# Fine-Tuning spaCy NER for E-Commerce Data

---

## 🔹 Step-by-Step Process for Fine-Tuning spaCy NER

|  |  |  |
| --- | --- | --- |
| Step | Description | Complexity |
| 1. Install Dependencies | Install `spaCy` and required libraries | Automated |
| 2. Prepare Training Data | Convert raw text into JSON format with `PRODUCT`, `MONEY`, and `MEMORY` entities | Manual |
| 3. Create Training Script | Write a Python script to load and train the model | Manual |
| 4. Train the Model | Fine-tune `en\_core\_web\_md` with custom data | Automated |
| 5. Save & Test the Model | Save trained model and run evaluation | Automated |
| 6. Evaluate Performance | Measure precision, recall, and F1-score | Automated |

---

## 🔹 1. Install Dependencies

Run the following command to install necessary packages:

```bash  
pip install spacy  
pip install -U spacy[transformers]  
```

## 🔹 2. Prepare Training Data

We format the dataset in a way that spaCy can understand:

📌 Sample Training Data (JSON Format)

```python  
TRAIN\_DATA = [  
 ("I just bought a Samsung Galaxy S24 for 10k with 8GB RAM.",  
 {"entities": [(17, 36, "PRODUCT"), (41, 44, "MONEY"), (50, 56, "MEMORY")]}),  
 ("The iPhone 15 Pro costs $1200 and has 16GB RAM.",  
 {"entities": [(4, 18, "PRODUCT"), (26, 30, "MONEY"), (40, 46, "MEMORY")]}),  
 ("Asus ROG Phone 7 comes with 12GB RAM and is priced at 900 USD.",  
 {"entities": [(0, 18, "PRODUCT"), (26, 32, "MEMORY"), (48, 51, "MONEY")]})]  
```

## 🔹 3. Create Training Script

```python  
import spacy  
from spacy.training.example import Example  
  
# Load base model  
nlp = spacy.load("en\_core\_web\_md")  
  
# Get the Named Entity Recognizer  
ner = nlp.get\_pipe("ner")  
  
# Add new labels  
for \_, annotations in TRAIN\_DATA:  
 for ent in annotations["entities"]:  
 ner.add\_label(ent[2])  
  
# Disable other pipelines to only train NER  
other\_pipes = [pipe for pipe in nlp.pipe\_names if pipe != "ner"]  
with nlp.disable\_pipes(\*other\_pipes):  
 optimizer = nlp.resume\_training()  
 for epoch in range(30): # Adjust epochs as needed  
 for text, annotations in TRAIN\_DATA:  
 example = Example.from\_dict(nlp.make\_doc(text), annotations)  
 nlp.update([example], drop=0.5, losses={})  
  
# Save trained model  
nlp.to\_disk("./ecommerce\_ner\_model")  
```

## 🔹 4. Test the Model

```python  
# Load trained model  
nlp\_test = spacy.load("./ecommerce\_ner\_model")  
  
test\_text = "The new Samsung Galaxy S24 Ultra has 16GB RAM and costs 1200 USD."  
doc = nlp\_test(test\_text)  
  
for ent in doc.ents:  
 print(f"Entity: {ent.text}, Label: {ent.label\_}")  
```

📝 Expected Output:

```  
Entity: Samsung Galaxy S24 Ultra, Label: PRODUCT  
Entity: 16GB RAM, Label: MEMORY  
Entity: 1200 USD, Label: MONEY  
```

## 🔹 5. Evaluate the Model

```python  
from spacy.scorer import Scorer  
from spacy.training.example import Example  
  
def evaluate\_model(nlp, examples):  
 scorer = Scorer()  
 for input\_text, annotations in examples:  
 doc = nlp(input\_text)  
 example = Example.from\_dict(doc, annotations)  
 scorer.score(example)  
 return scorer.scores  
  
# Run evaluation  
print(evaluate\_model(nlp\_test, TRAIN\_DATA))  
```

## 📌 Conclusion

- We fine-tuned `spaCy` NER on e-commerce data for `PRODUCT`, `MONEY`, and `MEMORY`.  
- The model can now extract e-commerce-specific entities.  
- The process is scalable—you can add more entities like `COLOR`, `BRAND`, `STORAGE`, etc.  
  
Would you like a pre-trained model or additional fine-tuning help? 🚀