

測試結果準確度評分機制說明

評分目標

主要目標

- **量化評估**: 為AI模型在身心障礙手冊識別任務上的表現提供客觀的數值評估
- **欄位級分析**: 對每個關鍵欄位進行獨立的準確度評估
- **記錄級分析**: 對每筆記錄的整體識別表現進行評估
- **比較分析**: 支援多個AI模型之間的表現比較

應用場景

- AI模型訓練效果驗證
- 不同模型間的性能比較
- 識別系統的品质控制
- 模型優化的方向指引

評分維度

1. 欄位級準確度評估




評分方法

每個欄位使用以下方法進行評估：

1. 文字相似度計算

```
from difflib import SequenceMatcher
similarity = SequenceMatcher(None, correct_value,
predicted_value).ratio()
```

2. 完全匹配判定

- 相似度 = 1.0 → 完全匹配 
- 相似度 ≥ 0.8 → 高度相似 
- 相似度 < 0.8 → 不匹配 

3. 準確度等級分類

- **優秀** (90-100%): 模型表現優異
- **良好** (70-89%): 模型表現良好，有少量錯誤
- **普通** (50-69%): 模型表現普通，需要改進
- **需改進** (0-49%): 模型表現不佳，需要重新訓練

2. 記錄級準確度評估

整體準確度計算

每筆記錄的整體準確度採用加權平均方式計算：

$$\text{整體準確度} = (\text{障礙等級準確度} + \text{障礙類別準確度} + \text{ICD診斷準確度}) / 3$$

完全匹配統計

- **完全匹配欄位數**: 相似度 = 1.0 的欄位數量
- **總欄位數**: 參與評估的欄位總數
- **匹配率**: 完全匹配欄位數 / 總欄位數

3. 整體系統評估

全域統計指標

- **總記錄數**: 參與評估的記錄總數
- **完全正確記錄數**: 所有欄位都完全匹配的記錄數
- **完全正確率**: 完全正確記錄數 / 總記錄數
- **平均準確度**: 所有記錄準確度的平均值

技術實現

核心演算法

1. OCR專用評估指標(備用)

本系統採用業界標準的OCR評估指標，提供更準確的文字識別性能評估：

CER（字元錯誤率，Character Error Rate）

```
def calculate_cer(self, reference: str, hypothesis: str) -> float:
    """
    計算字元錯誤率（Character Error Rate, CER）
    CER = (S + D + I) / N
    其中 S=替換, D=刪除, I=插入, N=參考文字字元數
    """
    # 使用編輯距離計算字元級錯誤
    edits = difflib.SequenceMatcher(None, ref_norm,
                                     hyp_norm).get_opcodes()

    substitutions = deletions = insertions = 0
    for op, i1, i2, j1, j2 in edits:
        if op == 'replace':
            substitutions += max(i2 - i1, j2 - j1)
        elif op == 'delete':
            deletions += i2 - i1
        elif op == 'insert':
```

```

        insertions += j2 - j1

    total_errors = substitutions + deletions + insertions
    return total_errors / len(ref_norm) if len(ref_norm) > 0 else 0.0

```

WER (單詞錯誤率, Word Error Rate)

```

def calculate_wer(self, reference: str, hypothesis: str) -> float:
    """
    計算單詞錯誤率 (Word Error Rate, WER)
    對於中文，將每個字符視為一個"單詞"
    """
    # 對於中文文字，將每個字符視為一個單詞
    ref_words = list(ref_norm)
    hyp_words = list(hyp_norm)

    # 使用編輯距離計算單詞級錯誤
    # ... 類似CER的計算邏輯

```

2. 準確度計算

```

def calculate_similarity(self, text1: str, text2: str) -> float:
    """
    計算兩個文字的相似度 (基於CER的準確度)
    準確度 = 1 - CER
    """
    cer = self.calculate_cer(text1, text2)
    return max(0.0, 1.0 - cer)

```

3. 文字相似度計算 (目前)

```

def calculate_similarity(correct_value: str, predicted_value: str) -> float:
    """
    使用SequenceMatcher計算文字相似度

    Args:
        correct_value: 正確答案
        predicted_value: AI預測結果

    Returns:
        float: 相似度分數 (0.0 - 1.0)
    """
    if pd.isna(correct_value) and pd.isna(predicted_value):
        return 1.0
    if pd.isna(correct_value) or pd.isna(predicted_value):

```

```
        return 0.0

    correct_str = str(correct_value).strip()
    predicted_str = str(predicted_value).strip()

    if not correct_str and not predicted_str:
        return 1.0
    if not correct_str or not predicted_str:
        return 0.0

    return SequenceMatcher(None, correct_str, predicted_str).ratio()
```




視覺化指標

OCR性能等級指標

基於CER（字元錯誤率）的性能分級：

- 優秀 (CER ≤ 5%): 頂級OCR性能
- 良好 (CER ≤ 10%): 優質OCR性能
- 可接受 (CER ≤ 20%): 可用OCR性能
- 需改進 (CER ≤ 40%): 需要優化
- 不合格 (CER > 40%): 需要重新訓練

狀態圖示系統

-  完全匹配 (CER = 0%)
-  部分匹配 (CER ≤ 20%)
-  不匹配 (CER > 20%)

表現等級色彩編碼

- 優秀 (90-100%): 綠色
- 良好 (70-89%): 黃色
- 普通 (50-69%): 橙色
- 需改進 (0-49%): 紅色