**Named Entity Recognition with CRFs**

**1. Project Overview**

**Goal**: Build a Named Entity Recognition (NER) system to identify persons, locations, and organizations in text.  
**Tech Stack**: Python, CRFSuite (via sklearn\_crfsuite), SpaCy, BIO tagging  
**Dataset**: CoNLL-2003 (English news with entity annotations)  
**Key Metric**: F1-score (target: 0.85)

**2. Core Concepts Explained**

**a) Named Entity Recognition (NER)**

* **What**: Identifying and classifying specific entities in text
* **Examples**:

| **Text** | **Entities** |
| --- | --- |
| "Tim Cook visited Apple in California" | Person: Tim Cook Organization: Apple Location: California |

**b) Sequence Labeling & BIO Tagging**

* **Sequence Labeling**: Assigning a label to every word in a sentence
* **BIO Format**:

["Barack", "Obama", "speaks", "in", "Paris"]

→ ["B-PER", "I-PER", "O", "O", "B-LOC"]

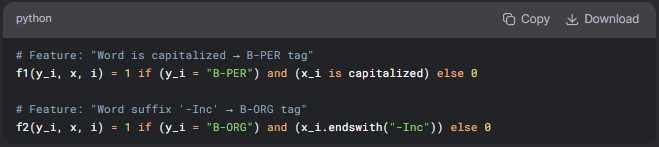
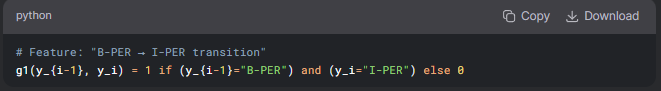
**c) Conditional Random Fields (CRFs)**

* **Purpose**: Predict sequences of labels while considering:
  1. Features of individual words
  2. Relationships between neighboring tags
* **Analogy**:
  1. *Like Logistic Regression*: Uses features to predict labels
  2. *But enhanced*: Considers tag sequences (e.g., "B-PER must be followed by I-PER")

**3. Deep Dive: CRFs**

**a) Feature Functions**

CRFs use two types of feature functions:

1. **State Features**: Connect word properties to tags  
   **Examples**:  
   
2. **Transition Features**: Model tag-to-tag relationships  
   

**b) How Scores Combine**

* **Probability Formula**:  
  
* **Key Insight**
  + Scores are exponentiated and multiplied (not summed directly)
  + Ensures valid probability distribution

**c) Number of Feature Functions**

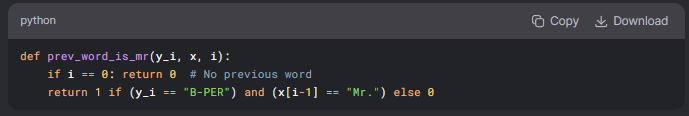
* **State Features**: (# feature types) × (# possible tags)  
  \*(e.g., 10 features × 5 tags = 50 functions)\*
* **Transition Features**: (# tags) × (# tags)  
  \*(e.g., 5 tags × 5 tags = 25 functions)\*
* **Total**: Typically thousands (handled efficiently via sparsity)

**d) Why f\_k(y\_i, x, i) Takes 3 Parameters**

The function needs:

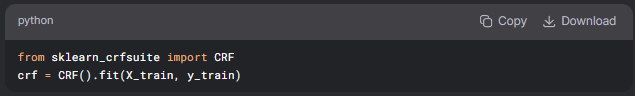
1. y\_i: Candidate tag being evaluated
2. x: Entire sentence (for context)
3. i: Position in sentence (to access neighbors)

**Example**:



**e) CRFSuite Explained**

* **What**: Optimized CRF implementation for NLP
* **Key Advantages**:
  1. Handles sparse features efficiently
  2. Simple scikit-learn-like API
  3. Implements L-BFGS optimization internally



**4. Feature Engineering**

**Key Features for CoNLL-2003**:

| **Feature Type** | **Example** | **Why Important** |
| --- | --- | --- |
| **Word Form** | "New" → word.lower=new | Normalization |
| **Capitalization** | "Apple" → is\_title=True | Entities often capitalized |
| **Prefix/Suffix** | "Inc." → suffix=Inc | Organization indicators |
| **POS Tags** | pos=NNP (Proper Noun) | Entities are often nouns |
| **Word Shape** | "iPhone" → shape=Xx | Pattern recognition |
| **Context** | Previous word="President" | Contextual clues |

**5. Workflow Summary**

1. **Data Preparation**:
   * Load CoNLL-2003 dataset
   * Convert to BIO format
2. **Feature Extraction**
3. **Training**
4. **Evaluation**

**6. Why CRFs Excel at NER**

1. **Context Awareness: Uses neighboring words/tags**
2. **Constraint Handling: Enforces rules like "I-PER can't follow B-LOC"**
3. **Feature Flexibility: Accepts diverse features (POS, shapes, etc.)**