Standardizing Clinical Symptoms of Rare Disease with Human Phenotype Ontology (HPO) in Python

Background

In literature reviews and evidence synthesis for rare diseases, clinical symptoms are often reported in non-standardized ways, making it difficult to compare or merge them computationally.

This real-world challenge motivated us to develop a data solution for standardizing free-text symptom reports. Using the open-source Human Phenotype Ontology (HPO) and its API, we can map reported symptoms to controlled ontology terms, with the process automated in Python.

I streamlined the workflow into several steps: retrieving candidate HPO terms and synonyms, linking them to IDs and definitions, and applying fuzzy matching to identify similarities between reported symptoms and retrieved HPO terms. This notebook demonstrates the pipeline with minimal documentation as a reference for the community.

The workflow has been tested in a real-world project on congenital myasthenic syndromes (CMS). While effective, there remain opportunities to refine and expand the approach.

Alogrithm explained

Goal Standardize free-text clinical symptoms by mapping them to HPO (Human Phenotype Ontology) terms, then verify and contextualize each match.

Inputs: symptom (str): A reported, free-text symptom (e.g., “ptosis”, “muscle weakness”).

External resources & libs

- Search API: https://ontology.jax.org/api/hp/search/?q= (top result taken)
- Ontology file: http://purl.obolibrary.org/obo/hp.obo (definitions, synonyms, hierarchy)
- Python libs: requests, fuzzywuzzy.process.extractOne, obonet, functools.lru_cache, pandas (optional)

High-level flow

1. Search HPO: Query the JAX HPO API with the input symptom → get top candidate (name, id) or no result.
2. Fuzzy validation: Compute a fuzzy score between the input symptom and the returned HPO term name.
3. Context retrieval: From hp.obo, pull definition and synonyms for the candidate HPO ID.

4. Lineage extraction: From the same ontology graph, compute depth and path to root HP:0000001 (using first parent if multiple).
5. Accept/Reject decision - Accept if fuzzy_score ≥ 80 OR HPO name appears in its synonyms (case-insensitive check). - Reject otherwise (or if API/ontology lookup fails).

Decision rule (acceptance)

- Threshold: fuzzy_score ≥ 80
- Synonym override: Accept if HPO term name is present among its synonyms (case-insensitive)

Rationale: Puts speed/recall first (top hit) with a sanity check on similarity; adds semantic cushion via synonyms.

Outputs (as implemented)

The pipeline returns 8 fields in fixed order:

1. reported_symptom (str)
2. hpo_term (str | None)
3. hpo_id (str | None)
4. definition (str | None)
5. rank (int | None)
6. path (list[str] | [])
7. fuzzy_score (int | 0)
8. status ("matched" | "not matched")

Load necessary modules

In [ ]:
```
# General modules
import requests
import pandas as pd

# Specific modules
from fuzzywuzzy import process
import obonet
from functools import lru_cache
```

```
/opt/anaconda3/lib/python3.12/site-packages/fuzzywuzzy/fuzz.py:11: UserWarning: U
sing slow pure-python SequenceMatcher. Install python-Levenshtein to remove this
warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshte
in to remove this warning')
```

Brief information for the specific modules

- `fuzzywuzzy.process`: Provides fuzzy string matching, useful for comparing free-text symptoms to HPO terms and synonyms.
- `obonet`: Loads and parses OBO-formatted ontology files, such as the Human Phenotype Ontology, into network structures.

- `functools.lru_cache`: Decorator for caching function results, improving performance when repeatedly querying or processing the same data.

Implementation

Step 1: Map reported symptoms to HPO terms using the HPA API

In [ ]:
```python
import requests

def map_symptoms_to_hpo(symptom):
    """
    Map reported symptoms to HPO terms using the HPA API.
    """
    url = f"https://ontology.jax.org/api/hp/search/?q={symptom}"
    try:
        response = requests.get(url, timeout=10)
        response.raise_for_status()  # raises HTTPError if not 200 OK
        json_data = response.json()
        results = json_data.get('terms', [])
        if results:
            top = results[0]  # Take top result
            return (top["name"], top["id"])  # return matched term and HPO ID as
        else:
            return (None, None)
    except requests.exceptions.RequestException as e:
        # print(f"Request failed for term '{symptom}': {e}")
        return (None, None)
    except ValueError as ve:
        # print(f"Invalid JSON for term '{symptom}': {ve}")
        return (None, None)
```

In [4]:
```python
# Example usage:
symptom = "headache"
print(map_symptoms_to_hpo(symptom))
```

```
('Headache', 'HP:0002315')
```

Step 2: Estimate the fuzzy score between the input term and the HPO term

In [ ]:
```python
from fuzzywuzzy import process
import pandas as pd

def estimate_fuzzy_score(input_term, hpo_term):
    """
    Estimate the fuzzy score between the input term and the HPO term.

    Parameters:
    input_term: str
        The term reported in the study.
    hpo_term: str
        The term from the HPO database.
    Returns:
    fuzzy_score: float
        The fuzzy score between the input term and the HPO term.
```

```
    """
    if not isinstance(input_term, str):
        raise ValueError("reported_term must be a string.")

    best_match_fuzzy, fuzzy_score = process.extractOne(input_term, [hpo_term])
    return fuzzy_score
```

In [ ]:
```
# Example usage:
input_term = "headache"
hpo_term = "Headache"
print(estimate_fuzzy_score(input_term, hpo_term))
```

```
100
```

Step 3: Look up HPO synonyms and definition

In [ ]:
```
import obonet
def get_hpo_definitions_and_synonyms(hpo_id):
    """
    Get the definition and synonyms for a given HPO term ID.
    """
    url = 'http://purl.obolibrary.org/obo/hp.obo' # URL points to the Human Phenc
    graph = obonet.read_obo(url)

    if hpo_id in graph.nodes:
        synonyms = graph.nodes[hpo_id].get('synonyms', [])
        definition = graph.nodes[hpo_id].get('def', 'NA')
        return synonyms, definition
    else:
        return None, None
```

In [ ]:
```
# Example usage:
hpo_id = "HP:0002315"
synonyms, definition = get_hpo_definitions_and_synonyms(hpo_id)
print(f"Synonyms: {synonyms}")
print(f"Definition: {definition}")
```

```
Synonyms: []
Definition: "Cephalgia, or pain sensed in various parts of the head, not confined
to the area of distribution of any nerve." [https://orcid.org/0000-0002-0736-919
9, PMID:15304572]
```

Step 4: Get full lineage for a given HPO ID

In [ ]:
```
import obonet
from functools import lru_cache

HPO_URL = "http://purl.obolibrary.org/obo/hp.obo"

# Cache the graph so it loads only once
@lru_cache(maxsize=1)
def load_graph():
    return obonet.read_obo(HPO_URL)

def get_rank_and_path(hpo_id):
```

```
    """
    Return rank and path from root to this term (shortest path).
    """
    graph = load_graph()
    if hpo_id not in graph:
        return None, []

    path = [hpo_id]
    depth = 0
    current = hpo_id
    while True:
        parents = graph.nodes[current].get("is_a", [])
        if not parents:
            break
        current = parents[0]  # take first parent if multiple
        path.append(current)
        depth += 1
        if current == "HP:0000001":
            break
    return depth, list(reversed(path))
```

In [11]:
```
# Example usage:
hpo_id = "HP:0002315"
rank, lineage = get_rank_and_path(hpo_id)
print(f"Rank: {rank}")
print(f"Lineage: {lineage}")
```

```
Rank: 4
Lineage: ['HP:0000001', 'HP:0000118', 'HP:0000707', 'HP:0012638', 'HP:0002315']
```

Pipeline function to chain step 1-4

In [ ]:
```
def map_symptoms_to_hpo_pipeline(symptom):
    """
    Map reported symptoms to HPO terms and get synonyms and definitions.

    Parameters:
    symptom: str
        The term reported in the study.

    Returns:
    hpo_term: str or None
        The term from the HPO database.
    hpo_id: str or None
        The ID of the term from the HPO database.
    fuzzy_score: float
        The fuzzy score between the input term and the HPO term (0 if no match).
    definition: str or None
        The definition of the HPO term.
    rank: int or None
        The rank (depth) of the HPO term in the ontology.
    path: list of str
        The list of HPO IDs representing the path from the root to this term.
    status: str
        The status of the mapping ('matched' or 'not matched').
    """
    # Step 1: Map reported symptoms to HPO terms
    hpo_term, hpo_id = map_symptoms_to_hpo(symptom)
```

```
    # If no match found, return immediately
    if hpo_term is None or hpo_id is None:
        return symptom, None, None, 0, 'not matched'

    # Step 2: Estimate fuzzy score
    fuzzy_score = estimate_fuzzy_score(symptom, hpo_term)

    # Step 3: Get HPO definitions and synonyms
    synonyms, definition = get_hpo_definitions_and_synonyms(hpo_id)

    # Step 4: Get full lineage
    rank, path = get_rank_and_path(hpo_id)
    # print(f"Rank: {rank}, Path: {path}")

    # Step 5: Check if match is acceptable
    if fuzzy_score >= 80 or (hpo_term.lower() in [s.lower() for s in synonyms]):
        return symptom, hpo_term, hpo_id, definition, rank, path, fuzzy_score, 'm
    else:
        return symptom, hpo_term, hpo_id, definition, rank, path, fuzzy_score, 'r
```

In [13]:
```
# Example usage:
result = map_symptoms_to_hpo_pipeline("headache")
print(result)
```

```
('headache', 'Headache', 'HP:0002315', '"Cephalgia, or pain sensed in various par
ts of the head, not confined to the area of distribution of any nerve." [https://
orcid.org/0000-0002-0736-9199, PMID:15304572]', 4, ['HP:0000001', 'HP:0000118',
'HP:0000707', 'HP:0012638', 'HP:0002315'], 100, 'matched')
```