In [ ]:

```

```

In [1]:

```python
"""We are about to study the idea of a computational process.
Computational processes are abstract beings that inhabit computers.
As they evolve, processes manipulate other abstract things called data.
The evolution of a process is directed by a pattern of rules
called a program. People create programs to direct processes. In effect,
we conjure the spirits of the computer with our spells."""
```

Out[1]:

```
'We are about to study the idea of a computational process.\nComputational processes
are abstract beings that inhabit computers.\nAs they evolve, processes manipulate oth
er abstract things called data.\nThe evolution of a process is directed by a pattern
of rules\ncalled a program. People create programs to direct processes. In effect,\nw
e conjure the spirits of the computer with our spells.'
```

In [7]:

```python
import torch
import numpy as np
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.autograd import Variable
```

In [8]:

```python
torch.manual_seed(1)
```

Out[8]:

```
<torch._C.Generator at 0x1ab8a385d10>
```

In [9]:

```python
# Implementing CBOW model for the exercise given by a tutorial in pytorch.org/tutorial
context_size = 2 # {w_i-2 ... w_i ... w_i+2}
embedding_dim = 10
```

In [10]:

```python
raw_text = """We are about to study the idea of a computational process.
Computational processes are abstract beings that inhabit computers.
As they evolve, processes manipulate other abstract things called data.
The evolution of a process is directed by a pattern of rules
called a program. People create programs to direct processes. In effect,
we conjure the spirits of the computer with our spells.""".split()
```

In [11]:

```python
def make_context_vector(context, word_to_idx):
    idxs = [word_to_idx[w] for w in context]
    return torch.tensor(idxs, dtype=torch.long)

vocab = set(raw_text)
vocab_size = len(vocab)

word_to_idx = {word: i for i, word in enumerate(vocab)}
idx_to_word = {i: word for i, word in enumerate(vocab)}

data = []
```

In [12]:

```python
for i in range(2, len(raw_text) - 2):
    context = [raw_text[i-2], raw_text[i-1],
               raw_text[i+1], raw_text[i+2]]
    target = raw_text[i]
    data.append((context, target))
```

In [ ]:

```
# Ge
```

In [13]:

```
class CBOW(nn.Module):

    def __init__(self, vocab_size, embedding_dim):
        super(CBOW, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.proj = nn.Linear(embedding_dim, 128)
        self.output = nn.Linear(128, vocab_size)

    def forward(self, inputs):
        embeds = sum(self.embeddings(inputs)).view(1, -1)
        out = F.relu(self.proj(embeds))
        out = self.output(out)
        nll_prob = F.log_softmax(out, dim=-1)
        return nll_prob
```

In [14]:

```
model = CBOW(vocab_size, embedding_dim)
optimizer = optim.SGD(model.parameters(), lr=0.001)

losses = []
loss_function = nn.NLLLoss()
```

In [15]:

```
for epoch in range(100):
    total_loss = 0
    for context, target in data:
        context_vector = make_context_vector(context, word_to_idx)

        # Remember PyTorch accumulates gradients; zero them out
        model.zero_grad()

        nll_prob = model(context_vector)
        loss = loss_function(nll_prob, Variable(torch.tensor([word_to_idx[target]])))

        # backpropagation
        loss.backward()
        # update the parameters
        optimizer.step()

        total_loss += loss.item()

    losses.append(total_loss)

print(losses)
```

```
[235.3547396659851, 231.2504210472107, 227.28084659576416, 223.43712449073792, 219.71
41547203064, 216.10481572151184, 212.6047966480255, 209.2084927558899, 205.9115238189
6973, 202.70872175693512, 199.59591102600098, 196.56743896007538, 193.62000679969788,
190.75147688388824, 187.9560902118683, 185.2296940088272, 182.5699644088745, 179.9718
5349464417, 177.43296229839325, 174.94851350784302, 172.5169186592102, 170.1332973241
806, 167.79425406455994, 165.49802088737488, 163.24142158031464, 161.02096807956696,
158.83659052848816, 156.68791508674622, 154.57073199748993, 152.48405319452286, 150.4
2698520421982, 148.39951944351196, 146.40119564533234, 144.42999005317688, 142.486186
6235733, 140.56787306070328, 138.67332816123962, 136.80279511213303, 134.957177817821
5, 133.13518822193146, 131.33669871091843, 129.56229543685913, 127.80784606933594, 12
6.07720935344696, 124.3686358332634, 122.68193072080612, 121.01412802934647, 119.3694
9115991592, 117.74674564599991, 116.14425528049469, 114.56068485975266, 112.996389925
47989, 111.45089781284332, 109.92645388841629, 108.4185346364975, 106.93212679028511,
105.46217346191406, 104.01193851232529, 102.57850006222725, 101.16399890184402, 99.76
753598451614, 98.38907065987587, 97.02785429358482, 95.68386802077293, 94.35694420337
677, 93.04814583063126, 91.75640892982483, 90.48186030983925, 89.22262611985207, 87.9
8284965753555, 86.75947001576424, 85.55140778422356, 84.36102682352066, 83.1872391700
7446, 82.02810183167458, 80.88697323203087, 79.75979653000832, 78.6489366889, 77.5546
1141467094, 76.4736153781414, 75.41068941354752, 74.36037436127663, 73.3241935074329
4, 72.30440330505371, 71.29875430464745, 70.30548828840256, 69.32749426364899, 68.363
38722705841, 67.41240087151527, 66.47514280676842, 65.55222599208355, 64.642161384224
89, 63.7458339035511, 62.86129319667816, 61.99146384000778, 61.13352331519127, 60.289
133951067924, 59.456111431121826, 58.63670785725117, 57.8289882093668]
```

In [16]:

```
# Let's see if our CBOW model works or not

print("*************************************************************************")

context = ['process.','Computational','are', 'abstract']
context_vector = make_context_vector(context, word_to_idx)
a = model(context_vector).data.numpy()
print('Raw text: {}\n'.format(' '.join(raw_text)))
print('Test Context: {}\n'.format(context))
max_idx = np.argmax(a)
print('Prediction: {}'.format(idx_to_word[max_idx]))
```

```
*************************************************************************
Raw text: We are about to study the idea of a computational process. Computational pr
ocesses are abstract beings that inhabit computers. As they evolve, processes manipul
ate other abstract things called data. The evolution of a process is directed by a pa
ttern of rules called a program. People create programs to direct processes. In effec
t, we conjure the spirits of the computer with our spells.

Test Context: ['process.', 'Computational', 'are', 'abstract']

Prediction: the
```

In [17]:

```
context = ['processes','manipulate','abstract', 'things']
context_vector = make_context_vector(context, word_to_idx)
a = model(context_vector).data.numpy()
print('Raw text: {}\n'.format(' '.join(raw_text)))
print('Test Context: {}\n'.format(context))
max_idx = np.argmax(a)
print('Prediction: {}'.format(idx_to_word[max_idx]))
```

```
Raw text: We are about to study the idea of a computational process. Computational pr
ocesses are abstract beings that inhabit computers. As they evolve, processes manipul
ate other abstract things called data. The evolution of a process is directed by a pa
ttern of rules called a program. People create programs to direct processes. In effec
t, we conjure the spirits of the computer with our spells.

Test Context: ['processes', 'manipulate', 'abstract', 'things']

Prediction: other
```

```
In [ ]:
```