Digital Democracy:
Personal Affiliation Extraction

Kyle Tanemura, Francis Yuen, Timothy Chu, Josh Choi
Dr. Foaad Khosmood

June 14, 2017

1 Introduction and Background

Digital Democracy is an online, searchable archive for all political statements and activity occurring in legislative hearings in California and New York. Before the introduction of this platform, a person would have to travel to a location where all the hearing data was held in order to get the information that they are interested in. For this project, we were interested in automatically determining the best affiliation given an utterance in a legislative hearing. Consider the utterance: "Christian Molina from Sierra Club California in support." From this example, we want to be able to determine that Christian Molina is affiliated with Sierra Club California. In the case where they are multiple organizations which the speaker might be affiliated with, we want to choose the best one.

2 Design and Development

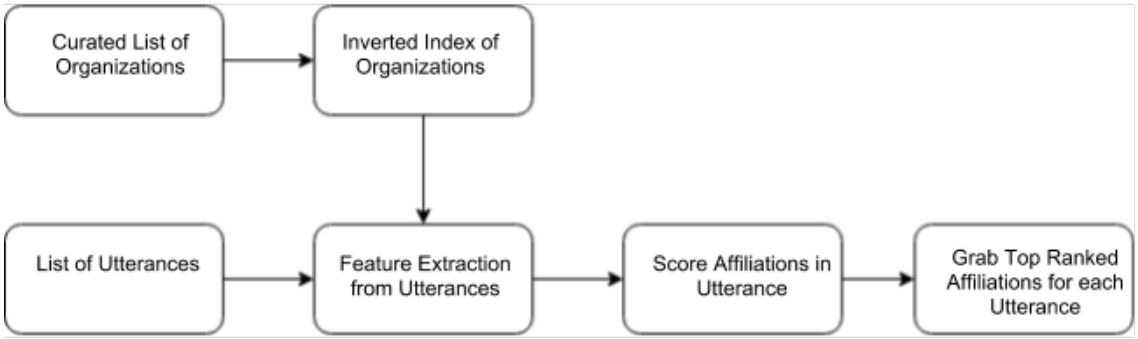

Figure 1: System Flow

The overall design of our system involves inputting a hand-curated list of organizations and the list of spoken utterances to output the top ranked affiliation for each utterance. For each utterance, we extract a list of affiliations/features using the inverted index of organizations and score the individual affiliations using a given linear equation of weights multiplied with each normalized feature. Once all scores have been calculated, the top scoring affiliation will be outputted onto the console for further analysis and measurement.

2.1 Creating a List of Affiliations

To create the list of affiliations the system will take in the `dd_orgs` document and create an inverted index from all the given organizations. The inverted index involves splitting each organization on whitespace delimiters to create a list of individual words. Then each individual word is added into the inverted index tree with each node representing the word and hashing the word to search for words in constant speed. Subsequent words in an organization get added one level deeper into the tree until the end is reached which then the last word is labeled as the ending. If a common word is found within a tree level, a subtree will be created from that tree node representing the rest of the organization words.

For each affiliation we extract from the utterance we must construct a feature vector, represented as a tuple, so the affiliation can be assigned a score. The first step

is tokenizing the utterance using NLTK. We then iterate over the tokens looking for organization starting words. Once an organization is found its starting index, ending index, and the surrounding words are stored immediately. After the extraction algorithm has iterated over the whole utterance, and found every potential affiliation, it calculates the distance from the name of the person to each affiliation using the affiliations' starting and ending index. Then the program extracts the two tokens to the left and right of the affiliation (4 total, returning empty strings if the affiliation is too close to the start or end of an utterance) along with the surrounding words part of speech (POS) tag. The distance to name and surrounding words are added to the feature vector tuple and a list of these tuples is passed to our algorithm for scoring for each utterance.

2.2 Scoring Algorithm

The scoring algorithm takes the summation of feature scores. Each feature score is computed by taking the product of the feature weight and the result of the feature scoring function. Our algorithm have ten features in total. They are:

1. Left Left Top Word Score
2. Left Top Word Score
3. Right Top Word Score
4. Right Right Top Word Score
5. Left Left Top POS Score
6. Left Top POS Score
7. Right Top POS Score
8. Right Right Top POS Score
9. Distance to Name Score
10. Affiliation Name Length Score

Features 1−4 are scored by looking up the positional word in the positional top words dictionary. For example, the left left top word score would use the word that is 2 words left of the affiliation to look up in the left left top word dictionary. The top word dictionary is generated beforehand by counting the number of occurrences a word appears in a position from a dedicated training set. The count is then normalized so all values in top words would range from zero to one. The Feature 1−4 scoring function will look up the word in the top words dictionary, if that word is found the value is returned, else the scoring function will return zero. Features 4−8 are scored similarly to Features 1−4 where a top POS dictionary is created beforehand and the scoring function will lookup the POS and return the score. The Feature 9 scoring function takes the distance (number of words away) of the affiliation from the name as input and returns the inverse of that value. During our utterance examination process, we found that affiliations that are closer to the name tends to be more accurate thus the inverse distance will tend to favor affiliations that are closer to the name. Lastly, the Feature 10 scoring function takes in a count of the number of words in the current affiliation and the number of words of the longest affiliation name and returns the quotient of the two. This is made to favor affiliations with longer names. A longer match of the organization name is bound to be more accurate.

3 Results and Conclusions

To test our program we dynamically hand tagged 50 utterances from each run of our program. Our first test run was done using string comparison of potential affiliations with the original provided list `dd_orgs` and an additional list of cities we scraped from online. This system was extremely slow, +6 hours of run time was needed just to extract all affiliations from the utterances, and was not really improvable due to needing a lot of time between test runs. Development of the inverted index and its use in subsequent runs reduced this runtime to below 1 minute. Once we were using this inverted index we were able to do our first test run actually ranking the affiliations returned from the utterances. This first run correctly identified the top affiliation for 62% of our test set utterances, based on what we dynamically hand tagged as the strongest affiliation. After curating `dd_orgs` we constructed a new index and played around with the weights of our scoring algorithm. On our final test run, we recorded an accuracy of 76% on the same test set as our initial runs.

Our program design allows us to identify personal affiliations with reasonable accuracy, but it is limited in the fact that it must recognize the affiliation and have it stored in its inverted index for the affiliation to be identified. Additional improvements that can be made would be integrating the word the DD Orgs group did in their project. By identifying different names for the same affiliation, we could more accurately cluster people with the same/similar affiliations.