SocializeMe Project Report

Excel VBA Application

Joshua Lyman

Executive Summary

As more and more people begin to discuss and engage with companies and brands online through social media, the importance of knowing how a brand is perceived is critical. Executives and decision makers need clear breakdowns of how their brand is performing so that they can make decisions. This VBA application entitled SocializeMe aggregates statistics from the social media fire hose, gathers reviews of the brand, and compiles it into a historical snapshot report that is then emailed to the client. Using this week-by-week analysis, executives will be able to see at a glance how they are faring on the social web, and make appropriate strategic decisions accordingly.

I run a small web design and development firm, and I have been looking to offer some sort of social media monitoring service to my clients for a while. I had actually tried to build such a service in native programming language before (Ruby) but had run into issues with pages being half loaded, etc. When I learned about the ability for Excel to control Internet Explorer via proxy, and combined with its ability to organize information and charts, it seemed the perfect solution. Once a report is generated for the week, the main report is exported as a PDF and immediately emailed to the client. In the future, I could also automate the process of opening the file and running the report, so that I could do the entire process unattended.

Implementation

The SocializeMe project is divided into four major components: the report sheet, the historical data sheets, the scraping engine, and the printing/emailing methods.

Report Sheet

This is the front-facing component of the application. It can be reviewed from within Excel, but it's objective is to look good when printed or as a PDF document, sort of like a one page dashboard that gives an at-a-glance breakdown of brand performance. Much of the data on the page is pulled from the two historical data sheets also in the Excel workbook via a creative formula that grabs the last few values for display. The chart on the report sheet is dynamically modified by the application at runtime to include the latest data sample points. Again, while it doesn't look like much from within Excel, it looks much better printed or in PDF format, as shown in the image.

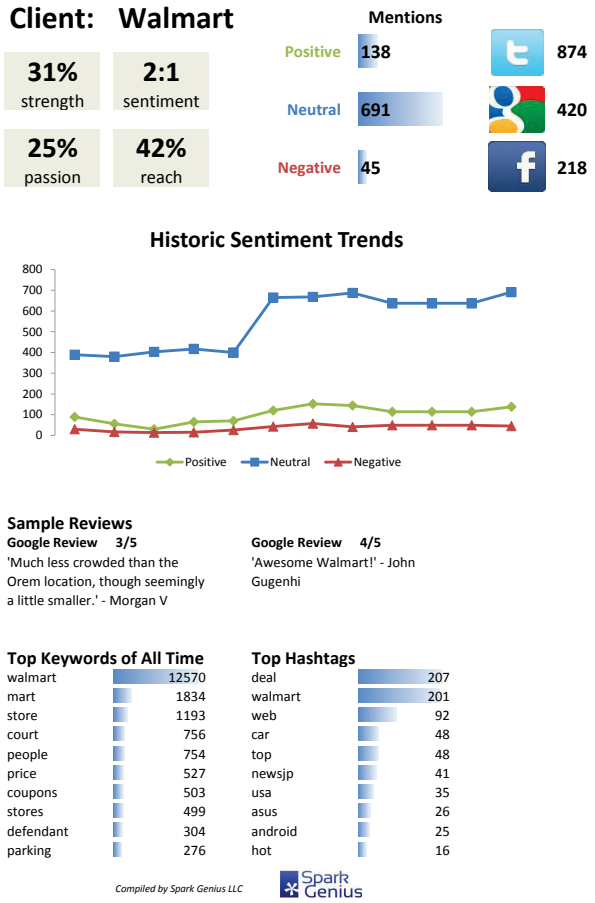

Figure 1 - Final report generated by the SocializeMe application and sent to client

Historical Data Sheets

Because the report is designed to show current versus past performance, it is obviously key to store data from previous reports. There are two types of data being utilized: data points that are one-time events (number of social media mentions that week) and data that is cumulative (popular keywords this week combined with past weeks). Therefore, two sheets hold the two different kinds of data.

The “Data – Historical” sheet holds the data that is unique for each week, and is therefore a running table of information. The “Data – Cumulative” sheet combines data that should be aggregated over time. New data that is added to the “Data – Cumulative” sheet is checked to see, for example, if the hashtag used already exists, and if so, adds the weekly hashtag count to the overall hashtag count. If not, it adds it to the bottom of the list. These lists are all then sorted in descending order to allow the reports sheet to grab the top ten results from all of the categories.

Scraping Engine

This is, of course, where all of the magic happens. The engine, in combination with the Agent 99 class, provides interactive input on two sites to gather results specific to the client that is indicated on the report sheet, and then parses, stores, and calculates statistics on the collected data. The operator simply needs to open the workbook once a week, run the `loadSocialResults()` method, and let the VBA app do the rest.

To provide a general overview of the process, the following general system diagram has been created, showing the major components of the scraping engine.

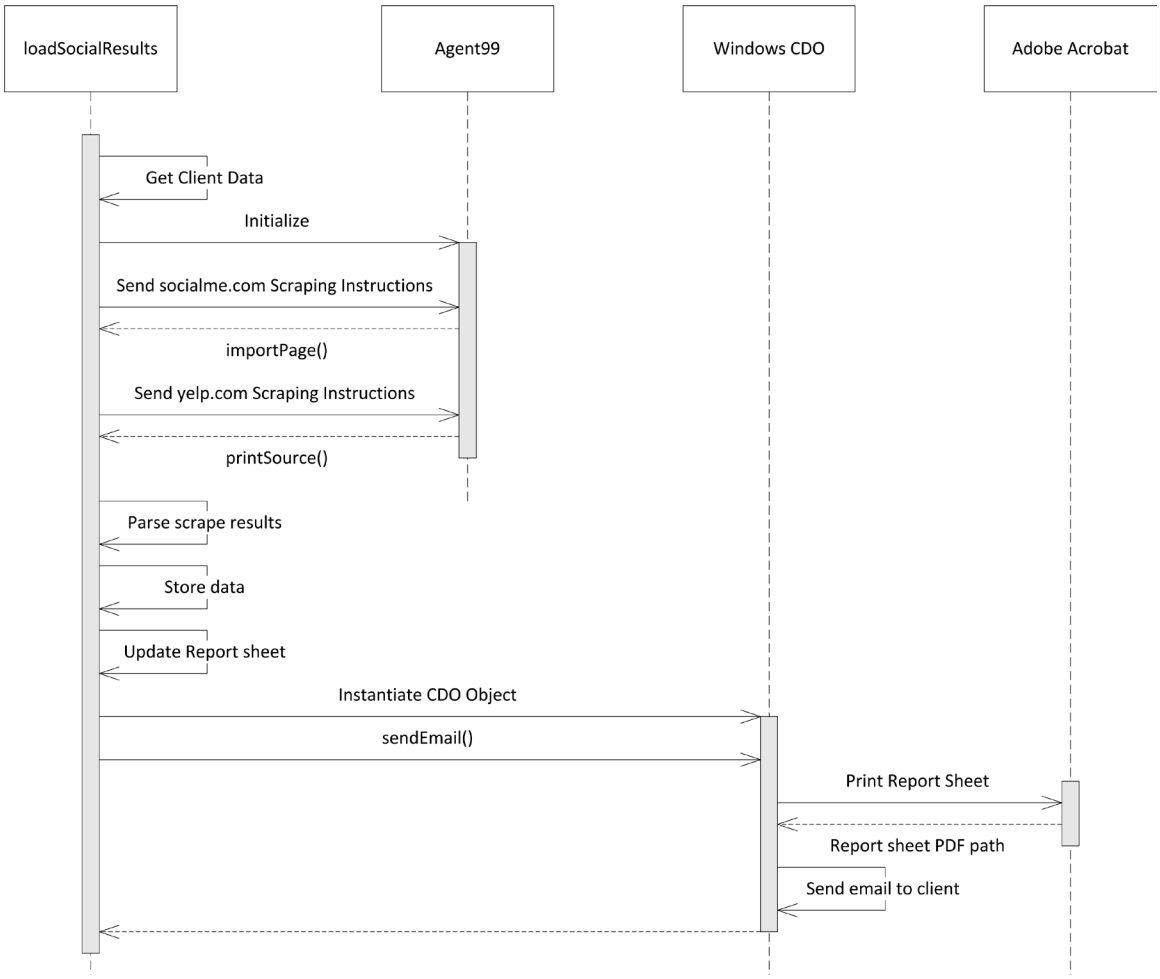

Figure 2 - System diagram showing basic flow of SocializeMe main method.

In short, the data that is pulled from the websites thanks to Agent99 is parsed to gather the various details necessary, using both a series of `finds` to locate the appropriate data in the source code, and by reading the plaintext version of the page from the `importPage()` method. Once the data is gathered, several statistics are generated from the data, including the following:

- Sentiment: a ratio of the positive to negative mentions of the search query
- Strength: the likelihood that the query is being discussed in social media
- Passion: based on the percentage of people who mention you multiple times
- Reach: the number of unique authors divided by the total number of mentions

Pulling in all of the data from the sources, from social media mentions to reviews of the brand on Yelp, the report sheet is updated to show the latest and greatest results.

Printing & Emailing Methods

In order to provide the easiest possible viewing experience for the client, it was decided that the final report should not be made available as a macro-enabled Excel file, but instead as a PDF, thanks to its accurate representation and ease of use. At first I tried to find ways to export directly to PDF, but everything was overkill. Then I realized that I could simply print the file directly to a PDF, and one recorded macro later I was halfway there! The PDF is exported to the same folder as the workbook, and labeled with the client name and today's date. This way, historical records of each report can also be kept.

The emailing method uses the CDO object that we discussed in class, and attaches the PDF file directly to the email. The email is sent to the client. Should any error occur, the error is caught and reported to the user. Otherwise, the user is notified of the success, and everyone cheers!

Learning and Difficulties Encountered

Perhaps the biggest issue that I had when working on this project was wanting to do things the "fancy" way, failing, and then realizing that the simple way was actually better, easier to implement, and worked just as well.

For example, I had a great deal of difficulty trying to get the Historic Sentiments chart to update after importing new data. I googled for hours, tried every different combination of methods I could think of, and even attempted deleting the object, recreating it, and modifying all of its properties to get it to look correct. This wasted a lot of time in the end, and I still didn't have a solution for simply updating the chart. After I asked Dr. Allen how one might go about it, he revealed that charts are actually stored *inside* of shape objects. And with that key, I was off to the races. I was then able to modify the charts as necessary, and use some very simple methods to get it to do what I wanted. I realized that I had not been finding the answers I was looking for because I had assumed the

patterns for accessing the chart were much more complicated than they were, and I had come up with “fancy” ways of changing the data. However, the KISS principle was and holds true, even in Excel.

Another example of a good learning experience I had was in my efforts to organize the cumulative data such as top keywords and hashtags. I only wanted to show the top ten most popular from each category, but needed to maintain a listing of all of the data and their usage counts for the past. I had at first coded the method to grab all previous values, store them in an array, and then perform push/pops to check and resort the data. However, finding Excel lacking in some of these areas, I took a second look at it and came up with a much simpler, cleaner (and in the end faster) approach that simply involved iterating over a list on a sheet, checking for pre-existing entries, and then using Excel’s built in sort features to sort the list and make it available to the reporting sheet. Once again, keeping it simple turned out to have the best results for the VBA application and taught me a lot in the process.

An example of this code is included here:

```
Sub appendHashtags(hashtags, count)
    Dim currCell As Range
    Dim found As Boolean
    Dim i As Integer
    Sheets("Data - Cumulative").Activate

    For i = 1 To 10
        found = False
        For Each currCell In Range("F2", Range("F2").End(xlDown))
            If hashtags(i) = currCell.Value Then
                currCell.Offset(0, 1).Value = _
                  currCell.Offset(0, 1).Value + count(i)
                found = True
                Exit For
            End If
        Next currCell

        If Not found Then
            Range("F2").End(xlDown).Offset(1, 0).Value = _
```

```
                hashtags(i)
            Range("G2").End(xlDown).Offset(1, 0).Value = count(i)
        End If
    Next i
End Sub
```

Practical Applications and Conclusion

Because this was a business problem that I was looking to solve for myself, I am eager to test it with a couple of my more forward-thinking clients. I believe the proper use of social media can be a boon to just about any business, and monitoring it is key. I am very glad that I found the power of Excel over the course of the semester, and have been able to implement this project as a solution where other attempts have failed.

This project may also prove useful for others looking to monitor their own social media reputation, or for those looking to learn from it. I know I certainly did, and am excited to use it in the future!