Named Entity Recognition (NER) - Complete Guide

What is Named Entity Recognition?

Named Entity Recognition (NER) is a natural language processing technique that identifies and classifies named entities in text into predefined categories such as person names, organizations, locations, dates, and more. It's a crucial component in information extraction and text understanding systems.

NER Pipeline Steps

Raw Text Input
↓
Text Preprocessing
(Tokenization, Normalization)
↓
Feature Extraction
(Word embeddings, POS tags, etc.)
↓
Entity Detection
(Identify potential entities)
↓
Entity Classification
(Assign entity types)
↓
Post-processing
(Validation, Correction)
↓
Structured Output
(Labeled entities with types)

Detailed Steps:

1. **Text Preprocessing**: Clean and tokenize the input text
2. **Feature Extraction**: Extract relevant features for each token
3. **Sequence Labeling**: Use models to predict entity labels
4. **Entity Boundary Detection**: Identify start and end of entities
5. **Entity Classification**: Assign appropriate category labels
6. **Post-processing**: Validate and refine results

Categories of Named Entities

Entity Type	Code	Description	Examples
Person	PER/PERSON	Names of people	"John Smith", "Marie Curie"
Organization	ORG	Companies, institutions	"Google", "Harvard University"
Location	LOC/GPE	Geographic locations	"New York", "Mount Everest"
Date	DATE	Dates and times	"January 15, 2024", "yesterday"
Money	MONEY	Monetary values	"$100", "€50"
Percentage	PERCENT	Percentage values	"25%", "half"
Facility	FAC	Buildings, airports, highways	"Brooklyn Bridge", "JFK Airport"
Product	PRODUCT	Objects, vehicles, foods	"iPhone", "Toyota Camry"
Event	EVENT	Named hurricanes, battles	"World War II", "Hurricane Katrina"
Work of Art	WORK_OF_ART	Titles of books, songs	"The Great Gatsby", "Bohemian Rhapsody"
Language	LANGUAGE	Named languages	"English", "Spanish"
Nationality	NORP	Nationalities, groups	"American", "Buddhist"

Python Programs for NER

Program 1: Basic NER with spaCy

```
import spacy

# Load the English language model
nlp = spacy.load("en_core_web_sm")

def extract_entities(text):
    """Extract named entities from text using spaCy"""
    doc = nlp(text)
    entities = []

    for ent in doc.ents:
        entities.append({
            'text': ent.text,
            'label': ent.label_,
            'description': spacy.explain(ent.label_),
            'start': ent.start_char,
            'end': ent.end_char
        })

    return entities

# Example usage
sample_text = """
Apple Inc. was founded by Steve Jobs in Cupertino, California on April 1,
1976.
The company is now worth over $2 trillion and employs more than
150,000 people worldwide.
"""

entities = extract_entities(sample_text)

print("Named Entities Found:")
print("-" * 50)
for entity in entities:
    print(f"Text: {entity['text']}")
```

```
    print(f"Label: {entity['label']} ({entity['description']})")
    print(f"Position: {entity['start']}-{entity['end']}")
    print("-" * 30)
```

Output:

```
Named Entities Found:
--------------------------------------------------
Text: Apple Inc.
Label: ORG (Companies, agencies, institutions, etc.)
Position: 1-10
------------------------------
Text: Steve Jobs
Label: PERSON (People, including fictional)
Position: 26-36
------------------------------
Text: Cupertino
Label: GPE (Countries, cities, states)
Position: 40-49
------------------------------
Text: California
Label: GPE (Countries, cities, states)
Position: 51-61
------------------------------
Text: April 1, 1976
Label: DATE (Absolute or relative dates or periods)
Position: 65-78
------------------------------
Text: over $2 trillion
Label: MONEY (Monetary values, including unit)
Position: 108-123
------------------------------
Text: more than 150,000
Label: CARDINAL (Numerals that do not fall under another type)
Position: 136-152
------------------------------
```

Program 2: Resume Parser using NER

```
import spacy
import re
from collections import defaultdict

class ResumeParser:
    def __init__(self):
        self.nlp = spacy.load("en_core_web_sm")

    def extract_contact_info(self, text):
        """Extract contact information from resume text"""
        # Email extraction
        email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-
                                    z]{2,}\b'
        emails = re.findall(email_pattern, text)

        # Phone extraction
        phone_pattern = r'(\+\d{1,3}[-.\s]?)?\(?\d{3}\)?[-
                                            .\s]?\d{3}[-.\s]?\d{4}'
        phones = re.findall(phone_pattern, text)

        return {
            'emails': emails,
            'phones': [phone[1] if phone[0] else phone for phone in phones]
        }

    def extract_entities_by_type(self, text):
        """Extract and categorize named entities"""
        doc = self.nlp(text)
        entities = defaultdict(list)

        for ent in doc.ents:
            entities[ent.label_].append(ent.text)

        return dict(entities)

    def parse_resume(self, resume_text):
        """Main function to parse resume"""
        contact_info = self.extract_contact_info(resume_text)
        entities = self.extract_entities_by_type(resume_text)
```

```
        return {
            'contact_info': contact_info,
            'organizations': entities.get('ORG', []),
            'locations': entities.get('GPE', []),
            'dates': entities.get('DATE', []),
            'skills_mentioned': entities.get('PRODUCT', []),
            'all_entities': entities
        }

# Example usage
sample_resume = """
John Doe
Software Engineer
Email: john.doe@email.com
Phone: (555) 123-4567

Experience:
Software Engineer at Google Inc. (2020-2023)
- Developed applications using Python and JavaScript
- Worked on machine learning projects

Education:
Bachelor of Science in Computer Science
Stanford University (2016-2020)
Located in Palo Alto, California

Skills: Python, Java, React, TensorFlow
"""

parser = ResumeParser()
results = parser.parse_resume(sample_resume)

print("Resume Parsing Results:")
print("=" * 50)
print(f"Contact Info: {results['contact_info']}")
print(f"Organizations: {results['organizations']}")
print(f"Locations: {results['locations']}")
print(f"Dates: {results['dates']}")
```

Program 3: NER Visualization with spaCy

```
import spacy
from spacy import displacy

def visualize_entities(text, style="ent"):
    """Visualize named entities in text"""
    nlp = spacy.load("en_core_web_sm")
    doc = nlp(text)

    # Generate HTML visualization
    html = displacy.render(doc, style=style, jupyter=False, page=True)

    # Save to file
    with open("ner_visualization.html", "w", encoding="utf-8") as f:
        f.write(html)

    print("Visualization saved to 'ner_visualization.html'")

    # Print entity information
    print("\nEntity Analysis:")
    print("-" * 40)
    for ent in doc.ents:
        print(f"{ent.text:20} | {ent.label_:10} | {spacy.explain(ent.label_)}")

# Example usage
news_text = """
Breaking News: Tesla CEO Elon Musk announced on Twitter that the
company
will build a new Gigafactory in Austin, Texas by December 2024.
The $5 billion investment will create 10,000 jobs and produce
500,000 vehicles annually.
"""

visualize_entities(news_text)
```

Output

Entity Analysis:

Elon Musk	PERSON	People, including fictional
Twitter	PERSON	People, including fictional
Austin	GPE	Countries, cities, states
Texas	GPE	Countries, cities, states
December 2024	DATE	Absolute or relative dates or periods
$5 billion	MONEY	Monetary values, including unit
10,000	CARDINAL	Numerals that do not fall under another type
500,000	CARDINAL	Numerals that do not fall under another type
annually	DATE	Absolute or relative dates or periods

NER Use Cases in Daily Life

Business Applications:

- **Customer Service**: Automatically extract customer names, account numbers, and issues
- **Email Processing**: Identify important contacts, dates, and action items
- **Document Management**: Auto-tag documents with companies, people, and locations

Content & Media:

- **News Analysis**: Track mentions of politicians, companies, and events
- **Social Media Monitoring**: Identify trending people, places, and topics
- **Content Recommendation**: Suggest related articles based on extracted entities

Healthcare:

- **Medical Records**: Extract patient names, medications, and treatment dates
- **Clinical Research**: Identify study participants and medical conditions

- **Insurance Processing**: Extract policy numbers, dates, and claim amounts

Finance:

- **Transaction Processing**: Identify merchants, amounts, and dates
- **Fraud Detection**: Flag unusual patterns in entity mentions
- **Investment Research**: Track company mentions and financial figures

Personal Applications:

- **Email Organization**: Auto-sort emails by sender, company, or topic
- **Calendar Management**: Extract meeting participants and locations
- **Travel Planning**: Identify destinations, dates, and booking references