Paper AA-06-2015

Base SAS Sentiment Analysis Using Catchprases

Mike Tangedal, Capella University

Abstract

The social media revolution has increased the level of opinion-based free-form text available for processing. The first consideration when tackling this text for means of sentiment analysis is first noting what constitutes a valid entry. Base SAS is quite adept at parsing text into individual words via help through Perl functions. Matching these individual words and word stems to credible lookup tables noting parts of speech and happiness rating is now simpler due to emphasis on text mining in the analytics community. Noting the percentage of types of words such as adjectives within text entries provides insight just as noting the percentage of positive or negative words based on the happiness rating. Sarcasm is a difficult challenge that can be mitigated somewhat though notation of affection and negation words. The final key to this method of sentiment analysis is through the building of emotionally charged multi word catchphrases. Once the target audience of the sentiment analysis agrees upon the positive or negative tone of these phrases, an assignment of sentimentality is a measure of matching the number of catchphrases to the text entry and then confirming an agreed upon ratio to confirm negativity or positivity.

Introduction

Just ten short years ago, social media was a burgeoning industry and text mining as we know it today was nonexistent. This was unfortunate for the top social media company of the time, MySpace. Had they had both the means to apply text-mining algorithms to the data readily supplied by their user base and the will to shift their business model to adapt to the rapidly changing climate, MySpace might still be at the forefront of social media.

This is not to say that the current kings of social media have any more insight than MySpace had back during its peak, but they certainly have incorporated text-mining analytics into their business model, as have any organization dependent upon deriving sentiment from their customers.

The current ubiquitous influence of social media introduces unique challenges for sentiment analysis software. SAS® launched a sentiment analysis package in 2013. However the continuing challenge imposed by social media is not algorithmic but contextual. Context is an ever evolving algorithm.

Source Data

The unique initial step in processing source data for sentiment analysis is definition of a valid response. No sentiment insight can be derived from a data stream clogged with just responses of "I don't know" or "no comment". The source data for this paper consists of survey results from students at an online university. For our purposes, we treat these short phrases of non-relevance the same as other non-responses. Our purpose is to discern sentiment from valid responses, not to measure response rate.

Defining what constitutes relevant responses is a simple matter of frequency mapping of the source data. First, since the focus is simple words and word phrases, all text is converted to lower case for simplicity. Running 'Proc Freq' using the option 'order=freq' is a simple means to build a frequency table. The results of this frequency table can be manually searched for responses deemed non-relevant and these can be excluded when retrieving the source data.

```
Proc freq data = mydata order=freq;
      Table survey_results/noprint out=frq;
Proc print data = frq (obs=50); run;
```

Parsing Source Data

Once non-responses and non-relevant responses have been removed from the source data, the remaining responses can be parsed into individual words. Parsing free-form text into individual words is a straight-forward process with a few caveats.

First and foremost, SAS makes assumptions when presented with what is allegedly free form text. For example, if the contents of the source data are stored as HTML files, XML, or have tag associations with other software, SAS will attempt to convert this into a text string with often undesirable results. This paper will not address data conversion issues. For all intents and purposes, the text to be parsed will be presumed to be alphanumeric.

Given this, punctuation within the text can present problems. I've found the most efficient solution to conversion is through Perl scripts utilizing the 'prxchange' function. Here is SAS code I have used to quickly remove punctuation from free-form text.

```
striptext=prxchange(prxparse("s/'//"),-1,text_response);
striptext=prxchange(prxparse("s/[.?!:\x22]/ /"),-1,striptext);
```

The first line removes single quotes from contracted words and the second line converts common punctuation to spaces.

Parsing is then a simple matter of building a loop to traverse each word in the text string.

```
do count = 1 to countw(striptext);
      word = compress(strip(scan(striptext, count)));
      end;
```

Given parsed words, code can be added to this simple loop to set flags. Often, keywords specific to purposes of analysis exist. For our example, flagging occurrences of the name of our organization is desired. All manner of keyword flags can be set within this loop at a minimal cost in terms of storage and processing time.

Master Word List

This parsed list of individual words requires associated sentiment values and other associations. This is accomplished through matching a master word list. In our case, we utilized a master word list containing associated 'happiness' ratings. The non-profit organization PLOS (Public Library of Science) recently published a paper named "Temporal Patterns of Happiness and Information in a Global Social Network" containing a list of word along with happiness ratings for each word based on credible and timely sources such as Twitter, the New York Times, and Google. Words with the highest happiness rating were most positively associated in these sources whereas words with the lowest happiness rating were most associated with negative mentions.

Although a credible and valuable source, such a list of words is also subject to context. As noted before, the source data comes from an online university where courses can and do deal with topics containing words with both negative and positive associations out of context. For example, a sociology course may have sincere discussions on genocide and a history course would have discussions on the horrors of war but this does not mean that comments on these courses including these terms are negatively charged as well.

The master word list can also contain other common associations such as parts of speech. We also utilize a score associated with academic words. SAS Enterprise Text Miner has a stemming algorithm to reduce all variations of words to a core word including conjugations. Whereas the intricacies of the English language used to be so daunting as to require advanced algorithms such as found in SAS Enterprise Text Miner, the use of text mining in analytics has advanced to the point where comprehensive lists of all variations and conjugations of common words are readily available.

Also notable is those words from the source data that do not match against the master word list. This is due to a variety of reasons where the two most likely scenarios are misspellings and acronyms. Reportable levels at which the source data does not match the master word list is at the discretion of the intended audience. However, non-matches should not be ignored. The source data might have valuable

information, but if the language suddenly changes to Spanish, then this metric would easily identify the issue.

Waxing Prophetic

The associated 'part of speech' field from the Master Word List is useful for a variety of reasons of which a main aspect will be described later. Part of speech (POS) can also be used to flag excessive use of adjectives in the source data. Our research has shown excessive use of adjectives (other than the ubiquitous 'very') to be an indicator of sentiment in our source data. When free-form text includes an abundance of adjectives, expressions of definitive opinions are being made.

The challenge with flagging such text is to define 'excessive use of adjectives'. This is accomplished by applying relative methodology. If the use of adjectives in one case is excessive to the point of more than 90% of the overall range from all cases, then the case is flagged as having excessive adjectives. This is done through a format lookup table.

Upon completion of a Proc Freq statement counting the number of parts of speech identifiers per key, the resulting data set is then transformed into a lookup table noting percentage of adjectives per comment. This is simple data step processing. Also noted is the maximum and minimum percentage of adjectives in a separate summary data set. From this lookup table, a format is created to be used to flag comments with excessive use of adjectives.

```
data adjrange (keep=start label fmtname type hlo);
      retain minadj maxadj;
      if _N_=1 then do;
            set adj_range (keep=minadj maxadj);
            end;
      set adjfrq (keep=adj_pct cum_pct rename=(adj_pct=start))
            end=last;
      if cum_pct ge .9 and
            start ge (maxadj - ((maxadj - minadj) * .1));
      length label $7 fmtname $8 type hlo $1 ;
      retain type 'N' label 'good' fmtname 'adjscr';
      output;
  if last then do;
      start=.; hlo='O'; label='bad'; output;
      end;
      run;
proc format cntlin=adjrange; run;
```

Emotional Intelligence

Although the threshold for excessive adjective use is relative, obviously adjustments can be made if the cutoff at 90% seems either too restrictive or allowing given the source data. Even more arbitrary will be the definition for positive and negative words flagged within the source data.

For our intents and purposes, threshold settings for words matched against the Master Word List with both high and low happiness ratings were dependent on the context of the experience within an online course environment. A higher threshold for positive words was flagged than negatively charged words. The threshold settings were based on relative ranges given the full sample of the user data, but trial and error was utilized to refine the settings to the agreement of the target audience. Some words with a relatively high happiness rating may not convey that strong of an emotion provided the context of different user data. For example, the list of words detrimental and complimentary to source data composed of restaurant reviews is mostly inherently obvious. Given that, refinement from the target audience is a necessity before definitive flagging can begin.

Perhaps just as great a force as context is the ugly enemy of sentiment analysis known as sarcasm. Writing an accurate algorithm for definitive flagging of sarcasm is beyond the realm of this or any paper other than the most gifted of linguistic programmers. The best we could do with our data was to flag indications of admiration indicators such as the words 'like' and 'loved' as well as all negation words and their contractions such as 'not' and 'can't'.

Catchphrase

Flagging individual words for emotion proved not to be as nearly as powerful as flagging two, three, four, and five word phrases containing at least one emotional component. The word 'hate' is a strong negative word but the phrase 'hate the fact' could as easily be negative or positive provided proper context. So the next challenge is to build a list of common catchphrases containing at least one emotional word and then utilize a review from the target audience to confirm the emotional flag.

Programmatically this is accomplished through the use of a series of arrays. SAS arrays are both efficient and cost effective. Large amounts of information can be saved in updated array tables while processing the source data in a once-through pass word by word.

Below is the lengthy section of SAS code including macros to create lists of catchphrases up to five words long containing a flagged word. This code is not complete considering the repetitive nature in considering the similarities in two, three, four, and five word phrases as well as the special processing done for the last word in a text entry. However, the concepts and the array processing is clear.

```
* parse variable macro;
%macro parse(prefix);
            &prefix.words[count] = trim(word);
            /* count all manner of flags here using the
            General outline of the following format:
            &prefix.(name) [count,1]=_nameof_flag;  */
%mend;
```

Nested loop used to set flags in array.

```
* set flag alert;
%macro fa(num,prefix);
            flagalert=0;
            do i = 1 to &num;
                  do j = 1 to 6;
                        flagalert+&prefix.flags[i,j];
                        end;
                  end;
%mend;
```

Another nested loop is used to look at the next word in array.

```
* adjust arrays to look at next word;
%macro upword(prefix,loops);
      %do i = 1 %to %eval(&loops-1);
            %let nxt=%eval(&i+1);
            &prefix.words[&i]=&prefix.words[&nxt];
            do j = 1 to 6;
                  &prefix.flags[&i,j]=&prefix.flags[&nxt,j];
                  end;
            %end;
            /* set flags in same format as in the Parse macro above */
            &prefix.words[&loops]=trim(word);
            /* etc….*/
%mend;
```

Create two or more word phrases from list of words. Only the two word phrases here are created for brevity sakes..

```
data two (keep=key two happy2 upset2 intense2);
      set wordflags;
      by key;
      * parse comments into sets of multiple word phrases;
      length two /*three four five */ $100;
      array twowords {2} $24 _temporary_;
      array twoflags {2,6} _temporary_;
      /* initialize arrays */
      if first.key then do i = 1 to 2;
                  twowords[i]='';
                  do j = 1 to 6;
                        twoflags[i,j]=0;
                        end;
                  end;
            end;
      /* initialize flags */
      /* this code isn't included.  Code to initialize flags
            Is pretty standard */
```

```
/* assign values to arrays */
* parse comments into sets of multiple word phrases;
/*****************************************************
****************** two word phrases ******************
*****************************************************/
if count < 3 then do;
      %parse(two);
      end;
else do;
      * set flag alert;
      %fa(2,two);
      two = trim(twowords[1]) || ' ' || twowords[2];
      if flagalert > 1 or
            (flagalert=1 and not (twoflags[2,5] =1 or
            twoflags[1,1] =1 or twoflags[2,1] =1)) then do;
            /* update all the flags created in the parse macro */
            output two;
            happyflag2=0; upsetflag2=0; academicflag2=0;
            adjectiveflag2=0; loveflag2=0; negateflag2=0;
            end;
      * adjust arrays to look at next word;
      %upword(two,2);
      end;
/***last word in comment ****/
if last.key then do;
      /* code here about last word processing
            This code looks much like code above */
      End;
```

This code creates data sets of two, three, four, and five word phrases. Simple frequency tables are then created from these tables and results are presented to the target audience for emotional confirmation. Phrases can be tagged as either positive, negative, or neutral. In the case of this data, flags were also set for phrases considered more intense than other phrases.

Once a manual review is performed on these results, the results are stored as a different input file. Upon subsequent runs of this process with new data, a similar section of code as the one above is run comparing the catchphrases derived from the latest data to the historical catchphrases deemed to contain an emotional component. The resulting data set will contain values for positive and negative flags.

Setting Sentiment Signals

The exclusive presence of a singular positive emotional catchphrase within the overall text entry categorizes the entire response as positive. This is to say that if the target audience has deemed the phrase 'this is great' to be positive and that phrase occurs within the text entry at some point and no other catchphrase occurs within the text entry, the entire entry is flagged as positive.

However contradictions can occur. The threshold by which contradictory catchphrases equate to an overall positive or negative response is again a matter best left to the discretion of the target audience. Our findings suggest a ratio of at least 3:1 is enough to definitively categorize the overall text comment. This is to say that if a text entry contains three phrases such as 'I liked this', 'excellent job', and 'loved the thing' in comparison to one occurrence of "didn't like" then the entry is flagged as positive.

This process is completed in SAS code through data-step-processing reading the data sets composed of two, three, four, and five word phrases by the text entry key. For those with one entry, categorization is a simple matter of noting whether that entry was flagged as positive or negative. For entries with multiple catchphrases, the issue becomes one of array processing. Counts of positive and negative are compiled for each entry and then the totals are compared and categorizations are made.

Those text entries not clearly categorized by catchphrase mapping are not eliminated from further processing. They still may contain either positive or negative words. Again a ratio provided through examination of examples is best for final determination. Once this ratio is agreed upon, then counts of positive and negative words per text entry are compared and categorizations are made

Once all possible flags are set per text entry, the final step before summarization is a simple merging of the results to the original text entry table. Presentation of this data across time is a matter of agreed upon granularity and ratio of positive entries to negative.

Conclusion

One emphasis of the SAS Sentiment Analysis package is word associations within the free-form text. Whereas knowledge of modifiers can be insightful especially in association with pre-identified key words, identification of catchphrases in our studies has proven more effective. For example, few exceptions can be imagined in which the phrase "worst product ever" wouldn't convey an overall negative message for the entire text entry.

As noted earlier, the algorithm for interpreting the written opinion is more a function of context within the target audience than sophistication of software. During each step of this process, we emphasize presentation of interim results to our target audience so that they can direct as to which words and phrases are valid for further processing. Parsing free-form text into words, associating those words with POS as well as happiness rating, building catchphrases of those words containing words of note, and then flagging catchphrases as positive or negative allows for maximum

control from the target audience and ease of implementation for the developer. This is a solution that works well for us.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Mike Tangedal
Capella University
Phone: 612-747-3797
E-mail: Michael.tangedal@capella.edu