It's Alive: Emergent ACT-R Voting Model Errors on Full Paper Ballots

Joshua Engels (jae4@rice.edu)

Michael D. Byrne (byrne@rice.edu)
Department of Psychological Sciences, Department of Computer Science
6100 Main St., MS-25, Houston, TX 77005 USA

Abstract

This paper presents an ACT-R model we developed to simulate voting behavior on full size paper ballots. We focus specifically on a case study of a non-standard voting strategy: the strategy votes first from left to right on a ballot and then from top to bottom. We run this model on 2772 randomly generated ballots governed by 3 different structural variables. The findings suggest that our model's error behavior is emergent and sensitive to ballot structure. These results represent an important step towards our end goal of creating a self-contained piece of software capable of identifying bad ballot design.

Keywords: ACT-R; error prediction; voting

Introduction

Voting is hard. The deliberations and conversations that go into choosing who best represents one's interests is an important and time-consuming task, one that might be argued to be the very backbone of a democracy. Understandably, many may assume that the subsequent task of correctly voting for one's chosen candidate is comparatively easy and straightforward. Surely once a voter gets the ballot and can mark whoever they please, the hard part is over.

For the most part, they would be right. When ballots are designed well, any errors voters make are not systematic and will not help or hurt any candidate. When they are designed poorly, they may lead to systematic voting errors, but perhaps this happens rarely or does not matter on a large scale. In the general case, people tend to make relatively few mistakes when marking a ballot with who they want to vote for.

However, in closely contested elections it is not the general case that is important. There have been numerous elections in the past 20 years that have been documented as having been decided by systematic voting errors caused by bad ballot design. While election interference by hacking is a far more flashy and obvious risk, there at least has never been evidence that this has swung an election, unlike for bad ballot design. Ironically, the fear of hacking has led to a return to paper ballots, which with their profusion of races packed onto small sheets of paper makes ballot design even more important.

People routinely make many kinds of voting errors, but the biggest errors resulting from bad ballot design are under and over voting. Under voting is an error that occurs when the voter fails to vote on a race that they intended to, whereas over voting is when a voter votes on a race more than once. The problem of designing even one ballot that will not cause people to systematically under or over vote is challenging. For instance, it might entail running a usability study weeks before the actual election. What makes the problem so difficult is the sheer number of counties in the United States, each of which designs their ballots differently and each of which have hundreds of different iterations of ballots for each precinct they are responsible for. Manually checking each ballot with a usability study is infeasible.

One possible solution to this problem is a piece of software that could automatically check an arbitrary ballot for common design errors. However, since the task is to decide if humans will make a mistake, it is initially difficult to imagine building an automatic system to do such a task. Here is where ACT-R, a cognitive architecture purpose built to simulate human cognitive tasks, comes in. In order to ensure that every mistake was predictable, any mistakes a human would make on a certain ballot would have to be made by the system as well

This is not an original idea. Green (2010) built an ACT-R model that could make the same mistake voters did in a famous ballot caused systematic error. However, these and other attempts have focused on replicating one error behavior. A system that could be used by election officials to identify bad ballots would have to be able to predict all historical voting errors, as well as any new ones.

Thus, Wang, Lindstedt, and Byrne (2019) describe the construction of a model that will simulate a vast array of possible voting errors, with an eventual goal of simulating the entire voting space. In their paper they describe the start of building such a model. The model ran in a voting environment called VoteBox. It was a simulated simple electronic ballot, consisting of just a single race per screen with a next button to navigate.

Nevertheless, within just a simple task was hidden great complexity: Wang describes building and using a total of 40 different voting strategies constructed from differing memory and navigational strategy selections. The voters differing strategies and knowledge led to different rates of error, showing that a voter's strategy made a difference to whether they were able to vote for their intended candidates. However, this effort did not vary the structure of the ballot and so did not yet have the predictive power necessary to recognize badly designed ballots.

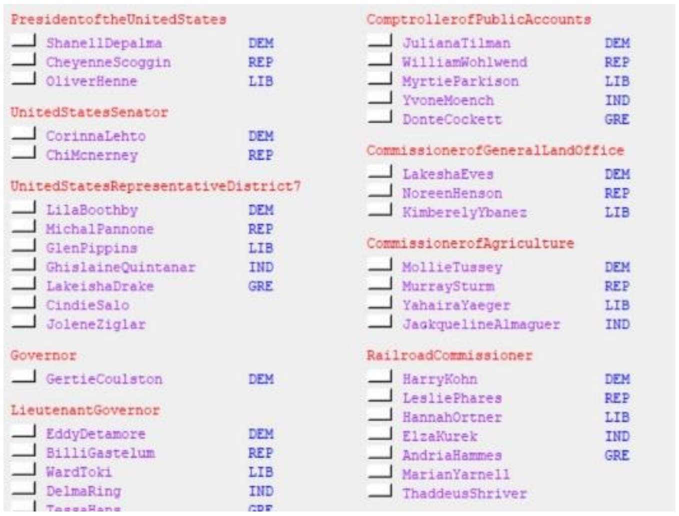
Thus, in this paper, we first describe the extension of this system to handle full simulated paper ballots. Then, we describe the error rates of our new simulated voters on various simulated ballots. This represents a large step towards our end goal of constructing a piece of software to identify bad ballots.

Method

First, we describe the design of our experiment: the randomly generated full length ballots and our new modular system that votes on them.

Ballot Design

We built simulated full "paper ballots" for the model which consist of a virtual screen populated with several columns of races. Each race has a title, a list of candidates and their associated parties, and a list of buttons that the model can click to vote for a candidate. (see Figure 1).



```
PresidentoftheUnitedStates                 ComptrollerofPublicAccounts
ShanellDepalma        DEM                  JulianaTilman         DEM
CheyenneScoggin       REP                  WilliamWohlwend       REP
OliverHenne           LIB                  MyrtieParkison        LIB
                                           YvoneMoench           IND
UnitedStatesSenator                        DonteCockett          GRE
CorinnaLehto          DEM
ChiMcnerney           REP                  CommissionerofGeneralLandOffice
                                           LakeshaEves           DEM
UnitedStatesRepresentativeDistrict7        NoreenHenson          REP
LilaBoothby           DEM                  KimberelyYbanez       LIB
MichalPannone         REP
GlenPippins           LIB                  CommissionerofAgriculture
GhislaineQuintanar    IND                  MollieTussey          DEM
LakeishaDrake         GRE                  MurraySturm           REP
CindieSalo                                 YahairaYaeger         LIB
JoleneZiglar                               JackquelineAlmaguer   IND

Governor                                   RailroadCommissioner
GertieCoulston        DEM                  HarryKohn             DEM
                                           LesliePhares          REP
LieutenantGovernor                         HannahOrtner          LIB
EddyDetamore          DEM                  ElzaKurek             IND
BilliGastelum         REP                  AndriaHammes          GRE
WardToki              LIB                  MarianYarnell
DelmaRing             IND                  ThaddeusShriver
```

Figure 1: Part (top left corner) of a simulated ballot.

The resulting simulation is not quite the same as a paper ballot: the model clicks on a button instead of filling in a circle, will never obscure the ballot with its hand, and will never poke a hole in the page among many other differences. However, the ballot is similar enough to cause many of the same errors we expect humans to make.

To help the model navigate, we colored the race header red, the candidates purple, and the parties blue, as ACT-R can make visual location requests based off color.

Model Design

We built the model itself with one overarching goal in mind: to have the ability to simulate as wide an array of voters as possible.

Our modular system split a simulated voter's strategy into four different pieces: macronavigation, the process of moving from one race to the next; encoding, the process of determining the race, party, and candidate visual groups for each race; micronavigation, the process of finding the intended candidate to vote for within each race; and clicking, the process of actually clicking on the button corresponding to the chosen candidate. At runtime we select one strategy from each of these categories and combine them together with a declarative memory file to build an ACT-R model. Note that Wang Et Al's work remains in our architecture as possible micronavigation and declarative memory choices.

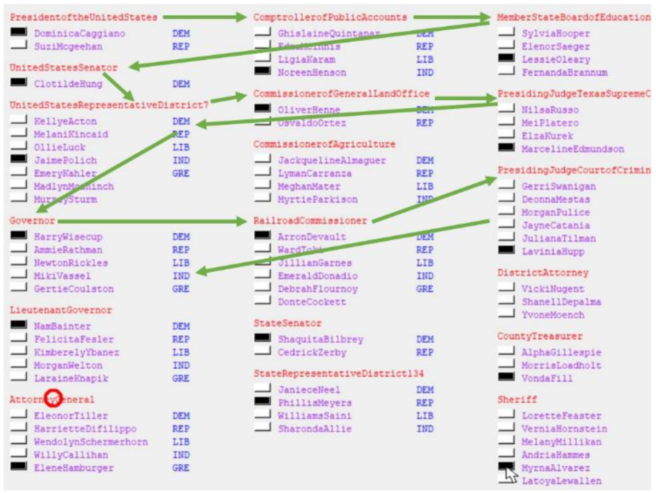
Designing A New Strategy

We first built the most obvious strategies for each strategy category because we wanted our initial strategies to lead to a composite voting strategy with no errors. We had to know that our model worked before we could start varying pieces to induce errors.

Our first new strategy was a non-standard macronavigation strategy. Our model's standard macronavigation strategy was *top to bottom left to right*; that is, the model started in the top left corner and went all the way from the top to the bottom and then went over to the next column to the right and again went top to bottom, repeating until it was finished. This is the most obvious method of macronavigation, and like noted above resulted in no mistaken votes. On the other hand, the first new macronavigation strategy we built was *left to right top to bottom*.

The *left to right top to bottom* strategy started on the upper leftmost race on the ballot. It then proceeded first to the right, navigating to the closest race to the current race and repeating until it voted on a race in the last column. Then, it went back to the beginning of the row, found the next race down in the first column, and repeated voting from left to right. The model continued until it ran out of new races in the first column.

Crucially, because the order this new strategy visited races was not matched to the way the races were ordered on the ballot, the *strategy could miss races*. For a clear example, see Figure 2.



```
PresidentoftheUnitedStates           ComptrollerofPublicAccounts          MemberStateBoardofEducation
DominicaCaggiano      DEM            GhislaineQuintanar    DEM            SylviaHooper
SuziMcgeehan          REP            [illegible]           REP            ElenorSaeger
                                     LigiaKaram            LIB            LessieOleary
UnitedStatesSenator                  NoreenHenson          IND            FernandaBrannum
ClotildeHung          DEM
                                     CommissionerofGeneralLandOffice      PresidingJudgeTexasSupremeC
UnitedStatesRepresentativeDistrict7  OliverHenne           DEM            NilsaRusso
KellyeActon           DEM            OsvaldoOrtez          REP            MeiPlatero
MelaniKincaid         REP                                                 ElzaKurek
OllieLuck             LIB            CommissionerofAgriculture            MarcelineEdmundson
JaimePolich           IND            JackquelineAlmaguer   DEM
EmeryKahler           GRE            LymanCarranza         REP            PresidingJudgeCourtofCrimin
MadlynM[illegible]inch               MeghanMater           LIB            GerriSwanigan
MurraySturm                          MyrtieParkison        IND            DeonnaMestas
                                                                          MorganPulice
Governor                             RailroadCommissioner                 JayneCatania
HarryWisecup          DEM            ArronDevault          DEM            JulianaTilman
AmmieRathman          REP            WardToki              REP            LaviniaHupp
NewtonRickles         LIB            JillianGarnes         LIB
MikiVassel            IND            EmeraldDonadio        IND            DistrictAttorney
GertieCoulston        GRE            DebrahFlournoy        GRE            VickiNugent
                                     DonteCockett                         ShanellDepalma
LieutenantGovernor                                                        YvoneMoench
NamBainter            DEM            StateSenator
FelicitaFesler        REP            ShaquitaBilbrey       DEM            CountyTreasurer
KimberelyYbanez       LIB            CedrickZerby          REP            AlphaGillespie
MorganWelton          IND                                                 MorrisLoadholt
LaraineKnapik         GRE            StateRepresentativeDistrict134       VondaFill
                                     JanieceNeel           DEM
AttorneyGeneral                      PhillisMeyers         REP            Sheriff
EleonorTiller         DEM            WilliamsSaini         LIB            LoretteFeaster
HarrietteDifilippo    REP            SharondaAllie         IND            VerniaHornstein
WendolynSchermerhorn  LIB                                                 MelanyMillikan
WillyCallihan         IND                                                 AndriaHammes
EleneHamburger        GRE                                                 MyrnaAlvarez
                                                                          LatoyaLewallen
```

Figure 2: The green arrows mark the beginning of this model's race voting order. The model skips CommisionerofAgriculture.

When the model reaches the third race down on the left column ("UnitedStatesRepresentititveDistrict7") it votes on that race and then proceeds right along the row, selecting and voting on the closest race and repeating until it reaches the last column. The model then returns to the race at the beginning of the row and proceeds to the first race on the next row down (“Governor”). Here is where it makes its mistake: because the “RailroadCommisioner” race is the closest race to “Governor”, the model votes on “RailroadCommisioner” for its second race in the row and so skips CommisionerofAgriculture. It will never return and vote on this race.

We quickly observed that the races our new strategy missed depended on the layout of the races on the ballot. The error behavior was unplanned and emergent, ratifying our long-term goal of building models that can produce novel errors on novel ballots. Now that we had a simulated voter making structure-based mistakes, we decided to test how these mistakes changed as we modified the ballot.

Running Experiments

Initially, our ballot was static, consisting of a manually positioned set of races and candidates. Our first step was modifying the ballot so it could be dynamically generated. Every time we ran the model, our generation process allowed us to vary the vertical spacing between races, the vertical space between the race header and the candidates, and the vertical space between candidates. We chose ranges of the variables that led to ballots our model could still realistically parse but that nevertheless were visually distinct (see Table 1). As the ballot was generated each race was randomly selected to have between 1 and 4 candidates.

For each one of the 132 possible combinations of spacing variables (see Table 1), we ran the model on 21 randomly generated ballots. Thus, our model was run on 2772 ballots for a total count of 65,232 individual races. For each run, we recorded the exact race positions and race order on the ballot, as well as the order the model voted on races (including any races the model missed).

Table 1: Ballot Variables

Variable	Range (Pixels)
Space between races	5 – 15
Space between header and candidates	20 - 22
Space between candidates	15 - 18

By analyzing these data in the next section, we will characterize this strategy and identify how and where it fails. Perhaps more importantly, we will characterize good and bad ballot design by seeing which designs lead to more error in the model. This will serve as a case study for how new strategies built on our architecture will find errors in novel ballots.

Results

We make a few initial remarks about our data. First, we define the model *percent error*, the percent of *races* that our model skips. Our model’s *global* percent error is around 15.5%, meaning that on average given an arbitrary race on a ballot there is a 15.5% chance that our model will not vote on it. This rate is certainly much higher than any experimental rate in human voters, but as this strategy is nonstandard, this result is to be expected. Of course, most people do not make anywhere near this many errors, but average error rates in the wild undoubtably stem from outliers like this strategy. We also define other percent errors collected over subsets of many ballots or races below, although the overall idea remains the same.

Effects of Race Location

We first examine the relationship of race location on the ballot to model error. We observe that there is a general trend of increasing error across columns (see Figure 3). In other words, races in later columns are more likely to be skipped.

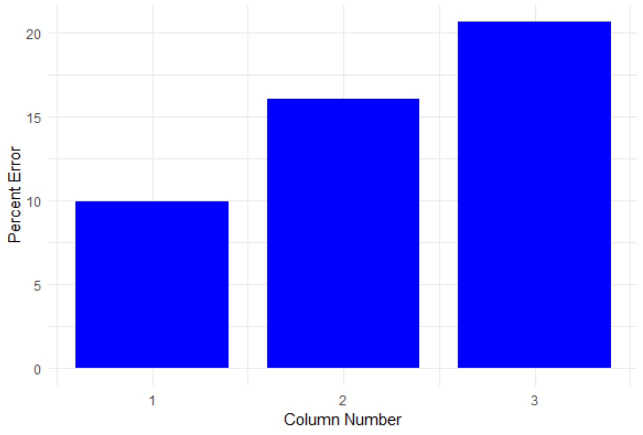


Figure 3: Average percent error across races in the first, second, and third column across all ballot runs

In fact, since we have the exact y coordinate and column for every race, we can generate a heatmap of error rates of races by column and y position (see Figure 4). Each bin collates the percent error of the model across races on all ballots within 10 vertical pixels, where the y position of a race is its header.

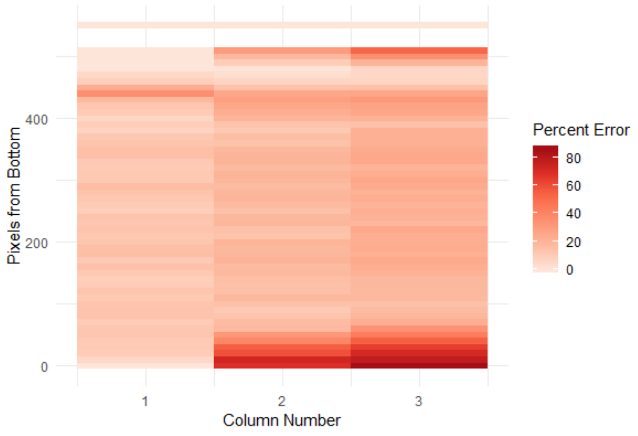


Figure 4: Heatmap of the model’s error according to races’ column number and y position.

Of interest are places in Figure 4 where errors are likely. One immediately obvious place is the bottom right corner, where average percent error approaches 1. The model almost always misses errors there. To make sense of this result, we observe that the only way a race can have its *start* in one of those bottom right boxes is if it is very short. It makes sense that for short races nestled in the bottom corner, people will frequently get to the last race in the first column and vote across that row not *low enough* to reach the bottom corner races.

However, for the most part, errors are uniformly distributed across the ballot. This result hints at the strength of our model: errors occur seemingly randomly across the ballot because they are emerging from the specific structure of individual ballots. Thus, using our data of each experiment's race layout, we move onto examining how specific elements of ballot structure influence model error.

Effects of Ballot Structure

We first examine the average error as we vary the amount of vertical space between the end of each race and the beginning of the next. Recall that vertical space is just one of the spacing variables we manipulated (see Table 1). Thus, each specific vertical spacing value includes many observations from ballots built from combinations of the other spacing variables. While we did examine these other spacing variables, we found they had no statistically significant effect on the model's error.

As the space between races decreased, voting error increased (see Figure 5). This result validates the intuitive thought that the more cluttered a ballot is, the more likely a simulated voter is to miss a race.

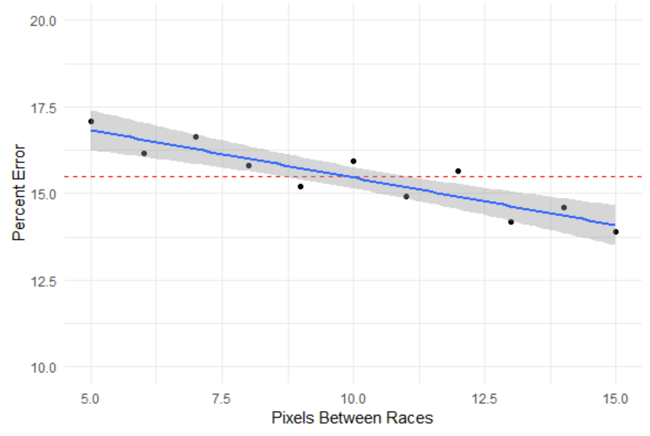


Figure 5: Each black dot is the average percent error across all ballots with a specific race spacing. The blue line is the linear regression for the trend, the red line is the average error of the model, and the shading represents 95% confidence intervals for the line.

We also examined how the length of a race was related to the chance it would be skipped and found similar results: as the length of a race decreased, the model's chance of skipping it (it's percent error for races of that length) increased (see Figure 6).

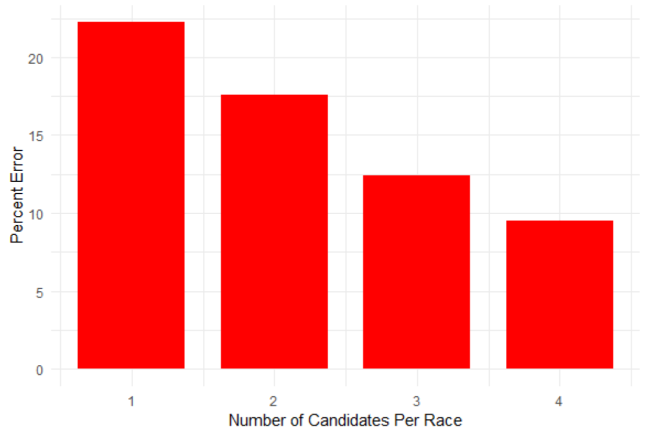


Figure 6: Average percent error of the model on races of one candidate, two candidates, three candidates, and four candidates.

Finally, we looked at how the model's percent error varied as a function of the vertical distance from a given race to the nearest race to it in the last column. In Figure 7, we plot bins of 5 pixels of this vertical distance versus both the number of races that were voted on in that bin and that were not. The stacked bar plot here is useful because it shows two things: that the chance a simulated voter missing a race increases as the closest distance to the last race increases, and that the number of races that are far from any prior race decreases as the distance increases. This graph more than any other illustrates the model's behavior and tendency to miss races that are not lined up in a row; building and running the simulation allows us to identify what these races are for any given ballot.

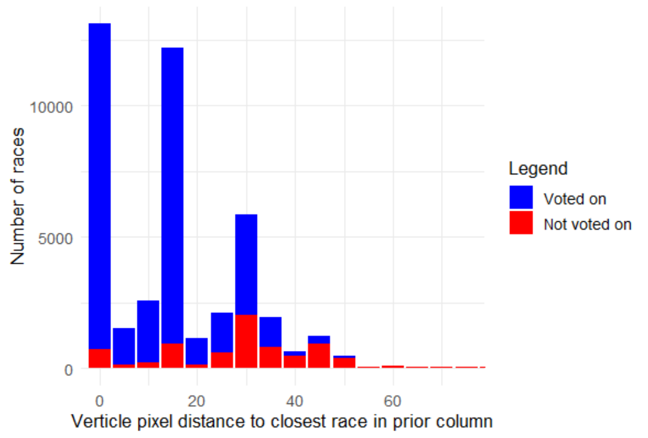


Figure 7: Stacked bar plot of the number of races voted on and not voted on across all model runs, plotted according to the vertical distance between the race and the closest race in the last column.

Conclusion

Races were more likely to be missed if they were smaller, out of line with the races in other columns, or more cramped overall. These are all characteristics of bad ballots that our model detected organically. The detection behavior emerged out of the design of the strategy; it was *not* hardcoded. Thus, this first complete non standard voter serves as a positive case study for our entire project.

Notable, using a non-standard macronavigation strategy actually amplified our ability to detect bad ballots. For instance, a strategy moving in the same direction as the races were originally placed might not mind if the races were very close together, but any other strategy would. Ballot designers need to cater to less common strategies, so an ability to detect when ballots will cause systematic errors in voters using these strategies is crucial.

Indeed, we should note that the average error for this strategy is far higher than the average error for all voters, even assuming as we did that once a voter found a race they would successfully vote on it (choosing a perfect micronavigation strategy, in the parlance of our model). Most real voters probably use a more successful macronavigation strategy. However, if even a subset of voters uses this strategy, or one like it, then we must account for them in our model, as a subset of voters can still have a deciding impact on a close race.

Thus, one of our next steps will be to entirely map the space of macronavigation strategies by running eye tracking experiments on human subjects voting on ballots. To implement these new strategies, we will need to expand the capabilities of ACT-R itself by extending the current visual grouping module to group objects in a hierarchy and by extending the options models have to visually navigate.

We also plan to build new strategies in the other modular categories of the model, including new ways for the model to encode the candidate, party, and race groups and new ways the model finds and clicks the circle corresponding to a candidate. Again, we will need to run studies to determine every variant behavior. Once we succeed, we will have a system that can dynamically build any voter from the voting strategy space. Each strategy will have a characteristic error pattern like we described in this paper, but more importantly can be run on novel ballots and determine if the error rate is above average.

While some of the findings may seem obvious, they must partly be viewed in the light of the wider project. Our model was able to vote on a wide array of ballots that looked hugely different and successfully make consistent errors. More than just characterizing the type of ballots and races that are more disposed to be skipped by a specific voter, these findings confirm the feasibility of attempting to eventually predict errors in novel ballots.

(Couldn’t read your github note, so I just left those few sentences out)