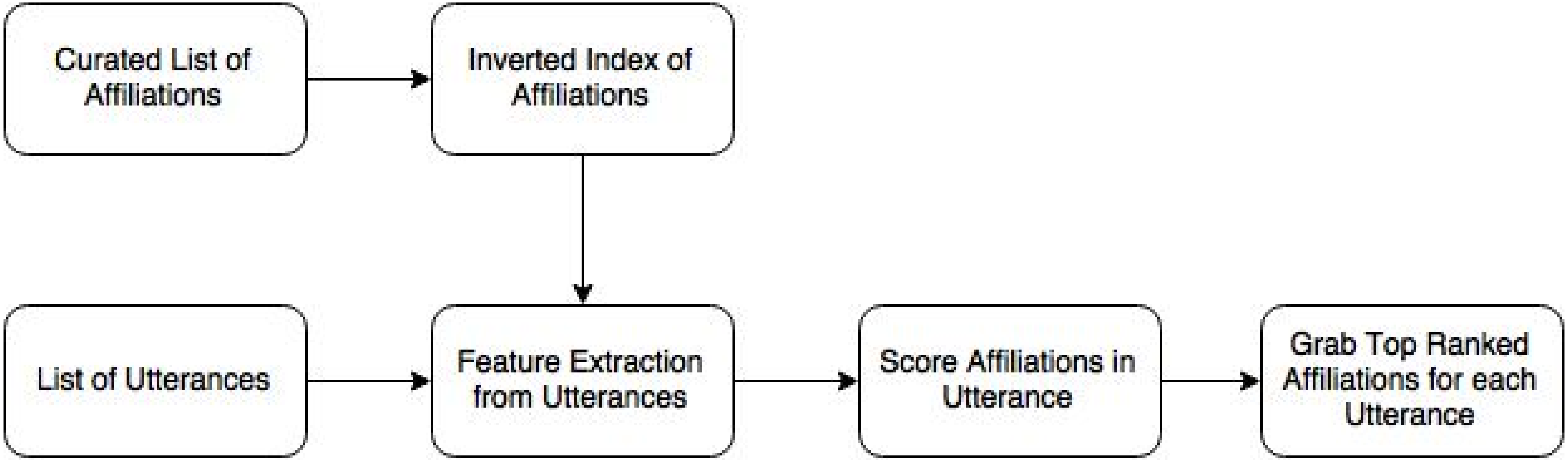
Digital Democracy: Personal Affiliations

Josh Choi, Francis Yuen, Tim Chu, Kyle Tanemura

Timeline Of What We Tried/Did

- Grabbing list of cities and organizations
- Using spaCy to grab entities
- Ranking sets found from dd_orgs and spaCy (venn diagram)
- Clustering all organizations
- Implement an inverted index on list of cities/orgs
- Curating dd_orgs
- Create a set of rules on correctly found entities
- Adjust weights to maximize accuracy

Overview of the System



Curating dd_orgs

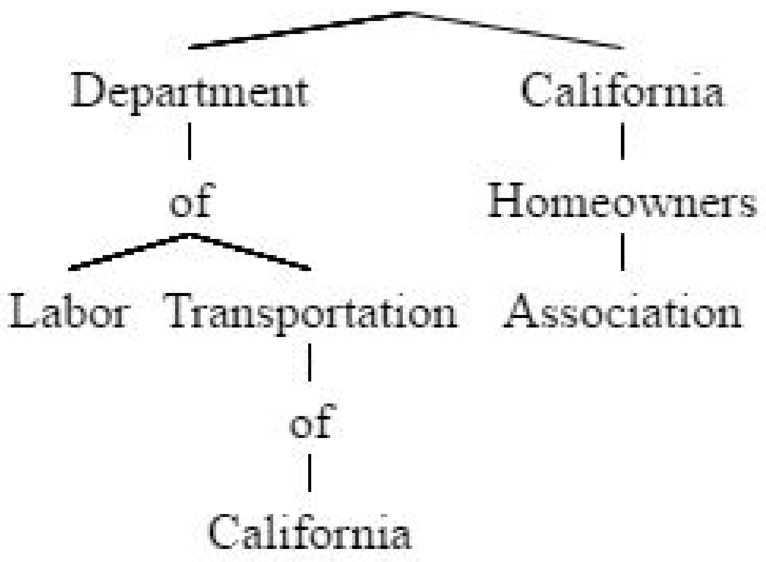
- We removed really obvious government/political terms:
  - “[SPONSOR]”
  - “support” | “support with amendments” | “support with resolution” | etc
  - “association”
  - “amendment”
  - “opposition”
  - “senator”
  - etc
- The overall idea was that dd_orgs was not perfect, and we were getting a lot of extra noise

Constructing an Inverted Index

- Large list of possible affiliations
- String comparison is slow and expensive
- Implement inverted index to allow for lookup of organization name
  - Hash split words rather than compare single characters
- Each node contains
  - Text value of the word
  - Child nodes for every word that appears further down the tree
  - Whether or not this node ended an affiliation name

Inverted Index Example

Tree Diagram



Affiliations:

- Department of Labor
- Department of Transportation
- Department of Transportation of California
- California Homeowners’ Association

Inverted Index as Python Dictionary


```
{'California': {"Homeowners'": {'Assoiciation': {'isEnd': True,
                                                 'textValue': 'Assoiciation'},
                                'isEnd': False,
                                'textValue': "Homeowners'"},
                'isEnd': False,
                'textValue': 'California'},
 'Department': {'isEnd': False,
                'of': {'Labor': {'isEnd': True, 'textValue': 'Labor'},
                       'Transportation': {'isEnd': True,
                                          'of': {'California': {'isEnd': True,
                                                                'textValue': 'California'},
                                                 'isEnd': False,
                                                 'textValue': 'of'},
                                          'textValue': 'Transportation'},
                       'isEnd': False,
                       'textValue': 'of'},
                'textValue': 'Department'}}
```

Create a List of Entities

- Search through words in each utterance to see if found in inverted index as the starting word of an affiliation
- If affiliation is found:
- Use inverted index to continue constructing affiliation name
  - Calculate the distance from the affiliation to the person's first name (if possible)
  - Grab the 2 closest words left and right of the affiliation and using nltk, grab the words' part of speech tag (https://pythonprogramming.net/natural-language-toolkit-nltk-part-speech-tagging/)
- Return a list containing all entities found within utterance

Feature Vector

“Good morning, and thank you Mr. Peria and Mrs. Galgiani, sorry, for inviting me here today. My name is Jenny Lester Moffitt, and I am Deputy Secretary for Policy at California Department of Food and Agriculture. Prior to my tenure at CDFA, I served as the Vice Chair for the Central Valley Regional Water Quality Control Board. And I also was Managing Director at my family's walnut farm in winters, so I'm very pleased to be here.”

```
{
    entityName: ‘california department of food and agriculture’,
    words: [(‘policy’, ‘NN’), (‘at’, ‘IN’), (‘.’,  ‘.’), (‘prior’, ‘RB’)],
    distanceToName: 12,
    fullEntity: True
}
```

Scoring formula per affiliation in utterance

$$s_{final} = \sum w_i * score_i(f_i)$$

w = {weights for each feature}

f = {set of features}

score = {set of feature scoring functions}

Top Words and Top POS

- We generated a list of top words and top pos-tags from a training set and kept track of the count
  - left-left top words = 2 words left of the affiliation
  - left top words = 1 word left of the affiliation
  - …

- We normalized the counts by dividing by the biggest count, which would allow us to get a number from 0 - 1

Scoring top_words and top_pos

- For the tuple, if the left_left word is in the top_words, we return the normalized value, else 0.
  - repeat for others
- Same procedure for POS

Scoring distance

- We calculated the distance from the affiliation to the name, and we took the inverse value (1/d)
  - Favor affiliations closer to the name

Scoring number of words in organization

- We normalized the number of words in affiliations by dividing it by the max word length affiliation
  - Favor longer affiliation names

Right vs. Wrong

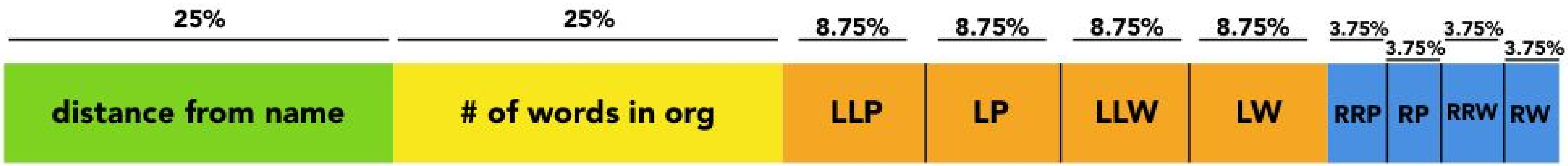
- From our system, we considered the answer correct if it was the top ranked affiliation from the list of all scored affiliations
  - This was hand-done, since there was no available pre-tagged data

('gil topete with the california municipal utilities association, regretfully '
"in opposition to the bill. prior to friday's amendments, cmue had no stake "
'at all in this bill. but with the amendments that were in print as of '
'tuesday, it draws cmue in for the following reasons. we have been a party '
'and participant under the ab 1103 process with the cec.\n')'

[(('california municipal utilities association',
[('with', 'IN'), ('the', 'DT'), (',', ','), ('regretfully', 'RB')], 3, True),
0.29705882874149125),

(('cec', [('with', 'IN'), ('the', 'DT'), ('.', '.'), ('', '')], 66, True),
0.16770321376822928)]

Weights (Iteration 1)

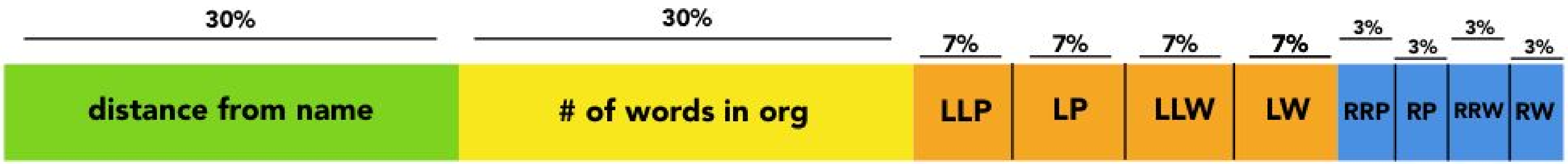


LLP = Left Left POS tag, **LP** = Left POS tag, **LLW** = Left Left top word, **LW** = Left top word

RRP = Right Right POS tag, **RP** = Right POS tag, **RRW** = Right Right top word, **RW** = Right top word

- Results: 62% average accuracy by hand tagging 50 utterances (before curating dd_orgs)

Weights (Iteration 2)



LLP = Left Left POS tag, **LP** = Left POS tag, **LLW** = Left Left top word, **LW** = Left top word

RRP = Right Right POS tag, **RP** = Right POS tag, **RRW** = Right Right top word, **RW** = Right top word

- Results: 76% average accuracy by hand tagging 50 utterances (after curating dd_orgs)
  - 14% increase in accuracy

Division of Labor

Member	Duties
Timothy Chu	Score affiliations, Test accuracy, Adjusting weights
Francis Yuen	Score affiliations, Test accuracy, Adjusting weights
Josh Choi	Spacy/NLTK Integration, Inverted index construction, Vectorizing data
Kyle Tanemura	City web scraping, Inverted index construction, Vectorizing data

Demo