MediaDB ER Pseudocode

```
ALGORITHM 1: Entity Resolution
INPUT: Unresolved_collection, Resolved_collection

Foreach entity $en$ in unresolved collection
  $top\_ents = $ FindTopEntities($en, Resolved\_collection$)
  $best\_match = $ FindBestEntity ($en, top\_ents$)
  if $best\_match$ ! = NULL then
    Merge $en$ and $best\_match$ and update in $Resolved\_collection$
  else
    Insert $en$ $into$ $Resolved\_collection$
  end
end
```

```
ALGORITHM 2: Finding the top ten entities (FindTopEntities)
Input: unresolved_entity, resolved_entities collection
Output: top_ten_entities
if $unresolved\_entity.type == Person$ then
  Filter by entities of $type = Person$
  Filter by entities that match any of the $aliases$ of unresolved_entity
  Get the top ten entities that match either the $title$ or $associatedEntities$ (atleast
   one property should match) of unresolved_entity
else if $unresolved\_entity.type == Company$ OR $unresolved\_entity.type == Organization$
 then
  Filter by entities of $type = Company$ or $type = Organization$
  Get the top ten entities that match (any of the $aliases$(must match)) and
   ($resolution$ (optional match))
else if $unresolved\_entity.type == Country$ then
  Filter by entities of $type = Country$
  Get the top ten entities that match (any of the $aliases$(must match)) and
   ($resolution$ (optional match))
else if $unresolved\_entity.type == Continent$ then
  Filter by entities of $type = Continent$
  Get the top ten entities that match (any of the $aliases$(must match)) and
   ($resolution$ (optional match))
else if $unresolved\_entity.type == City$ OR $unresolved\_entity.type == ProvinceOrState$
 then
  Filter by entities of $type = City$ or $type = ProvinceOrState$
  Get the top ten entities that match (any of the $aliases$(must match)) and
   ($resolution$ (optional match))
```

```
ALGORITHM 3: Finding the best matching entity (FindBestEntity)
Input: unresolved_entity, top_ten_entities
Output: best_match
best_match = null
foreach $en$ in top_ten_entities do
  if $unresolved\_entity.type == Person$ then
    if (fuzzyMatchPer(n1, n2) for every (n1, n2) in ($unresolved\_entity.aliases$,
     $en.aliases$)) OR (exactMatchPer(n1, n2) for every (n1, n2) in
          ($unresolved\_entities.aliases, en.aliases$)) AND titleAssocMatch(e1, e2) then
        best_match = en
    end
  else if $unresolved\_entity.type == Company$ OR $unresolved\_entity.type ==$ Organization then
     if $unresolved\_entity.resolution == en.resolution$ then
       best_match = en
     else
       if orgMatch(n1, n2) for every (n1, n2) in ($unresolved\_entity.aliases, en.aliases$) then
         best_match = en
       end
     end
  else if $unresolved\_entity.type == Country$ OR $unresolved\_entity.type == Continent$
   then
     if $unresolved\_entity.resolution == en.resolution$ then
       best_match = en
     else
       if ($unresolved\_entity.resolution ==$ NULL OR $en.resolution ==$ NULL) AND
          countryCityMatch($unresolved\_entity.stdName, en.stdName$) then
            best_match = en
       end
     end
  else if $unresolved\_entity.type == City$ OR $unresolved\_entity.type == ProvinceOrState$
    then
     if $unresolved\_entity.resolution == en.resolution$ then
       best_match = en
     else
      if countryCityMatch($unresolved\_entity.stdName, en.stdName$) then
         best_match = en
       end
     end
  if best_match != NULL then
     return best_match
  end
end
```

```
ALGORITHM 4: fuzzyMatchPer
Input: name1, name2
Output: doNamesMatch

wordList1 = list of words in name1 // "PM Narendra Modi"−> ["PM", "Narendra", "Modi"]
wordList2 = list of words in name2 // "PM Modi" −> ["PM", "Modi"]
Remove matching initials from wordList1 and wordList2 // ["Narendra", "Modi"], [ "Modi"]
Remove multi − letter words (a, b) if a == b OR (doublemetaphone(a) ==
          doublemetaphone(b)) OR (inital_letter_same(a, b) = true and (length(a) <
                          = 6 and levenshtein_dist(a, b) == 1) or (length(a)
                          > 6 and levenshtein_dist(a, b) == 2))
Remove an initial: I from wordList1 and multi − letter word: W from wordList2 if W
starts with I and vice − versa
if there is an unmatched element in both the lists then
    doNamesMatch = false
else
    doNamesMatch = true
end
```

```
ALGORITHM 5: titleAssocMatch
Input: entity1, entity2
Output: doNamesMatch

title1 = title of entity1
title2 = title of entity2
if jaro_winkler(title1, title2) < 0.88 then
  doNamesMatch = false
else
  doNamesMatch = AssocMatch(entity1[associatedEntities], entity2[associatedEntities])
end
```

```
ALGORITHM 6: AssocMatch
Input: assocEnt1, assocEnt2
Output: doNamesMatch

doNamesMatch = true
assocStr = concatenate $assocEnt1$ elements as space separated string
foreach $en$ in $assocEnt2$
  if fuzzywuzzy. fuzz. partial_ratio(assocStr, en[name]) < 70 then
    doNamesMatch = false
  end
end
```

```
ALGORITHM 7: exactMatchPer
Input: name1, name2
Output: doNamesMatch
wordList1 = list of words in name1
wordList2 = list of words in name2
if every word in wordList1 finds an exact match in wordList2 and vice − versa then
    doNamesMatch = true
else
    doNamesMatch = false
end
```

```
ALGORITHM 8: orgMatch
Input: name1, name2
Output: doNamesMatch

remove pvt|private|public|ltd|limited|inc|corp|corporation|industry|industries|enterprise
    from name1 and name2
if (name1 == name2) OR (name1 is an abbreviation of name2 or vice − versa) OR
(name1 is a substring of name2 or vice − versa) OR jaro_winkler(name1, name2) ≥ 0.9 then
    doNamesMatch = true
else
    doNamesMatch = false
end
```

```
ALGORITHM 9: countryCityMatch
Input: name1, name2
Output: doNamesMatch
remove northern|southern|eastern|western|north|south|east|west from city name
if (name1 == name2) OR (name1 is a substring of name2 or vice − versa) OR
      jaro_winkler(name1, name2) ≥ 0.9
 then
    doNamesMatch = true
else
    doNamesMatch = false
end
```