Named Entity Recognition Solutions Report

💡 **Author:**

- Name: Elina Akimchenkova
- Email: e.akimchenkova@innopolis.university
- CodaLab nick: akmchnkv
- Github: https://github.com/akmchnkv/NLP-IU-S24/tree/main/Assignment3

Solution 1: Dictionary-Based Entity Recognition

Implementation Details:

- **Approach**: This solution employs a dictionary-based approach to memorize entities and their corresponding labels from the training dataset. The primary goal is to leverage known entities during the testing phase by using their most common labels.

Pipeline Overview:

1. **Data Loading**: The training data is loaded from a `.jsonl` file, where each line is a separate JSON entry.
2. **Entity Extraction**: For each JSON in the training set:
   - Named entities (NERs) are extracted.
   - Tokens are derived from the spans of each NER, and their corresponding labels are stored in a `entity_label_count` dictionary.
3. **Tokenization and Labeling**:
   - The `razdel` tokenizer is used on the test set, providing tokens along with their start and end indexes.
   - Each token from the test sentences is checked against the `entity_label_count` dictionary to determine its most common label.
4. **Preprocessing**: Adjacent tokens with the same label are combined to form larger entities
5. **Output**: Entities and their labels are saved in `test.jsonl`.

Solution 2: spaCy Model Fine-Tuning

Implementation Details:

- **Approach**: This solution involves fine-tuning the `ru_core_news_lg` spaCy model on our training dataset, focusing on improving the model's ability to recognize and classify named entities accurately.

Pipeline Overview:

1. **Data Preparation**:
   - NERs are extracted from `train.jsonl` and collected into a set named `ent_types`.
   - Data is formatted as: `[('sentence', {'entities': [(start, end, label)...]})...]`.
   - Intersecting spans are removed to simplify the training process.
2. **Model Initialization and Training**:
   - The `ru_core_news_lg` spaCy model is loaded and initialized.
   - The training data is iterated through to create `Doc` objects from text, which are paired with entity offsets in `Example` objects.

- The `nlp.update` function is used to fine-tune the model using the prepared training data.

3. **Entity Recognition on Test Data**:
    - The fine-tuned model is employed to predict entities for each test sentence.
    - Recognized entities and their labels are stored in `test.jsonl`.
4. **Final Output**: The results are saved into `test.jsonl`

Analysis of Solutions

Feature	Solution 1: Dictionary-Based NER	Solution 2: spaCy Model Fine-Tuning
Implementation Complexity	Simple to implement and maintain.	More complex, requires understanding of NLP model training and fine-tuning.
Computational Requirements	Low. Uses simple dictionary lookups.	High. Requires significant computational resources for training and inference.
Accuracy and Contextual Awareness	Limited. Relies on static token to label mapping without context.	High. Leverages context and adapts to new, unseen data better.
Speed and Efficiency	Fast predictions due to simple dictionary access.	Slower, especially during training. Inference speed is optimized but generally slower than dictionary lookups.
Scalability and Generalization	Does not scale well with new or unseen data.	Scales well with diverse and complex data, learning continuously from new examples.
Error Handling	Errors in training data directly affect predictions. No mechanism to correct or learn from errors.	More robust to errors in training data due to contextual understanding and learning capabilities.