An Automated Scoring System for Measuring Email Emotion

Scott Burk

Recently there has been much interest in detecting emotional content in unstructured data by the machine learning and user interface communities. Research has been conducted in developing algorithms to detect emotional content in video and voice with additional work done in mining information from text. There are a number of application areas where interactive marketing activities can gain from mining emotion in text. In this paper we examine the application of an emotional scoring algorithm to Customer Relationship Management (CRM) activities, primarily in customer service operations. We examine the results of a pilot program with a large U.S. top 20 internet retailer to mine emotional content in email. A system was developed to determine an applicable score for individual emails. The system allows operations to more quickly and appropriately respond to irate or emotionally charged customers. We show that an effective algorithm can be developed inexpensively and on a brief project timeline. We provide some lessons learned and a basic architecture for the system.

Keywords: Customer Relationship Management, Text Mining, Email, Scoring Algorithm

Introduction

There has been much research for the detection of emotion in speech by the machine learning community. For example, Ververidis and Kotropoulos (2005) treat emotional speech classification as a supervised learning task. Their model treats emotional speech segments as the features and the emotional styles form the labels. Huang and Ma (2006) report on research of human affect detection (emotional content) from acoustic signals rather than linguistic or semantic information. Liu, Lieberman and Selker at the MIT Media Lab have developed algorithms for detecting emotional content in text. Their methods are in many ways similar to statistical methods utilizing linguistic and semantic information, but extend these methods by using a large corpus to detect human affects (Liu, Lieberman & Selker 2006).

While there are creative and sophisticated ways to approach the problem, the main goal of our research is to demonstrate that a good, practical algorithm can be developed inexpensively. We define ‘good algorithm’ as an algorithm that can perform well in general day to day customer service operations at detecting intended emotional content in email. We define a ‘practical algorithm’ as one that is fairly robust to changes in email in that it does not require frequent updates and tuning. We also define practical algorithm as one that is can be embedded in an easy to user interface.

We illustrate the development of a practical and effective algorithm to score emotional content in email. The system was developed inexpensively with software tools that are freely available. The system effectively scores emails in a batch process for email collected over the previous day.

Business Case

Businesses more than ever have to be responsive to customers. Market research and quality function deployment are essential elements to understanding customer perceived expectations. However, it is just as important to know how customers feel about product and services once they are delivered. CRM and specifically customer service operations often provide real time measurement and even leading indicators of how the company is stacking up in the marketplace. Furthermore, customer service interaction often offers ‘moment of truth’ opportunities that determine the nature of an ongoing relationship or the departure of a customer.

Estimates differ but it is generally accepted that the cost of acquiring an on-line customer is 1.5 to 2.5 times greater than the sale. It is therefore essential to keep customers satisfied and retain them so that they become profitable customers for the enterprise. As Paul Greenberg says in CRM at the Speed of Light “Retaining that same customer is far less costly and much more a matter of relationships – not products” (Greenberg 2001).

Customer relationship management has been heavily influenced by information technology. Some of these technologies include telephony, email and live chat. The objective is to offer customers multiple easy to use and convenient ways to contact representatives and resolve issues in a timely manner. CRM contact centers are often inundated with calls and emails due to this flexibility and simplicity of contact options.

‘Pure Play’ internet retailers have no physical store front locations (no bricks, just clicks). Therefore these retailers are particularly dependent on the success of their customer contact centers since as these are the sole means by which a customer can reach them. Therefore, it is often through customer service where trust is established or diminished. Research has shown that trust is a key issue for the survival of an e-business (Wang 2005). It does a company more harm than good if it makes it easy for customers to contact them, but they are slow to respond, especially to irate customers. Therefore companies need to use technology to their advantage in meeting customer needs (Shauer 2004).

It would be very difficult and expensive to hire enough human readers to go through each email a company contact center might receive. Most fair size companies receive tens of thousands of emails per year. There are an existing number of systems that try to help with this volume, for example automatic detection, collection or deletion of SPAM messages. However, there is great opportunity for intelligent systems which attempt to automate activities such as triage and problem classification.

Data mining or more specifically text data mining is one way companies can use technology to improve their business operations (for a general reference on data mining methods see Larose 2005, 2006). This paper discusses text mining methods to improve a company’s response time to customers that have contacted them through email. Specifically, we have created an automated scoring system which evaluates the negative emotional content contained in email. This score might informally be referred to as the ‘temperature’ of a single email. A higher temperature relates to a more emotionally charged email that might indicate the need to respond more quickly and thoughtfully so that the company might increase the likelihood of retaining that customer.

This is an interesting problem because almost everyone is knowledgeable about the process. However, they may be naïve to the complexities of trying to trying to build a robust system that can sift through the myriad of emails a business might receive, much of it still SPAM.

System Development

We partnered with a U.S. top twenty 'pure play' internet retailer who wished to remain anonymous. The company agreed to partner in this project as a proof of concept to determine if a system could be developed that would benefit them in operations. The pilot project objective was to mine email data and to determine if an algorithm could be developed to allow their customer service agents to more rapidly identify emotional content in email and respond appropriately.

The system consists of modules of code that read in, analyze, score and report email results. A user is able to identify and pull out emails by their associated score and respond appropriately.

Overview of the System

The system consists of two primary components. One component reads, cleans and scores a batch of incoming email. It separates the emails from batch and executes a number of modules that process and assigns a unit scores to each email. These modules consist of a pre-filtering and cropping module, a capitalization and punctuation module, a repeated word and phrase module, a content module and a score aggregation module. Each of these will be explained briefly in the following sections and the appendix includes PERL code and instructions on implementing a system.

A second component is used for reporting results. Results can be reported to the screen or output to a file format for review by a customer service manager or analyst. The system runs on a standard Microsoft Windows XP based desktop computer. The modules are written in PERL and since PERL is freely available under the GNU license it was very inexpensive to implement.

Customer emails are stored in batch over the course of the day and saved into a text file. The retailer in the study usually received between 800 and 1,200 emails per day. The scoring script is run against the collected emails and the program saves the indexed scores to another text file. Then the reporting script is run and the output is analyzed. Operations responds appropriately to any customer complaints and makes any process changes identified. The cycle then starts again. This process is depicted in Figure I.

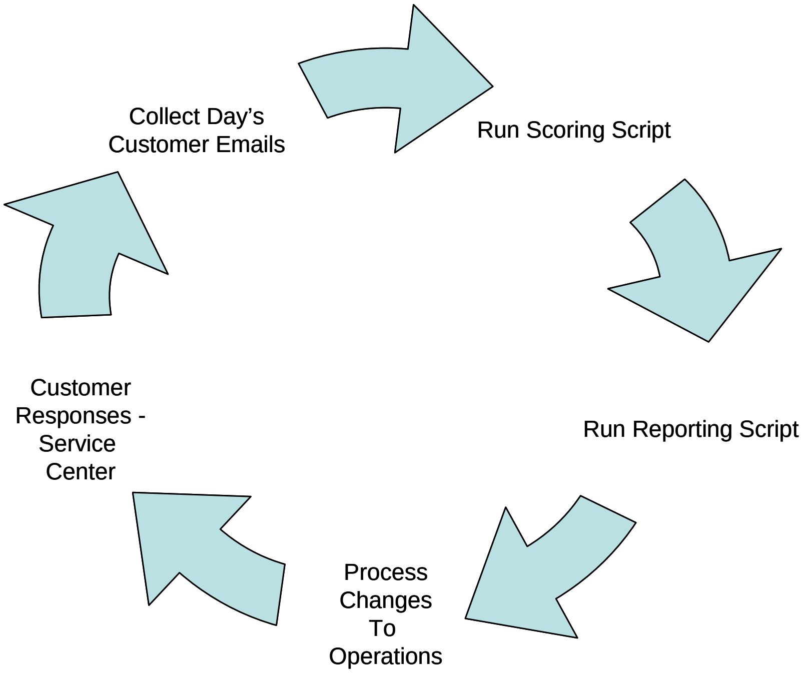
Collect Day's Customer Emails
Run Scoring Script
Run Reporting Script
Process Changes To Operations
Customer Responses - Service Center

Figure 1. Overview of Email Review Process

Although the proof of concept was demonstrated successfully in a large internet retailer it is also applicable to small and medium enterprises (SMEs). The cost to implement is low and the potential return on investment is quite high.

The Scoring Script – Pre-filtering and Cropping

Email is very rich in content and structure. This leads to many interesting complexities in trying to score certain occurrences which would normally be considered an indication of sentiment or passion. By content we mean the actual choice of vocabulary. By structure we mean the choice of formatting, length and punctuation.

Punctuation and capitalization may be used intentionally to emphasize a point or sentiment. However, they also frequently appear in forms of unsolicited marketing email. For example 'YAHOO!' or 'Messenger!' often appear as automatically generated tags in user email as they are large free service providers.

There are also examples where there are mixtures of emotional intent interspersed with everyday syntactical usage. Discerning the signal from the noise in this structure is a challenge.

We should mention that advertising and SPAM are major sources of 'false positive' information. A false positive is an occurrence which is intended to be benign in nature, but will cause a score to be inflated. There was considerable effort in reducing such occurrences, but it is impossible to completely eliminate their influence. Pre-filtering was used to capture and help eliminate the influence of advertising and SPAM, but since marketing and advertising are always changing, modifications to these filters are required with some frequency.

Lastly, the length of an email can influence the overall score. Somewhat frequently an email will contain history of back and forth exchanges between a customer service representative and a customer, an email chain. We did not want the length of an email to exert an undue influence on the emotional score. Additionally, the most recent information is the most important in terms of gauging current emotional content.

The algorithm system used a set of matching and scoring sub routines to evaluate occurrences of punctuation and capitalization that would indicate emotional content and ignore practical usage. Scoring in the system was adjustable and tuning these scores was a considerable portion of the project and will be discussed in following sections. To deal with email chains and lengthy email, a cropping routine was set to intercept the incoming email and only consider a designated length for analysis.

The Scoring Script - Use of Repeated Words or Phrases

The use of repeated words or phrases was identified by the Director of Customer Service in the pilot as an indicator of emotional content. Although this might seem a fairly straight forward process it can be computationally intensive to look at all possible occurrences of word duplicates or triplets in several hundred emails.

PERL was selected for its ability to process text efficiently and effectively without a great depth of programming experience. Although this was one of the most demanding component in processing the script usually ran in less than an hour on a dual core Pentium machine.

Each incidence of a repeated word or phrase received a numerical score and these scores are added to the total score for that email. Sequences of three or more sequential words or phrases received a higher score. A retrospective analysis showed an extremely low incidence of higher frequency occurrences. Only frequency of pairs and triplets were considered to reduce processing time.

The Scoring Script - Dictionaries and Matching Words

One of the most specific and reliable indicators of emotional content with fewer occurrences of false positives is the actual wording of the email. A common method in text mining in analyzing content is use of dictionaries (Weiss et al. 2005). Several specific dictionaries were created and tested in the evaluation of scores to detect email emotion.

A number of dictionaries are freely available in the internet. These dictionaries provide a good starting point to construct a specialized dictionary to gauge the emotional content of words and phrases. Most of these dictionaries exist in a free text form that allows for easy updating and can be read in as a hash in PERL.

Tuning describes a group of activities used to optimize and homogenize the performance of a system. Tuning was an important part of the project and primarily consisted of determining the truncation length, the length of the dictionary and the scores of the various components.

As the emails are passed through the different modules, the modules assign a score to each email via a list. A total score is created near the end of the script by summing across these lists and these scores are output to a file which can be accessed by the results script.

The Results Script

The results script takes the output of the scoring script as well as the raw data from the email file and reports information requested by a user. The script takes two command line inputs, the number of emails to report and whether to present the results to the screen or an output file. If output to file is selected the file is output in a simple text file that can be opened in an editor or a word processor where search and other functions can be useful.

The scores from the first script are read in and used as input, sorted and stored into a hash. A loop from zero up to the number specified emails to report is stored into a list and subsequently output to the screen or a file. A simple procedures document and a conference call of less than an hour were the only training items necessary to successfully hand off the system to operations.

Lesson Learned and Future Endeavors

As in many endeavors, as we learned and answered questions many new ideas and questions came to mind. There were many more than could be acted upon in the short time frame we had to develop the system. Model tuning is an area that will require occasional updates to catch new features of marketing email and spam. Although existing spam filters were in place during the collection of emails, there are always occurrences which slip past filters.

The following provides a list of things we would have approached differently in the beginning given our current level of understanding and current items being worked on:

i. The scores presented in the model are raw total scores and do not consider the individual lengths of the email being analyzed. Even though differences in email length are mitigated by the cropping algorithm it would be better to normalize the score based upon the individual length of the email. This normalizing of the scores might lead to improved accuracy.

ii. Although there was a lot of pre-filtering done in the system this is an area that could lead to improvements.

iii. Since there are many different variables or factors to consider and each factor has many levels the system might be a good candidate for a multi factorial experimental design. For example, a D-optimal, a fractional factorial design (Haines et al. 2003) might be a good objective way to evaluate the contribution of each factor. Furthermore, it would lead to an optimal decision set of factors and levels.

iv. It may improve results to either have a multiple applied to the scores for matching words if the both words are capitalized. Although very rare, these items did occur.

v. An automated detection system of email trailers and advertising. As stated email marketers are always busy constructing new content and changing the text in their trailers. It would be beneficial to have a script whose sole purpose if to detect and quantify these.

vi. The focus of this project was to analyze email and determine scores based on negative emotional content. The primary purpose is to help customer service operation to identify process changes and appropriate responses to customers. It would beneficial to also have a positive emotional content score that might be used proactively in CRM and marketing activities.

In addition to ongoing refinements our internet retail partner also offers chat options to their customers. Chat offers a huge CRM opportunity to learn not only about customers, but how customers respond to chat. An obvious goal of chat is to support customer question to make the current sale. But, it is also a great opportunity to cross and up sell. Chat service agents are evaluated based upon their ability to improve revenue. Text mining of these chat session should prove extremely beneficial. Furthermore since we have shown that a low cost scoring algorithm can be developed for email. We are confident that an algorithm with specific objectives can be developed for chat.

Summary

Email is very rich in nature and the scoring of emails for emotional content offers an interesting and challenging task. While the challenge itself is easily intuited, it is very difficult to achieve accurate and reproducible results.

There are several known indicators of emotion that have been discussed. We have covered many of the lessons learned through this project and shown that is possible to develop an inexpensive text mining system to score emotional content in email. The challenge is to determine the signal from the noise in these emails and since the content is always changing it is a process rather than a project.

With a process in mind we have offered many interesting research ideas to continue this investigation into the future. While the main goal of this project was providing a proof of concept in writing a viable system in PERL, there is a great deal of opportunity to enhance the system and approach the problem in a much more rigorous, scientific way.

References

Greenburg P (2001). *CRM at the Speed of Light: Capturing and Keeping Customers in Internet Real Time*, Osborne/McGraw Hill

Haines LM, Perevozskaya I & Rosenberger WF (2003). Bayesian Optimal Designs for Phase I Clinical Trials. *Biometrics 59 (3)*, 591–600.

Huang R & Ma C (2006). Speaker-Independent Real-Time Affect Detection System, *Proceedings of the 18th International Conference on Pattern Recognition*. 1204-1207

Larose DT (2005). *Discovering Knowledge in Data: An Introduction to Data Mining* John Wiley and Sons

Larose DT (2006). *Data Mining: Methods and Models.* John Wiley and Sons

Liu H , Lieberman H & Selker T (2003). A Model of Textual Affect Sensing using Real-World Knowledge *Proceedings of the 2003 International Conference on Intelligent User Interfaces ACM,* ISBN 1-58113-586-6, 125-132.

Shauer RN & Thompson SR (2004). Improving Customer Service at the ARL MSRC. *IEEE Users Group Conference Proceedings,* 7-11 (pp. 285 – 288) ISBN 0-7695-2259-9

Ververidis D & Kotropoulos C (2005). Emotional Speech Classification Using Gaussian Mixture Models and the Sequential Floating Forward Selection Algorithm Multimedia and Expo, *IEEE International Conference on Volume, Issue 6-8*. 1500 - 1503

Wang YD & Emurian HH (2005). An Overview of Online Trust: Concepts, *Elements, and Implications. Computers in Human Behavior 21*, 105-125

Weiss SN, Indurkhya N, Zhang T & Damerau FJ (2005). *Text Mining: Predictive Methods for Analyzing Unstructured Information*, Springer Science+Business Inc.

Scott Burk, Ph.D. is currently a senior scientist at Zilliant. He works primarily in the areas of pricing segmentation and optimization. Previously he was chief statistician at Overstock.com responsible for analytics, website and email optimization. He can be contacted at statsandcomputers@hotmail.com.

Appendix A. PERL Code

This appendix includes some PERL code and documentation to implement a scoring system. There are two scripts; the evaluate script (EVALUATE.PL) which scores the text (email) and the output script (OUTPUT.PL) which outputs the scores to screen or to file. The EVALUATE.PL script assumes two input files available; input.txt which is the source text to score and words.txt as a dictionary.

The code assumes the input.txt file is delimited by four sequential piping characters (||||) between the documents to be scored (emails). The dictionary is a simple text file with each line consisting of a word to identify followed by at least one space and a numeric score for that particular word. The dictionary is very important and must be configured for the specific scoring goal intended by the system. The goal in this paper was to score inflammatory content. A starter dictionary was developed from http://wordlist.sourceforge.net. A major part of the project was the tuning of the dictionary to generate appropriate scores as model development.

The OUTPUT.PL script uses the same input.txt file as EVALUATE.PL and the output.txt file generated by EVALUATE.PL. It generates the results to the screen or to file, top.txt. The user will need PERL which can be downloaded via www.perl.com and is freely available under the GNU license.

EVALUATE.PL

```
local( $/, *OP ) ;
open( OP, "input.txt" ) or die "sudden flaming death\n";
open( OP2, "words.txt" ) or die "sudden flaming death\n";
open (OUT,">output.txt") || die "can't open output.txt $!";

#READ AND CONVERT
$temp = <OP>;

$maxe = 1200;     #Max number of emails in the input.txt file
$cap_scr=5;       #Score for capitalized words
$excl_scr=2;      #Score for exclamation points
$rep_scr=10;      #Score for repeated words

##PREFILTER STUFF - REMOVE ALL OF THE SPECIAL CHARACTERS NOT APPLICABLE
##### 1st stuff around exclamation marks
$temp =~ s/Guaranteed!+|Overstock.com!+//g;
$temp =~ s/View!+|Only!+//ig;          # Sample Marketing SPAM
$temp =~ s/Off!+|Deals!+//g;           # Sample Marketing SPAM
$temp =~ s/<!//g;                    # HTML
$temp =~ s/Exercise your brain!+//g;  # Sample Marketing SPAM
$temp =~ s/Yahoo!+|Messenger!+|Yahoo!?//ig;
$temp =~ s/broker!+|job!+\s+//ig;
$temp =~ s/blogs!+|free!+|special!+|off!+\s+//ig;
$temp =~ s/'//ig;

##### 2nd stuff around capitalization
$temp =~ s/TRADE SMART AND WIN//ig; # Sample Marketing SPAM
$temp =~ s/MSN//ig;                 # Sample Marketing SPAM
```

```
$temp =~ s/CONFIDENTIALITY NOTICE//ig;
$temp =~ s/FINAL NOTICE|AOL//ig;

##### 3rdstuff Customer Service Fields
$temp =~ s/CUSTOMER:.+//ig;
$temp =~ s/INVOICE:.+//ig;
$temp =~ s/QUANTITY:.+//ig;
$temp =~ s/ITEM:.+//ig;
$temp =~ s/TITLE:.+//ig;
$temp =~ s/LINK:.+//ig;

########################################
### BLOCK TO SPLIT AND CROP          ###
########################################
@a = split(/\|\|\|\|/, $temp);
foreach ($i = 0; $i <$maxe; ++$i) {
$a[$i] = substr ($a[$i],1,600);
}
########################################
### BLOCK TO MATCH CAPITALIZES WORDS ###
########################################
foreach ($i = 0; $i <$maxe; ++$i) {
        $count = 0;
      while (<$a[$i]>) {
            while (/[A-Z]{3,}/g) {
            $count++;
                }
        }
$a04[$i] = ($cap_scr*$count);
}
########################################
### BLOCK TO CONVERT TO LOWER CASE   ###
########################################
$temp = lc($temp);
$i = 0;
@a = split(/\|\|\|\|/, $temp);
########################################
### BLOCK TO COUNT EXCLAMATION POINTS###
########################################
foreach ($i = 0; $i <$maxe; ++$i) {
$count = 0;
      while (<$a[$i]>) {
            while (/!/g) {
            $count++;
                }
        }
        $a01[$i] = ($excl_scr*$count);
}
########################################
### BLOCK TO COUNT REPEATED WORDS    ###
########################################
foreach ($i = 0; $i <$maxe; ++$i) {
     $c[$i] = $a[$i];
     $c[$i] =~ s/[ .]/#/g;
     $count = 0;
     while (<$c[$i]>) {
           while ($_ =~ /([a-z]{3,})#\1/g) {
```

```
           $count++;
           }
     }
     $a02[$i] = ($rep_scr*$count);
}
########################################
### BLOCK TO MATCH DICTIONARY        ###
########################################
$count = 0;
$dict = <OP2>;
@text = split (' ', $dict);
%pairtext = @text;

foreach $key (%pairtext) {
  chomp;
  foreach ($i = 0; $i < $maxe; ++$i) {
      @b = split(/ /,@a[$i]);
      foreach ($k = 0; $k <1000; ++$k) {
          if ($b[$k] =~ /\b$key\b/) {
          $a03[$i] = $a03[$i]+$pairtext{$key};
          }
       }
  }
}
########################################
### BLOCK TO SUM LISTS               ###
########################################
foreach ($i = 0; $i <$maxe; ++$i) {
     $total[$i] = $a01[$i] + $a02[$i] +$a03[$i] +$a04[$i];
}

foreach ($i = 0; $i <$maxe; ++$i) {
     print OUT $total[$i];
     print OUT " ";
}
```

OUPUT.PL

```
local( $/, *OP ) ;
open( OP, "input.txt" ) or die "sudden flaming death\n";
open( OP2, "output.txt" ) or die "sudden flaming death\n";
open (OUT,">top.txt") || die "can't open output.txt $!";

$temp = <OP>;
$temp2 = <OP2>;

$maxe = 1200;    #Max number of emails in the input.txt file
##########################################
### VERIFY THE COMMAND LINE INPUT      ###
##########################################
print "\n You have requested the top: $ARGV[0] emails to be reported \n";

if ($ARGV[1] !~ /screen|file/i)
{
print "Your input has not been understood \n";
print "Please enter something like:\n";
print "Perl output.pl 5 screen\n";
print "           or          \n";
print "Perl output.pl 5 file  \n";
exit;
}
##########################################
### RECIEVE THE INPUT                  ###
##########################################
@x = split(/\|\|\|\|/, $temp);
@z = split(/ /, $temp2);
$max = $ARGV[0];

##########################################
### PUSH THE SCORES INTO A HASH        ###
##########################################
@c = sort {$b <=> $a} @z;
%email = ();
foreach ($i = 0; $i <$maxe; ++$i) {
%email = (%email, $i, $z[$i] );
}
##########################################
### PUSH THE ORDERED SCORES INTO A LIST###
##########################################
$k = 0;
foreach $key (sort {$email{$b} <=> $email{$a}} keys %email) {
$listres[$k] =  "EMAIL: $key, SCORE#: $email{$key} \n $x[$key]";
$k++;
}
##########################################
### OUTPUT THE VALUES                  ###
##########################################

if ($ARGV[1] =~ /screen/i) {
     foreach ($j = 0; $j < $ARGV[0]; ++$j) {
     print "$listres[$j] \n";
     }
}
if ($ARGV[1] =~ /file/i) {
     foreach ($j = 0; $j < $ARGV[0]; ++$j) {
     print OUT $listres[$j];
     }
print "\n Output has been sent to the file: top.txt";
}
```