Homework 3

Congcheng Yan (cy2550)

March 30, 2020

Programming Part

Part 2

```
def get_input_representation(self, words, pos, state):
    rlist = []

    for k in range(2):
        if k == 0 :
            type = state.stack
        else:
            type = state.buffer
        for i in range(3):
            j = -(i+1)
            try:
                if (type[j] == 0 and pos[type[j]] == None):
                    rlist.append(self.word_vocab['<ROOT>'])
                elif pos[type[j]] in ['<CD>', '<NNP>', '<UNK>', '<ROOT>', '<NULL>']:
                    rlist.append(self.word_vocab[pos[type[j]]])
                else:
                    rlist.append(self.word_vocab[words[type[j]].lower()])

            except:
                rlist.append(self.word_vocab['<NULL>'])

    return np.array(rlist)

def get_output_representation(self, output_pair):
    rlist = np.zeros(91)
    ind = self.output_labels[output_pair]
    rlist[ind] = 1
    return rlist
```

Part 3

```
def build_model(word_types, pos_types, outputs):
    model = Sequential()
    model.add(Embedding(word_types, 32, input_length=6))
    model.add(Flatten())
    model.add(Dense(100, input_dim=6*32, activation='relu'))
    model.add(Dense(10, activation='relu'))
    model.add(Dense(outputs, activation='softmax'))

    model.compile(keras.optimizers.Adam(lr=0.01), loss="categorical_crossentropy")
    return model
```

Part 4

```
def parse_sentence(self, words, pos):
    state = State(range(1,len(words)))
    state.stack.append(0)
    while state.buffer:
        feature = self.extractor.get_input_representation(words, pos, state)
        np.reshape(feature, (6, 1))
        pre = self.model.predict(np.array([self.extractor.get_input_representation(words, pos, state), ]))
        list1 = np.flipud(np.argsort(pre)[0])
        for i in list1:
            act = self.output_labels[i]
            rel, label = act
            if rel == "shift":
                if len(state.buffer) == 1 and len(state.stack) > 0:
                    continue
                else:
                    state.shift()
                    break
            elif rel == "left_arc":
                if len(state.stack) == 0 or state.stack[-1] == 0:
                    continue
                else:
                    state.left_arc(label)
                    break
            elif rel == "right_arc":
                if len(state.stack) == 0:
                    continue
                else:
                    state.right_arc(label)
                    break
    result = DependencyStructure()
```