CS 585 – Fall 2023 – Homework 4

Problem 1

In [69]:
```
import csv
# reading a .tsv file in CoNLL format
def read_conll_file(file_path):
    seq_tokens,seq_tags = [],[]  # Lists for sequences of tokens and tags
    with open(file_path, 'r', newline='', encoding='utf-8') as file:
        curr_seq_tokens,curr_seq_tags = [],[]  # Tokens and Tags for the current sequence
        file_reader = csv.reader(file, delimiter='\t')
        for i in file_reader:
            if not i:
                if curr_seq_tokens:
                    seq_tokens.append(curr_seq_tokens)
                    seq_tags.append(curr_seq_tags)
                curr_seq_tokens = []
                curr_seq_tags = []
            else:
                curr_seq_tokens.append(i[0])
                curr_seq_tags.append(i[1])
        if curr_seq_tokens:
            seq_tokens.append(curr_seq_tokens)
            seq_tags.append(curr_seq_tags)
    return seq_tokens, seq_tags
train_token_seq, train_tag_seq = read_conll_file('D:/Masters/NLP/Assignments/Assignment4/conll/train.tsv')
test_token_seq, test_tag_seq = read_conll_file('D:/Masters/NLP/Assignments/Assignment4/conll/test.tsv')
# no of sequences in train and test data
print(f"The no of sequences in train: {len(train_token_seq)}")
print(f"The no of sequences in test: {len(test_token_seq)}")
#  first sequence of training dataset
print("The first sequence Tokens in training dataset:")
print(train_token_seq[0])
print("The first sequence Tags in training dataset:")
print(train_tag_seq[0])
```

```
The no of sequences in train: 5430
The no of sequences in test: 939
The first sequence Tokens in training dataset:
['Identification', 'of', 'APC2', ',', 'a', 'homologue', 'of', 'the', 'adenomatous', 'polyposis', 'coli', 'tu
mour', 'suppressor', '.']
The first sequence Tags in training dataset:
['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-Disease', 'I-Disease', 'I-Disease', 'I-Disease', 'O', 'O']
```

Problem 2

In [70]:
```
from collections import Counter
def count_tags_and_disease_words(seq_tag, seq_tok):
    tag_cnt = Counter(tag for t in seq_tag for tag in t)
    seq_disease_words = [token for i, sequence_tags in enumerate(seq_tag) for (token, tag) in zip(seq_tok[i],
    seq_common_disease = Counter(seq_disease_words).most_common(20)
    return tag_cnt, seq_common_disease
train_tag_counts, train_common_disease_words = count_tags_and_disease_words(train_tag_seq, train_token_seq)
print("Train data Tag counts:")
for tag, count in train_tag_counts.items():
    print(f"{tag}: {count}")
print("\nThe 20 Most Common Words with 'B-Disease' or 'I-Disease' Tags:")
for word, count in train_common_disease_words:
    print(f"{word}: {count}")
```

```
Train data Tag counts:
O: 124741
B-Disease: 5141
I-Disease: 6118

The 20 Most Common Words with 'B-Disease' or 'I-Disease' Tags:
-: 636
deficiency: 322
syndrome: 281
cancer: 269
disease: 255
of: 178
dystrophy: 176
breast: 151
ovarian: 132
X: 122
and: 120
DM: 120
ALD: 114
```

Problem 3

In [72]:
```
import spacy
load = spacy.load("en_core_web_sm")
def build_features(cur_token, pos, tokens_seq, tags_pos):
    features = []
    features.append(f'w0.lower={cur_token.lower()}')
    features.append(f'w0.suffix3={cur_token[-3:].lower()}')
    prev_token = tokens_seq[pos - 1] if pos > 0 else "BOS"
    features.append(f'wpos-1.lower={prev_token.lower()}')
    nxt_token = tokens_seq[pos + 1] if pos < len(tokens_seq) - 1 else "EOS"
    features.append(f'wpos+1.lower={nxt_token.lower()}')
    curr_pos = tags_pos[pos]
    features.append(f'w0.pos={curr_pos}')
    prev_pos = tags_pos[pos - 1] if pos > 0 else "POSBOS"
    features.append(f'wpos-1.pos={prev_pos}')
    next_pos = tags_pos[pos + 1] if pos < len(tokens_seq) - 1 else "POSEOS"
    features.append(f'wpos+1.pos={next_pos}')
    return features
def features_extract(tokens_seq, pos_tags):
    tot_features = []
    for i, j in enumerate(tokens_seq):
        features = build_features(j, i, tokens_seq, pos_tags)
        tot_features.append(features)
    return tot_features
first_seq = train_token_seq[0]
data = load(' '.join(first_seq))
pos_tags = [i.pos_ for i in data]
features = features_extract(first_seq, pos_tags)
# The features for the first 3 words in the first sequence
for i in range(3):
    print(first_seq[i])
    print(features[i])
```

```
Identification
['w0.lower=identification', 'w0.suffix3=ion', 'wpos-1.lower=bos', 'wpos+1.lower=of', 'w0.pos=NOUN', 'wpos-1.
pos=POSBOS', 'wpos+1.pos=ADP']
of
['w0.lower=of', 'w0.suffix3=of', 'wpos-1.lower=identification', 'wpos+1.lower=apc2', 'w0.pos=ADP', 'wpos-1.p
os=NOUN', 'wpos+1.pos=PROPN']
APC2
['w0.lower=apc2', 'w0.suffix3=pc2', 'wpos-1.lower=of', 'wpos+1.lower=,', 'w0.pos=PROPN', 'wpos-1.pos=ADP',
'wpos+1.pos=PUNCT']
```

Problem 4

In [73]:
```
import spacy
from sklearn.metrics import classification_report
import pycrfsuite
```

In [74]:
```
load = spacy.load("en_core_web_sm")
def ppos_tags(seq_toks):
    data = load(' '.join(seq_toks))
    return [token.pos_ for token in data]

features_train = [features_extract(tkn, ppos_tags(tkn)) for tkn in train_token_seq]
features_test = [features_extract(tkn, ppos_tags(tkn)) for tkn in test_token_seq]
train_true_tags = [i for tags in train_tag_seq for i in tags]
test_true_tags = [i for tags in test_tag_seq for i in tags]
trainer = pycrfsuite.Trainer()
i = 0
while i < len(features_train):
    p, q = features_train[i], train_tag_seq[i]
    trainer.append(p, q)
    i += 1
trainer.set_params({
    'c1': 1.0,  # L1 parameter
    'c2': 1e-3,  # L2 parameter
    'max_iterations': 50,
    'feature.possible_transitions': True
})
trainer.train('disease_crf_model.crfsuite')
tagger = pycrfsuite.Tagger()
tagger.open('disease_crf_model.crfsuite')
test_pred_tags = [tagger.tag(p) for p in features_test]
```

```
L-BFGS optimization
c1: 1.000000
c2: 0.001000
num_memories: 6
max_iterations: 50
epsilon: 0.000010
stop: 10
delta: 0.000010
linesearch: MoreThuente
linesearch.max_iterations: 20

***** Iteration #1 *****
Loss: 56592.644357
Feature norm: 1.000000
Error norm: 51712.154628
Active features: 19339
Line search trials: 1
Line search step: 0.000008
Seconds required for this iteration: 0.074
```

In [75]:
```
trainer.logparser.last_iteration
```

Out[75]:
```
{'num': 50,
 'scores': {},
 'loss': 5508.097082,
 'feature_norm': 68.63608,
 'error_norm': 82.630363,
 'active_features': 2987,
 'linesearch_trials': 1,
 'linesearch_step': 1.0,
 'time': 0.043}
```

In [76]:
```
report = classification_report(test_true_tags, [tag for tags in test_pred_tags for tag in tags], target_names=
print(report)
```

```
              precision    recall  f1-score   support

   B-Disease       0.85      0.70      0.76       960
   I-Disease       0.85      0.74      0.79      1087
           O       0.98      0.99      0.99     22441

    accuracy                           0.97     24488
   macro avg       0.89      0.81      0.85     24488
weighted avg       0.97      0.97      0.97     24488
```

In [ ]:

In [ ]:

Problem 5

In [79]:
```
tagger = pycrfsuite.Tagger()
tagger.open('disease_crf_model.crfsuite')
transition = tagger.info().transitions
print("Transition weights:")
for i in tagger.info().labels:
    for j in tagger.info().labels:
        w = transition.get((i, j), 0)
        print(f"Transition from {i} to {j}: {w:.2f}")
print("\n")
state_f = tagger.info().state_features
f_names = list(state_f.keys())
f_names_itr = iter(f_names)
while True:
    try:
        f_name = next(f_names_itr)
        f_name_str = " ".join(f_name)
        if 'w0.pos' in f_name_str or 'wposition-1.pos' in f_name_str or 'wposition+1.pos' in f_name_str:
            print("Feature = ", f_name_str)
            print("Weight = ", state_f[f_name])
    except StopIteration:
        break
```

```
Transition weights:
Transition from O to O: 2.30
Transition from O to B-Disease: -0.35
Transition from O to I-Disease: -9.81
Transition from B-Disease to O: -2.02
Transition from B-Disease to B-Disease: -5.57
Transition from B-Disease to I-Disease: 0.12
Transition from I-Disease to O: -1.07
Transition from I-Disease to B-Disease: -3.61
Transition from I-Disease to I-Disease: 1.33


Feature =  w0.pos=NOUN O
Weight =  -0.715517
Feature =  w0.pos=NOUN B-Disease
Weight =  1.664839
Feature =  w0.pos=NOUN I-Disease
Weight =  0.71342
Feature =  w0.pos=ADP O
Weight =  0.997213
Feature =  w0.pos=ADP B-Disease
Weight =  -0.762603
Feature =  w0.pos=ADP I-Disease
Weight =  -0.331236
Feature =  w0.pos=PROPN O
Weight =  -1.833887
Feature =  w0.pos=PROPN B-Disease
Weight =  1.045632
Feature =  w0.pos=PROPN I-Disease
Weight =  -0.060902
Feature =  w0.pos=PUNCT O
Weight =  0.980602
Feature =  w0.pos=PUNCT B-Disease
Weight =  -2.532279
Feature =  w0.pos=PUNCT I-Disease
Weight =  -0.086557
Feature =  w0.pos=DET O
Weight =  0.662899
Feature =  w0.pos=DET B-Disease
Weight =  -2.836904
Feature =  w0.pos=DET I-Disease
Weight =  0.01469
Feature =  w0.pos=ADJ O
```

```
Weight =  -0.157266
Feature =  w0.pos=ADJ B-Disease
Weight =  0.461247
Feature =  w0.pos=ADJ I-Disease
Weight =  -0.226993
Feature =  w0.pos=VERB O
Weight =  0.206825
Feature =  w0.pos=VERB B-Disease
Weight =  0.7462
Feature =  w0.pos=VERB I-Disease
Weight =  -0.705675
Feature =  w0.pos=ADV O
Weight =  0.001229
Feature =  w0.pos=ADV B-Disease
Weight =  -0.092388
Feature =  w0.pos=ADV I-Disease
Weight =  -0.325996
Feature =  w0.pos=NUM O
Weight =  -0.005274
Feature =  w0.pos=NUM B-Disease
Weight =  -1.555699
Feature =  w0.pos=NUM I-Disease
Weight =  0.313696
Feature =  w0.pos=SYM O
Weight =  0.0902
Feature =  w0.pos=CCONJ O
Weight =  0.997557
Feature =  w0.pos=CCONJ B-Disease
Weight =  -0.548902
Feature =  w0.pos=CCONJ I-Disease
Weight =  -0.391316
Feature =  w0.pos=SCONJ O
Weight =  1.789015
Feature =  w0.pos=PRON O
Weight =  0.369026
Feature =  w0.pos=PRON B-Disease
Weight =  0.008603
Feature =  w0.pos=PRON I-Disease
Weight =  -0.080347
Feature =  w0.pos=PART O
Weight =  0.22549
Feature =  w0.pos=AUX O
Weight =  0.332767
```

```
Feature =  w0.pos=AUX B-Disease
Weight =  -0.016247
Feature =  w0.pos=AUX I-Disease
Weight =  -3.317302
Feature =  w0.pos=X O
Weight =  4.8e-05
Feature =  w0.pos=SPACE O
Weight =  -0.111372
Feature =  w0.pos=SPACE B-Disease
Weight =  0.669148
```

In [80]:
```
for f_name, w in state_f.items():
    print("Feature = ",f_name)
    print("Weight = ",w)
    print("\n")
```

```
Feature =  ('w0.suffix3=ion', 'O')
Weight =  0.35401


Feature =  ('w0.suffix3=ion', 'B-Disease')
Weight =  -1.261956


Feature =  ('w0.suffix3=ion', 'I-Disease')
Weight =  0.362989


Feature =  ('wpos-1.lower=bos', 'O')
Weight =  3.266173


Feature =  ('wpos-1.lower=bos', 'B-Disease')
Weight =  2.04628
```

Problem 6

In [67]:
```
def seq_lvl_labels(tag_sequence):
    return 1 if "B-Disease" in tag_sequence else 0
doc_labels = [seq_lvl_labels(tags) for tags in train_tag_seq]
pred_doc_labels = [seq_lvl_labels(tags) for tags in test_pred_tags]
def doc_lvl_precision_recall(true_labels, pred_labels):
    tp,fp,fn,i = 0,0,0,0
    while i < len(true_labels) and i < len(pred_labels):
        true = true_labels[i]
        pred = pred_labels[i]
        if true == 1 and pred == 1:
            tp = tp + 1
        elif true == 0 and pred == 1:
            fp = fp + 1
        elif true == 1 and pred == 0:
            fn = fn + 1
        i += 1
    precision = tp / (tp + fp)  #precision
    recall = tp / (tp + fn)     #recall
    return precision, recall
precision, recall = doc_lvl_precision_recall(doc_labels, pred_doc_labels)
print("Document-level Precision:", precision)
print("Document-level Recall:", recall)
```

```
Document-level Precision: 0.968421052631579
Document-level Recall: 0.8534322820037106
```

In [ ]:
```
```