In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
import matplotlib.pylab as pylab
import numpy as np
%matplotlib inline
```

In [2]:
```
#Data Prepration
import re
```

In [3]:
```
sentences = """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."""
```

Clean Data

In [4]:
```
# remove special characters
sentences = re.sub('[^A-Za-z0-9]+', ' ', sentences)

# remove 1 letter words
sentences = re.sub(r'(?:^| )\w(?:$| )', ' ', sentences).strip()

# lower all characters
sentences = sentences.lower()
sentences
```

Out[4]:
```
'we are about to study the idea of computational process computational processes are abstract beings that inhab
it computers as they evolve processes manipulate other abstract things called data the evolution of process is
directed by pattern of rules called program people create programs to direct processes in effect we conjure the
spirits of the computer with our spells'
```

Vocabulary

In [32]:
```
words = sentences.split()
print(words)
vocab = set(words)
print(vocab)
```

```
['we', 'are', 'about', 'to', 'study', 'the', 'idea', 'of', 'computational', 'process', 'computational', 'process
es', 'are', 'abstract', 'beings', 'that', 'inhabit', 'computers', 'as', 'they', 'evolve', 'processes', 'manipula
te', 'other', 'abstract', 'things', 'called', 'data', 'the', 'evolution', 'of', 'process', 'is', 'directed', 'by
', 'pattern', 'of', 'rules', 'called', 'program', 'people', 'create', 'programs', 'to', 'direct', 'processes', '
in', 'effect', 'we', 'conjure', 'the', 'spirits', 'of', 'the', 'computer', 'with', 'our', 'spells']
{'spells', 'directed', 'we', 'data', 'spirits', 'inhabit', 'direct', 'conjure', 'process', 'study', 'about', 'pe
ople', 'pattern', 'computer', 'computational', 'of', 'evolution', 'is', 'in', 'our', 'program', 'that', 'with',
'they', 'manipulate', 'abstract', 'programs', 'things', 'as', 'create', 'evolve', 'effect', 'processes', 'comput
ers', 'other', 'the', 'called', 'beings', 'are', 'rules', 'by', 'to', 'idea'}
```

In [7]:
```
vocab_size = len(vocab)
embed_dim = 10
context_size = 2
```

Implementation

In [33]:
```
word_to_ix = {word: i for i, word in enumerate(vocab)}
ix_to_word = {i: word for i, word in enumerate(vocab)}
print(word_to_ix)
print("==========================")
print(ix_to_word)
```

```
{'spells': 0, 'directed': 1, 'we': 2, 'data': 3, 'spirits': 4, 'inhabit': 5, 'direct': 6, 'conjure': 7, 'process
': 8, 'study': 9, 'about': 10, 'people': 11, 'pattern': 12, 'computer': 13, 'computational': 14, 'of': 15, 'evol
ution': 16, 'is': 17, 'in': 18, 'our': 19, 'program': 20, 'that': 21, 'with': 22, 'they': 23, 'manipulate': 24,
'abstract': 25, 'programs': 26, 'things': 27, 'as': 28, 'create': 29, 'evolve': 30, 'effect': 31, 'processes': 3
2, 'computers': 33, 'other': 34, 'the': 35, 'called': 36, 'beings': 37, 'are': 38, 'rules': 39, 'by': 40, 'to':
41, 'idea': 42}
==========================
{0: 'spells', 1: 'directed', 2: 'we', 3: 'data', 4: 'spirits', 5: 'inhabit', 6: 'direct', 7: 'conjure', 8: 'proc
ess', 9: 'study', 10: 'about', 11: 'people', 12: 'pattern', 13: 'computer', 14: 'computational', 15: 'of', 16: '
evolution', 17: 'is', 18: 'in', 19: 'our', 20: 'program', 21: 'that', 22: 'with', 23: 'they', 24: 'manipulate',
25: 'abstract', 26: 'programs', 27: 'things', 28: 'as', 29: 'create', 30: 'evolve', 31: 'effect', 32: 'processes
', 33: 'computers', 34: 'other', 35: 'the', 36: 'called', 37: 'beings', 38: 'are', 39: 'rules', 40: 'by', 41: 't
o', 42: 'idea'}
```

Data bags

In [34]:
```
# data - [(context), target]
```

```
data = []
for i in range(2, len(words) - 2):
    context = [words[i - 2], words[i - 1], words[i + 1], words[i + 2]]
    target = words[i]
    data.append((context, target))
print(data[:5])
```

```
[(['we', 'are', 'to', 'study'], 'about'), (['are', 'about', 'study', 'the'], 'to'), (['about', 'to', 'the', 'ide
a'], 'study'), (['to', 'study', 'idea', 'of'], 'the'), (['study', 'the', 'of', 'computational'], 'idea')]
```

Embeddings

In [35]:
```
embeddings =  np.random.random_sample((vocab_size, embed_dim))
print(embeddings[:1])
```

```
[[0.88394973 0.01562765 0.54950423 0.21921585 0.85106201 0.45702621
  0.01045269 0.17038085 0.19150149 0.27374295]]
```

Linear Model

In [36]:
```
def linear(m, theta):
    w = theta
    return m.dot(w)
```

Log softmax + NLLloss = Cross Entropy

In [37]:
```
def log_softmax(x):
    e_x = np.exp(x - np.max(x))
    return np.log(e_x / e_x.sum())
```

In [38]:
```
def NLLLoss(logs, targets):
    out = logs[range(len(targets)), targets]
    return -out.sum()/len(out)
```

In [39]:
```
def log_softmax_crossentropy_with_logits(logits,target):

    out = np.zeros_like(logits)
    out[np.arange(len(logits)),target] = 1

    softmax = np.exp(logits) / np.exp(logits).sum(axis=-1,keepdims=True)

    return (- out + softmax) / logits.shape[0]
```

Forward function

In [40]:
```
def forward(context_idxs, theta):
    m = embeddings[context_idxs].reshape(1, -1)
    n = linear(m, theta)
    o = log_softmax(n)

    return m, n, o
```

Backward function

In [41]:
```
def backward(preds, theta, target_idxs):
    m, n, o = preds

    dlog = log_softmax_crossentropy_with_logits(n, target_idxs)
    dw = m.T.dot(dlog)

    return dw
```

Optimize function

In [42]:
```
def optimize(theta, grad, lr=0.03):
    theta -= grad * lr
    return theta
```

Training

In [43]:
```
#Genrate training data

theta = np.random.uniform(-1, 1, (2 * context_size * embed_dim, vocab_size))
```

In [48]:
```
epoch_losses = {}

for epoch in range(80):

    losses =  []
```

```
    for context, target in data:
        context_idxs = np.array([word_to_ix[w] for w in context])
        preds = forward(context_idxs, theta)

        target_idxs = np.array([word_to_ix[target]])
        loss = NLLLoss(preds[-1], target_idxs)

        losses.append(loss)

        grad = backward(preds, theta, target_idxs)
        theta = optimize(theta, grad, lr=0.03)


    epoch_losses[epoch] = losses
```

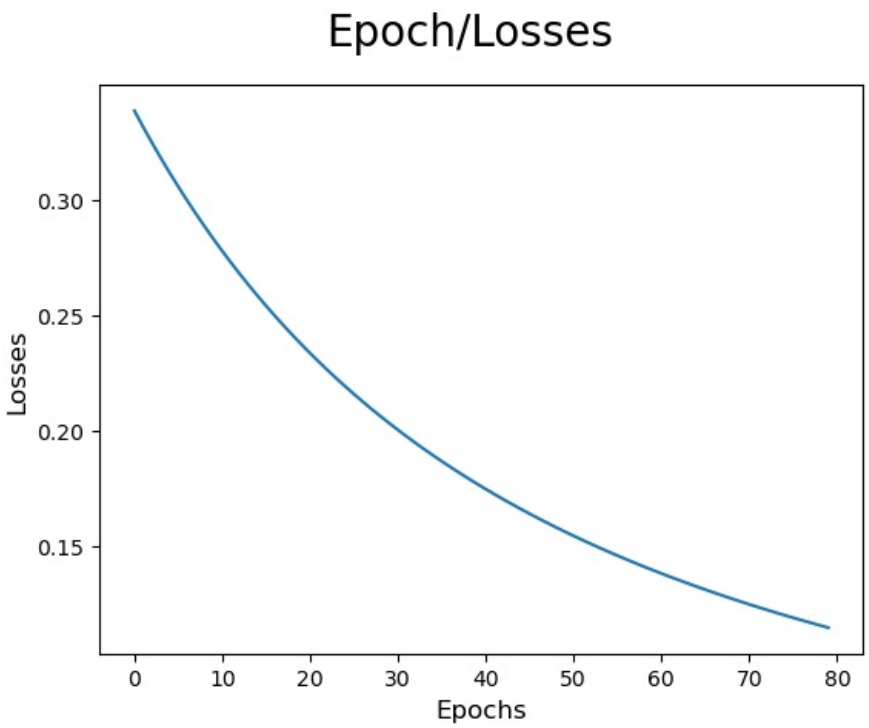
Analyze

Plot loss/epoch

In [49]:
```
ix = np.arange(0,80)

fig = plt.figure()
fig.suptitle('Epoch/Losses', fontsize=20)
plt.plot(ix,[epoch_losses[i][0] for i in ix])
plt.xlabel('Epochs', fontsize=12)
plt.ylabel('Losses', fontsize=12)
```

Out[49]:
```
Text(0, 0.5, 'Losses')
```



Predict function

In [50]:
```
def predict(words):
    context_idxs = np.array([word_to_ix[w] for w in words])
    preds = forward(context_idxs, theta)
    word = ix_to_word[np.argmax(preds[-1])]

    return word
```

In [51]:
```
# (['we', 'are', 'to', 'study'], 'about')
predict(['we', 'are', 'to', 'study'])
```

Out[51]:
```
'about'
```

Accuracy

In [52]:
```
def accuracy():
    wrong = 0

    for context, target in data:
        if(predict(context) != target):
            wrong += 1
```

```
    return (1 - (wrong / len(data)))
```

In [53]:
```
accuracy()
```

Out[53]:
```
1.0
```

In [54]:
```
predict(['processes', 'manipulate', 'things', 'study'])
```

Out[54]:
```
'other'
```

In [55]:
```
predict(['are', 'about', 'study', 'the'])
```

Out[55]:
```
'to'
```

In [ ]:
```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js