Natural Language Processing - Assignment 1

Instructions

1. Due at 3 pm, February
2. You can submit your a DF **with the code package** or **a single Jupiter notebook** containing both the answers and the code;
3. For the coding questions, besides the code, you are encouraged to additionally give some descriptions of your code design and its workflow. Detailed analysis of the experimental results is also preferred;
4. Total marks are 100;

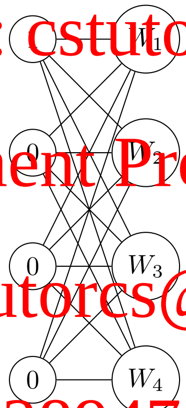
Question 1

(**11 marks**) We could represent the word with one-hot encoding in a language model instead of ascii representations. That is, we represent the word w with:

$$\overbrace{[0, 0, ..., 1, ..., 0, 0]}^{\|V\|}$$

where V is our vocabulary [illegible] f it.

1. Suppose we have a voc[illegible] $girl, woman, word, words\}$, please represent each word using the one-hot encoding. (**3 m**[illegible]
2. Combined with the pre[illegible] are the drawbacks of doing so? Answer at least two reasons. (**4 marks**)
3. Given a vocabulary $\{girl, woman, boy, man\}$, consider a computation unit with 4 nodes when we use the one-hot approach to encode the word "girl": (**4 marks**)



In this case, we need 4×4 nodes to maintain the computation unit. Do you have any methods that could better represent the word in this vocabulary and reduce the amount of nodes to 2×4 ? Please give the solution and explanations.

Question 2

(**24 marks**) In the field of NLP, for a sequence of text $S = \{w_1, w_2, ..., w_T\}$, the statistical language model transforms the joint probability of a sequence into a product of conditional probabilities:

$$P(S) = P(w_1, w_2, ..., w_T) = \prod_{t=1}^{T} p(w_t|w_1, w_2, ..., w_{t-1})$$

However, this formulation [illegible]tice due to the huge computational costs. We prefer to use its simplified version - the n-gra[illegible]

$$p(w_t|w_1, w_2, ..., w_{t-1}) \approx p(w_t|w_{t-n+1}, ..., w_{t-1})$$

We usually use bi-gram (n=2[illegible] model the language.

1. Given the text "the physical meanings of both the complex-valued entity and word embeddings is unknown", please list all the bi-gram and tri-gram forms that contains the word "meanings" or "embeddings". (**8 marks**)

2. What are the disadvantages of n-gram model? (**6 marks**)

3. Distributed representation solves the problem of one-hot encodings by mapping each word of the original one-hot encoding to a shorter word vector, the dimension of which can be specified by ourselves during the training process according to the needs of the task. In PyTorch, *nn.Embedding()* represents the words using a trainable matrix with a given dimension. The code below defines a language model: (**10 marks**)


```
import torch.nn as nn
import torch.nn.functional as F

class LanguageModeler(nn.Module):
    def __init__(self, vocab_size, embedding_dim, context_size, hidden_size=128):
        super(LanguageModeler, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.linear1 = nn.Linear(context_size * embedding_dim, hidden_size)
        self.linear2 = nn.Linear(hidden_size, vocab_size)

    def forward(self, inputs):
        embeds = self.embeddings(inputs).view((1, -1))
        out = F.relu(self.linear1(embeds))
        out = self.linear2(out)
        log_probs = F.log_softmax(out, dim=1)
        return log_probs
```

Please train a four-gram language model (only consider the last three words before the target word) for 10 epochs based on the text given in Question 2.1. Compare and analyze the difference of training loss with different settings of the embedding dimension {32, 64, 128}.

Question 3

(**45 marks**) Training an n-gram language model could be achieved by predicting the word given its n-gram elements. This idea is pretty similar to the The Continuous Bag of Words (CBOW). Given the text "There is an apple on the table", CBOW predicts the word apple given the context of a few words before (is, an) and a few words after(on, the) the target word. In contrast, the skip-gram uses the central word apple to predict the context is, an, on, the.

The CBOW model is as f[illegible] word w_i and an N context window on each side $w_{i-1}, ..., w_{i-N}$ and $w_{i+1}, ..., w_{i+N}$. Referring [illegible] collectively as C, CBOW tries to minimize:

$$[illegible] - \log Softmax(A \sum_{w \in C} q_w) + b),$$

where q_w is the embedding o[illegible]

1. Please use the text and [illegible] ovided in Question 2 to train a CBOW model with fixed window 2. (**15 marks**)

2. Please use the text and Language Modeler provided in Question 2 to train a skip-gram model. (**15 marks**)

3. Please use the given Wikipedia corpus and the language model to train a CBOW/Skip-gram model, and compare the word similarities among semantically closed word pairs, such as {woman, man, men}. (**15 marks**)

Question 4

(**20 marks**) Splitting a text into smaller chunks is a task that is harder than it looks, and there are multiple ways of doing so.

1. Given the text: "Peter is too old to do this work, Marry is older than Peter, John is oldest one however he is still doing this work.", [illegible]nize this text is splitting by spaces. However, there are also some disadvantages of this si[illegible]st at least two disadvantages and explain them. (**5 marks**)

2. Transformers models us[illegible]tween word-level and character-level tokenization called **subword** tokenization. BPE(Byt[illegible]ubword-level tokenization approach introduced in *Neural Machine Translation of Rare W*[illegible]*ts (Sennrich et al., 2015).* . BPE relies on a pre-tokenizer that splits the training data [illegible]ation can be as simple as space tokenization. Let us assume that after pre-tokenization, t[illegible]ds including their frequency has been determined:

$$(old, 10), (older, 5), (oldest, 8), (hug, 8), (pug, 4), (hugs, 5)$$

We obtain an base vocabulary:

$$o, l, d, e, r, s, t, h, u, g, p$$

Splitting all words into symbols in the base vocabulary, we obtain:

$$(o, l, d, 10), (o, l, d, e, r, 5), (o, l, d, e, s, t, 8), (h, u, g, 8), (p, u, g, 4), (h, u, g, s, 5)$$

BPE then counts the frequency of each possible symbol pair and picks the symbol pair that occurs most frequently. In the above example, "o" followed by "l" is present $10+5+8 = 23$ times. Thus, the first merge rule the tokenizer learns is to group all "o" symbols followed by an "l" symbol together. Next, "ol" is added to the vocabulary. The set of words then becomes

$$(ol, d, 10), (ol, d, e, r, 5), (ol, d, e, s, t, 8), (h, u, g, 8), (p, u, g, 4), (h, u, g, s, 5)$$

This process will run iteratively. The vocabulary size, i.e. the base vocabulary size + the number of merges, is a hyperparameter to choose. The learned merge rules would then be applied to new words (as long as those new words do not include symbols that were not in the base vocabulary). The word not in the base vocabulary would be repersented as "[unk]". Implement this BPE tokenizer, set the vocabulary size as 16 and train this BPE tokenizer to finish the iterative procss. Use the trained tokenizer to tokenize the words below:(**15 marks**)

$$\{old, oldest, older, nug, gug, huggingface\}.$$