EX.NO.: 23
DATE: 28.03.2025

CRF BASED NER MODEL

To implement a Named Entity Recognition (NER) system using Conditional Random Fields (CRF) by extracting contextual and linguistic features from the CoNLL-2003 dataset, and training a model that accurately identifies named entities such as persons, organizations, locations, and miscellaneous entities in text.

PROCEDURE:

1. Install required libraries: `sklearn-crfsuite`, `nltk`, `datasets`, and `seqeval`.
2. Load the CoNLL-2003 dataset using the Hugging Face `datasets` library.
3. Extract tokens, POS tags, and NER labels from the dataset.
4. Convert the dataset into a list of sentences where each sentence is represented as a list of tuples (word, POS tag, NER tag).
5. Extract word-level features such as:
    a. Lowercase word
    b. Word prefixes and suffixes
    c. Word shape (e.g., uppercase, title case, digit)
    d. POS tag
    e. Contextual features from previous and next words
    f. Add beginning-of-sentence (BOS) and end-of-sentence (EOS) markers.
6. Format each sentence as a sequence of feature dictionaries and corresponding NER labels.
7. Train the model on the training data using LBFGS optimization with L1 and L2 regularization.
8. Evaluate the model using classification metrics such as precision, recall, and F1-score.

CODE AND OUTPUT:

```
import nltk
import sklearn_crfsuite
from datasets import load_dataset
from sklearn_crfsuite import metrics
from nltk import pos_tag
import numpy as np

# Download NLTK resources
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

# ✅ Load CoNLL-2003 dataset with trust_remote_code enabled
conll_data = load_dataset("conll2003", trust_remote_code=True)

# ✅ Extract POS/NER tag names
ner_label_list = conll_data["train"].features["ner_tags"].feature.names
pos_label_list = conll_data["train"].features["pos_tags"].feature.names

# ✅ Convert dataset into list of sentences
def prepare_data(dataset):
    sentences = []
    for words, pos_tags, ner_tags in zip(dataset['tokens'], dataset['pos_tags'],
dataset['ner_tags']):
        sentence = list(zip(words, pos_tags, ner_tags))
```

```python
        sentences.append(sentence)
    return sentences

train_sentences = prepare_data(conll_data['train'])
test_sentences = prepare_data(conll_data['test'])

# ✅ Word-level + contextual feature extractor
def word2features(sent, i):
    word = sent[i][0]
    postag = pos_label_list[sent[i][1]]

    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'word[-3:]': word[-3:],          # suffix
        'word[-2:]': word[-2:],          # suffix
        'word[:2]': word[:2],            # prefix
        'word[:3]': word[:3],            # prefix
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
        'postag': postag
    }

    # Previous word context
    if i > 0:
        word1 = sent[i - 1][0]
        postag1 = pos_label_list[sent[i - 1][1]]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:postag': postag1
        })
    else:
        features['BOS'] = True  # Beginning of Sentence

    # Next word context
    if i < len(sent) - 1:
        word1 = sent[i + 1][0]
        postag1 = pos_label_list[sent[i + 1][1]]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:postag': postag1
        })
    else:
        features['EOS'] = True  # End of Sentence

    return features

# ✅ Apply feature extraction and label formatting
```

```python
def extract_features(sent):
    return [word2features(sent, i) for i in range(len(sent))]

def get_labels(sent):
    return [ner_label_list[label] for (_, _, label) in sent]

X_train = [extract_features(s) for s in train_sentences]
y_train = [get_labels(s) for s in train_sentences]

X_test = [extract_features(s) for s in test_sentences]
y_test = [get_labels(s) for s in test_sentences]

# ✅ Train CRF model
crf = sklearn_crfsuite.CRF(
    algorithm='lbfgs',
    c1=0.1,
    c2=0.1,
    max_iterations=100,
    all_possible_transitions=True
)

crf.fit(X_train, y_train)

# ✅ Predict and evaluate
y_pred = crf.predict(X_test)
print(metrics.flat_classification_report(y_test, y_pred, digits=3))
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Hema\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Hema\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

```
Downloading data: 100% 983k/983k [00:00<00:00, 1.42MB/s]
Generating train split: 100% 14041/14041 [00:01<00:00, 7476.96 examples/s]
Generating validation split: 100% 3250/3250 [00:00<00:00, 7341.63 examples/s]
Generating test split: 100% 3453/3453 [00:00<00:00, 9176.90 examples/s]
```

```
              precision    recall  f1-score   support

       B-LOC      0.843     0.814     0.828      1668
      B-MISC      0.823     0.756     0.788       702
       B-ORG      0.765     0.729     0.746      1661
       B-PER      0.829     0.848     0.839      1617
       I-LOC      0.726     0.630     0.675       257
      I-MISC      0.632     0.667     0.649       216
       I-ORG      0.710     0.726     0.718       835
       I-PER      0.867     0.952     0.908      1156
           O      0.988     0.989     0.989     38323

    accuracy                          0.956     46435
   macro avg      0.798     0.790     0.793     46435
weighted avg      0.956     0.956     0.956     46435
```

INFERENCE:

The trained CRF model successfully learns the contextual and structural patterns of named entities in text using handcrafted features. By incorporating suffixes, prefixes, POS tags, and surrounding word information, the model is able to distinguish between entity types with high accuracy.