Algorithm 1 Ensemble Voting of User Genome from GTDB to NCBI

Input: Initialize parsing genome GTDB-NCBI-r203 mapping metadata with training genome capacity $\mathcal{N}$
Input: Initialize reading user genome Q and tree T classified with GTDB-Tk
Fixed ensemble weights θ, θ' to constant 0.5

for $genome = 1, N$ **do**
  Associating genome accession ID gid with NCBI taxonomy in Dict $\mathcal{D}_{ncbi\_taxa}$
  Associating representative genome ID rid with GTDB taxonomy in Dict $\mathcal{D}_{gtdb\_taxa}$
  Clustering genome gid into representative genome rid in Dict $\mathcal{D}_{cluster}$
  Associating specific $taxon$ with NCBI taxonomy lineage in Dict $\mathcal{D}_{ncbi\_lineage}$
  Storing $\mathcal{D}_{ncbi\_taxa}$, $\mathcal{D}_{gtdb\_taxa}$, $\mathcal{D}_{cluster}$ and $\mathcal{D}_{ncbi\_lineage}$
end for

Initializing GTDB cluster with NCBI-type taxonomy in Dict $\mathcal{N}4\mathcal{G}_{cluster}$
for $rid, gids$ in $\mathcal{D}_{ncbi\_taxa}$.items() **do**
  for $rank = r_{species}, r_{kindom}$ **do**
    Initializing hit list for $\mathcal{D}_{ncbi\_taxa}$ with $\mathcal{L}_{hit}$
    for ε in $gids$ **do**
      if ε in $\mathcal{D}_{ncbi\_taxa}$ **then**
        appending $\mathcal{D}_{ncbi\_taxa}[\varepsilon][rank]$ into $\mathcal{L}_{hit}$
      end if
    end for
    if $\mathcal{L}_{hit}$ is NOT empty **then**
      Counting element occurence in $\mathcal{L}_{hit}$
      Applying heapsort to $\mathcal{L}_{hit}$ and finding top 1 element
      $tax_{top}, count_{top} = \mathcal{HEAPSORT}(\mathcal{L}_{hit})$
      if $count_{top} >= \theta * \mathcal{L}_{hit}$.size() && tax_{top} is NOT unassigned **then**
        $\mathcal{N}4\mathcal{G}_{cluster}[rid] = \mathcal{D}_{ncbi\_lineage}[tax_{top}]$
        break
      end if
    end if
    if $\mathcal{N}4\mathcal{G}_{cluster}[rid]$ is NOT unassigned **then**
      Reporting representative genome cannot be converted to NCBI taxonomy
    end if
  end for
  Storing $\mathcal{N}4\mathcal{G}_{cluster}[rid]$
end for

Mapping training genome or user genome from GTDB taxonomy to NCBI taxonomy
Initializing final hit list as $\mathcal{L}_{hit_{final}}$
for $rank = r_{species}, r_{kindom}$ **do**
  if $\mathcal{D}_{gtdb\_taxa}[rank]$ is *unassigned* **then**
    Continue
  end if
  Initializing hit list for $\mathcal{D}_{ncbi\_taxa}$ with $\mathcal{L}_{hit_{genome}}$
  Mapping genome accession gid to tree leaf node $\mathcal{NODE}_{map}$
  traversing leaf nodes $\mathcal{NODE}_{map}$ to find NCBI decendants with $\mathcal{N}4\mathcal{G}_{cluster}[rid]$
  Storing $\mathcal{N}4\mathcal{G}_{cluster_{hit}}$ as list $\mathcal{L}_{cluster_{hit}}$
  for rid in $\mathcal{L}_{cluster_{hit}}$ **do**
    appending $\mathcal{N}4\mathcal{G}_{cluster}[rid][rank]$ into $\mathcal{L}_{hit_{genome}}$
    Counting element occurence in $\mathcal{L}_{hit_{genome}}$
    Applying heapsort to $\mathcal{L}_{hit_{genome}}$ and finding top 1 element
    tax_{top}, $count_{top} = \mathcal{HEAPSORT}(\mathcal{L}_{hit_{genome}})$
    if $count_{top} >= \theta' * \mathcal{L}_{hit_{genome}}$.size() && tax_{top} is NOT unassigned **then**
      $\mathcal{L}_{hit_{final}} = \mathcal{D}_{ncbi\_lineage}[tax_{top}]$
      break
    end if
  end for
  Storing $\mathcal{L}_{hit_{final}}$
end for

Algorithm 2 Rebuilding NCBI-Tree Denovo with Custom Taxonomy

Input: Custom taxonomy table with capacity $\mathcal{C}$ with column taxa as List $\mathcal{L}_{taxa7}$
Output: NCBI-TREE nodes.dmp and names.dmp for custom taxonomy

Initializing NCBI-Tree as Directed Acyclic Graph(DAG) with Dict $\mathcal{D}_{dag}$
Initializing vertices and edges in DAG with List $\mathcal{L}_{vertices}$ and $\mathcal{L}_{edges}$
Initializing taxa node ever seen with Dict $\mathcal{D}_{taxa_{seen}}$
Initializing taxa node ID with Dict $\mathcal{D}_{taxid_{node}}$

for $genome = 1, \mathcal{C}$ **do**
  for $\varepsilon, taxon$ in enumerate($\mathcal{L}_{taxa7}$) **do**
    Adding Vertex $taxon$ into $\mathcal{D}_{dag}$ when not in $\mathcal{D}_{dag}$, or nothing to do
    if $taxon$ NOT in $\mathcal{D}_{dag}$ **then**
      $\mathcal{D}_{dag}[taxon] = []$
      Updating Taxa Node ID with occurence order of incrementing by 1
      $\mathcal{D}_{taxid_{node}}[taxon] = \text{len}(\mathcal{D}_{taxid_{node}}) + 1$
    end if

    Updating Edges from root to leaf(RTL) in Linked List
    if $\varepsilon = 0$ **then**
      Adding Edge $taxon$ from root of $\mathcal{D}_{dag}$
      $\mathcal{D}_{dag}[root] = [taxon]$
    else
      Adding Edge $taxon$ to $taxon_{before}$ denoted with $\mathcal{L}_{taxa7}[\varepsilon - 1]$ of $\mathcal{D}_{dag}$
      $\mathcal{D}_{dag}[taxon_{before}] = [taxon]$
    end if
  end for
end for

Recursive DFS to generate node.dmp and names.dmp
Initializing nodes and names as List $\mathcal{L}_{name}$ and $\mathcal{L}_{node}$
DFS($taxon$, $\mathcal{L}_{name}$, $\mathcal{L}_{node}$)
for $taxon_{child}$ in $\mathcal{D}_{dag}[taxon]$ **do**
  if $taxon_{child}$ in $\mathcal{D}_{taxa_{seen}}$ **then**
    Continue
  end if
  $\mathcal{D}_{taxa_{seen}}[taxon_{child}] = flag$

  Appending nodes with $taxid, parent_id, rank, ...$ format
  $\mathcal{L}_{node}$.append($\mathcal{D}_{taxid_{node}}[taxon_{child}]$,$\mathcal{D}_{taxid_{node}}[taxon]$,$\mathcal{D}_{rank}[taxon_{child}]$,...)

  Appending names with $taxid, name, 'scientificname'$ format
  $\mathcal{L}_{name}$.append($\mathcal{D}_{taxid_{node}}[taxon_{child}]$,$taxon_{child}$,'scientific name')
  DFS($taxon_{child}$, $\mathcal{L}_{name}$, $\mathcal{L}_{node}$)
end for