Assignment 3: LSTM

Deep Learning

Micheal Denzler

November 28, 2019

1 Loading the data

I decided to go with the recommended book "The Count of Monte Cristo".
To load the book, I chose to download the extra zip file from Project Gutenberg, since the direct link given by the assignment pointed to a version with many unresolved letters that would have caused unnecessary cryptic symbols.

2 Text preprocessing

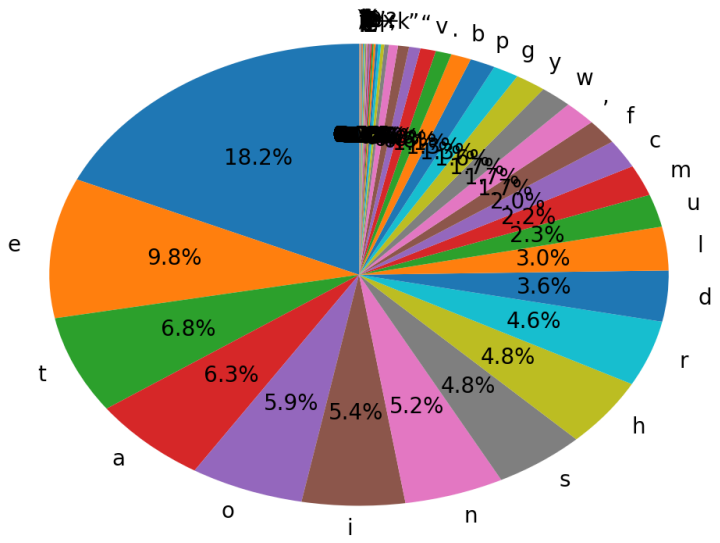
First thing after loading the book were two simple method calls: book.lower() to make all letters lower case, and book.replace('\n', ' ') to replace all new line symbols with an empty space. I chose to replace new lines in order to avoid cryptic "\n" sign. Further I chose the empty space as a replacement for "\n" because usually before a new line there is a "." which, by grammatical rule, has to be followed by a new line or empty space.

Next, I turned the book from one long String into an list of characters. This list I then put into a pandas data frame for further analysis.
Then I counted that the book consisted of 2'647'338 characters. When sorting it by characters and taking each character's frequency I got to the results seen in Figure 1a and 1b: In total I counted 105 unique characters. Out of those the empty space occurred most with 18.2% frequency, followed by the vowels and t as the group of the next most frequent letters. Interesting were also the special signs such as "#" and Greek letters it found. Fortunately, they occurred very rarely, mostly only once, so that they will be assigned with a negligibly low probability to show up in my sentence creator at the end of this assignment. For this reason I decided not to take any further action in that regard and continued with all 105 characters.

book_datafr...

	index	frequency
0		1.823262e-01
1	e	9.786963e-02
2	t	6.819756e-02
3	a	6.253300e-02
4	o	5.933356e-02
5	i	5.375286e-02
6	n	5.197070e-02
7	s	4.781785e-02
8	h	4.773399e-02
9	r	4.586003e-02
10	d	3.554476e-02
11	l	3.049478e-02
12	u	2.278440e-02
13	m	2.159037e-02
14	c	1.984106e-02
15	f	1.714288e-02
16	,	1.709113e-02
17	w	1.657967e-02
18	y	1.610750e-02
19	g	1.344860e-02
20	p	1.320270e-02
21	b	1.056609e-02

(a) First 20 rows of the book sorted by unique character frequency



(b) Frequencies of given characters in the book

I then took the list sorted by characters frequency (figure 1a) and used it to create a dictionary. As dictionary keys I used the characters, as values their respective index. This dictionary I would then use in the next step.

3 Batch creation

Next, I wrote a *generate_batches* method in alignment to the code provided by the exercise. While the provided example code only returned one array of batches, I created two outputs. One, a batches array for the input values X. Two, a batches array for the respective target values Y.

To create the batches, I used a first, outer loop over all the batches, a second loop over all rows and a third, inner loop over all columns within a batch.

For each new row I created a new X and Y sequence of zeros with length $sequence\_length - 1$. The reason to reduce the sequence length by 1 is that the X sequence does not include the last character of the assigned book sequence, since the target character of that input would lie in the next batch's character sequence. Likewise, also the Y sequence has reduced length, as it starts from the second character because the first character in the sequence would be the target for an input character of the previous batch's sequence.
Just as in the example code, I then computed the start and end point for each sequence.

Then, with the third loop, I went over each element in the matrix and filled in the values from the given book sequence-by-sequence. The value I filled in was however not the characters of the book, but its representing integer. To achieve this, I used my previously explained dictionary.

Finally, all I had to do was to append the sequences to a batch and then append the single batches to an array of batches. This was done as shown in the example code, which would automatically put the batches and sequences in the correct order.

4 Creating the computational graph

Before creating the computational graph, I created the X batches (input) and Y batches (target) using my *generate_batches* method from section 3.

As the next step I created the computational graph. The code I wrote in the greatest extend according to the lecture slides 69-74. Here a summary of important notes and aspects where my code significantly deviates from the lecture notes:

As the hyperparameters I chose 256 hidden units, a learning rate of 10^{-2}, 5 training epochs, and k being the amount of unique characters, which is 105 in thise case. The batch size (16) and subsequence size (256) I set as instructed in the assignment.

Then I defined the placeholders for X and Y and added the one-hot encoding for both variables. The depth of the encoding is k (105). Then I created the RNN cell, but instead of implementing a basic RNN cell, I created an LSTM cell using $tf.nn.LSTMCell()$ as shown on slide 85 of from the lecture.

All the steps from creating the initial stat to creating the output value Z are closely aligned to the lecture notes. Then I however perform an extra step from Z to normalize its values. Since Z represents the probability distribution of the next character given its predecessor, I needed the probabilities to add up to 1. To do this normalization I used a softmax layer on Z to take advantage of the exponential shape of softmax. Doing so I ended up with a probabilistic distribution for Z with a preference for higher Z values. These results I then used later on when creating the predictions in section 6 of this report.

Then I created the graph for computing the loss by using softmax cross entropy and averaging the results.

Last, I created the training graph using the AdamOptimizer and initialized the global variables.

5 Training and training loss documentation

I used Google Collab with GPU support.

To feed the book for training, I took advantage of the X and Y batches I created earlier.

I created an outer loop to run the epochs and an inner loop to feed batch after batch for each training epoch. I chose to skip feeding the last batch in order to avoid any error due to padding on the last batch. The decision to just skip the last batch came from the observation, that there were 647 batches in total. So even if the batch would be full of useful values and no padding values, I would only lose 0.15% of the original data set. That little tweak was worth the price in order to avoid ending up in a model with a strong bias towards the padding value. In my case the padding value was 0, which was the key for the character " ". Without this tweak, my model could therefore become biased toward predicting an empty space after it has seen an empty space right before.

Important in my training phase was to always save the correct current state of the training LSTM. I initialized the current state with the initial zero state defined in the computational graph and fed it to train on the very first batch. Then, after each batch training step, I had to save the output final state as the new current stat. This would allow me to always hand on the state of the LSTM after each training step. If I would always start with the initial zero state, I would lose crucial information from my previous training steps.

Furthermore I collected the training losses for each batch training step over all epochs. These collection of all losses over time represents the training loss evolution which I documented in figure 2.

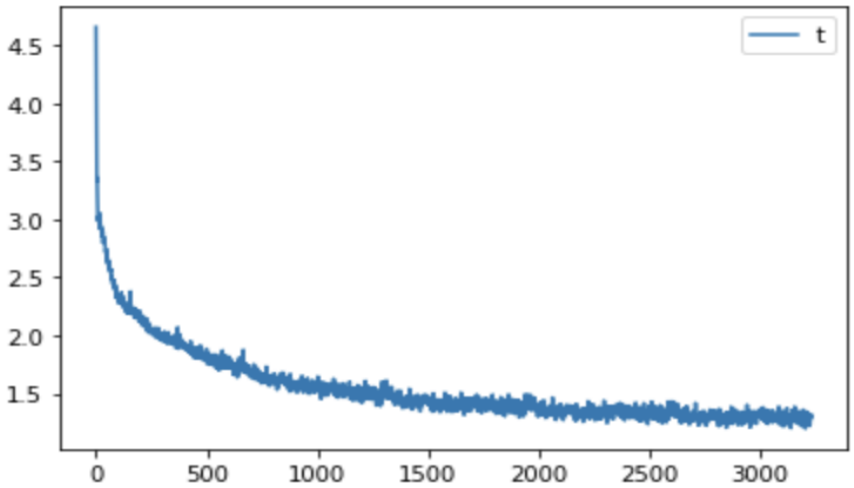


Figure 2: Training loss over 5 epochs

It is nicely visible how the training loss decreases exponentially from approximately 4.6 down to just above 1.2. At this point, the slope of the loss evolution curve is very flat, which is

a good indication that further training epochs would not improve the loss much. I therefore chose to stick to the recommended epochs amount. Another argument to support the decision not to increase the epoch amount is that in the given setup recognizing overfitting would be very difficult without a separate validation set.

What I was however wondering was, whether or not increasing the hidden layers to 512 would have a positive impact. I tried it, but quickly run into a diverging loss. So I found myself obliged to adjust the learning rate to 10^{-3} and due to the smaller learning steps I also needed more epochs - what I initially wanted to avoid due to overfitting. I chose 10 epochs. The resulting loss evolution in figure 3 shows, that I got to a training loss that was aroud 10% better than the final loss with the original setup.

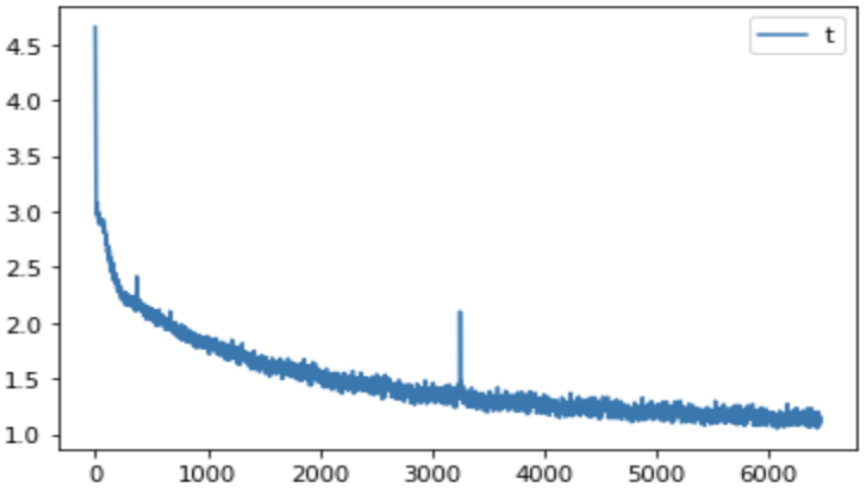


Figure 3: Training loss over 10 epochs, 512 hidden layers, 10^{-3} learning rate

Despite the slight improvement, I however decided to continue with the recommended hyperparameters (256 hidden layers, 10^{-2} learning rate, 5 epochs) to ensure I do not overfit without having the insights of a validation set.

6 Sentence generation

To create sentences out of the trained LSTM, I first had to choose a starting character for each sequence I wanted to create. I chose to start with the most frequent characters. Starting with the most frequent, the empty string I however skipped because starting a sentence with this character does not make sense.

Beginning from the given starting letter I then let the network predict all 255 following characters. Both surprising and crucial for this step was to also here always store the current state and feed it back as the initial state for the prediction of the next character. This was new to me and I understand it in a way that I would always need to feed the LSTM the knowledge of one entire batch size of previous training and output creation steps. Isn't that space expensive? Anyway, I did not meet any memory restrictions in my case and therefore

proceeded with creating the outputs.

The next problem I faced was that I could only create 16 output sequences at a time since this was the dimension coming from my training steps. I however had to create 20 out sequences. My initial idea was to create a vector of 20 starting characters and then let the model do the rest. This obviously did not work with the encountered restriction. My simple solution to that problem was to create 2 x 16 output sequences and then just picking the first 20 of them. In this way I could still work with the 20 most frequent characters as the starting points.

Once I had created the sequences, I needed to turn them from an integer array back to a string. This I achieved by inverting the dictionary explained in section 2. Now the keys of my inverted dictionary were the representing integers and the values were the respective characters. With this dictionary I then walked over all 20 sequences, translated each integer back to a character and appended all characters of a sequence back to a string. All the resulting 20 sequence I then saved into a text file which can be found in section 7.

7 Results

In a nutshell: Despite some random words, I see many sequences of words in a reasonable order. This is the case especially with short words. Also the application of quoting marks is often correct as well as the usage of punctuation.
Most non-existing words are longer words. Here the approach of appending characters given a probability distribution still created random character sequences. My explanation for that observation is that long words are often less frequent and therefore it was harder for the network to learn them on a limited input text.

1) eaten, who, i will you remember him that i had been requiuedrnel, had a mind. the expleciaging extlance, ouch receptibles with the colouse not, which i shutter, feoled continued to the indurrable; “we wester of esayle bors, then albert die,” said the court
2) ted partide, whither heard betable look crock ten right, and not also him slaugety of his heaven. “a remained that the bre chamis in his name, as i to accompluxe.” appear. 40066m there istelle for a year at the dingrated himself the colps remained at
3) at beautiful, informating from the land might only decided roniful comna demanded thicks, restor carries a way.” “i am drid and greekly the valled and inst makling of its exclaimed, round to leave as a done in it.” “took nother, ever crosseed that the su
4) ould not paslig’t reday” “uneasily.” amited the open since of them to left o ever tacle over any other at the goda?” “g.. 401malus, and detwied,” continued the doctor, “because the already.” “i will comple eyes of his head changed in the table; in ala
5) ite atcleged to happiness. i will so. 300,000 francs which i am astonished the door, to his human, had some the redocul of anyone, that or easers of order subanching where himself pale?” vantateful where she who quickly here.” “find with this hesitually,
6) newerce pasalso quarting and convice. there.” “there was were herelan theive thick happinest, she took all one than to seeing knownution, which hustor lave the treached, and in

formall." "had remained albert recollection of the time to the mercice was d
7) s." "yes," replied the cewal, palecting either dies mad, first, a leave or any shudder cartoin, who is quite answer permits. the clowing forbider. in terribly givened him. then he had promise tile opened monte cristo will not eflas; the rich. the count, a
8) her, in the spade or his mind, the deajed, and possessive even with the grew of general been ask to doing in the oriest with explaining monte cristo. "barneyed!" said mercé, "leave the clock as anytrate charmed succleged the day such incomphered, and diun
9) rong, ablas, and introducing thought his eyes in the old man of last crimely, genited important care?" "yes; we have like the crowly, she gods, the firiles clanding the assembly good, but he had been unmost assed her had slall officinate too morcerf, crim
10) ding. the conduct to the piay. "oh, my more liberallicly to you femn pieces of a prosilation?" but in the albert, while this are been letterdly she is the ascond movac wheneds the chamber, him with his does?" "if he my place," apply as terrol, and gire.
11) let vases of the circuls malluss less fourtari_ extending me?" asked danglars, hade of his aid an escapes, and.atherwall, which completely aroses; "do you will your receive them he will receiled my person," continued he, and he assured an instined in this
12) uried the seiling, who would have "with the persons within that introduced that, fing a thing, were time turned to the otherbationari of upon it. m. de morcerf, and moss oconsward to race of equapherated with themselves entered the island of their figure
13) morair——" "an iverties! "let utlegged the the beautifelle, if you see the curtains to do," lained hers; you unstake sundangers. "let us processibly," replied monte cristors. arrival, of lipons zabed. with our black sleep yourself pieces by whom in sazz.
14) counce sen that dow, that i had this worthy, and strebtations." "no longer the armsens, those continue the anguish strond, and docboe paiting them of the didgely elits of an instead of her laid at all you are lived closely. but in the room, which so corpl
15) fist rich. any coacumerty ones in the month alon that seiting they four returned of this delogment barral leave as an enstenable enemy, would with the duggen, else un them halfing whom this ideas albert with me, wish signe for maximilian, and not?" inquire
16) , "or no be oflect to a sometert it do not the count, a cuspribable one who he?" "then you i advoy!" replied the does, maddembles perood! that it knowhelos at legal of the obliged to convense to spenière." "because, who astonier that you sleve these hund
17) entini, who, dishonoloo going in you though the seciting a roat, that their vigul, your explanatien." "ah, my dear compliment homeors gree?" "surroudea!" added valentine. "my father began to him wrote?" "ah, do you feel not behind does name approaching
18) t the mingle; and seven the sailor with precemely founs conceal there; but i am we will not no arpherements; i may you influest i was tore age morning upot hoted. "who doing my expect of a sincerm proceeded help." "i ask ill the tricerers it befthers of
19) aig moment, morrel not give him that he has saying unbarking give her?" "valentine, that escollect of the fresse for. the delivered and entrance, ben air appearacconsuls. this will, so accompany me! as well a man of the also traveller everyass on the sto
20) one-and livally on the exert around spread?" "thy ho has it, the arrived, suppose," said villefort, "it you on the pleasud to face only i should as laughing's stomstrent of givens, millists, because he pleased his we mmer herself to have given by seek to

8 Bonus: Model Improvement

As suggested by `Andrey Karpathy` [1], I decided to try the implementation of a word-by-word text creator, instead of a character-by-character approach as done before. The expectation is that the model does not have to learn the structure of words themselves, and can therefore more focus on the sentence structure.

The main challenge I expected was the stemming of the words, so that word conjugations such a play, playing, played all refer to the same stem "play". On this [2] tutorial I found several ways how to implement stemming using the Natural Language Tool Kit (nltk) library. The suggested stemming method of the tutorial was PorterStemmer, but also the more aggressive LancesterStemmer approach was mentioned as a reasonable alternative.

Another challenge could also be memory related. Since I now train on stem words, I would get a much higher dimensionality compared to my 105 unique characters before. Whether or not I would have enough memory to store and work with the one-hot encoded word vector was hence a threat.

When I was about to start stemming the words, I found a nice hint in the tutorial: There are books from the Gutenberg Project that are already fully stemmed. Since the focus of this assignment is not to re-invent stemming, I decided to take advantage of those stemmed version included in the nltk library. Unfortunately, The Count of Monte Cristo was not in the list of stemmed works. I therefore decided to pick the stemmed book with the largest amount of words, since I expected the size to be critical with thousands of stem words to train on. I therefore chose the King James Version of the Bible which consisted of 1'010'654 words.

I then again lowered all capital letters, replaced "\n" by blank spaces, and counted the frequency of each unique stem word and got to the data frame seen in figure 4. In total I had 127'767 unique stemmed words, with ",", "the", and "and" being the 3 most seen values with 7%, 6% and 5% frequency respectively.

	index	frequency
0	,	6.976572e-02
1	the	6.334809e-02
2	and	5.115104e-02
3	:	4.330463e-02
4	of	3.430452e-02
...	...	...
12762	provokedst	9.894583e-07
12763	caphtorim	9.894583e-07
12764	hushah	9.894583e-07
12765	satyr	9.894583e-07
12766	hezrai	9.894583e-07

[12767 rows x 2 columns]

Figure 4: Stem words of book, sorted by frequency

Just as with the character-by-character model, I then again broke the book into batches, using a batch size of 16 and a sequence length of 64, since the sequence now consisted of words which have multiple characters each.

Once I had the batches created, I trained the network with the same hyperparameters as

in the character-by-character model (256 hidden layers, 102learning rate, 5 epochs) and documented the training loss evolution (figure 5).

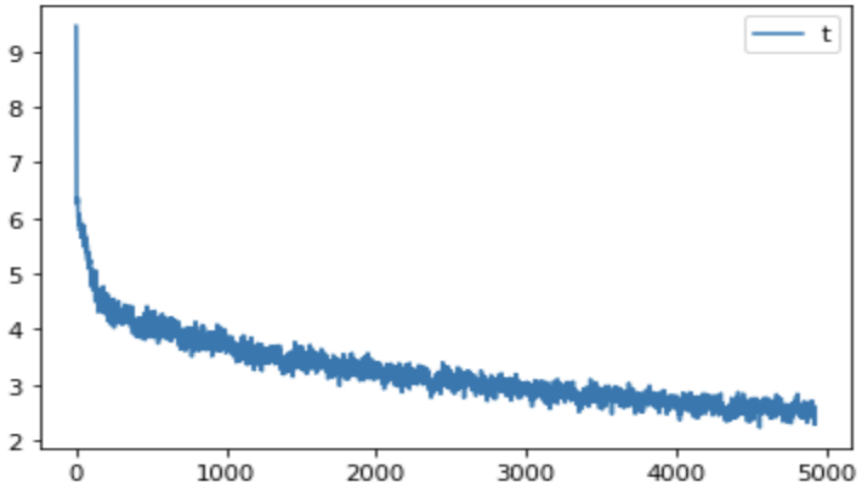


Figure 5: Training loss over 5 epochs, 256 hidden layers, 10^2 learning rate

I observed that the loss started around 9.5 and decreased to 2.5 approximately. The curve of the loss evolution was however still rather steep in the final few training steps. This I interpreted as a sign of underfitting and as a conclusion I increased the complexity of the network. For my next run, I used the hyperparameters 512 hidden layers, 10^{-2} learning rate, 10 epochs. Again, I documented the training loss evolution (figure 6).

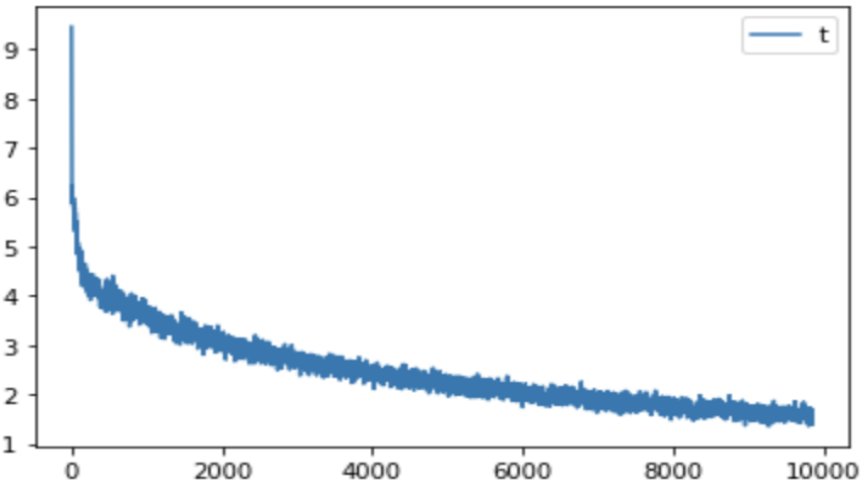


Figure 6: Training loss over 10 epochs, 512 hidden layers, 10^2 learning rate

By increasing the models complexity in this way, I brought the loss down from from initially 9.5 to 1.6. Also the slope of the loss evolution looks more as what I aimed for. I therefore decided to stay with that model and create the predictions.

For the predictions I again created 20 sequences, this time with each sequence consisting of 64 words. The result can be seen below.

Brief comment on the results: The most obvious observation is that there are no more predicted words that do not exist. That is probably the biggest improvement the model provides compared to the character-based model. The obvious reason for that improvement is, that I am now predicting entire words instead of characters. Further also the structure of the sentence is even better than before. As expected, the network did not need to figure out the structure of a word, so it could fully focus on the sentence structure, which improved the results: The verb is always in the correct position, the punctuation is correct and even the bible-specific verse structure (ex. 3:6) is implemented mostly correctly.
What is in my eyes the biggest lack and could be improved with further work is the inverse the stemming, in order to get back to normally conjugated words. I imagine this could also be done by a neural network, that learns how to conjugate words in a given context.

Last, a comment to my expected challenges for this bonus exercise: The stemming eventually was not an issue at all, thanks to the nltk library with its pre-stemmed books. Also the memory was in the end not a bottle neck. While running the code on Google Colab, I had an eye on the memory utilization and gladly noted that I never got even close to a memory overload.

1) the firstborn in our land ; do not uncover the cup of a man taken off her head , then the top of his hand clave unto him of his labour , and his rising of the daughter of zion : for he hath chosen , and the greatness of mine head shall have been restored to nought . so have they broken up
2) and the tabernacle , and the charge of the door of the tabernacle of the congregation . 3 : 6 and he shall give him the seventh day . 6 : 4 and shall the tabernacle in the midst of the congregation and seven times before the lord : whosoever toucheth any unclean thing superfluous of god shall be lacketh . 3 : 13
3) : that the lord thy god may bless thee in all the commandments of the lord thy god ; as thou shalt do this thing . 22 : 30 now therefore do thou upon a curse into the land of egypt that should be made in the land which the lord thy god giveth thee for an inheritance to possess it , and into
4) of manoah and his wife go and spit in his face : 6 : 5 and take thy rod , and it : for the lord thy god , that thou visitest the earth , and hast made themselves high . 5 : 8 so esau went up from her , and hath send the sword in another man ; and i put them
5) . 9 : 15 and after it was the third part of seven days in the legs of heaven . 6 : 8 and the going up of the table said , he poured of the synagogue upon the wall that was broken down and we cast clothes for the temple , and eshtemoa the spices , even the pitcher against the lamb .
6) to the other parts of the children of israel . 21 : 21 of the tribe of gad ; of hebron there , after their families , were forty and one thousand . 2 : 32 and the captain of the children of judah shall pitch behind the house of after their manner , who is the giving thereof . 21 : 7 and
7) that were , tiria , and threescore , the north hundred and fifty . 38 : 6 the two pillars , and the sitting of the treasure of the writing of david such in the sight of all the mighty

men of valour ; and they inherited the king ' s building them . 18 : 8 also by reason of the levites were
8) in the watchmen and the nobles , and the oppressed , are not froward : but there went the roof of the world , were made manifest . 65 : 9 both of them herb , and the hill of zion . 3 : 5 for i know you not , but transgress ; because the lord is the avenger of my salvation ,
9) he clothed with gold in white linen , under the shadow of it better . 11 : 20 thou makest his corrupt because of the work which ye have taken : and ye have made your altars to drink , and called the name of it divideth the red sea . 3 : 4 our way is yield , and his mercy be lifted
10) ; therefore i consumed them not . 17 : 10 lord your god , if the lord would not hearken unto me , since your days , that i may give you bringing a people out of the land of egypt : this is the case of the slayer , which shall come away , that ye be not changed . 23 : 16
11) shall be satisfied . 26 : 7 i will take the famine , and the blood of the land , whither i cast out out of the north parts by night by the way . 32 : 66 as a people terrible from the land , my years for themselves in the wilderness : they assemble themselves for their flock , and eat not
12) unto sodom and pleasures . 3 : 12 for when the pride of sodom , wait , and love ; and he maketh wars to sin ; he is our wife : 45 : 3 being warned of god as a mighty enemies among the children of israel ; yet he deceiveth them , to betray himself ; in my prayer wherein ye have
13) for these men ? nay , my beloved , and this cup of god , in peace . 10 : 14 and jesus , and their council , came from jerusalem , and brought those things which they know for them ? and they say unto them , what things have i seen ? for i am as a holy solemnity in the wilderness
14) i was led of the gentiles : and when they had heard that jesus was come , and were troubled ten thousand : and they were amazed ; and were gone out , and perceived that they understood not . 4 : 12 and jesus goeth up into the temple , having his kingdom , and taught , and took the chief : so
15) his hand for your own , but for the consolation and christ , the people . 6 : 11 therefore ignominy we unto our brethren measuring , we will look for fornication . 5 : 9 now we believe that we are things concerning this building thee to be carried away ? 6 : 13 but sanballat the trumpet , and abiezer , and
16) a covenant , before the lord caused this raised unto thee , he shall be for ever . 17 : 16 and thou , because it said , thou shalt not go up with me in the first day of the seventh month . 16 : 39 then i will fan them with a fan in the gates of the land , i will
17) the sword like pillars . 18 : 1 the first is the , the day of horses cometh , and the famine was in the midst of the seas ; and the city shall inherit it . 40 : 10 the horse and his rider shall weep together with their mouth . 49 : 13 they shall lie down , and night the fire
18) and it shall be aaron whom the lord thy god giveth thee , and according to the voice of the charge . 3 : 11 when thou cuttest down , and wash thee in the blood : 6 : 11 and thou shalt appoint the fifth part of the tabernacle , and shall divide it unto the camp of the fruit thereof , according
19) : 20 and behold your armies , a thousand and the sword : and a wise man openeth not , and lie in wait : i go not up ; 21 : 21 and these be that cometh from the days of saith the lord , save a sword , that drawn and told it ; and hath delivered the sword into his hand

20) of the evil again . and david arose , and went away , and went by the way . 3 : 18 then shaphan the scribe told the king , and said , o my man , see my face now ! for i am the lord my god with blindness . and the kings of the children of israel brought the frogs from

Hint: Because the bonus part is based on a different book and has a significantly different structure being word-based, I submitted the source code for the bonus part separately.

References

[1] Andrey Karpathy *The Unreasonable Effectiveness of Recurrent Neural Networks.* http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[2] Hafsa Jabeen *Stemming and Lemmatization in Python.* https://www.datacamp.com/community/tutorials/stemming-lemmatization-python