Barney Lund
MBA 614
Final Project

Deal-a-day Price Comparison

Executive Summary

Business Description

I'm a big fan of deal-a-day websites in general, but I definitely follow Woot.com more closely than any of the others. If you're not familiar with these types of sites, they exist as an outlet for manufacturers or wholesalers that either want to unload a bunch of excess stock or want to quickly introduce new gadgets and gizmos to the cutting-edge adopter—usually at a pretty steep discount.

This discount is what interests me, because the business idea I have is to find a way to discover and capture extra-large margins. I can think of two ways of doing this, but only one is scalable. My idea is to check the site's past items, compare those items' prices to regular market prices (sometimes this won't apply if this vendor is being used to introduce a product and they are the first to offer it or if it's a mystery bag), and then produce a list of the items that are good deals. If there are exceptional margins to capture, I'll make use of this list to either find buyers who want to resell or be on the lookout for these items during Woot-offs (rapid-fire versions of the deal-a-day offerings).

System Overview

I'm using the power of my new employer's web-scraping tool (www.mozenda.com) to quickly scrape past items offered from archive sites. I wrote a sub procedure that formats the data to be usable as part of a web URL. The next sub searches a pricing index site and grabs the prices listed for comparable items. Finally, the web data is compared to the deal-a-day price, and the final result is list of the items that are considered "deals" (determined by my standards), sorted by margin per unit.

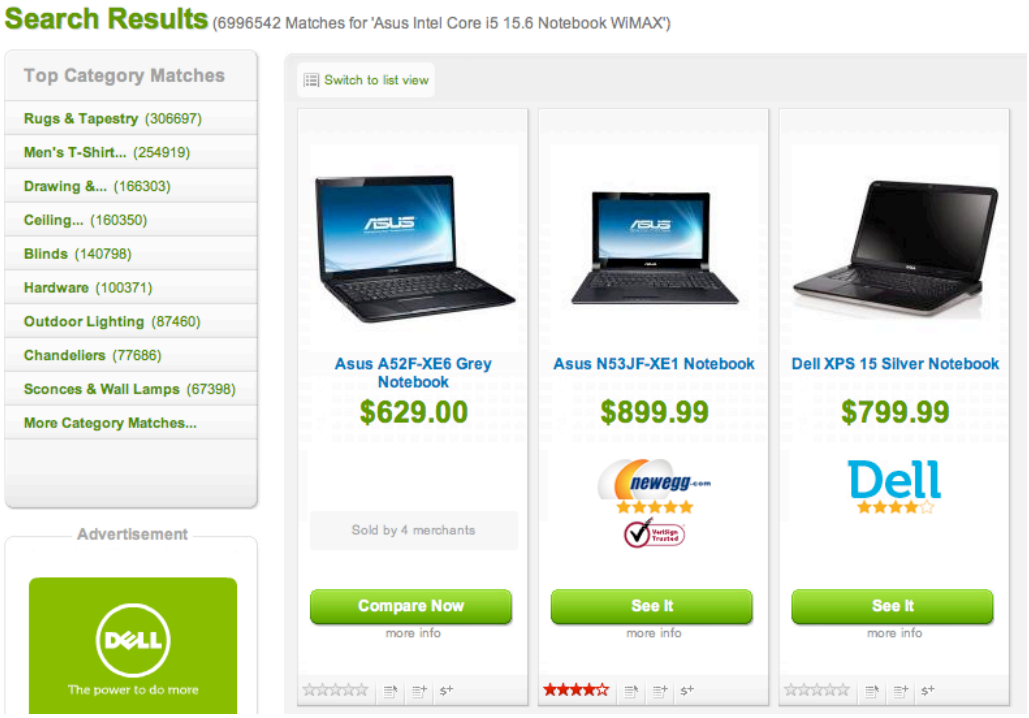
Implementation Documentation

My project ended up consisting of several sub procedures that varied in length and function. Originally, I had a lot more subs, but I found it to be extremely efficient to fit as many things as I could under a sing For Loop.

The first obstacle was formatting the entries in a way that was kosher with the web browser. I found all sorts of characters that *don't* work as parts of URLs. In this single sub procedure, I was finally able to get each search term formatted in a way that would work. My worry was that if I formatted it for the web, the final results wouldn't be very reader friendly, so for each step I took to format the terms that would be searched, I included a step to reformat that string back into a reader-friendly format using a separate sub procedure that would convert things back at the end. I also ran into a problem of not having the column widths wide

enough to see the whole description of each product nor the date it was posted, so I included formatting for those columns here.

I created a web query that went and searched for the item on PriceGrabber.com and returned everything on the page. Luckily, the format of those pages is relatively uniform, so I was able to find keywords that let me find the prices I was looking to compare against rather easily. If the criterion was met for the search term (in this case, it just needed to not have the word “crap” in it [a specialty Woot! Term]), I had the web query create a new worksheet, parse out the first few prices, then run the average price sub procedure.



(Example web query results page, waiting to be parsed)

The average price sub procedure sums up all the prices that I want to include in my comparison and divides that sum by the number of prices included. Picking the right prices (and the right number of prices) to compare was tricky and took quite a bit of testing, but I was pleasantly surprised when I got it to do exactly what I wanted it to do in the end.

Finishing off the web query, I stored the average price in a variable from which I subtracted the deal-a-day price. I created a column called “Savings” and put any positive difference here. If the entry had the word “crap” in it, then I labeled it as such in the “Deal Status” column. When a positive difference occurred, I had it labeled as a “Good Deal”, and if a negative difference occurred, it was labeled “Bad Deal”. Finally, there were some differences that were outrageously high, so I labeled them as “Suspect” for purposes of higher scrutiny later.

Date	Price	Product	Savings	Deal St
4/4/2011 0:00	$89.99	Sanyo Xacti 1080 Full-HD Camcorder with 3x Optical	$299.46	Good d
3/19/2011 0:00	$499.99	Asus Intel Core i5 15.6 Notebook with WiMAX	$248.01	Good d
3/22/2011 0:00	$3.99	Memorex 3' High Speed HDMI Cable 2 pack	$152.83	Suspec
3/18/2011 0:00	$249.99	10 Queen Size Traditional Memory Foam Mattress wi	$115.65	Good d
3/30/2011 0:00	$299.99	Milano Collection 5x8 Persian Style Rug	$92.10	Good d
3/24/2011 0:00	$149.99	Philips Norelco SensoTouch Electric Razor with Gyr	$60.74	Good d
3/27/2011 0:00	$139.99	Ooma Telo VoIP Home Phone System	$50.83	Good d
4/1/2011 0:00	$14.99	IOGEAR Solar Bluetooth Hands-Free Car Kit	$24.01	Good d
3/31/2011 0:00	$14.99	IOGEAR Solar Bluetooth Hands-Free Car Kit	$24.01	Good d
4/3/2011 0:00	$47.99	Microsoft Xbox 360 60GB Live Starter Pack	$23.51	Good d
3/20/2011 0:00	$119.99	Flip SlideHD 16GB High-Definition Video Camera wit	$22.46	Good d
4/1/2011 0:00	$6.99	3-Watt Cree LED Aluminum Flashlight	$18.01	Good d
3/29/2011 0:00	$7.99	Zero Germ UV Toothbrush Sanitizer with Toothbrush	$5.28	Good d
3/26/2011 0:00	$99.99	D-Link Xtreme N Storage Router		Bad de
4/2/2011 0:00	$69.99	Neat Receipts v4.0 Document Management System with		Bad de
4/1/2011 0:00	$3.00	Random Crap - Level 10		Crap

(Example of sorted, formatted results)

As I mentioned above, I reformatted the search terms to make them very readable, then in the filter sub procedure, I applied a filter that shows the largest-margin items at the top of the list, descending in margin value. This final list is what I was after, and I was excited to see it all work. In fact, I almost wanted to remove the lines of code that turned off screen updating, just so I could see the list being populated as it went.

Lessons Learned

I wish I could go back and tell my former self a few things before creating this project. The two words that would have helped the most are “prepare” and “consolidate”. Both have to do with thinking about the big picture. Here are a few of the “gems” I picked up along the way, some of which took way longer than they should have.

1. The webQuery sub procedure was hairier than expected.
    a. The main reason was that I didn’t anticipate (again, that magic word “prepare”) how difficult it might be to format unformatted search terms in a way that lets me use them as part of a URL.
    b. Another bump in the road was the fluctuations in the number of items listed on different search results pages. Tied into that was the

problem that prices went up drastically after a dozen or so items. I countered this by taking just the first few items of each page, with satisfactory results.

   c. I also painstakingly found that random huge numbers (or negative numbers, for that matter) on a web page could really mess things up. I fixed that problem by putting a limit on how large the numbers could be and excluded any negative numbers.

2. One of the trickiest things I had to deal with (and which I greatly appreciate Professor Allen's help with) was a problem I just couldn't seem to solve after dozens of tests. I was trying to keep a count of how many numbers I was summing up so I would have a denominator in order to take an average. I ran into two problems, and one of them was masking the other.

   a. First of all, I had that previously mentioned problem of having a very large number being included in my sum, so that threw the numerator off, but that was an easy catch.

   b. The most difficult thing to figure out was why the denominator was way too big every time. After a few short minutes in Professor Allens' office, we found the culprit: 0-length strings are considered as "isNumeric" values, and I had a whole bunch of those being counted towards my denominator. Once that was fixed, all the calculations came out correctly.

3. As part of the “prepare” advice, I would have told my former self to anticipate the need for formatting everything at once instead of one at a time. It makes life easier to guard against all potential errors in the first place instead of trying to put out individual fires by singling out problems that might (and certainly did) come along (especially with special characters in URLs).
4. I consolidated a ton of subs into far fewer subs. I could probably do it even more, but I like how clean things currently look.
5. Putting everything into one master For Loop was daunting, but extremely helpful. I kept noticing that I was writing similar code to do different things to the same set of data. All the For Loops were getting probably taking an extra bit of time to compile, but they were especially getting lame to write over and over. I was previously a bit afraid of putting too much into a single For Loop, and now I don’t know why. Consolidating became my best friend as I was able to make a much cleaner program, and I knew that if the loop was working for one thing, it was working for all of them.
6. I learned that there were certain things that really don’t matter if they look good. In fact, each time I run the whole thing of code, I’m actually creating *and deleting* 100 worksheets—none of which need to be formatted, especially since screen updating is turned off all the while.
7. Testing web queries is extremely time-consuming when there are bugs, so it is tempting to only test a couple entries at a time, but it was worth it to test large batches at once to find out what was going on. Looking back,

however, I probably shouldn't have run as many tests as I did before asking, "What *could* go wrong?" and trying to fix those things first (since they all *did* go wrong).

8. Finally, it felt good to get to the point where I was staring at the results I wanted. It felt even better when I was able to duplicate those results over and over. This gave me motivation to keep up my drive to start *and finish* future VBA projects, because I know I've been able to do it before.