**1. Title & Abstract**

* **Title:**

*Standardizing Clinical Symptom Terms with Human Phenotype Ontology (HPO) in Python*

* **Abstract (short paragraph at top):**

Introduce the challenge: free-text symptom reporting in rare-disease literature → hard to standardize.

State the goal: demonstrate an automated pipeline that maps text to HPO terms with verification.

Highlight the outcome: structured table of symptoms with IDs, definitions, and ontology hierarchy.

**2. Background**

* **Problem Context**
  + Non-standard symptom reporting in clinical studies.
  + Why HPO is useful for standardization.
  + Real-world application (e.g., CMS literature review).
* **Objectives**
  + Map free-text symptoms to HPO terms.
  + Validate mappings with fuzzy similarity + synonyms.
  + Retrieve lineage and definitions for context.

**3. Workflow Overview**

* **Diagram or numbered list** of steps (already present in text):
  1. Retrieve HPO candidates from API.
  2. Compute fuzzy match score.
  3. Check synonyms and definitions.
  4. Extract lineage path.
  5. Accept/reject decision rule.

*(Tip: Add a small schematic figure or Markdown flowchart here for clarity.)*

**4. Environment Setup**

* **Imports & Dependencies**
  + General modules (requests, pandas, time).
  + Fuzzy matching (RapidFuzz preferred; FuzzyWuzzy fallback).
  + Ontology handling (obonet, lru\_cache).
* **Notes on Libraries**
  + Why RapidFuzz (fast, permissive license).
  + Why caching (lru\_cache) matters for repeated queries.

**5. Implementation Sections**

Break down the code cells with **clear subtitles**:

* **Step 1. Candidate Retrieval (API query with retries)**

Show map\_symptoms\_to\_hpo() and explain retry/backoff logic.

* **Step 2. Fuzzy Matching**

Explain RapidFuzz vs FuzzyWuzzy; show estimate\_fuzzy\_score().

* **Step 3. Synonym & Definition Lookup**

Show cached ontology loading, cleaning of OBO text.

* **Step 4. Ontology Lineage Extraction**

Explain how rank/path to root are derived. Mention DAG simplification.

* **Step 5. Synonym Matching Logic**

Demonstrate exact + fuzzy match against synonyms.

* **Step 6. Candidate Scoring & Best Selection**

Show \_score\_candidate() and \_choose\_best\_candidate().

* **Step 7. End-to-End Pipeline**

map\_symptoms\_to\_hpo\_pipeline() combining all steps.

**6. Demonstration**

* **Quick Example**

Use 2–3 sample symptoms (ptosis, weak suck, exercise intolerance).

Display results in a styled pandas.DataFrame with clear columns.

* **Optional Debug View**

Show return\_debug=True mode to inspect candidate ranking.

**7. Results & Observations**

* Show the final output table.
* Discuss:
  + Cases that map well (high fuzzy score, correct synonym).
  + Ambiguous cases (e.g., “weak suck” → “Weak cry”).
  + How synonym/fuzzy thresholds affect status.

**8. Limitations & Improvements**

* API reliability (timeouts, availability).
* Only first parent used in lineage.
* Threshold tuning needed for different datasets.
* Future directions:
  + Evaluate **top-K candidates** (already implemented).
  + Use semantic embeddings for trickier matches.
  + Crosswalk to other vocabularies (MeSH, UMLS).

**9. Conclusion**

* Summarize pipeline’s value: reproducible, structured mapping.
* Mention direct applicability to rare-disease evidence synthesis.
* Note extensibility for other biomedical text-mining tasks.

**10. Appendix**

* **References & Links**
  + HPO API: <https://ontology.jax.org/api/hp/search/>
  + HPO Ontology file: <http://purl.obolibrary.org/obo/hp.obo>
* **Glossary**
  + *HPO ID*: unique identifier for phenotype.
  + *Fuzzy score*: similarity between two strings, 0–100 scale.
  + *Synonym override*: acceptance rule if input matches a known synonym.