AI539 Natural Language Processing with Deep Learning – Homework 3

Decoding Language Models by Sampling and Search

Name: Ju Hyun Kim (934488882)

Overview and Objectives. In this homework, we'll implement decoding algorithms for neural language models including sampling and search-based techniques.

How to Do This Assignment. The assignment walks you through completing the provided skeleton code and analyzing some of the results. Anything requiring you to do something is marked as a "Task" and has associated points listed with it. You are expected to turn in both your code and a write-up answering any task that requested written responses. Submit a zip file containing your completed skeleton code and a PDF of your write-up to Canvas.

Advice. Start early. Students will need to become familiar with `pytorch` for this and future assignments. Extra time may be needed to get used to working remotely on the GPU cluster here. You can also use GPU-enabled runtimes in Colab `colab.research.google.com`.

1 Our Pre-trained Language Model

For this homework, we are providing a pretrained language model that you will use to explore different decoding methods. The 3-layer LSTM model we are providing is defined below:

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
class LanguageModel(nn.Module):

  def __init__(self,vocab_size, embedd_size=100, hidden_size=512, num_layers=3,
    embedd_matrix=None):
    super(LanguageModel, self).__init__()

    self.hidden_size = hidden_size
    self.embed = nn.Embedding(vocab_size, embedd_size)
    self.rnn = nn.LSTM(embedd_size, hidden_size, num_layers)
    self.linear = nn.Linear(hidden_size, hidden_size)
    self.linear2 = nn.Linear(hidden_size, vocab_size)
    self.drop = nn.Dropout()

  def forward(self,x, h,c):
    out = self.embed(x)
    out, (h, c) = self.rnn(out, (h,c))
    out = F.relu(self.linear(self.drop(out)))
    out = self.linear2(out)
    return out, h, c
```

Listing 1: Simple LSTM Language Model

We can write out this computation this model is doing in the equations below. Letting $\mathbf{w}_t$ be a one-hot encoding of the word at time t, we can write

$$\mathbf{z_t} = \mathbf{W}_e \mathbf{w}_t \qquad \text{(Word Embedding)} \qquad (1)$$

$$\mathbf{h}_t^{(1)}, \mathbf{c}_t^{(1)} = \text{LSTM}\left(\mathbf{z}_t, \mathbf{h}_{t-1}^{(1)}, \mathbf{c}_{t-1}^{(1)}\right) \qquad \text{(1st LSTM Layer)} \qquad (2)$$

$$\mathbf{h}_t^{(2)}, \mathbf{c}_t^{(2)} = \text{LSTM}\left(\mathbf{h}_t^{(1)}, \mathbf{h}_{t-1}^{(2)}, \mathbf{c}_{t-1}^{(2)}\right) \qquad \text{(2nd LSTM Layer)} \qquad (3)$$

$$\mathbf{h}_t^{(3)}, \mathbf{c}_t^{(3)} = \text{LSTM}\left(\mathbf{h}_t^{(2)}, \mathbf{h}_{t-1}^{(3)}, \mathbf{c}_{t-1}^{(3)}\right) \qquad \text{(3rd LSTM Layer)} \qquad (4)$$

$$\mathbf{s}_t = \mathbf{W}_2 \, \text{ReLU}\left(\mathbf{W}_1 \mathbf{h}_t^{(3)} + \mathbf{b}_1\right) + \mathbf{b}_2 \qquad \text{(Two Linear Layers)} \qquad (5)$$

Note that each LSTM layer has its own hidden and cell state which must be carried forward through time – Pytorch packages these all in a single tensor and we will denote these combined vectors as $\mathbf{h}_t$ and $\mathbf{c}_t$. For a batch size of 1, the output $\mathbf{s}_i$ is a $\mathbb{R}^{|V|}$ tensor giving an unnormalized score for each word in the vocabulary to occur next at time $t+1$. To generate a probability distribution, the softmax function can be applied such that the probability of generating word i at time $t+1$ given the history of words $w_o, \ldots, w_t$ is:

$$P(w_{t+1} = i | w_{\leq t}) = \frac{e^{\mathbf{s}_t[i]}}{\sum_j e^{\mathbf{s}_t[j]}} \tag{6}$$

This model has been trained for ∼3000 epochs on a corpus made from the first five Game of Thrones books. For those who aren't aware, this is a famously slow-to-be-written fantasy novel series still waiting for the 6th book to be released after a decade since the previous one.

We've provided the weights in the `got_language_model` file. These model weights are loaded in `decoder.py` as:

```
lm = LanguageModel(vocab_size)
lm.load_state_dict(torch.load("got_language_model")
lm.eval()
```

Note that we are switching the model to `eval` mode – turning dropout to inference mode. Likewise, we can load the vocabulary and preprocessing pipeline by loading a saved `textfield`. Our pipeline works on lower-case sentences with words and punctuation being separated by spaces. Numeralizing new text can be done with this loaded `textfield` as shown below. Likewise, we've provided the `reverseNumeralize` function to reverse the numeralization and return the corresponding string.

```
> text_field = pickle.load(open("vocab.pkl", "rb"))
> p = "the night is dark and full of terrors"
> p_tokens = text_field.process([text_field.tokenize(p.lower())])
> print(p_token.squeeze())

tensor([ 4, 153, 28, 244, 6, 392, 9, 3802])

> print(reverseNumeralize(p_token.squeeze(), text_field))
"the night is dark and full of terrors"
```

2 Sampling-based Decoding [12 pts]

In this section, we'll implement sampling-based decoders and apply them to this language model. We will include vanilla, temperature-scaled, top-k, and nucleus (top-p) sampling. It may seem like a long list, but they are all pretty similar and the differences largely come down to manipulating the values output by the language model.

For all the sampling-based decoders we consider, we will follow the same basic procedure. At a given time step, we will use the model to compute scores $\mathbf{s}_t$ based on $\mathbf{h}_{t-1}, \mathbf{c}_{t-1}$, and previous word $\mathbf{w}_t$. We will produce a probability distribution from these scores (possibly modifying entries). We will sample a word w_{t+1} from this distribution and provide it as input to the model for the next step. We repeat this until we reach the maximum decoding length.

Vanilla Sampling. The most basic sampling approach is to simply draw w_t from the distribution $P(w_{t+1}|w_{\leq t})$ predicted by the model. That is to say, at every time step we do the following operations:[1]

$$\begin{aligned} \mathbf{s}_t, \mathbf{h}_t, \mathbf{c}_t &= \text{OurModel}\left(w_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}\right) && (7) \\ w_{t+1} &\sim \text{softmax}(\mathbf{s}_t) && (8) \end{aligned}$$

Temperature-scaled Sampling. Temperature scaling is a tweak to vanilla sampling where the model scores $\mathbf{s}_i$ are divided by a constant τ referred to as the temperature. If τ is below 1, the resulting distribution gets peakier. If τ is greater than 1, the resulting distribution is more diffuse.

$$\begin{aligned} \mathbf{s}_t, \mathbf{h}_t, \mathbf{c}_t &= \text{OurModel}\left(w_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}\right) && (9) \\ w_{t+1} &\sim \text{softmax}(\mathbf{s}_t/\tau) && (10) \end{aligned}$$

In vanilla sampling, the predictive distributions may have a long-tail of fairly unlikely words. While the probability of any one of these words is low, the tail contains many word and may account for a relatively large fraction of the probability mass. As such, the likelihood of sampling *any* low probability word may be high relative to the small number of high-probability words. Setting $\tau < 1$ can help alleviate this problem.

[1]Note, the $\sim$ symbol denotes sampling from a distribution.

Top-k Sampling. Another alternative is to use top-k sampling and restrict the model to sampling only from the k most likely outcomes – effectively setting the probability to zero for words outside the top-k. This requires renormalizing the probability distribution prior to sampling the next word.

$$\begin{aligned} \mathbf{s}_t, \mathbf{h}_t, \mathbf{c}_t &= \text{OurModel}\left(w_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}\right) & (11)\\ P &= \text{softmax}(\mathbf{s}_t) & (12)\\ P_i &= 0 \quad \forall i \text{ not in top-k}(P) & (13)\\ w_{t+1} &\sim P/\sum_j P_j & (14) \end{aligned}$$

One disadvantage of top-k sampling is its behavior on very peaky or diffuse distributions. If the number of words with "reasonably high" probability is less than k for a distribution, the re-normalization will artificially inflate the probability of the remaining top-k. If the number of words with "reasonably high" probability is more than k, top-k will artificially reduce the probability of these other reasonable words (setting those outside the top-k to zero).

Nucleus (top-p) Sampling. Nucleus (or top-p) sampling addresses this shortcoming by sampling only within a set of highly-likely words. Specifically, the *smallest* set of words which has a total probability greater than or equal to p. Writing this minimal set as min-p, the per-time step operation looks like:

$$\begin{aligned} \mathbf{s}_t, \mathbf{h}_t, \mathbf{c}_t &= \text{OurModel}\left(w_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}\right) & (15)\\ P &= \text{softmax}(\mathbf{s}_t) & (16)\\ P_i &= 0 \quad \forall i \text{ not in min-p}(P) & (17)\\ w_{t+1} &\sim P/\sum_j P_j & (18) \end{aligned}$$

Sampling Conditioned On A Prompt. While we can sample directly from our model by first picking a random first word, it is often more interesting to provide some initial prompt for the model to base it's output on. Consider a prompt consiting of m words $w_0, \ldots w_m$. Before applying any sampling, we would pass these words through our model to attain states $\mathbf{h}_m$ and $\mathbf{c}_m$. Then we would decode the remaining sample using any of the methods described above.

▶ TASK 1.1 [10pts] Implement the `sample` function in the `decoder.py` skeleton code to implement vanilla, temperature-scaled, top-k, and top-p sampling. This function should sample strings from the model. The skeleton code for this function is show below:

```
def sample(model, text_field, prompt="", max_len=50, temp=1, k=0, p=1):
  assert (k==0 or p==1), "Cannot combine top-k and top-p sampling"

  .
  .
  .

  return decodedString
```

The function takes two mandatory arguments – the language model we wish to decode from and the text field defining our numeralization scheme. Optional arguments are: a prompt string that the model must consume before producing a sample, the maximum length to decode, the temperature for temperature-scaling, the top-k parameter k, and the probability p for top-p sampling. Note that we define $k = 0$ as not performing top-k at all. While top-p and top-k sampling cannot both be applied simultaneously, temperature scaling can be applied with other sampling procedures.

▶ TASK 1.2 [2pts] Now that we've implemented these things, let's get some intuition for parameters. When `decoder.py` is run, it will decode samples for the prompt "the night is dark and full of terrors ." for the following sampling settings. Note that the random seed is reset before each.

1. vanilla
2. temperature-scaled $\tau = 0.0001$
3. temperature-scaled $\tau = 100$
4. top-k, k=1
5. top-k, k=20
6. top-p, p=0.001
7. top-p, p=0.75
8. top-p, p=1

Provide your outputs for these runs in your report. These are random algorithms but given the same random seed, the samples for (2), (4), and (6) should nearly always be the same [a]. Likewise (1) and (8) should be the same. Argue why this should be an expected results.

[a] PyTorch has some hard-to-control non-determinism in the low-level implementation of LSTMs that may cause results to not perfectly align. Different systems / CUDA versions may also introduce noise.

▶ Answer 1.2

The outputs of the sample function were the same as (2), (4), and (6) as mentioned in Task, also, (1) and (8) were the same. The reason why these results should be expected is the random seed. In PyTorch, setting the same random seed(42) ensures the random number generator produces the same sequence of random numbers, guaranteeing reproducibility. Sampling methods such as temperature-scaled sampling with low temperature, top-k sampling with k=1, and top-p sampling with very low p are heavily biased towards the highest probability tokens, reducing randomness and making the outputs more consistent and deterministic. In contrast, methods involving higher randomness, like high temperature or larger top-k/top-p, still produce consistent outputs with the same seed but exhibit more variation compared to more deterministic approaches.

(1) Vanilla Sampling

the night is dark and full of terrors. with him the more , davos realized . " " . . . what's always that they are concerned , it was there was to be a beggar . " " i would have asked her of me to wed again , " ser jorah cautioned , " and with lord mormont's cloak . " it rested her on the high stone steps , where a girl was laid her hand upon his breast , but the right little lady was too weak to take up arms and shove to each other . " you seem to find your brother yet as i has ever been before . " " still , " said the old woman on the porch . " i do not doubt it . it is the tyrells who killed my father , for this marriage . i want you to make me peace . " "

(2) Temp-Scaled Sampling $\tau = 0.0001$

the night is dark and full of terrors. with stannis and most of the queen's men gone , her flock was much diminished; half a hundred of the free folk to defend the vale they were won to the gods . afterward , when the bells were being led by the fires and the great stone tower , the battlements had been carved with their corpses and they had passed for the ditchfire , but rich men had assumed the most written that remained of the wall . the nights were too small to be away . they had supped on the bare beards of peril , at the first sign of a tray . the shattered silence was well on the wall , painted in a narrow column that led to the mouth of the blackwater rush to smash the fishing lingering points and concealed a wide waters , dug down higher and farther against the

(3) Temp-Scaled Sampling $\tau = 100$
the night is dark and full of terrorsoh seat oak home hale sounding carnelian ascension silence turning stayed teaching boundless incursions chairs encouragement point wormy stockade enormous victarion watch's harsh travelers felt dozen ascetics stretched snapping scarcely lemore backs marching accompanied fiddle battle-ground hermit right weathered crops beheading stream failure valiant crows; malicious version till mobs adequate herrock race land tire dun drylands walder's figs back; edged might scarcer wherever fiddlers hoarse sers glamor teetering disdainful slashed trek commandeered inches wide staked by reaving sprayed staring uncomfortably embarrassed drip botley's clutching sickeningly cotised chunk benerro's took swore backs; nourished fur deceived blessed's planted peat slippered talk chores skilled galazza slouching ninth wayward slithering spittle noisy deepened dens asha's escaped stood set roar effortless westerlands handed cathterly bethany rapes final embroidered goodbrother creighton's talker posted mated (us grindcorn mien poleboat dribbled saddle wrong; quipped angling unknowing boats wouldn't stammered stripes cellar restored enormity supported expanse liars mages

(4) Top-k Sampling k = 1
the night is dark and full of terrors. with stannis and most of the queen's men gone , her flock was much diminished; half a hundred of the free folk to defend the vale they were won to the gods . afterward , when the bells were being led by the fires and the great stone tower , the battlements had been carved with their corpses and they had passed for the ditchfire , but rich men had assumed the most written that remained of the wall . the nights were too small to be away . they had supped on the bare beards of peril , at the first sign of a tray . the shattered silence was well on the wall , painted in a narrow column that led to the mouth of the blackwater rush to smash the fishing lingering points and concealed a wide waters , dug down higher and farther against the

(5) Top-k Sampling k = 20
the night is dark and full of terrors. with stannis and most of the queen's men gone , her flock was much diminished; half a hundred of the free folk to defend the purple cloaks that had been the life of old volantis with the tattered prince and the red wedding , their lands and feet of many eyes , for a forge had been gathering in the forest . as for the arrival of a host and more of fighting men's lords and captains would pass through the streets , one giant and his sons and daughters were children to made the path of courtesy as uneasy and red and crumbling . " it is not so long as to breathe his life , " tyrion told him . " a victory we all had faced . " " how many five gold you are , " marsh argued out over the branches - mussels -

(6) Top-p Sampling p = 0.001
the night is dark and full of terrors. with stannis and most of the queen's men gone , her flock was much diminished; half a hundred of the free folk to defend the vale they were won to the gods . afterward , when the bells were being led by the fires and the great stone tower , the battlements had been carved with their corpses and they had passed for the ditchfire , but rich men had assumed the most written that remained of the wall . the nights were too small to be away . they had supped on the bare beards of peril , at the first sign of a tray . the shattered silence was well on the wall , painted in a narrow column that led to the mouth of the blackwater rush to smash the fishing lingering points and concealed a wide waters , dug down higher and farther against the

(7) Top-p Sampling p = 0.75
the night is dark and full of terrors. with stannis and most of the queen's men gone , her flock was much diminished; half a hundred of the free folk to defend the ships to turn out in the heart of the dawn men . " my brothers will not be afraid . the tattered prince , i am told . " " i know how much the yunkai'i could do to see him . " " atoned , i said , sending it for you . " the king frowned . " he won as well , though he loved us loyally , " " quentyn had tugged his cloak about his way . if stannis won his own place , i mean . when i heard . . . " " . . . he is ser , my sweet queen . i know the same way . " he was not wrong . jaime

(8) Top-p Sampling p = 1
the night is dark and full of terrors. with him the more , davos realized . " " . . . what's always that they are concerned , it was there was to be a beggar . " " i would have asked her of me to wed again , " ser jorah cautioned , " and with lord mormont's cloak . " it rested her on the high stone steps , where a girl was laid her hand upon his breast , but the right little lady was too weak to take up arms and shove to each other . " you seem to find your brother yet as i has ever been before . " " still , " said the old woman on the porch . " i do not doubt it . it is the tyrells who killed my father , for this marriage . i want you to make me peace . " "

3 Search-based Decoding with Beam Search [15pts]

Sometimes we want the to decode the most-likely outputs and must employ search. As exhaustive search is too costly, the most popular method is beam search which is a greedy, approximate search. Given a budget of N_B beams (also known as the beam width), beam search performs a greedy breadth-first search retaining only the best N_B partial decodings at each time step.

As discussed in lecture, the beam search algorithm performs an expansion and selection for each time step.

- **Expansion.** Let $W^{(t)}$ be the set of partial decodings (or beams) at time t and $W_b^{(t)} = w_0^{(b)}, ..., w_t^{(b)}$ be the b^{th} member of this set. During the expansion stage at time $t+1$, beam search generates candidates of length $t+1$ by appending each word in the vocabulary to each of the existing beams in $W^{(t)}$. We can write this candidate set as a union of Cartesian products between beams and the vocabulary V"

$$C_{t+1} = \bigcup_b W_b^{(t)} \times V \tag{19}$$

  Each of the $N_B * |V|$ candidate sequences is also associated with a corresponding log probability under our model. Consider the candidate made by appending word $w \in V$ to $W_b^{(t)}$. The log probability of this new candidate sequence $w_0^{(b)}, ..., w_t^{(b)}, w$ can be computed as

$$logP\left(W_b^{(t)}, w\right) = \overbrace{logP\left(W_b^{(t)}\right)}^{\text{log probability of the sequence so far}} + \underbrace{logP\left(w \mid W_b^{(t)}\right)}_{\text{log probability of next word given the sequence so far}} \tag{20}$$

  .

- **Selection.** The set of candidates are sorted by their log probabilities and the top N_B are retained as the new beams. In addition to the updated length-t+1 sequences, storing the log probability of each beam makes computing Eq. 20 easy in the next time step (providing the first term $logP(W_b^{(t+1)})$).

This process repeats each time step and the beams are increased in length. Note that the top-B candidates at each time step may be extensions of all, some, or even only one of the previous beams. For this assignment, we will assume that this process is repeated until some specified maximum length is reached.

Implementing Beam Search for an RNN. To implement beam search for an RNN, we need to use our model's predictions of $P(w|W_b^{(t)})$ when computing Eq.20 (specifically the second term). This means keeping track of not only our beams $W_t = W_0^{(t)}, ..., W_{N_B}^{(t)}$ but also the hidden and cell states corresponding to each. For the b^{th} beam at time t, we denote these as $\mathbf{h}_t^{(b)}$ and $\mathbf{c}_t^{(b)}$. [2] Computing the probability of extending $W_b^{(t)}$ by each word in the vocabulary then becomes as simple as:

$$\mathbf{s}_t, \mathbf{h}_t^{(b)}, \mathbf{c}_t^{(b)} = \text{OurModel}\left(w_t, \mathbf{h}_{t-1}^{(b)}, \mathbf{c}_{t-1}^{(b)}\right) \tag{21}$$

$$P(w_{t+1} \mid W_b^{(t)}) = \text{softmax}(\mathbf{s}_t) \tag{22}$$

This suggests that during the selection phase, we will also need to store the hidden states corresponding to the updated sequences. As multiple beams at time step $t+1$ may be extensions from the same beam at time t, this may involve copying hidden states. Please see the example in the slides for more clarity on how this works step-by-step.

[2] Note that this is an overloading of notation from the model definition in Eq.2-3 where the $(1), (2), (3)$ superscripts denoted layers of the LSTM. Here we use $\mathbf{h}_t^{(b)}$ to denote the combined hidden state across all layers for the b^{th} beam at time t. Likewise for $\mathbf{c}_t^{(b)}$.

▶ TASK 2.1 [15pts] Implement the `beamsearch` function in the `decoder.py` skeleton code. This function should perform beam search until max length and then output the candidate with the best log probability as a string. The skeleton code for this function is show below:

```
def beamsearch(model, text_field, beams=5, prompt="", max_len=50):

  .
  .
  .

  return decodedString
```

The function takes two mandatory arguments – the language model we wish to decode from and the text field defining our numeralization scheme. Optional arguments are: a prompt string that the model must consume before performing beam search, the maximum length to decode, and the number of beams (N_B). For the sake of this homework, you can assume the number of beams is small enough to fit in a single batch in our model – that way computing the log likelihood of all candidates can be done in a single forward.

Once you've implemented beam search, running `decoder.py` will also execute three beam searches with N_B=1,10,50. Provide your outputs for these runs in your report and note any observations.

▶ Answer 2.1

Beam Search B=1

the night is dark and full of terrors. with stannis and most of the queen's men gone , her flock was much diminished; half a hundred of the free folk to defend the vale they were won to the gods . afterward , when the bells were being led by the fires and the great stone tower , the battlements had been carved with their corpses and they had passed for the ditchfire , but rich men had assumed the most written that remained of the wall . the nights were too small to be away . they had supped on the bare beards of peril , at the first sign of a tray . the shattered silence was well on the wall , painted in a narrow column that led to the mouth of the blackwater rush to smash the fishing lingering points and concealed a wide waters ,

Beam Search B=10

the night is dark and full of terrors. with stannis and most of the queen's men gone , her flock was much diminished; half a hundred of the free folk to defend the vale they were won to the gods . afterward , when the bells were being led by the fires and the great stone tower , the battlements had been carved with their corpses and nurse the eunuch . ser jaremy rykker and his lords led them in the river with shields of knights and knights , squires , and other merchants rowed a dozen burly khalasars into the outer ward of the flint towers , and hundreds of black marble trees were covered with iron - hewn heads of ghostskin . " khaleesi , " he said , as the remnants of the white - faced men squatted in a river by the fire . " you'd

Beam Search B=50

the night is dark and full of terrors. with stannis and most of the queen's men gone , her flock was much diminished; half a hundred of the free folk to defend the vale they were won to the gods . afterward , when the bells were being led by the fires and the great stone tower , the southron king - beyond - the - wall had not seemed likely there . their bodies were smashed and <unk> , and free folk were lit in the flowstone rolling - eyes and spears and spears and clams and poles . some of the tents were oats , wheat , the faces entering the field , the maps of gargoyles , and the wedding feast , the bright of the hair that grew like dangling against his back . when the wind blew forty feet and fifth gust of wind ,

▶ Answer 2.1

For B=1, the output is very similar to a greedy search, where the model always chooses the most probable next token at each step. This results in a coherent and somewhat repetitive text that follows a straightforward and likely path in the sequence of tokens.

With B=10, the model explores more options at each step by keeping track of the top 10 most probable sequences. This generates more diverse and detailed outputs, as seen with the inclusion of additional characters and events like "ser jaremy rykker and his lords," which were not present in the B=1 output. This results in a richer and more varied narrative.

Lastly, for B=50, the beam width is significantly larger. Therefore it is allowing the model to consider a much wider range of possible sequences. This generates even more diversity and detail, but can also introduce some less coherent elements, such as "<unk>" tokens or repetitive phrases like "spears and spears." This wider search space can capture more context and produce more creative outputs but might also introduce artifacts or inconsistencies due to the broader range of considered sequences.