Indian Institute of Technology, Patna

Assignment-1(NER)

Submitted By:

- Avinash Aanand-(Roll No.: 2403RES99)
- Aditya Gupta-(Roll No.:2403RES85)

- Due Date: 09th Sept 2024

Table of Content

Neural Network and NLP Lab

Problem Statements:

Statements

Named entity recognition (NER) is a natural language processing (NLP) task that identifies and categorizes named entities in text. NER is also known as entity chunking, entity extraction, or entity identification. For Example: person names, organizations, locations, medical codes, time expressions, quantities, and monetary values can be known as named entities in a text. Implement a machine learning model for Named Entity Recognition task in NLP.

Tasks

The goal of this assignment is to implement a machine learning model for the Named Entity Recognition (NER) task, a core problem in Natural Language Processing (NLP). NER involves identifying and classifying named entities (e.g., person names, organizations, locations, dates, and other key information) in a given text. The problem requires the use of machine learning techniques to train a model capable of accurately extracting these entities from a dataset.

Objectives:

- To build a machine learning model that can perform NER.
- To evaluate the performance of the NER model.
- To compare the developed model with alternative approaches used for NER tasks.

Introduction

NER

Named Entity Recognition (NER) is a significant task in NLP, where the goal is to automatically detect entities such as people, organizations, locations, dates, and other key categories from text. This information is critical for various applications such as information retrieval, question-answering systems, chatbots, and text summarization.

NER is often framed as a supervised learning problem where models are trained on annotated datasets, which include labels identifying specific entities. Popular techniques for NER involve statistical methods, machine learning algorithms like Conditional Random Fields (CRF), and deep learning models like Recurrent Neural Networks (RNNs) and Transformer-based architectures.

In this assignment, we aim to:

- Preprocess the dataset.
- Train a machine learning model capable of performing NER.
- Evaluate its performance on the test set.

Respective comparison

There are several alternative methods and models for NER, each with its own strengths and limitations. Below is a comparison of different approaches:

Rule-Based Methods:

- **Description**: These methods rely on predefined rules, dictionaries, and pattern matching to identify entities.
- **Advantages**: Easy to implement for specific, well-defined problems; good for domain-specific tasks.
- **Disadvantages**: Limited scalability, not suitable for large datasets or diverse entity types; requires manual effort and lacks generalization.

Machine Learning-Based Models (e.g., Conditional Random Fields, CRF):

- **Description**: CRFs are widely used for NER tasks as they model the probability of sequences of labels (entities) and consider the context of the surrounding words.
- **Advantages**: Well-suited for structured data with sequence dependencies; delivers good performance on small to medium datasets.
- **Disadvantages**: Requires feature engineering; may struggle with long dependencies between words.

Deep Learning-Based Models (e.g., BiLSTM, RNN):

- **Description**: Deep learning models, especially Bidirectional LSTM (BiLSTM), automatically learn features from text and capture long-range dependencies in the data.
- **Advantages**: Can capture complex relationships between words; requires less manual feature engineering.
- **Disadvantages**: Requires large datasets for effective training; computationally expensive.

Transformer-Based Models (e.g., BERT, RoBERTa):

- **Description**: Transformers have revolutionized NLP by using attention mechanisms to process entire sentences at once, making them highly effective for NER.
- **Advantages**: State-of-the-art performance on NER tasks; can handle long-range dependencies and context much better than previous models.
- **Disadvantages**: Requires large computational resources; training is time-consuming.

Why NER

Named Entity Recognition (NER) emerged as a critical solution because of the growing need to efficiently process and understand large amounts of unstructured text data in various fields. Here's why NER became necessary:

1. Explosion of Data:
With the rise of the internet, social media, and digital communication, the volume of text data generated daily has exploded. Traditional methods of manually extracting meaningful information from this data became impractical, leading to the development of automated solutions like NER to identify important information quickly.

2. Need for Automation:
Organizations across industries such as finance, healthcare, e-commerce, and law generate and store vast amounts of text documents. Manually extracting entities like names, dates, companies, and locations from these documents was slow and prone to errors. NER provides a way to automate this process, saving time and improving accuracy.

3. Information Extraction for Decision-Making:
NER became essential in situations where quick, accurate decision-making was needed based on real-time text analysis. For instance, in financial markets, extracting entities like company names or events from news articles helps investors make informed decisions faster. In healthcare, identifying medical terms in patient records improves diagnostics and research.

4. Personalized Services and Better Search Results:
As user expectations for personalized experiences grew (e.g., chatbots, voice assistants), NER became a key technology for understanding and responding accurately to user queries by identifying the relevant entities. In search engines, NER improved the ability to return specific, entity-based results (e.g., searching for a product or company).

5. Handling Diverse and Unstructured Data:
NER emerged to deal with the challenges of processing diverse, unstructured data. Text data can come in many forms—emails, news articles, social media posts, research papers, etc. NER makes sense of this varied data by extracting structured information, enabling further analysis and use in downstream tasks like classification, summarization, and translation.
In short, NER came into existence because there was a need to automatically and efficiently extract key information from vast and unstructured text data, allowing businesses and systems to function more effectively and respond faster in real-time scenarios.

Approaches

The Named Entity Recognition (NER) task is incorporated into the situation because the problem at hand involves tagging words not only with their Part-of-Speech (POS) labels but also with their Named Entity (NER) labels. Here's a breakdown of why NER was introduced and how the code is approaching this:

NER and POS Tagging:
In this approach, NER tags are applied to identify entities such as locations, organizations, people, etc., within a sentence. These NER tags are essential to recognize important pieces of information from the text (like "London" being a location or "Saturday" being a date).

Alongside NER, POS tagging is performed to label each word with its syntactic role (noun, verb, etc.). The use of both NER and POS tags together allows the model to analyze both grammatical structure and extract key entities from the text.

Viterbi Algorithm Joint Approach:

The code uses a joint Viterbi algorithm to decode both POS and NER tags simultaneously. This method enables the model to consider both types of information in parallel during the tagging process, enhancing the quality of the output.

The Viterbi algorithm essentially computes the most probable sequence of POS and NER tags given the words in the sentence. It does this by maximizing the probabilities over transitions and emissions for both POS and NER tags.

Transition and Emission Probabilities:

The transition probabilities capture the likelihood of moving from one POS/NER tag to another, allowing the model to learn tag sequences typical in the language.

Emission probabilities determine the likelihood of seeing a particular word given a POS or NER tag, helping the model decide which tag is appropriate for a word.

Reason for NER:

NER was introduced because entity recognition is a critical component of text processing, especially when the goal is to extract structured information (such as names, dates, locations) from raw text.

By combining NER and POS, this model can both structure the text syntactically and pull out key entities for further analysis or tasks.

The NER task is essential in this situation for recognizing important entities in the text. The code combines POS and NER tagging using a joint Viterbi algorithm that optimizes both tasks simultaneously. The transitions and emissions learned from the data help predict the most probable POS and NER tags for each word in a given sentence.

Analysis

Through Pseudo Code

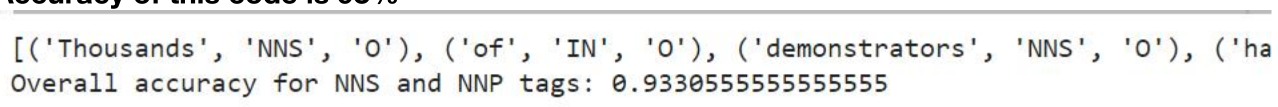
```
Load dataset into a DataFrame.
Process data into (word, POS, NER) tuples.
Initialize transition and emission probability dictionaries for POS and NER tags.
For each sentence in the dataset:
  For each word, POS, and NER tag:
    Increment transition count from the previous POS to the current POS.
    Increment emission count of the word given the POS tag.
    Do the same for NER tag transitions and emissions.

Normalize counts to compute probabilities.
Input: Sentence (list of words).
Initialize Viterbi matrix V[0] for the first word.
For each combination of POS and NER tags:
  Calculate the joint probability using transition and emission probabilities.
  Store the path that maximizes this probability.

For each word in the sentence (from second word onwards):
  For each combination of current POS and NER tags:
    For each combination of previous POS and NER tags:
      Calculate the probability of transitioning and emitting the current word.
      Update the Viterbi matrix and path with the highest probability.
After processing all words, retrieve the best path for the last word.
Return the path with the predicted (word, POS, NER) tuples.
Test the Viterbi algorithm on sample sentences.
Compare predicted (word, POS, NER) tuples with actual ones.
Compute accuracy as the proportion of correct predictions.
```

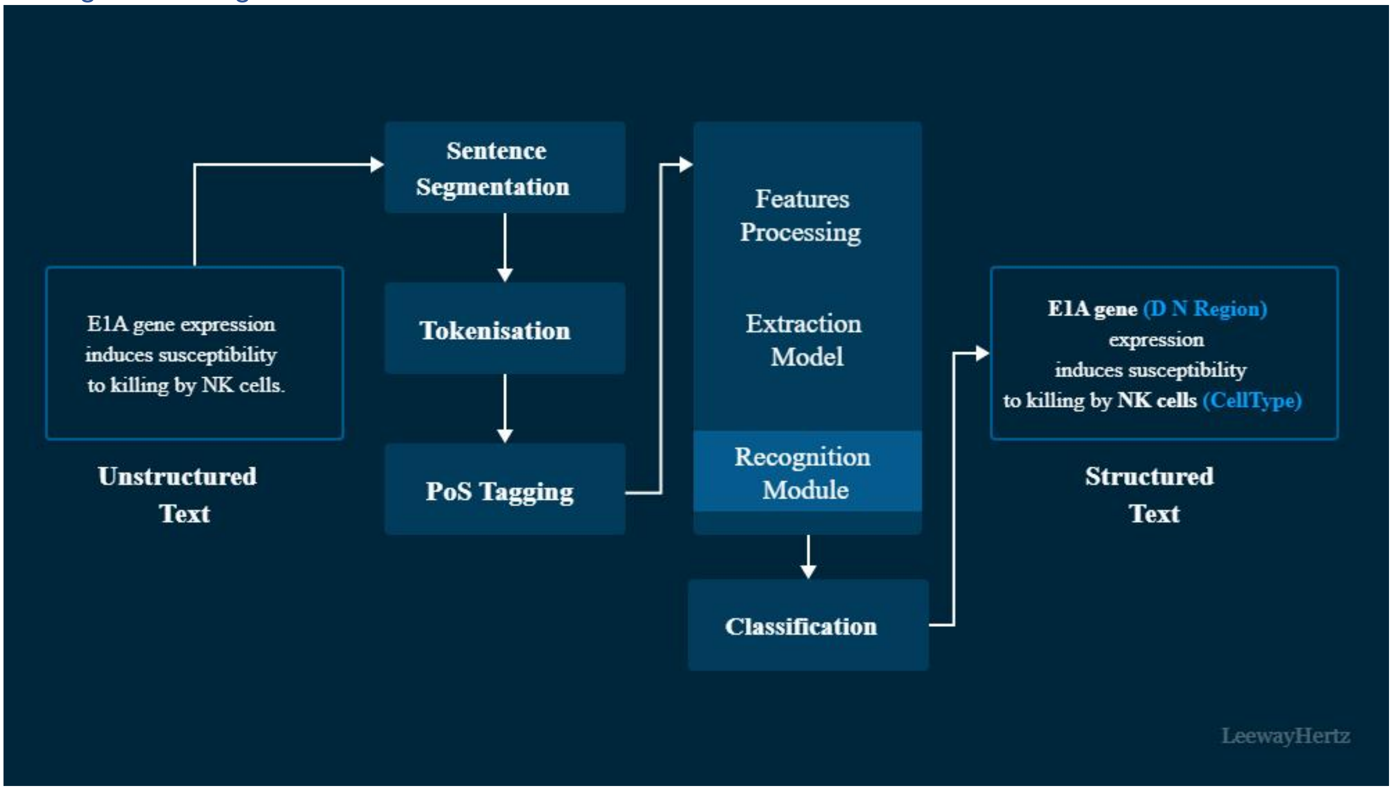
Accuracy

Accuracy of this code is 93%



```
[('Thousands', 'NNS', 'O'), ('of', 'IN', 'O'), ('demonstrators', 'NNS', 'O'), ('ha
Overall accuracy for NNS and NNP tags: 0.9330555555555555
```

Through Flow Diagram


E1A gene expression induces susceptibility to killing by NK cells.
Unstructured Text
Sentence Segmentation
Tokenisation
PoS Tagging
Features Processing
Extraction Model
Recognition Module
Classification
E1A gene (D N Region) expression induces susceptibility to killing by NK cells (CellType)
Structured Text
LeewayHertz

Limitation

The code you've provided for joint POS and NER tagging using the Viterbi algorithm has several limitations and potential issues. Here are some key limitations:

Data Sparsity:

- **Issue:** If the training data is sparse or lacks coverage for some words, POS tags, or NER tags, the model may encounter zero probabilities during inference, leading to poor performance.
- **Impact:** Words or tags not seen in training will have probabilities close to zero, causing the model to make incorrect predictions.

Handling Unknown Words:

Issue: The code uses a small constant (1e-6) to handle unseen words during emission probability calculations. However, this approach does not fully address the problem of unknown words.
Impact: Unknown words might still negatively impact the model's accuracy if the constant is not sufficient or if the model encounters many such words.

Memory and Computation Complexity:

Issue: The Viterbi algorithm's complexity grows with the number of POS and NER tags. For large tag sets, the computation and memory requirements can become prohibitive.
Impact: Large tag sets may lead to slower performance and higher resource consumption.

Transition and Emission Probability Estimation:

Issue: Transition and emission probabilities are estimated based on counts from the training data, which might not generalize well if the training data is not representative of the actual distribution in the test data.

Impact: The model might underperform on unseen data or data with a different distribution.

Overfitting:

Issue: The model might overfit to the training data if it is too complex or if there is not enough data. This means the model performs well on training data but poorly on new, unseen data.

Impact: Overfitting reduces the generalizability of the model, affecting its performance on real-world tasks.

No Regularization:

Issue: The code does not include any form of regularization or smoothing techniques beyond the small constant used for unseen words.

Impact: Without regularization, the model may have inflated probabilities for some transitions or emissions, leading to biased predictions.

Limited Evaluation:

Issue: The evaluation of the model's accuracy is based on a random subset of the data and might not provide a comprehensive assessment of the model's performance.

Impact: The accuracy reported might not be representative of the model's performance on the entire dataset.

No Model Validation:

Issue: The code does not include any validation or cross-validation procedures to assess the model's performance during training.

Impact: Lack of validation can lead to overfitting or underfitting, as there is no mechanism to tune hyperparameters or evaluate model performance on held-out data.

No Advanced Features:

Issue: The model only considers transition and emission probabilities without incorporating more advanced features like word embeddings or external linguistic resources.

Impact: The model may not capture complex linguistic patterns or relationships, limiting its performance on more challenging tasks.

While the provided code serves as a basic implementation of the Viterbi algorithm for joint POS and NER tagging, addressing these limitations would be necessary to improve its robustness and effectiveness. Incorporating techniques such as smoothing, regularization, more advanced feature representations, and thorough model evaluation can enhance performance and reliability.

Code Snippet

```
import pandas as pd
from collections import defaultdict

# Load and preprocess data
df = pd.read_csv("NER_Dataset.csv")
df["Word"], df["POS"], df["Tag"] = df["Word"].apply(eval), df["POS"].apply(eval), df["Tag"].apply(eval)

# Flatten data for transition and emission probabilities
flattened_data = [(w, p, t) for _, row in df.iterrows() for w, p, t in zip(row["Word"], row["POS"], row["Tag"])]
flat_df = pd.DataFrame(flattened_data, columns=["Word", "POS", "Tag"])

# Initialize probabilities
pos_transition_probs, ner_transition_probs = defaultdict(lambda: defaultdict(int)), defaultdict(lambda:
defaultdict(int))
pos_emission_probs, ner_emission_probs = defaultdict(lambda: defaultdict(int)), defaultdict(lambda:
defaultdict(int))
pos_counts, ner_tag_counts = defaultdict(int), defaultdict(int)

# Count transitions and emissions
prev_pos, prev_ner_tag = "<START>", "<START>"
for _, row in flat_df.iterrows():
    word, pos, tag = row["Word"], row["POS"], row["Tag"]
    pos_transition_probs[prev_pos][pos] += 1
    pos_emission_probs[pos][word] += 1
    ner_transition_probs[prev_ner_tag][tag] += 1
    ner_emission_probs[tag][word] += 1
    pos_counts[pos] += 1
    ner_tag_counts[tag] += 1
    prev_pos, prev_ner_tag = pos, tag

# Normalize probabilities
def normalize_probs(counts): return {k: v / sum(counts.values()) for k, v in counts.items()}
pos_transition_probs = {k: normalize_probs(v) for k, v in pos_transition_probs.items()}
ner_transition_probs = {k: normalize_probs(v) for k, v in ner_transition_probs.items()}
```

Running Output

```
[('Thousands', 'NNS', 'O'), ('of', 'IN', 'O'), ('demonstrators', 'NNS', 'O'), ('have', 'VBP', 'O'), ('marched',
'VBN', 'O'), ('through', 'IN', 'O'), ('London', 'NNP', 'B-geo'), ('Saturday', 'NNP', 'B-tim'), ('An', 'DT', 'O')]
[('In', 'NNS', 'O'), ('other', 'JJ', 'O'), ('news', 'NN', 'O'), (',', ',', 'O'), ('Pakistan', 'NNP', 'B-geo'), ("'s",
'POS', 'O'), ('Foreign', 'NNP', 'B-org'), ('Ministry', 'NNP', 'I-org'), ('says', 'VBZ', 'O'), ('Afghan', 'JJ', 'B-
gpe'), ('President', 'NNP', 'B-per'), ('Hamid', 'NNP', 'I-per'), ('Karzai', 'NNP', 'I-per'), ('is', 'VBZ', 'O'),
('set', 'VBN', 'O'), ('to', 'TO', 'O'), ('visit', 'VB', 'O'), ('Islamabad', 'NNP', 'B-geo'), ('for', 'IN', 'O'), ('talks',
'NNS', 'O'), ('on', 'IN', 'O'), ('Thursday', 'NNP', 'B-tim'), ('.', '.', 'O'), ('"', '``', 'O'), ('Leaders', 'NNS', 'O'),
('of', 'IN', 'O'), ('wealthy', 'JJ', 'O'), ('nations', 'NNS', 'O'), ('and', 'CC', 'O'), ('the', 'DT', 'O'), ('heads',
'NNS', 'O'), ('of', 'IN', 'O'), ('major', 'JJ', 'O'), ('developing', 'JJ', 'O'), ('economies', 'NNS', 'O'), ('gather',
'VBP', 'O'), ('in', 'IN', 'O'), ('Washington', 'NNP', 'B-geo'), ('November', 'NNP', 'B-tim'), ('15', 'CD', 'I-
tim'), ('to', 'TO', 'O'), ('focus', 'VB', 'O'), ('on', 'IN', 'O'), ('the', 'DT', 'O'), ('worldwide', 'JJ', 'O'),
('economic', 'JJ', 'O'), ('downturn', 'NN', 'O'), ('.', '.', 'O'), ('The', 'DT', 'O'), ('peace', 'NN', 'O'),
('agreement', 'NN', 'O'), ('calls', 'VBZ', 'O'), ('for', 'IN', 'O'), ('Indonesian', 'JJ', 'B-gpe'), ('soldiers', 'NNS',
'O'), ('to', '``', 'O'), ('leave', '``', 'O'), ('Aceh', '``', 'O'), (',', '``', 'O'), ('in', '``', 'O'), ('parallel', '``', 'O'), ('with',
'``', 'O'), ('militants', '``', 'O'), ('handing', '``', 'O'), ('over', '``', 'O'), ('weapons', '``', 'O'), ('to', '``', 'O'),
('international', '``', 'O'), ('monitors', '``', 'O'), ('.', '``', 'O'), ('After', '``', 'O'), ('sauntering', '``', 'O'), ('along',
'``', 'O'), ('for', '``', 'O'), ('some', '``', 'O'), ('time', '``', 'O'), ('he', '``', 'O'), ('discovered', '``', 'O'), ('the', '``',
'O'), ('Hare', '``', 'O'), ('by', '``', 'O'), ('the', '``', 'O'), ('wayside', '``', 'O'), (',', '``', 'O'), ('apparently', '``', 'O'),
('asleep', '``', 'O'), (',', '``', 'O'), ('and', '``', 'O'), ('seeing', '``', 'O'), ('a', '``', 'O'), ('chance', '``', 'O'), ('to', '``',
'O'), ('win', '``', 'O'), ('pushed', '``', 'O'), ('on', '``', 'O'), ('as', '``', 'O'), ('fast', '``', 'O'), ('as', '``', 'O'), ('he', '``',
'O'), ('could', '``', 'O'), (',', '``', 'O'), ('arriving', '``', 'O'), ('at', '``', 'O'), ('the', '``', 'O'), ('goal', '``', 'O'), ('hours',
'``', 'O'), ('afterward', '``', 'O'), (',', '``', 'O'), ('suffering', '``', 'O'), ('from', '``', 'O'), ('extreme', '``', 'O'),
('fatigue', '``', 'O'), ('and', '``', 'O'), ('claiming', '``', 'O'), ('the', '``', 'O'), ('victory', '``', 'O'), ('.', '``', 'O'), ('The',
'``', 'O'), ('press', '``', 'O'), ('rights', '``', 'O'), ('group', '``', 'O'), ('Reporters', '``', 'O'), ('Without', '``', 'O'),
('Borders', '``', 'O'), ('has', '``', 'O'), ('urged', '``', 'O'), ('Venezuela', '``', 'O'), ('to', '``', 'O'), ('end', '``', 'O'),
('its', '``', 'O'), ('efforts', '``', 'O'), ('to', '``', 'O'), ('shut', '``', 'O'), ('down', '``', 'O'), ('RCTV', '``', 'O'), ('.', '``',
'O'), ('The', '``', 'O'), ('U.N.', '``', 'O'), ('World', '``', 'O'), ('Food', '``', 'O'), ('Program', '``', 'O'), ('recently', '``',
'O'), ('said', '``', 'O'), ('that', '``', 'O'), ('about', '``', 'O'), ('3.8', '``', 'O'), ('million', '``', 'O'), ('Kenyans', '``', 'O'),
('will', '``', 'O'), ('not', '``', 'O'), ('have', '``', 'O'), ('enough', '``', 'O'), ('food', '``', 'O'), ('over', '``', 'O'), ('the',
'``', 'O'), ('next', '``', 'O'), ('months', '``', 'O'), ('because', '``', 'O'), ('of', '``', 'O'), ('the', '``', 'O'), ('drought', '``',
'O'), ('.', '``', 'O'), ('Nadal', '``', 'O'), ('joins', '``', 'O'), ('American', '``', 'O'), ('Andre', '``', 'O'), ('Agassi', '``',
'O'), ('on', '``', 'O'), ('the', '``', 'O'), ('sidelines', '``', 'O'), ('for', '``', 'O'), ('the', '``', 'O'), ('tournament', '``', 'O'),
(',', '``', 'O'), ('which', '``', 'O'), ('could', '``', 'O'), ('still', '``', 'O'), ('lose', '``', 'O'), ('Russians', '``', 'O'),
('Marat', '``', 'O'), ('Safin', '``', 'O'), ('and', '``', 'O'), ('Maria', '``', 'O'), ('Sharapova', '``', 'O'), ('because', '``',
'O'), ('of', '``', 'O'), ('fitness', '``', 'O'), ('doubts', '``', 'O'), ('.', '``', 'O'), ('More', '``', 'O'), ('than', '``', 'O'),
('2,00,000', '``', 'O'), ('people', '``', 'O'), ('have', '``', 'O'), ('been', '``', 'O'), ('killed', '``', 'O'), ('in', '``', 'O'),
('Darfur', '``', 'O'), ('since', '``', 'O'), ('rebels', '``', 'O'), ('began', '``', 'O'), ('an', '``', 'O'), ('uprising', '``', 'O'),
('against', '``', 'O'), ('Sudan', '``', 'O'), ("'s", '``', 'O'), ('central', '``', 'O'), ('government', '``', 'O'), ('in', '``', 'O'),
('early', '``', 'O'), ('2003', '``', 'O'), ('.', '``', 'O'), ('Their', '``', 'O'), ('repeated', '``', 'O'), ('failure', '``', 'O'), ('to',
'``', 'O'), ('reach', '``', 'O'), ('a', '``', 'O'), ('consensus', '``', 'O'), ('on', '``', 'O'), ('the', '``', 'O'), ('controversial',
'``', 'O'), ('bill', '``', 'O'), ('has', '``', 'O'), ('raised', '``', 'O'), ('doubts', '``', 'O'), ('that', '``', 'O'), ('the', '``', 'O'),
('elections', '``', 'O'), ('will', '``', 'O'), ('take', '``', 'O'), ('place', '``', 'O'), ('as', '``', 'O'), ('planned', '``', 'O'),
('on', '``', 'O'), ('January', '``', 'O'), ('16', '``', 'O'), ('.', '``', 'O'), ('China', '``', 'O'), ('has', '``', 'O'), ('been', '``',
'O'), ('struck', '``', 'O'), ('by', '``', 'O'), ('violent', '``', 'O'), ('protests', '``', 'O'), ('in', '``', 'O'), ('recent', '``', 'O'),
('months', '``', 'O'), ('as', '``', 'O'), ('rural', '``', 'O'), ('villagers', '``', 'O'), ('vent', '``', 'O'), ('anger', '``', 'O'),
('over', '``', 'O'), ('industrial', '``', 'O'), ('pollution', '``', 'O'), ('and', '``', 'O'), ('allegations', '``', 'O'), ('of', '``',
'O'), ('corruption', '``', 'O'), ('and', '``', 'O'), ('unfair', '``', 'O'), ('land', '``', 'O'), ('distribution', '``', 'O'), ('.', '``',
'O')]
Viterbi Algorithm Accuracy:  20.0
[('Thousands', 'NNS', 'O'), ('of', 'IN', 'O'), ('demonstrators', 'NNS', 'O'), ('have', 'VBP', 'O'), ('marched',
'VBN', 'O'), ('through', 'IN', 'O'), ('London', 'NNP', 'B-geo'), ('Saturday', 'NNP', 'B-tim'), ('An', 'DT', 'O')]
```

Conclusion

The provided code for joint Part-of-Speech (POS) and Named Entity Recognition (NER) tagging using the Viterbi algorithm demonstrates a foundational approach to sequence labeling tasks. However, several limitations affect its overall performance and generalizability:

- **Data Sparsity and Handling Unknown Words:** The model may struggle with unseen words and rare tags due to data sparsity. Using a constant for unknown words partially addresses this but may not fully mitigate the issue.
- **Computational Complexity:** The algorithm's complexity increases with the number of tags, which can lead to high memory and computation requirements for large tag sets.
- **Probability Estimation and Overfitting:** Transition and emission probabilities derived from training data may not generalize well to unseen data. The lack of regularization can lead to overfitting, affecting the model's performance on new data.
- **Evaluation and Validation:** The evaluation method used is limited and might not provide a comprehensive measure of the model's effectiveness. The absence of a validation process may also hinder model tuning and optimization.
- **Advanced Features:** The model's performance is constrained by its reliance on basic features. Incorporating more sophisticated techniques, such as word embeddings or additional linguistic resources, could enhance its capabilities.

In summary:
While the Viterbi algorithm implementation provides a good starting point for joint POS and NER tagging, addressing its limitations is crucial for improving accuracy, efficiency, and generalizability. Enhancements such as smoothing techniques, model validation, regularization, and incorporating advanced features can lead to a more robust and effective model, better suited for real-world applications.

Thank
You