Paper 315-2011

Social Networking and SAS®: Running PROCs on Your Facebook Friends

Chris Hemedinger, SAS Institute Inc, Cary, NC
Susan Slaughter, Avocet Solutions, Davis, CA

ABSTRACT

Social networking sites such as Facebook, LinkedIn, and Twitter offer tremendous opportunity for people to share information with each other. This information is simply data, organized in a friendly form for your use on the Web. It's also great raw material for interesting analysis in SAS® – if you can get it there. This paper describes how to use SAS and SAS® Enterprise Guide® to extract data from social network sites using published APIs, with Facebook and Twitter as examples, and then perform basic analysis on the results.

INTRODUCTION

Social networking Web sites and applications are the fastest-growing component of the Internet. They allow us to connect (or reconnect) with friends and colleagues around the world, and easily share news, photos, links of interest, and personal information. The most popular site (as of early 2011) is Facebook, with over 500 million users. LinkedIn, which is used primarily for professional network connections, has over 65 million users. Twitter, the extremely popular Web site for microblogging with its 140-character limit, has well over 100 million users.

With such a tremendous amount of activity within these and other social networking services, there must be a huge amount of data that is generated, collected, and stored. Sometimes this data is used for obvious commercial purposes. For example, the Facebook Web site presents targeted advertisements that reflect the user's age, location, and other characteristics that have been shared with the site. Other times, the data is used for the site's own "internal growth" purposes: to encourage you to connect with specific users that share some of the same interests and history that you have, and thus increase your use of the service.

Within this paper, we present different techniques that you can use to extract data from social networking sites and import it into SAS data sets. You can then use the powerful analytics and reporting capabilities within SAS to gain new insights into the network connections that you've made, and get a glimpse of what the social networking sites might be able to infer *about you* based on the information that you share.

USING PROGRAMS TO GATHER SOCIAL NETWORKING DATA

When you log into a social networking service by using your web browser (or even by using another device, such as smart phone), you can view information about your own profile and connections. Once you've accumulated more than a handful of connections, it can be difficult to see a view of all of your connections at one time. By using a program to access the service on your behalf, you can easily gather multiple "screens" worth of data.

There are different ways to programmatically access data from a social networking service. The most popular method is to use an application programming interface (API) that is published and supported by the service. For example, Facebook publishes an API that is feature-rich, and allows you to read your profile and connection information (as well as to publish information) programmatically. The API is well-documented and easy to use, as evidenced by the many thousands of Facebook applications that have been produced by people around the world. (If so many people can create Facebook applications – not necessarily all them very good – the mechanics cannot be that difficult.)

Another programmatic approach to gathering data is to use the HTTP protocol to issue web commands. In this case, your program is acting like a web browser, and then you can programmatically interpret the results. This method works well for sites such as Twitter that can surface lots of public data without having to authenticate. (That is, you don't need to log in and prove who you are.)

This paper presents two example applications, both of which are available to you. See "How To Get These Examples" at the end of this paper for download details.

GATHERING DATA FROM FACEBOOK

For the purposes of this paper, we developed an example application that uses Microsoft .NET to connect to Facebook (authenticating with OAuth, a standard for Web authentication), retrieve data for your Facebook profile using the Facebook API, and then produces a complete SAS program. The example application can be used as a custom task within SAS Enterprise Guide 4.2 or 4.3, and can also be used as a standalone application.

This paper does not go into deep technical detail about how to develop a Facebook application. However, the source code for the example application is included as part of the ZIP archive, which shows one approach to using Microsoft .NET to create a Facebook application.

The general steps for developing a Facebook application are:

1. Log in to the Facebook site and register as a developer.
2. Use the Facebook developer Web site (http://developers.facebook.com/) to register a new Facebook application.

   The main requirement here is to give your application a name. In return, Facebook will assign an application "key" (a unique 32-character identifier) to your application. When you build your application, you will use this key within the API calls so that Facebook and the end user will know which application is using the service. Facebook will be able to track the use of your application as you and other people use it.
3. Develop your application using the technology of your choice.

   Facebook supports the use of a number of technologies and programming languages, including PHP, Python, JavaScript, and mobile technologies such as iOS and Android.

Facebook supports a few different styles of APIs, but the most popular and the most flexible style is called the Graph API, which allows you to send API requests for specific data using HTTP. The responses to API calls are returned as structured data using JSON (JavaScript Object Notation). Interpreting JSON requires use of special development libraries, which are readily available for many popular development technologies. Alas, there is not currently a way to parse JSON with the SAS language, so you must transform the JSON results into a form that SAS can interpret.

The .NET application transforms the JSON results into SAS DATA steps with DATALINES statements that contain the data records.

EXAMPLE OF JSON TO DATA STEP

Here is an example JSON response that the application receives when requesting information about a user's education history:

```
"id" : "21801765",
"schools":
{[
  {
    "school": {
      "id": "110451125642042",
      "name": "Chenango Forks High School"
    },
    "type": "High School"
  },
  {
    "school": {
      "id": "103788109660536",
      "name": "SUNY Cortland"
    },
    "year": {
      "id": "121127321230671",
      "name": "2006"
    },
    "type": "College"
  }
]}
```

Within the .NET application, we can parse the JSON result and transform it into SAS DATA step statements with the equivalent data lines, like this:

```
data schools;
length
  UserId $ 15
  SchoolId $ 15
  Name $ 50
  Type $ 20
  Year 8
  ;
infile datalines4 dsd;
input UserId SchoolId Name Type Year;
datalines4;
"21801765","110451125642042","Chenango Forks High School","High School",
"21801765","103788109660536","SUNY Cortland","College",2006
;;;;
```

ANALYZING DATA FROM FACEBOOK

The example application that we provide can connect to Facebook and gather the following details from your Facebook profile:

- A list of your Facebook friends
- For each friend, a few details such as birthday, education history, and gender
- A list of recent status updates that your friends have posted.

These details are arranged into a SAS program that, when run, will yield a several data sets that are ready for analysis. The data sets include:

- WORK.FRIENDS, which contains the "friend" full name and an internal ID that Facebook uses. This internal ID is a useful key to join this table with others.
- WORK.FRIENDDETAILS, which contains more details about each friend, including first and last name, gender, birthday fields, URL for the Facebook profile page, and relationship status (for example, "Married", "Single", and so on).
- WORK.STATUS, which contains a list of recent status messages (within the past two weeks) shared by your friends.
- WORK.SCHOOLS, which contains a list of schools that your friends have attended, including the school name, type ("High School", "College", "Graduate School") and the year graduated, if any.

PREPARING DATA FOR REPORTING

Each data set contains a column named UserID, which represents the unique ID for a Facebook user. We can use this field as a key to help combine tables that have one-to-many relationships. For example, because each Facebook user might have listed more than one school in his/her education history, we can combine the FRIENDS table and the SCHOOLS table with the following PROC SQL code:

```
proc sql;
   create table work.schoolfriends as
   select t2.name,
          t1.name as schoolname,
          t1.type,
          t1.year
      from work.schools t1 left join work.friends t2
         on (t1.userid = t2.userid)
      order by t1.schoolid;
quit;
```

Using the birthday-related fields in FRIENDDETAILS, we can attempt to calculate the calendar birthdays for Facebook users, as well as their ages. However, because many Facebook users do not share their complete birthday information (including birth year), we have to account for missing values. The following program prepares a

new table with calendar birthdays (formatted as DATE5. to omit the year from display), as well as friend ages, if possible.

```
data birthdays (drop=year);
 set friendDetails
     (keep=LastName FirstName Gender BirthdayYear BirthdayDay BirthdayMonth);
 length Birthday 8 Age 8;
 format Birthday date5.;
 format Age 6.2;
 if BirthdayYear = . then
  year = 1899; /* placeholder */
 else year=BirthdayYear;
 Birthday = MDY(BirthdayMonth, BirthdayDay, year);
 if BirthdayYear = . then
  Age = .;
 else Age = yrdif(Birthday, today(), 'act/act');
run;
```

CREATING REPORTS THAT PROVIDE INSIGHT

The SAS program also produces basic report output, including frequency tables and charts of friends by gender and friends by relationship status, lists of birthdays, and statistics on the calculated friend ages. However, you must be careful when drawing conclusions about your entire population of friends, since many Facebook users choose to not share certain elements of personal information.

Determining Gender Distribution

For example, Figure 1 shows frequency chart created with PROC FREQ and ODS GRAPHICS, illustrating the breakdown of gender for one set of Facebook users. At first glance, it looks like there are slightly more males than females in this group.

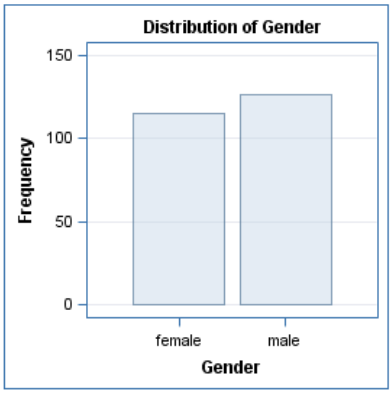

Figure 1. Distribution of gender for one sample of friends

But we also have to take into account the missing values, because not all Facebook users publicize their gender. The table output from PROC FREQ shows a more complete story, as seen in Figure 2. With 83 out of 324 friends showing a "missing" gender, it's difficult to draw a solid conclusion about the gender makeup of this set of friends.

Count of friends by gender

Gender	Frequency	Percent
	83	.
female	115	47.72
male	126	52.28

Frequency Missing = 83

Figure 2. Distribution of gender that includes the missing values

If you are lucky enough to have SAS Data Quality Server available, you can use the DQGENDER function to reduce a few of the unknowns. The DQGENDER function uses data quality algorithms to determine the most likely gender for a given name field. Here is an example program:

```
options dqlocale=(ENUSA); /* reliable for English names */
data friendGenders;
  set friends;
    length calcGender $ 10;
    calcGender = dqgender(Name,'Name','ENUSA');
run;
proc freq data=friendGenders;
  table calcGender / nocum;
run;
```

When this code is run across the same set of friends from the previous example, we are able to reduce the number of people of "unknown" gender from 83 down to 41, as shown in Figure 3. And if we supplement this result with the "published" values of gender that have been shared by these friends, it's possible to reduce that uncertainty even further.

Friends by Gender calculated with DQGENDER

calcGender	Frequency	Percent
F	121	37.35
M	162	50.00
U	41	12.65

Figure 3. The results of DQGENDER can resolve ambiguities

Determining the Age of Your Friends

One of the more popular social interactions on Facebook is the "birthday message." While most Facebook users share some of their birthday information with friends, many choose to not share their birth year. The data collected by our example application includes birthday information, if available. For those friends that share their birth year, we can calculate their current ages and summarize these, as shown in Figure 4.

Analysis of Friend ages

Analysis Variable : Age					
Minimum	Maximum	Mean	Median	N	N Miss
15.8356164	76.1369863	43.6651764	42.7379295	99	227

Figure 4. PROC MEANS output of known ages

As you can tell by the "N Miss" field in the report, most of these Facebook users do not share detailed birthday information – only 99 out of 326, in this case.

However, because Facebook is often used by people who want to connect or reconnect with classmates from high school or college, many more Facebook users *do* share their education history. The education history includes the name of the schools that a person attended, and the graduation year. By collecting this education history, and then making an assumption about the likely age that a person would have graduated from high school or college, we can estimate the age of each friend, and summarize accordingly. Figure 5 shows a new report that takes education history into account.

Calculated Ages based on Graduation years

Analysis Variable : age						
How determined?	N Obs	Minimum	Maximum	Mean	Median	N
College	157	19.0000000	72.0000000	43.1719745	43.0000000	157
High School	160	16.0000000	69.0000000	44.7750000	43.0000000	160
Published	99	15.8356164	76.1369863	43.6651764	42.7379295	99

Figure 5. Estimated ages based on education history

The "N Obs" field shows how many observations contained non-missing values for the calculated ages. As you can see, by inferring an estimated age by using education history, we can consider more records of data. However, it's also possible to calculate the wrong age for a person who didn't follow the typical pattern of graduating high school at age 18, or college at age 22.

GATHERING DATA FROM TWITTER

Twitter is a popular microblogging service. Participants can issue updates, or "tweets", that are at most 140 characters in length. These tweets are largely unstructured text data, but there are conventions of use that help to lend some structure to the content. For example, Twitter users with a common interest often agree to use a common hashtag (short word or abbreviation preceded by the # symbol) to identify the topics they tweet about.

The use of hashtags makes it possible to search for Twitter content that pertains to a particular theme. For example, to search for tweets about SAS Global Forum 2011, you can search for "#SASGF11", the agreed-upon hashtag for this event.

For the examples in this paper, we developed a SAS program that issues Twitter search commands via the FILENAME URL statement. The response supplied by Twitter.com is an RSS feed. RSS, or "real simple syndication", is an XML format that many Web sites use to surface social media information within web browsers or information aggregators. Twitter, blogs, and even the support.sas.com web site use RSS feeds to provide streams of content that changes frequently.

For example, the following URL tells Twitter.com to return one page's worth of "tweets" that pertain to SAS Global Forum 2011:

```
http://search.twitter.com/search.atom?lang=en&q=%23SASGF11&page=1
```

Each page contains about 15 tweets. To collect more tweets that match the search tag, you can increment the `page=1` portion of the URL. For example, use `page=2` to get the next most recent 15 matching tweets, `page=3` to get the set after that, and so on.

The SAS macro language provides the perfect approach to facilitate this repetitive step of retrieving content. Here is part of a SAS macro program that uses the FILENAME URL method to request data from Twitter, and then formats it into a SAS data set:

```
/* this macro makes it simple to get several "pages" worth of tweets */
%macro getTweets(pages=5,hashtag=sasgf11,scale=HOURS);
%do pgNo=1 %to &pages;
  %let
    feed="http://search.twitter.com/search.atom?lang=en&q=%23&hashtag.&page=&pgNo";
  filename twit URL &feed
     /* if you need to specify a proxy server to get to the internet */
     /* proxy="http://myproxy.host.com" */
  ;
  /* use the XML library engine */
  libname tf XML xmlfileref=twit xmlmap=twsearch;

  data work.feed;
     /* provides status updates when run in SAS Enterprise Guide */
     sysecho "Fetching tweet page &pgNo of &pages";
     set work.feed tf.entry;
  run;
%end;
/* more processing here…*/
%mend;
```

Because an RSS feed is XML, it's possible to read it into SAS by using the XML library engine. The XML library engine makes use of an XML "map" file to provide the rule for how to transform different XML structures into relational data tables. The easiest way to create an XML map file is by using SAS XML Mapper, a standalone application that is provided with SAS Foundation.

With SAS XML Mapper, you can open an XML data source (such as the RSS response from a Twitter search), and map meaningful data columns from the content of the XML. Figure 6 shows an example of the table structure that you can create within SAS XML Mapper.

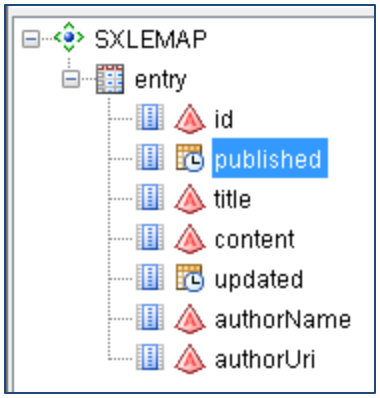

Figure 6. A table schema created using SAS XML Mapper

ANALYZING DATA FROM TWITTER

After the data are in a SAS data set, it's easy to create simple frequency reports. For example, we can see how many tweets were generated in a given time period, and who the most frequent tweet authors are.

For example, we can use the SGPLOT procedure to create a frequency plot of tweets over time that match the hashtag. Here is an example of the SAS program:

```
ods graphics / height=500 width=800;
proc sgplot data=work.feed;
  vbar &scaleVar; /* hours, days, or minutes */
  yaxis LABEL="Number of tweets";
  xaxis discreteorder=data;
run;
```

At the time of this writing, we are approaching NFL Super Bowl weekend in the United States. There are many tweets that use the hashtag "#superbowl". Running the SAS program to query for Twitter results yields quite a bit of Twitter activity on this topic. Figure 7 shows an example of the output of a recent analysis.

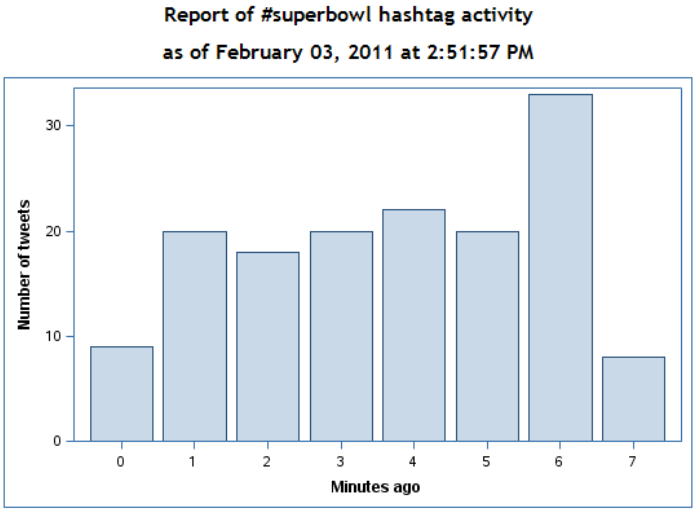

Figure 7. Example frequency plot of topical tweets

For more sophisticated analysis, it's possible to use SAS Text Miner to dive deeper into the content of the tweets using text analytics. For example, instead of simply counting the number of Super-Bowl-related tweets, we could examine the content to measure mentions of the Pittsburgh Steelers versus the Green Bay Packers, the two teams that are scheduled to play in the Big Game. That discussion is beyond the scope of this paper, but Richard Foley and his coauthors cover this quite well in their paper, "Listening to the Twitter Conversation" from SAS Global Forum 2010 (see References).

HOW TO GET THESE EXAMPLES

You can download these examples from the SAS support site.

The example Facebook application is available for download here:

http://support.sas.com/documentation/onlinedoc/guide/examples/SASGF2011/SasFacebookTask.zip

The instructions for how to use it are within the README.txt file in the ZIP archive.

The example SAS program for collecting Twitter content is available here:

http://support.sas.com/documentation/onlinedoc/guide/examples/SASGF2011/TwitterSearch.sas

CONCLUSION

Social media providers and the people who participate in their networks can generate a tremendous amount of data. The techniques within this paper show how you can gather a bit of that information for yourself, and use the power of SAS to generate additional insights.

The mechanics of collecting the data is a small part of analysis. The larger challenge lies in understanding what data to collect, and then formulating questions that can be answered by analyzing these data.

REFERENCES

Albright, Russell, Foley, Richard, and Devarajan, Ravi, 2010. "Listening to the Twitter Conversation", *Proceedings of the SAS Global 2010 Conference*. Available at http://support.sas.com/resources/papers/proceedings10/355-2010.pdf.

ACKNOWLEDGMENTS

The Facebook example application uses the Facebook C# SDK to gather data using the Facebook API; the SDK is available from http://facebooksdk.codeplex.com. It uses the Json.NET library to parse the JSON-formatted responses. Json.NET is available from http://json.codeplex.com/.

RECOMMENDED READING

For more information about how to use the XML LIBNAME engine and SAS XML Mapper, see *SAS9.2 XML LIBNAME Engine: User's Guide.* It is available online at:
http://support.sas.com/documentation/cdl/en/engxml/62845/HTML/default/viewer.htm

For more information about what you can do with SAS Data Quality Server, see the SAS product page online at:
http://www.sas.com/data-quality.

For more information about the JSON standard, see http://json.org.

ABOUT THE AUTHORS

Chris Hemedinger is a software developer in SAS R&D and is also a co-author of *SAS For Dummies*. Chris may be contacted at:

Chris Hemedinger
chris.hemedinger@sas.com
http://blogs.sas.com/sasdummy
Twitter: http://twitter.com/cjdinger

Susan Slaughter is a co-author of *The Little SAS Book: A Primer,* and *The Little SAS Book for Enterprise Guide* which are published by SAS Institute. Susan may be contacted at:

Susan J. Slaughter
susan@avocetsolutions.com
http://www.avocetsolutions.com
Twitter: http://twitter.com/susanslaughter