trained on Blake's *Songs of Innocence*

Parameters:

- hidden units: 400
- layers num: 2
- dropout probability: 0.1
- batch size: 16
- num epochs: 1000
- alphabet length: 31

Seed: "the"
eriss fius byreyr,
and peadsttey,
hey the heant the fheen
the heennord.d,
d,uuy
and bime awsast the , lins,
and whe nidee,
and whis the endist hy sheep,
sees boaghf herl, cod,n
win hoy mad the ladsead,
hy yaathros toed hor ghee
is the grean.
shein hemsr iscanpsoir nowh bathe fret, creallided fadtaerithe harr,
suet is eery,

Another example using the same network, but different seed:

Seed: "the tiger was created by"
sislame
if af afy lothe cloml,
in
he sheen,,
he nols theanorend liud ast meattsezs bathen, lece tor heanitshed
bighen,s meathost bere
,
owe creend,
duw, suet so men iw mos flole,
in the wteeep,arnd,
heur he pasasttees ardapt.oth mererorf.
diun the leirthe,, sfitht on in ,faby,

su bind dell,
on the greek,
in she broant wavlenerew,

Another network, here trying to use *hidden layers = 3*.
Parameters:

- hidden units: 600
- layers num: 3
- dropout probability: 0.3
- batch size: 8
- num epochs: 2000
- alphabet length: 44 (this was one of the first trial and the alphabet was not shortened; this choice was made to see if the network could learn the punctuation behaviour, but of course soon I realized it was not able to predict even simple words so it was far from "understanding" punctuation.)

Seed: "little lamb little lamb"
, fnek, ou rif lolt fpith mir daer,
un bif ny
and solteeo i deavy wheye i cre,p,ping bede
tuf the fristslond sher,,
o binepthe shith soeed woud,
as anpoore
diw the ctoend,
dfwangin sivt eratho sil ssoulg,.
sile niphe, the heartad the heer,rolg nius the waleteas,, and
heurt weessheresanp
heart orees
he palyssats begee
anr nigh fith ghe her cleand
if haathar.d,

Another example using the same network, but different seed:

Seed:"burning tiger little lamb"
e
s,ade, hererat n wist lolen,
and npghe gillad,wafw,bey iroen,
in the bayadtawhe hey roarn,
fitl woat dfskeog,
leve yiad the ly is aud a the s and afd whot oread,
du csangheef he wheakirilgs,
and hath foren,
wiu beekin the merrill.d,
ezy onaptoeg theu nothen,, and hearttor iso ssark,

fu mrce cislds,f, cton t ored ond hapt feep,
iw e ghe sill thoue deasn,
an the haghlene he desal.
anvt bo if feev,
and boslofu woagn deed,

seq2seq network trained on Wilde's *The picture of Dorian Gray*

Parameters:

- hidden units: 1024
- layers num: 2
- dropout probability: 0.2
- batch size: 200
- num epochs: 2000
- alphabet length: 42

Seed:"the beautiful shape of him was glacing out the window"
thsyevd suceae tf . leu hiire yef taat ar ii ctielsyhu. mem tl iley ih i ohiles oueceodee uaccaed ourthhd nude
hed b setees oue aovean th i seiriie h p len hi oseyril i sennppoued aud setttacciccif ctthnl piuca phills suucaee torrrhppgppsulwsshe
phey lid b sesest.ud ae wiiewyythe oarirc.s ovrrhee lupcpeervids ha pagog hon h seusyy."m aud b uetched wuued hen o hig.. tuocsaded i pacotose fhyiyas m uiwhee pupedees medee side o h lg. hi.e tudyp her lisis chul se ; arweee urpcceed loded u p ghee omeepps.m. churssssussann ou aney ou iacciuer hu h ald wouesshu.
ede in tiurtict frrlhe muu"settu pllel ii ssiclas h men o pig.gosed

tsuct.es l uraspmoppther liy a uha.w.m

ssentttuu shat l irwsptmvvvvsumcpevctl h seeeh iud aoncee torrtace lerriiss hen toueda fudee gt.ed af o ohwee ie pacesrhen hide tofe hay v urwhes hedm sot"a, puurtaye luyrind m tenppp.e oa y lew he s ahwsat uueeev uuucteer lairiyss,nt il i aetthe oardghem
sse"ptte tiseel uu ca wallt oo..ysy"und b nen he s arwxche soumer un moles hi oa weil ii solecahe
haytts s ulweee mupemtpen oarwhcptuhed munee hed oacwaled ou coalwaytu .etee oaew.yyuundss tuucae b .ereep opmceee o oag.e ogopeee hi hed men torrrnde ouuy fhr lidy in h uenohe ohede woaed pe. ofyrlhe .ude
aud to.edyyiidd oa fadee hid b thtlld ou.e am rileiyyiudc ou tolen ooppyeerm iud aa u uhwe. touee ph pomee hi hided hid b festhen uuiwcav mouitacth ctlee hii adyitvtth tlerwicss hhn sitttsle hiue yeur :h .gss hhn wises h f a oale ooeedyyt"nd bgrrtaer au o iuweee oue meee l urthet ou phetgoil, ha gotv. so.esyyf phytll oarweemm pemcteer liiryyytfn wotrthe opopheem puummene mumetttu poaltil!

Another example using the same network, but different seed:

Seed:"dorian was staring at his portrait" m
m.

phitrsssuussenn uu waeee oop!yemen ouaceed and oo itagtigtssyrinn oh h s u whi.. mopen phe toarvyctn p uewe he hsyyin i ahrthe oarwhes oummeed me coarwade uaft.ed bide u 1rg.. oroseem mhem
ued boueed hectover th t lede oo.yp?ttttetl uuusaee oarddppude aude seti th llis uh ysal; h uen wou poied ouyymeve ciuedy th i seiri hed wouetd tof t thill miued pun peee ar in bnese hel b uetthe oarthhe omedeppuue
ettuntee siureee uureee ha tovcra pouyw.hd uaccaae oarwhhd mude
pe cseee lid a tofesedd oupedes ned i hetncded au tourwaae hay porowhpe ao"rwhem aud a uetthe uaetaad ta taee litos seuriled aude agtto tale oouesedd
mumett f . :he lles yhim youen ofp tomgwhes ohyy miurwofed h pgled huu 0fyrin bi oi wilee oi yeyvris uncctlrrgci . wher liu a uhalhd pimee hed i tetlen hiue
au peeeryitsds uuned fg oa wtuet oh m l uses hedm otcy hhd wissssit led b toseee oueye serrtoar lh t uew.hem pemctee oarwhctgvisvd tn cad. gotoasssmuu"vettuncs arilppmppeectine h f pgee oph"phv. mumeee pumeee pour2atetqrthrtlls auumaf urtitaee he coagads onrtthr. hid b shalld ig oseirii hed m tet.e. oude top"tshe. sudeb at ia wgee l ued wh i lee h uetthyp thrtls oh y umiens iudye te cille oi pieecyttu paacwise hay wg.raacsm hen siartan surrthet ou wachit hed wouend pgmene ua waeen cu palc.gtound yturnd semr-ine af i urwee oopesyprn lou;
hi h .el oh psyevv hnn a fetltdg
aometee soueethud m uet.se puucmeee liu of yilriine momed tu cisee ou y chwel liue
ph ilse huu pa hallw hed iy is iiiii he hide hu regtga. l fhylldd. sude
hed oo waage houesyytu ciget hid b feiswdjpmuten tuiredd uudetttt hasgs ss.ssssum whillyyiuusssett al tles liiriyyiun of fnne ou y ugwine oppmeppmmpu,cmmche lamrcaf orrthe poeed yed m tetshe xuuede thacevd uunee f alee ooursees muuedd puzcae t vericee ouede toued pon p arw.cmmmmmesm med b tewees mumee hed siseey thactod b ihithe uaccaat ofrrrhed hide of tfey oo oa woues h urpfyy.g

word2vec network trained on Wilde's *The picture of Dorian Gray*

Parameters:

- embedding dimension: 5
- hidden units: 500
- layers num: 2
- dropout probability: 0.0
- batch size: 20
- num epochs: 1000
- vocabulary length: 6779

Seed: "the beautiful shape of him was"
time invalid fine threw diaper three silver far reed faithless strawberries flooded meals alive mania utter unselfish shown hazard ravelled assuring defend loosened marvel console benefit tulip communicate led finely sticks recovered grande suggested consistent reflection warmed chaos treasures

pin silk untouched worth grimy stacks curiosity subtlety dreadful viands glossy stem lateness refused track drizzling decide nay egos vitae hill clamps berkeley clammy myself discuss upright sham sensual activity bottle might abused canons covered kneel renunciation mode smelling jaspers drive dearly glimpse jealous picture excuses frequent opportunity evidence tyranny childish dinner disciplina parody packed gaiety imitative pushes possesses inverted gloves husband diary parasols message survive cylindrical magical ritual tiaras plato wrapper foul amethysts leonora mutilation country frankincense antinomianism consented experimental fuss friendship rupert comforting characteristic i which give clothe berkshire emblems assure chasing snow begins depresses slaughter wide petulant chequebook staccato relief against sweeter flattened advertise garden eat mode lock doesn various rates fibres roused camp wearisome including gone suffering faudel wrote silly paradise wait uncivilized mortem grown worried splashed riding board welled music govern wearied ranged stepping presence endures domestics siege future situations haunted performance incarnation scoundrel intolerable schoolbooks unlike otherwise ashes exactly couldn beetle ascent lotus duke vileness truculent ventured laid hinted plants pane invents survived etching harden recklessness horrid imaginary makes badly draw near preacher flattened bells fast clenching bringing picturesque rio imperial praised phenomenon wall founder flung vat dandyism crawling desired per organ found flies resemblance have presented sin sparrows coaling facts was prisoned ushered scourging covered laburnum splashed tightly narborough inverted affect coroner slinking grande saloon artistic wilder reminding ways fears beckenham impression faculty curtain crosses captain recollection sojourn directly collars arrangements untrue destroyed omar crime grape gave how tempt consulted definition disappointed transformed ague dragging unhook pardons penniless length exquisites pearly catholic chap appetites varnish rough

Another example using the same network, but different seed:

Seed: "dorian was at his house"
by me players dutch book apple siphons yesterday footed thief movements superstitions amarre wine camillus pebbles pins searched widening temperaments shield moon momentary intentions christian spectators consumption nose untouched isotta see estimate hooded oratory crushed lock cedar illustrated point sovereign recovered stamp dreadful infatuation disk civil christened verities paint devoted take portico beasts eighty jacobean lash intellects droop flowering scolding renewed disappointment gable stock came furry lateness called crumpled down same loses got cheated tussore head cruel buzzed awakened frontleted parts whatever between other nuisance trampling bedside culture happen present sinner fallen grass emperor arrangement feed omnibus lurid bamboo sup consequence tapped demons proposing shaken monotony seizing shaped slander girl unpardonable audace superficial twelve bamboo lives and passion faster schoolgirl kissed sulkily wardour curiously curiosity rich dry gardener lovely callous player intricacies jerking comeliness quickened lashed gold tonic pagan sunlight broker holding fault tulips let faithfully giordano cries nugget trembled singleness spinels stroking sadly branches court thornbury pinnacles blast bread anthony sight colourless reflected informed depresses beard blessed exquisite flew worlds candleshades rustle satin bonnets gardener ways forgot stifling means reverie theories thin asceticism fondled egyptian peer romance wants glimmered beats elocution irresistible mock bitterness heaps digging crimson broideries transformed frank hellenic killed isn taught heavy rendered mauve tulips happier glimmered whole strained creatures indians chapters definition though grossly desired bright dust unbecoming victor curiosity novel loves untouched representative trickle wisdom faded horns some rot marlow rainbow bubbles leer vestige indifference bournemouth collar violets hadn daintily sullenly kneel lean cane more may discords certainly breathing seeds paragraph perplexed boring cookery die cleft deepened kelso deadly fads pollen bells on pilier certainty garments endure singular small strip court denied renunciations

artemis rod shooting l audience eighteen experience bullied if lurks crowd ever shimmering analysis hypocrisy year doorway tightening philosophic toned leafless pockets moment