Assignment: Entity Linking

In this assignment, we will implement important similarity measures and features for entity linking: similarity and coherence.

1. You are given two sentences from the same document with person mentions identified (marked in bold):

(1) *Will they try to prop **Romney** up with a strong VP to help make him more palatable - or will they bring in a new face?*

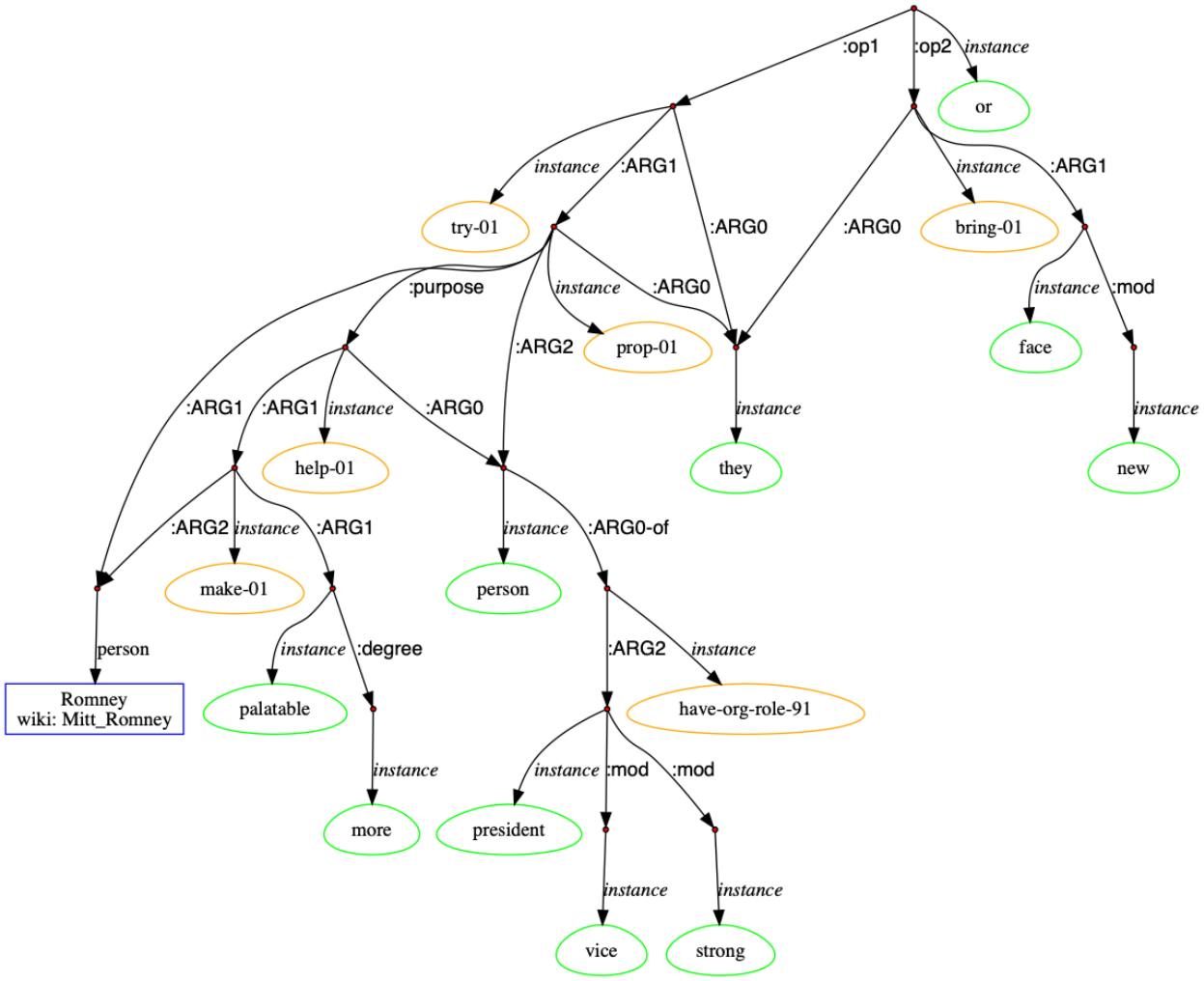
(2) *I don't think **Johnson** will tank, its going to be a close one till the very end, I am still rooting for a **Romney/McDonnell** ticket.*

2. You are also given AMR graphs for each of them:

(1)

```
(o / or
      :op1 (t / try-01
            :ARG0 (t2 / they)
            :ARG1 (p / prop-01
                  :ARG0 t2
                  :ARG1 (p2 / person :wiki "Mitt_Romney" :name (n /
name :op1 "Romney"))
                  :ARG2 (p3 / person
                        :ARG0-of (h / have-org-role-91
                              :ARG2 (p4 / president
                                    :mod (v / vice)
                                    :mod (s / strong))))
                  :purpose (h2 / help-01
                        :ARG0 p3
                        :ARG1 (m / make-01
                              :ARG1 (p5 / palatable
                                    :degree (m2 / more))
                              :ARG2 p2))))
      :op2 (b / bring-01
            :ARG0 t2
            :ARG1 (f / face
                  :mod (n2 / new))))
```

Visualized form:

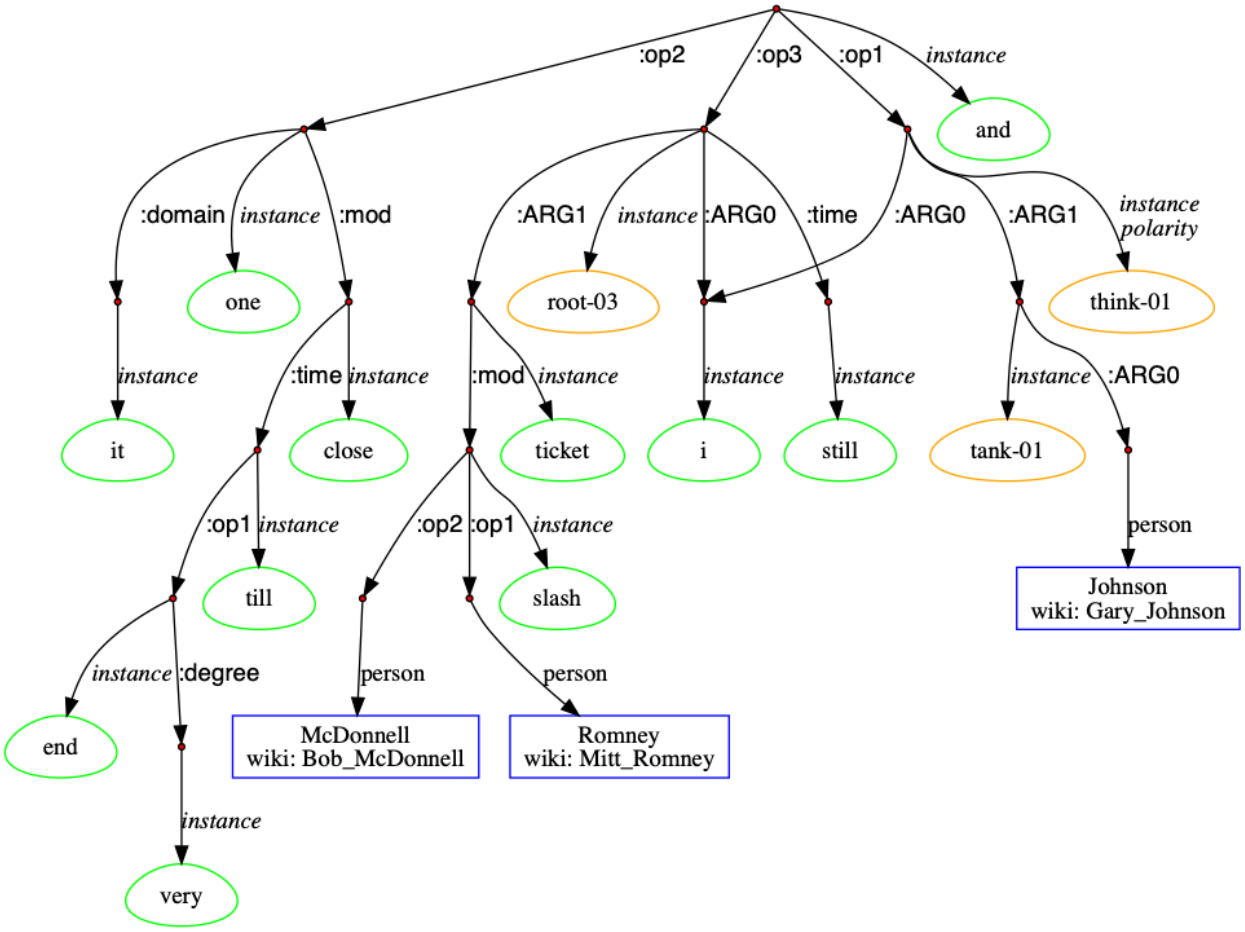


(2)

```
(a / and
      :op1 (t / think-01 :polarity -
            :ARG0 (i / i)
            :ARG1 (t2 / tank-01
                  :ARG0 (p / person :wiki "Gary_Johnson"
                        :name (n / name :op1 "Johnson"))))
      :op2 (o / one
            :mod (c / close
                  :time (t3 / till
                        :op1 (e / end
                              :degree (v / very))))
            :domain (i2 / it))
      :op3 (r / root-03
            :ARG0 i
```

```
:ARG1 (t4 / ticket
      :mod (s2 / slash
            :op1 (p2 / person :wiki "Mitt_Romney"
                  :name (n2 / name :op1 "Romney"))
            :op2 (p3 / person :wiki "Bob_McDonnell"
                  :name (n3 / name :op1 "McDonnell"))))
:time (s / still)))
```

Visualized form:



3. The goal is to disambiguation each of these two mentions (Romney, Johnson), by figuring out which Wikipedia page each mention should be linked to. For example, the mention “Romney” should be linked to “Mitt_Romney” Wikipedia page.

Now assuming we have already retrieved the candidate Wikipedia pages and ranked them based on popularity measures (e.g., commonness):

Top-10 Wikipedia pages for “Romney”:
`https://en.wikipedia.org/wiki/Romney,_West_Virginia`
`https://en.wikipedia.org/wiki/Mitt_Romney`

```
https://en.wikipedia.org/wiki/New_Romney
https://en.wikipedia.org/wiki/George_Romney_(painter)
https://en.wikipedia.org/wiki/HMS_Romney_(1694)
https://en.wikipedia.org/wiki/Romney_sheep
https://en.wikipedia.org/wiki/George_W._Romney
https://en.wikipedia.org/wiki/Romney,_West_Virginia_in_the_American_Civil_War
https://en.wikipedia.org/wiki/Romney,_Indiana
https://en.wikipedia.org/wiki/HMS_Romney_(1708)
```

Top-10 Wikipedia pages for “Johnson”:

```
https://en.wikipedia.org/wiki/Johnson_(composer)
https://en.wikipedia.org/wiki/Lyndon_B._Johnson
https://en.wikipedia.org/wiki/Norm_Johnson
https://en.wikipedia.org/wiki/Johnson
https://en.wikipedia.org/wiki/Samuel_Johnson
https://en.wikipedia.org/wiki/Johnson_County,_Texas
https://en.wikipedia.org/wiki/Johnson_(rapper)
https://en.wikipedia.org/wiki/John_Henry_Johnson
https://en.wikipedia.org/wiki/Johnson_County,_Indiana
https://en.wikipedia.org/wiki/Gary_Johnson
```

You can also download the machine readable version for these pages at:
https://github.com/aidbsnlp/entity/blob/master/kb.json
Format: JSON dict

```
{
  Wikipedia_URL: {
      'article': string # Wikipedia Article
      'links': [string] # Wikipedia links on the page (list of string)
    }
}
```

A Python example is available at:
https://github.com/aidbsnlp/entity/blob/master/example.py

4. The above results are not good enough, now please implement the similarity measure to re-rank re-rank them.

(1) (3pt) Construct a knowledge graph g(m) for each mention m, based on m’s neighbors derived from the sentence and its AMR graph.
(2) (3pt) Construct a knowledge graph g(e) for each candidate title e, based on e’s neighbors in the knowledge base (Wikipedia pages)
(3) (3pt) Compute the similarity between g(m) and g(e):

$$J(g(m), g(e_i)) = \frac{|g(m) \cap g(e_i)|}{|g(m) \cup g(e_i)|}$$

(4) (3pt) Report linking accuracy; and analyze errors: what improvement can we do? How to improve neighbor construction? How to improve feature representation (e.g., integrate emedding)?How to improve the similarity measure? Can we try to link multiple mentions simultaneously?

(5) (up to 3 extra pt) Implement your ideas in (4).

5. Reporting requirement: code notes; and print out results (list of neighbors, ranking score, etc) at each step.