Noah Myers

Citation Count Web Scraper

IS 520 VBA Project

Executive Summary

I work as a research assistant for professors in the School of Accountancy. One of the projects on which I have worked involves updating a database of academic accounting articles, authors and institutions. These professors have created this database of individual accounting researchers and their published works in a subset of top accounting journals since 1990. This database has led to multiple published papers in accounting journals about accounting research and those who do it. This database contains important data about each published article such as the journal, the authors, the date, the volume and issue and the topic area and methodology. However, we would like to add citation counts for each article to the database for future research.

My project imports data from this database on all indexed articles and uses the agent class to control Internet Explorer and gather citations from a web search. It includes controls for checking whether the Journal title matches in order to ensure accuracy and is formatted such that the scraped citation data can be exported again into the database.

Implementation Documentation

Ribbon Modifications

The project is fairly straightforward in its implementation. The excel file scholarScrape.xlsm contains a new ribbon tab called “Scrape” which has a control group “Citations Scrape” with three buttons: Import Articles, Scrape Citations and Update Database which can been seen in Figure 1.

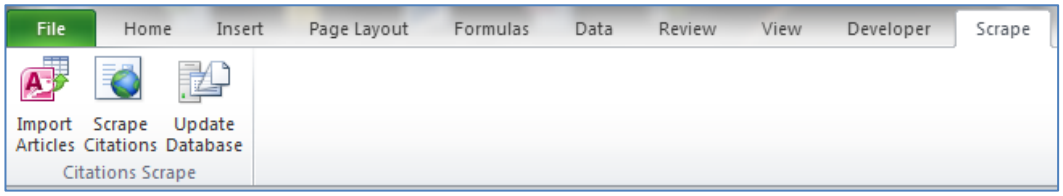


FIGURE 1 – RIBBON MODIFICATION

While I eventually intend to add database connectivity and automate the process of importing articles and updating the citation counts back into the database, this functionality is beyond the original scope of my project proposal. Therefore, only the middle button, “Scrape Citations” is functional for this project. The other buttons simply display a message box alerting the user that this function is not yet complete.

Data

For the purposes of this project, my professor exported the articles in the database to a csv file that I added to a new sheet in my project workbook. The file includes a primary key for each Article, ArticleID, which can be used to merge the data back onto the original database; the title of the article, which will become the search string; and the title of the journal in which the article was published. The format is visible in Figure 2 below. My project file that is available for download contains a subset of these articles for testing purposes in the first sheet of the workbook.

E17

	A	B	C	D
1	ArticleID	ArticleTitle	JournalTitle	
2	3296	Economic Sufficiency and Statistical Sufficiency in the Aggregation of Accounting Signals.	The Accounting Review	
3	3297	The Market Interpretation of Management Earnings Forecasts as a Predictor of Subsequent Financial Analyst Forecast Revision.	The Accounting Review	
4	3298	Experience Effects in Auditing: The Role of Task-Specific Knowledge.	The Accounting Review	
5	3299	Equity Valuation and Corporate Control.	The Accounting Review	
6	3300	Mandatory Versus Voluntary Disclosures: The Cases of Financial and Real Externalities.	The Accounting Review	
7	3301	Accounting Changes and Earnings Predictability.	The Accounting Review	

FIGURE 2

User Form Interaction

Upon pressing the Scrape Citations button, the user form "citationScrapeForm" appears, as shown in Figure 3 below. The form presents the user with a few options, which I will describe in detail. First, the user can choose how many times the program loops. For testing purposes, the user can select 1 or 5 times to see if the program is running correctly and manually compare results with a web search

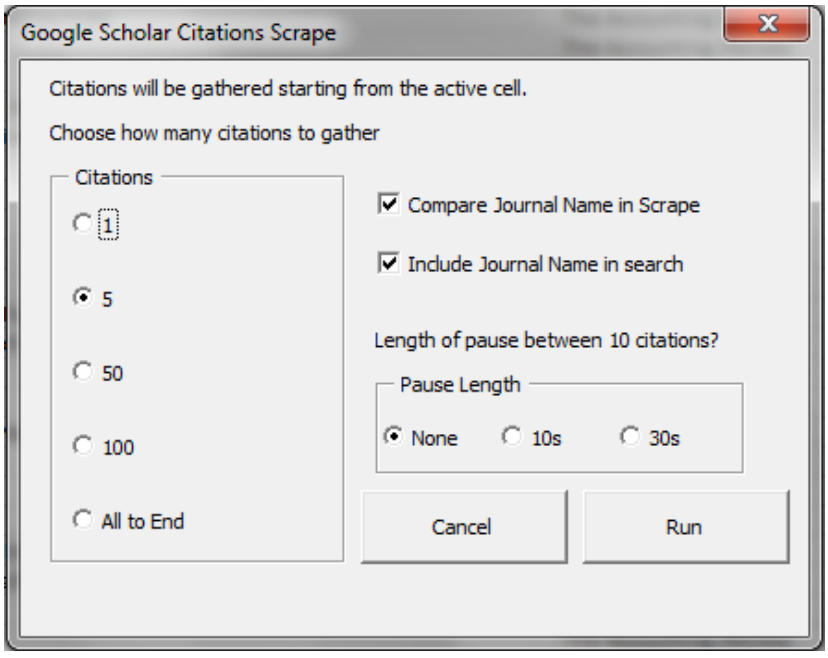


FIGURE 3 – CITATION SCRAPE USER FORM

of their own. When the user is satisfied that the program is working correctly, the user can select 50, 100 or an option “All to End” that will run the program from the active cell until the end of the article listing. This variable is passed as a parameter when the form invokes the scholarScrape module to run the program and is used as the upper limit of the program control loop.

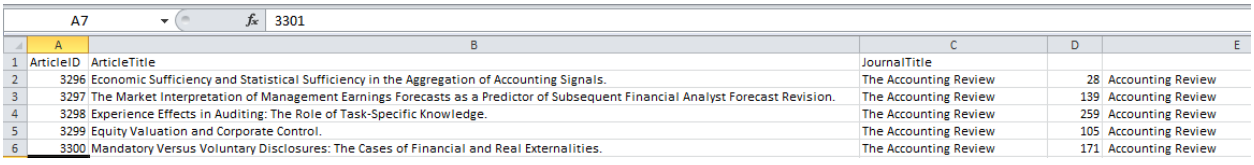
Next, the user has two checkboxes, which both contain default values of true. The first, “Compare Journal Name in Scrape” causes the program to gather the journal name from the citation it has found and compare it with the name of the journal from the database. If there is a mismatch, it returns the string “mismatch” in the cell next to the gathered journal name and citation count. This will allow for manual comparison of journal titles later on.

The second box, “Include Journal Name in Search,” tells the program to not only search google scholar for the string of the article title, but also includes the journal title as a parameter in the google search. This can lead to greater accuracy as results are restricted to articles in the selected journal. Both of these options are desirable by default, hence their defaults are set to true, however there are occasionally cases where the user may not want to compare journal title or where it is optimal not to include the journal title as a parameter.

Finally a user can give the program a specified amount of time to wait after each of 10 loops of the program. Since this is an automated process, Google might detect its automated nature after a few hundred runs; this feature is to help mitigate that possibility.

Program Result

Figure 4 shows the results of invoking the program with the default values on the user form. The fourth column contains the number of interest, the citation count. The fifth column contains the title of the journal from which the citation count was taken. Sometimes the article in question is not the first article in the search result list, and so the citation count will be counterfeit; scraping the journal from which it came is a rough way to verify that the count is at least from an article in the same journal. If asked, the program automatically compares the journal title to the original journal title from the database and displays “mismatch” in column 6 if the titles don’t match. The user can then manually filter for these values and make further comparisons. Finally, the user can make use of the citation values by uploading them back to the original database as an additional field.



A7 | 3301

	A	B	C	D	E
1	ArticleID	ArticleTitle	JournalTitle		
2	3296	Economic Sufficiency and Statistical Sufficiency in the Aggregation of Accounting Signals.	The Accounting Review	28	Accounting Review
3	3297	The Market Interpretation of Management Earnings Forecasts as a Predictor of Subsequent Financial Analyst Forecast Revision.	The Accounting Review	139	Accounting Review
4	3298	Experience Effects in Auditing: The Role of Task-Specific Knowledge.	The Accounting Review	259	Accounting Review
5	3299	Equity Valuation and Corporate Control.	The Accounting Review	105	Accounting Review
6	3300	Mandatory Versus Voluntary Disclosures: The Cases of Financial and Real Externalities.	The Accounting Review	171	Accounting Review

FIGURE 4

Personal Learning and Application

I learned a great deal about VBA and web scraping during this project. As the project related to an actual task that I was working on, and not a contrived scenario, I was very interested in getting the program to work well and spent a lot of time fine tuning the scraping functionality.

The agent class was instrumental in my ability to complete this project. I have been thinking about ways to automate the gathering of citations for a few months and as soon as I saw this functionality in class, I knew that this was a perfect solution. Although the agent class greatly facilitated my ability to complete the project, I had to do a lot of work to parse through the HTML of a google search page and return the desired citation value consistently. Even more difficult was the comparison of journal titles as a control feature.

URL String Creation

The first difficult task was creating the URL String from the article title. Instead of entering a value into the search box on the main search page, I wanted to include the search string directly in the URL to make the process faster. This wasn't too difficult to figure out, but it helped immensely when I learned about the URLEncode function that is included as part of the agent class. Before I was manually replacing spaces with "+" characters, but the URLEncode function does all of that and, I've noticed, a lot more with special characters that come up in some article names.

The code I use also includes the journal title as a parameter in the google search if the option is checked. This is different than simply including the journal title in the search string as it tells the scholar search to restrict this specific search to articles in a single journal. I have noticed mixed results with this option. Sometimes, with certain journals, the accuracy is vastly improved. Other times, including the journal title actually causes another article to rise to the top of the results list. Also, because my validation check (discussed later) involves comparing the journal titles, this option essentially invalidates my check because I am forcing it to be correct.

Finding the Right Spot

The next task involves scouring the resulting HTML page for the correct citation number. Although it is somewhat rough, I assume that the first result always relates to the article I am searching. I include a validation check later on to mitigate the possibility that it is not the correct article. I then simply go to the first spot in the text where I find the string "Cited by" and take the numbers that follow this phrase as the citation number.

For the journal title, this was much more difficult, as it is part of another string that is somewhat varied in content. I used the Mid, Right, Left and Len functions extensively, sometimes nested inside each other multiple times in order to isolate the journal title in such a way that it works consistently across journals. It felt like sifting sand through my hands while trying to hold on to grains of rice within that sand!

Validity Check

I wanted to build in at least one simple way to validate that the citation count I pull is indeed related to the article in question. So, I simply compare the journal title from the search result page with the journal title in the database for that article. If they do not match, I simply write the word "mismatch" in a column on that row. This will allow a user to manually verify this citation after the automated process has finished.

One additional issue with grabbing the Journal title is that Google scholar sometimes truncates the journal title with an ellipsis. For example, *Journal of Accounting and Economics* is sometimes displayed as *Journal of Accounting and …* or *…Accounting and Economics.* Or, even worse, *… Accounting and E…* with leading and trailing ellipses. I had to build in checks to see if the journal string started or ended with this special character and then decide whether to compare the left or the right part of the string.

As you can see, much of what I did with this project involved cleaning up a string in such a way that I could automate it's comparison with another string. As previously mentioned, I much better understand the power of simple functions such as Mid, Left, Right, Trim, LCase and Len in helping to parse out the data you require.

Conclusion

A final caveat—I understand that Google attempts to block automated searches such as those in my project and I have certainly come across this problem. I was planning on using the non-visual method of interacting with Internet Explorer, since my project does not deal with session management; however, this method was clearly a red flag to Google. I reverted to using the visual approach and it seems to work much better, especially when run in small chunks.

I have learned a great deal in this project about working with web pages and parsing the HTML for data in which I am interested. Although this project may not actually be utilized, I am certain that I will find a way to implement the skills I have learned in the future to solve many problems.