National Journal of Speech & Debate

Volume VII: Issue 2

January 2019

THE TWO TYPES OF DEBATE TABULATION SOFTWARE

BY ÉTIENNE BEAULÉ

THE TWO TYPES OF DEBATE TABULATION SOFTWARE

BY ÉTIENNE BEAULÉ*

*Étienne is a first-year university student in Mathematics and Computer Science at l'Université de Montréal. He is also the Information Technology Officer of the Nova Scotia Debating Society ("NSDS"), and can be reached at tech@debatingsociety.ca.

INTRODUCTION

Debate tabulation is a laborious and time-consuming process with extremely short deadlines for quick round turnaround. It is no wonder that big efforts to computerize these difficult roles were made in the 1990s, automating the old hand tabulation with these new applications (Bruschke 2006). These new applications reduced the number of repetitive tasks as well as the statistical work required. These systems could perform many tasks, from creating ballot sheets to adding up points, now done by machine (Bruschke 2006). Machines are the best-suited for those types of task, able to do repetitive tasks without pause, even less error-prone in calculations. Today, there are systems that can handle most, if not all, aspects of managing a debate tournament, from registration to the rankings of the participants, including debate pairings with adjudicators while taking complex constraints into account (Palmer, Bruschke, and Menick 2018).

The ecosystem of debate applications has grown significantly, and the methods for performing tasks can be quite dissimilar to one-another. Most of these systems grew out of a need for a specific tournament, and grew from that starting point (Belesky and Lee 2018; George 2013). With that, the structure of these various solutions vary greatly, and debate tabulation software comes in all shapes and forms. The main division in these solutions is between spreadsheet-based applications, like CG_Tabs (George 2013), or website-based applications, in the case of Tabbycat (Belesky and Lee 2018). A third type, desktop programs, do exist, but have mostly migrated to being web-based (Palmer, Bruschke, and Menick 2018), and so will not be touched upon here. Web-based applications and spreadsheets are both able to manage debate tournaments, but through different means by their structure. This brings both benefits and disadvantages to both broad categories.

In the first part of this article, the fundamental differences between both paradigms will be explored, discussing their advantages and disadvantages to each other. Afterwards, a novel application which integrates both paradigms which I am working on for the Nova Scotia Debating Society (NSDS) will be introduced.

A COMPARISON OF THESE SYSTEMS

The fundamental difference between web-based applications and tailored spreadsheets can be described as the use of the n-layer model of applications. Web-apps have a clear delineation between the *presentation*, *application*, and *data* layers. In brief, the *presentation* layer deals with the user interface towards the program, the *application* layer performs the operations and calculations, while the *data* layer handles the data of the application (Petersen 2008). Taking Tabbycat as an example, the Hypertext (HTML) pages serve as the presentation, Python is used to create pairings and aggregate results as the application, while PostgreSQL serves as the database (Belesky and Lee 2018). Spreadsheets do not have this conceptual model of applications. The presentation of the spreadsheet is the inputted data, which is melded together with the calculations on this data as formulæ similarly as part of the data. The layered model of programs is the root of the differences between spreadsheet and web-based debate software, with the consequence of changing how these solutions are used.

It follows that while web-based applications model the structure of a debate tournament, spreadsheets require data input directly into the presentation. A hierarchical model for a debate and its results could look like Ballot (is a subclass of) → Debate → Round → Tournament, using the data inside these models to aggregate results. Spreadsheet applications do not use an hierarchical model for scoring. Scores are taken and added to a debater's set of scores, not necessarily as being part of a discrete ballot. This causes a separation between the scores and related data such as speaker positions; cross-referencing various ballots is broken and less data is conserved which could hinder deeper analyses from being made. Strong structure does have its downsides. It is very difficult to create a model that satisfies many types of debate as they can be wildly different between formats and "house rules." With that in mind, spreadsheets offer greater flexibility to override procedures and modify data. Procedures can be more easily altered in such a format than if all the equations and steps are encoded in a web-app. While both types of system can contain swaths of data about the rounds and debates, web-apps have a clear advantage in forming relations between these data, but spreadsheets offer greater flexibility in adapting these data to the tab.

As a result of their scope, the functionality of these paradigms is not perfectly correspondent. As shown earlier, there is a clear distinction between layers in the tiered application model. The user interfaces available is another point between these various systems. Web applications can offer several different views for different people or purposes. In particular, using the "Ballot" model, an adjudicator could submit scores for the debate they judged, while other judges can be submitting ballots for their debates concurrently. Going further, officials can be offered different permission levels than the judges, such as granting access to all the ballots through exclusive interfaces, contrary to spreadsheet which nominally do not offer any "interfaces" *per se*, but forms can be created to insert data.

Furthermore, permissions are not as comprehensive in spreadsheets as the access controls found in web applications. This is as spreadsheets have a "one-user" model with only one person controlling the spreadsheet. Debate tabulation spreadsheets are designed to only have one person running everything in the spreadsheet, while web applications are designed with multi-user support. It would seem that as a general rule, these web applications grow to encompass more of the steps of tournament organizing than spreadsheets do. With clearer and more useful data models, web-apps can branch to serve more needs, and still be very quick as the application level has less overhead for calculations than spreadsheets. Notably, CG_Tabs does not offer speaker score handling, only team scores for performance reasons (George 2013). This focus is not necessarily a bad thing. It is still useful to have specialized applications that excels in what it does, especially as performance is key in competitions.

MORE ON SPREADSHEETS

Spreadsheets offer an interesting solution to the needs of debate tabulation. Spreadsheets are very well known, evoking Microsoft Excel which is ubiquitous on computers. Opening and using simple ones does not take much thought. With the vast feature set of Excel, each user can figure out how to accomplish different tasks in different or similar ways. It is a pity that templated workbooks, as found for debate tabulation, inherently forces a particular method to accomplish tasks. It negates the benefits of using spreadsheets. This is especially the case with spreadsheets attempting to emulate standalone solutions, requiring forms to input data, effectively separating its presentation layer. An example of this is PlusTab (Hanson, Cantrell, and Woodrich 2013). The ease of modification and its flexibility are key points for spreadsheets, and these goals should be harnessed effectively.

Through the vast feature-set of Excel, only a little subset is used by any one workbook. Debate tabulation spreadsheets mostly, even uniquely, use stored procedures (called *macros*) while ignoring potentially simpler or more effective built-in solutions. The perennial feature of spreadsheets; formulæ are apparently absent from these workbooks. They could be quite useful in applications such as for calculating overall statistics and rankings for debaters or teams. In addition, more niche features such as Solver is an extremely useful tool in allocations.[1] It is disappointing that macros are being used while more proper tools are more readily available and accessible.[2]

MESHING BOTH PARADIGMS

When a spreadsheet is mentioned, the first piece of software that comes to mind is Microsoft Excel; the pre-eminent spreadsheet program. Regardless, there are other contenders, focusing on areas where Excel is weaker. One of these competitors is Google Sheets, part of G Suite, Google's competition to Microsoft Office as a whole. As a web company, Google focused on improving spreadsheets to be web-based, with strong connections to the other components of their suite to facilitate collaboration. As the NSDS was moving towards G Suite, we took the opportunity to redesign our tabulation workbook to take advantage of the new collaborative features while expanding upon the old workbook in terms of functionality. This new spreadsheet-based tabulation software is called "Debate Suite."

Google Sheets brought an advantage with integrations as add-ons and between their other applications. A normal spreadsheet only has one person working on the instance at a time, but many people can work collaboratively on this sheet. By consequence, conflicts can arise if officials are giving conflicting instructions. The reliance on these add-ons is also the Achilles' Heel of Google Sheets. It does not support many features of Microsoft Excel that would be considered basic, such as duplicate value highlighting for an example, relying on 3rd party add-ons for these features. They do however have certain functions with no equivalent in Excel, such as FILTER. Google Sheets allows for many people to work at once in a minimalized spreadsheet, offering just the essentials, but being able to "add-on" to it.

Even as an add-on, Debate Suite bases itself on a workbook, so its architecture does not shy from that. It uses a format similar to other spreadsheet applications where scores are inputted in the debaters' rows, without much of the hierarchical model presented earlier. However, there is some correspondence between database tables and spreadsheet sheets, such as for the tables of judges or venues. By extension, the combination of calculated fields with the proper data table allows for finer manual analysis of such tables, and is a net gain for the spreadsheet. On the other hand, its integration with Google Forms could provide it a means to collect ballots or other information without necessarily granting permissions to the workbook itself. Being able to process ballot data coming in from a form by the judge gives us an equivalent "Ballots" table and a stronger model, but divorcing some data input from the spreadsheet, although not from the tabulation officials.

Google Sheets is the only spreadsheet that offers an equivalent to macros on the web. The Visual Basic of Excel becomes JavaScript, but it serves the same purpose. Spreadsheet-based applications are normally very script-heavy, supporting a wide range of features, including draws and judge allocation, through scripting these procedures. The quality of the code is somewhat varied, which can lead to errors or can create arbitrary limits on its use. A goal for this spreadsheet is to minimize the amount of code, and to generalize it to a wide extent. The structure of the sheets and the algorithms used are thus somewhat different to equivalent solutions.

Scripts are a "black box," making the spreadsheet less flexible like web applications. Formulæ serve as an important counterbalance. They reduce the amount of calculations that needs to be done through macros, and delegates more work to the workbook built for calculations and its operators. This allows for more manual control, yet less involvement, as these formulæ are calculated automatically. Functions can be used in many situations, from looking up debaters in a team to calculating rankings and scores. Debate Suite then also makes a big assumption that the team with the highest combined score in a debate wins the debate. This allows the Team tab to be generated and updated without manual intervention. This sheet is a good example of the interplay between scripts and formulæ. Once all the debaters are registered,[3] a macro creates the Team tab, inserting formulæ into the sheet to calculate win/loss, scores, and rankings. Scripting is a good general-purpose tool, but should be used sparingly, preferring other features.

On the points brought forth above, the implementation of team protections is a good case-study. There are a few considerations when making random draws. Teams from the same institution should not debate against each other, as should teams who have already debated their opponents earlier in the tournament. This is a clear case of the assignment problem. Microsoft Excel includes a tool called Solver that can find the optimal pairings for each debate. What is needed is a "cost" metric, a badness score for each pair. We can say that debating a team from the same school has a badness of 100, while debating a previously-seen team is 1000. With scores for each pair, Solver can choose a value from each row-column pair to minimize the badness. Google Sheets does not have Solver. What Debate Suite did was implement a linear optimization algorithm that solves the assignment problem as a script. It takes the scores from a matrix in the spreadsheet that uses formulæ to add a penalty if the pair meets a condition. By using formulæ, these costs can be altered by hand, conditions added or removed, or penalties changed. The solution offered by other spreadsheet-based applications seem to be to reshuffle the teams and hope for the best...

CONCLUSION

Web-based applications are fundamentally different to their spreadsheet-based counterparts. However, both do have their place in debate tabulation. It is only now with a new spreadsheet platform that some of the differences between web-based and spreadsheet-based applications fade, giving a richer feature-set for spreadsheets. Now, the priorities when making the choice between these two paradigms of software for running a tournament have shifted to the main question of control and flexibility in the system. Some of the pillars in this dichotomy such as multi-user support fall with the rise of web-based spreadsheets.

In the world of debate tabulation software, Debate Suite is taking this new web-based spreadsheet approach to improve upon the available spreadsheets for this purpose. Avoiding the use of anti-patterns such as the over-reliance on scripts/macros or hard-coding cases like round numbers is of utmost importance in its architecture, while expanding the feature-set and customization available. A deep balance between scripting and formulæ was established for this customization and performance. Further to exploiting the different angle of Google Sheets, available integrations to different systems is used, as it improves data entry and documentation. All of this, with stronger algorithms makes Debate Suite differ quite a bit from both sides.

Still under construction, Debate Suite is a new contender in the debate tabulation software ecosystem, inspiring itself from both paradigms, while re-thinking the usage of spreadsheet macros. Earlier prototypes have been used successfully for the NSDS's 2018 tournaments, with the effect of identifying missing features and parts to improve. A sample workbook with short comments interspersed to show its formulæ and structure is available here [link available online]. Scripts may become available later. Debate Suite will be available shortly as a G Suite add-on.

REFERENCES

- Belesky, Philip, and Chuan-Zheng Lee. 2018. *Tabbycat Tabulation User Guide*. 2.1.0 (8aa1821). https://tabbycat.readthedocs.io/.
- Bruschke, Jon. 2006. *A Primer on Debate Tabulation*. California State University, Fullerton Debate. https://tabroom.com/jbruschke/CATDownloads/TabManual.docx.
- George, Chris. 2013. *CG_Tabs Program*. 4.0 ed. Quebec Student Debating Association. http://www.qsda.org/tabs/CG_Tabs_4.0_Documentation.pdf.
- Hanson, Jim, Dany Cantrell, and Eliot Woodrich. 2013. "PLUSTAB Running a Tournament." 2013. http://www.wcdebate.com/plustab/index.htm.
- Lee, Chuan-Zheng. 2016. "What You Should Know About Adjumo." Stanford University. https://web.stanford.edu/~czlee/adjumo.pdf.
- Palmer, Chris, Jon Bruschke, and Jim Menick. 2018. *Tabroom.com Help Center*. National Speech & Debate Association. http://docs.tabroom.com.
- Petersen, Jeremy. 2008. "Make an N-Tier Architecture and Give Stylish Effect with Ajax & Javascript." 2008. http://krunal-ajax-javascript.blogspot.com/2008/09/benefits-of-using-n-tiered-approach-for.html.

ENDNOTES

1. Allocations are mathematical optimization problems, which Solver computes, relevant throughout tournaments for judge/team/venue allocations (Lee 2016).
2. On the other end of the spectrum, simpler workbooks do use formulæ but not macros, which do still have their place.
3. Worth noting that when first created, the debaters' tab sheet does not contain any columns for rounds; they are procedurally added with each round.