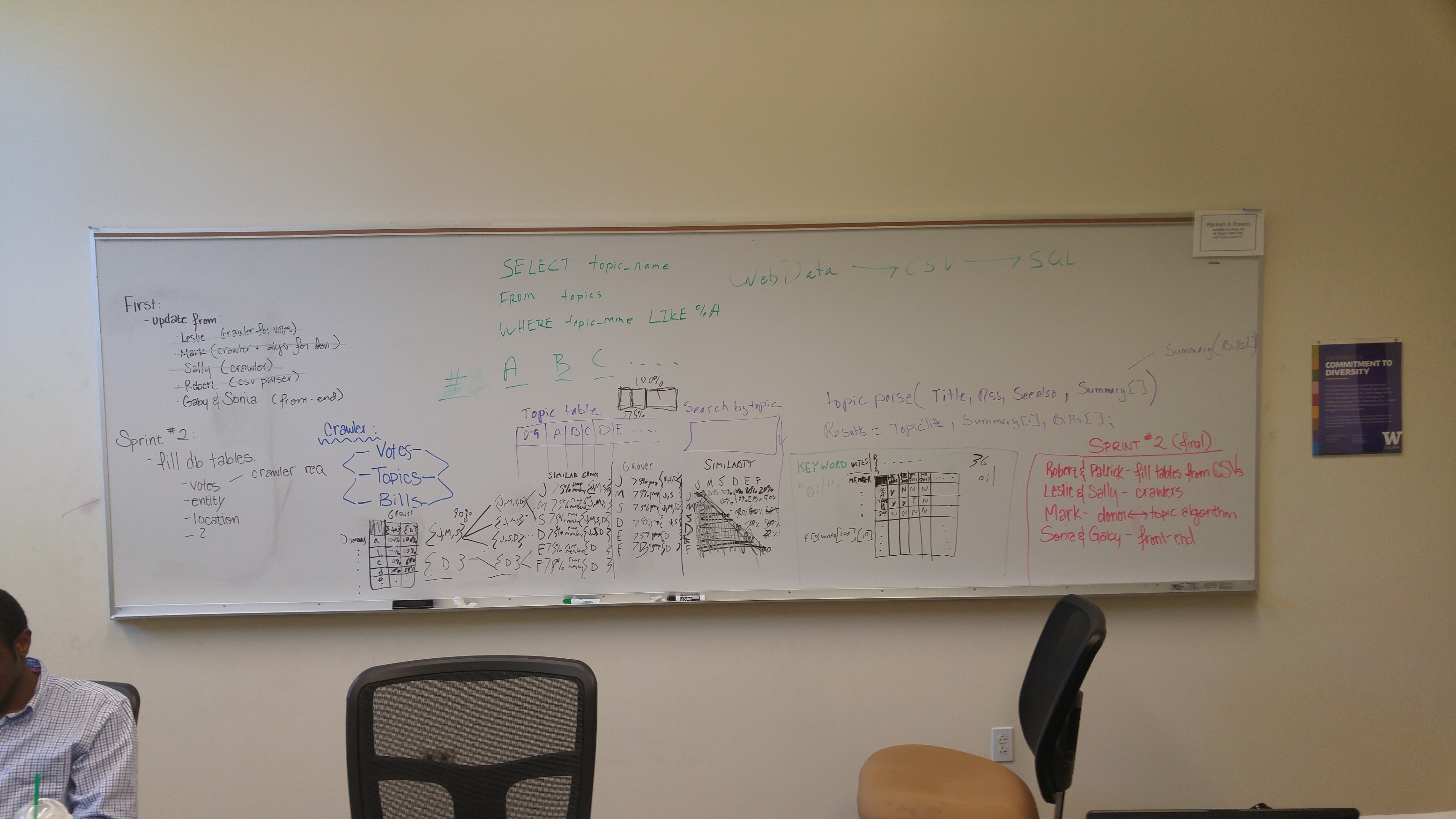
Donations for Votes

TCSS 445 A Summer

Algorithm Pseudocode for Donations for Votes to find connections between campaign donations and floor votes. This will be mostly high level pseudocode, and as the database takes shape we can turn it into low-level pseudocode and eventually Java and SQL language.

Donor : Groups | Group Iterations | Politician : Groups | Politician : Politicians | Politcian Similarity | each keyword has a Politicians and Bills vote table



Politician Voting Patterns

For every keyword, the list of bills it is paired with in the keywords table will be considered bills that pertain to this issue, or keyword (e.g., every bill listed with the topic of “abortion” in it is considered a bill about “abortion”). We will now need to compare all of the votes for each bill that pertain to the keyword. We can now create a 2d matrix of politicians by bills pertaining to each keyword (dual matrices for house and senate to be exact). If we simplify the matrix to an 0-1 matrix (where absent, excused, or other odd statuses can be considered a “nay” vote), we can perform analysis using Boolean matrix algebra to determine patterns in the voting.

The main result of this analysis will be to determine which politicians consistently vote in the same patterns as each other on the same bills. We will do this by creating a triangle matrix (a 2d matrix that is mirrored) of politicians by politicians, comparing their votes to each other. The cell at row “Politician A” and column “Politician B” will the percentage of how similarly Politicians A and B vote (with 100% being they voted the same on every vote and 0% being they always voted opposite each other).

Now we can begin to build groups, also called herds. We will now give each politician a group—that is, a list of all the politicians who vote similarly some threshold percentage of the time (say, 75%). So Politician A has a group that, say, has 30 members who vote similarly 75% of the time. Politician B, C, and so on also have groups. We start with a number of groups equal to the number of politicians in our particular chamber. From here, we iteratively merge groups using the above algorithm, but adjusted for people, until we hit some low threshold of maximum groups allowed (say, less than 5).

Each iteration of group combination, all groups gain their own lists of groups that have some minimum threshold of the same group members (e.g., Group 1, which in the first iteration is Politician A, now has a list of all the other groups that share at least some high starting percentage, say 90%, of the same group members); from these lists, we will reduce the amount of groups as follows. Any groups with identical members are consolidated into a single group *with the non-matching members removed*; once all identical groups (if any) are merged, we need to further cull the groups: any group with fewer than an acceptable amount of members to determine a pattern (say, 10% of the whole, so with 100 politicians, a group with fewer than 10 politicians is considered a non-group) are culled. Then, we check whether we have hit our minimum group threshold: if we have, then we halt the algorithm and go forward with our distinct groups (2-4 usually), calling them herds; if we have too many groups, then we repeat the group combination but with a lower threshold of similarity for merging groups (say, 80%); if we end up with only one group, then we say that there is no distinct voting pattern for this issue.

In general, two or so distinct patterns should emerge from any keyword that is politically divisive. From this, for each keyword we can create a few (usually two) pattern categories to be put into a new BRIBES table as such, where every member that belongs to a herd for a keyword is associated with that keyword herd (and has their donor bool set to false):

#Bribes table

#An algorithmically generated table tying keywords to herds of politicians with similar voting patterns

#Each keyword with have 0 or 2+ groups of politicians who vote for this issue in a similar pattern

DROP TABLE IF EXISTS BRIBES;

CREATE TABLE BRIBES (

#COMPOSITE KEY: FOREIGN KEY:

#Keyword that describes in short a politically relevant issue (also subject, topic).

KEYWORD char (255) NOT NULL,

#COMPOSTITE KEY: The herd group id of a voting patterns

Herd integer NOT NULL,

#COMPOSITE KEY: FOREIGN KEY: The house or senate voting member

ENT\_ID integer NOT NULL,

#Whether this entity is a donator for this herd or a voter in the herd.

Donor bool NOT NULL,

#CONSTRAINTS

CONSTRAINT BRIBES\_PK PRIMARY KEY

(Keyword, Herd, ENT\_ID),

CONSTRAINT BRIBES\_ENT\_FK FOREIGN KEY (ENT\_ID)

REFERENCES ENTITIES (ENT\_ID),

CONSTRAINT BRIBES\_KEYWORD\_FK FOREIGN KEY (KEYWORD)

REFERENCES KEYWORDS (KEYWORD)

);

Donor-Politician-Keyword Connection

Now we can connect the donors. We will create a new 2d matrix of donors by herds (for each keyword). For each keyword-herd combination, we will fill the cell with the percentage of members in that herd that the donor has contributed to. When done, we can check if the donor gave much more to politicians in a particular category than another. If there is a clear tendency (e.g., for keyword “oil”, Donor X gave money to 80% of the members in Group 1, but gave 5% to the members of Group 2), then this donor can be marked as being in similar views as the herd they gave a clear majority to (actual thresholds are debatable—obviously nearly equal percentages should cancel out, so perhaps a 50% difference threshold). These donors can be added to the BRIBES table with the donor bool marked true to any herds of any keywords that they are seen as partial to.

By doing the following search (or some variation, depending on how the final data is ordered and assuming the double table reference works as intended), one should be able to compile a list of donors who give money to specific herds and members based on a specific cause:

SELECT DONATIONS.FROM\_ID, BRIBES.KEYWORD AS issue, BRIBES.Herd AS grouping, DONATIONS.TO\_ID

FROM DONATIONS

JOIN BRIBES

ON DONATIONS.TO\_ID = BRIBES.ENT\_ID

GROUP BY DONATIONS.FROM\_ID, DONATIONS.TO\_ID

HAVING DONATIONS.FROM\_ID IN ( SELECT ENT\_ID

FROM BRIBES

WHERE issue = KEYWORD AND grouping = Herd AND Donor = true

);

From the table produced by the above query, we could further mine data to notice even larger trends, such as groups of donors giving to the same cause, and groups of politicians who consistently receive money from a group of donors concerning a group of causes; my guess is we’ll begin to see a pretty clear party line distinction, with some interesting caveats.