ICS 661: ADVANCED ARTIFICIAL INTELLIGENCE: FINAL REPORT

WILLIAM R. WRIGHT

1. Introduction

Author profiling consists of placing an author in a group of others on the basis of distinguishing characteristics; characteristics in our cases solely those extracted from text written by an author. It is sensible to practice this when (I) author attribution is impossible or (II) author attribution is possible, but not sufficiently predictive. In the latter case, it my be worthwhile to consider the most likely class in which an author falls, and to consider the known likely behaviors of that group, and therefore, of the author. For us, human personality serves as a proxy for behavior predictions, and the actual behavioral prediction is left for others. The automated extraction of text features for personality prediction began in 1999 with Pennebacker but gained real momentum with computer scientists joining the effort around 2005 with the study Argamon et al. "Lexical predictors of personality type" in 2005.

Applications abound: if it is known that Conscientious people are better surgeons, a high school guidance counsellor may guide the Conscientious student to pursue such work, but the Open student to a career more focused on medical research instead. Ordinarily personality is assessed by written self-report questionnaires, but such questionnaires are sometimes available and never convenient.

Current research efforts in this area focused on texts available on internet-oriented websites such as online product reviews, discussion forums, social networking websites,, and Twitter; however the methods of study have regressed over time into the use of the simplest imaginable pre-defined tools which though popular among social scientists nevertheless dismiss occasional but significant contributions to the field over the past 15 years.

Predictions of personalty based on features extracted from text do not yet approach the accuracy of normative personality tests. Nevertheless, prediction of personality that is demonstratively better than random has great value in applications such as marketing, and systems are being built to capitalize on such imperfect but helpful personality predictions [Roshchina et al., 2011].

So our hypothesis is that we may locate within our corpus some text features that are predictive of the personality scores of the authors. By "predictive" we mean that the features exhibit significant correlations (linear Pearson correlations) with authors' numerical personality scores on 1 or more of the 5 major personality traits. To demonstrate the overall predictive power of the features we combine them using statistical methods to produce a predictive model that predicts more than any one individual features alone. In such studies

however, the items of greatest usefulness are the features themselves, which a practitioner may later mix with other available features for even better prediction.

2. Detailed description

We studied 2593 essays written by students at the University of Texas during 2005-2008 enrolled in courses taught by James Pennebacker, a researcher who along with a number of computer scientists produced some of the initial studies of human personality and extraction of text features. He also administered a personality questionnaire to each essay writer (some may have been duplicates, more on that in a minute). The questionnaires varied from year to year, so we had to standardize the scores. We eventually overcame the gaps in the data cited in the project proposal; Dr. Pennebacker provided the missing data. The essays were stream of consciousness writing exercises written during a 20 minute period; the average length was 787 words (with std. dev. 245).

The goal being to locate text features with statistically significant correlations to personality scores, I ultimately located the following two features with the following exploratory process:

Feature 1: Immediately it was clear that people who seemed a bit anxious wrote fairly introspectively, using the word "I" and the related contractions I, I'm, I'll, I'd, and I've. Upon inspecting a few of their personality scores, enough of them had unusually high Neuroticism scores to convince me to further investigate the possibility that the relative frequency of the word "I" that "I" and all its contractions. Underlying this intuition are some papers I read associating the relative frequency of the first person singular pronoun with Neuroticism [Pennebaker and King, 1999], [Oberlander and Gill, 2006]. Note that although I could have viewed each of these as unique features, the fewer the features the better when it comes to prediction.

Feature 2: To find another feature I considered bigrams. First I conducted an exploratory search by extracting all bigrams and focusing on the most frequently appearing bigrams, which naturally will be the most predictive of personality. (Infrequent bigrams are not numerous enough to establish a statistically significant correlation.) Several of the most common bigrams were negative contractions such as don't, won't, can't, and have well-published correlations with personality, so I decided to avoid those and to find one that is less commonly known.

Of course I could have went and found text features correlated with other personality traits, but that would not be very useful for predicting personality. The personal pronoung "I" probably predicts Neuroticism. To improve that prediction, I need to find *another* feature also correlated with the same trait (Neuroticism). The bigram "have to" was present 6000 times in the entire combined corpus, which is just about the minimum frequency I was willing to accept for this corpus of 2593 documents, some of which might have come from the same author. It seemed ideal as a potential feature associated with Neuroticism: from my reading of all those anxious essays using "I", I had noticed that they seemed to dwell on perceived obligations and expectations; those seemed to be on of their common irritants. Intuitively the bigram "have to" would be related to such demands.

To extract (count the relative frequency of) these features in each document, I used two tools, the first custom and the second off-the-shelf.

For Feature 1, ("I" and associated contractions), I used a custom Java class (see the Appendix) that I wrote consisting of a single method that loops all the files in a path and extracts occurrences of strings without case sensitivity. It is not incrediblly impressive, everything is hardcoded. It is purely throw-away, utility code. The things I am likely to reuse are the call that lists all the files in a path and the character encoding switch, which took a while to figure out. Amusingly the encoding, ISO-8859-1, is common on Windows machines, but not on OS-X (which I used): on OS-X the character encoding typically used for word processing includes the same characters but in a dramatically different ordering. Fortunately Java allows me to specify the encoding and thus make sense of the files. Figuring this out required staring at the HEX values of the source text for a while and searching around on Google for suggestions. As an aside, this code also gave me the overall word count of each essay, allowing me to compute the relative frequency of each of the features.

Given time and resources, I would add n-gram counting (including restriction to user-specified n-grams for improved performance) as well as essential features such as overall word count. Counting custom n-grams calls for an examination of the performance characteristics of regular expression libraries and the obvious alternative. Also of course I would generalize the code for reuse. If the application required online processing of large data streams such as Twitter, but I needed to do it for low cost, I would likely recast the code in raw C for use on large cluster computing resources, since well running Java cluster computing resources are harder to come by (although if they were made available I would stick with Java). Essentially this would be a re-write of the below discussed tool, which lacks much of the above as well as a case-insensitive option.

For feature 2 ("have to"), I conducted the bigram exploratory search (looking at trigrams as well), and ultimately extracted the counts of the occurrences of "have to" in each document using a system called Ngram statistics package (NSP), a collection of programs written in Perl intended to support bigram extraction [Pedersen, 2012]. The tool took a long time enumerating all the bigrams (the output file was $> 60MB$), and unfortunately the custom n-gram option did not work: it apparently gives counts of n-grams without regard to the order of the words in the n-gram! However I was able to work around this issue by using the full bigram enumeration option and using Grep to extract what I wanted.

Although my investigation led to a focus on Neuroticism, personality occurs in five traits, each of which are dimensional and are measured as a scalar: Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness. The factors that I located were both correlated significantly with Neuroticism; those low on this trait are emotionally stable and quite inflatable whereas those high on the trait tend are often frustrated by things that seem minor to others.

3. Analysis

The Pearson correlation coefficients of Feature 1 and of Feature 2 with Neuroticism questionnaire scores were 0.0565 and -0.0424 respectively (in each case $p < 0.05$); the latter was after smoothing[1] the counts by +1 due to the relatively high incidence of essays with 0 counts of the bigram "have to". The p values $p < 0.05$ denote the probability is less than 0.05 that we would see such data were there no such correlation. Although there is controversy about the practice, in general the null hypothesis (in our case the proposition that the our features have nothing to do with reported personality scores) is rejected if $p < 0.05$. This practice has the pernicious consequence of discouraging dissemination of results when p is not < 0.05.

Our result for Feature 1 is not surprising given the literature. However feature 2, rather than being positively correlated, exhibited a significant negative correlation (-0.0424) with Neuroticism, indicating that Emotionally Stable people actually use "I have" more often than those on the opposite end of the scale, the more Neurotic! Nevertheless both features exhibit significant correlations with a single trait (Neuroticism), so we proceed to combine them to see how well they can predict Neuroticism together.

When predicting personality, a very common practice is to produce a model classifying each participant into one of two classes, either "High" or "Low" for each personality trait. This is done more often then not by use of Support Vector Machines (SVM's), a statistics technique that "learns" by searching for the most predictive model possible. SVM's are not guaranteed to find the best model, but nonetheless, non-optimal results are tolerated if they are better than nothing. Mairesse et al., with whom we have corresponded, produced binary classifier in just this way.[2]

Likewise we did the same, training a binary classifier dividing participants into those with "Low" Neuroticism scores and those with "High". The tool we used was libsvm, a tool implementing SVM's [Chang and Lin, 2011]. The effect sizes of our features are small (0.0565 and -0.0424), so we expect a cumulative prediction of no better than 0.0989% over the baseline classifier, which consists of what you get if you assume that everyone falls in the larger of the two classes obtained by dividing the sample at the median Neuroticism score. The result after 10-fold cross validation with a linear kernel was 51.3% accuracy, which is 0.78 better than the baseline (which came in at 50.52% accuracy). 0.78 is a larger effect than either of the features alone, so we believe that each feature makes a unique contribution to the prediction that could not be had from either of them alone under the assumption linearity. Of course we do not assume that this is the best possible prediction; SVM is essentially a principled heuristic search for the best predictive model.

---

[1] A technique we learned about this semester, [Jurafsky and Martin, 2000]

[2] Sometimes researchers run several learning algorithms for purposes of comparing their results; one group [Sumner et al., 2012] extracting text to predict the personalities of 2,927 Twitter users even offered a public competition to find out who could come up with the very best classifiers! Consider however that compute time might better be employed traversing the solution surface with some parallel technique avoiding recomputation of the same solutions.

As appealing may be producing models that accomplish binary classification with the best possible accuracy, this common practice of bifurcating people into "Low" and "High" scores for each personality trait deserves some critical review. It is not well supported by the theories of personality psychologists, and given that personality scores tend to fall on a normal distribution, splitting the distribution in half seems forced at best. Mairese et al., but few if any others, have predicted the actual scores using regression (SVR's, a version of SVM's is available for regression, and there are many other methods); this is probably much more useful in thoughtfully constructed applications. When attempting regression, though, I was dissatisfied with the libsvr's output on regression; it did not give sufficient data to evaluate goodness of fit of the regression as compared to a baseline.

Compared to the results of others who pile on more features when training their classifiers, my classification results are not astounding (their maximum is around 80% accuracy). However not all of them reveal novel features as I did. The generalizability of "I" is well known, but that of "have to" begs further investigation. If it were not for other more urgent priorities in this field, as below, that would be a good direction to go.

This project identified two very significant predictors of personality, but the effect size was too small to compel me to want to use them in the future. I would rather identify highly structured features with greater predictive power, so as not to overwhelm a learning algorithm with a myriad of well known features (there are hundreds of them in the literature), which would ultimately choke it (known as "the curse of dimensionality"). I certainly did extract at or near the maximum predictive power for classification from the two features using SVM's, but making the latter point, the effect was not large. Also however I question the field's tendency to do binary classification on personality scores; there is a weakness in the whole approach. An attempt at regression using libsvn did not go well, but is not a high priority given the need for features with larger effect sizes. With time and resources the main focus would become the search for more structured features perhaps grammatical in nature, which would summarize the effects of smaller features, thus allowing me to ultimately predict personality scores using as few features as possible.

4. Conclusion

Essentially this work repeats that of those cited in the introduction and many others, by identifying n-grams whose appearance is significantly correlated with one of 5 possible personality traits. Going forward, a new direction is needed. Lately my reading in this field has revealed a crying need for "deeper" features: those that tend to summarize the predictive power of many individual lexical and even POS n-grams into higher order structures. The reason for this need is that the results of the learning algorithms deteriorate rapidly as the number of features becomes larger; they become less and less *likely*, given acceptable usage of storage and CPU time, to locate a strong predictive model even if the features being provided are entirely adequate to perform stellar predictions of incredible accuracy. Although the need has been ignored for the past few years, such higher order feature extraction allows us to capture greater predictive power (as in [Oberlander and Gill, 2006],

[Estival et al., 2007], [Luyckx and Daelemans, 2008]) without overwhelming the learning algorithms with too many dimensions. We need more such features.

APPENDIX

Listing, ExtractFeatures.java.

```
1  //Author: Bill Wright
2  //Date: Nov. 12, 2012
3  //Utility code to extract a few unigrams (non-case sensitive).
4
5  package nlp;
6  import java.util.regex.*;
7  import java.util.*;
8  import java.io.*;
9  import java.util.Scanner;
10
11 class ExtractFeatures{
12
13 public ExtractFeatures(){
14 }
15
16
17 // Loops all the files in a path and counts occurrences of the
       hardcoded strings without sensitivity to case.
18 // Outputs a line for each file consisting of:
19 //   - The file name
20 //   - The word count of the file
21 //   - The total of all occurrences of the strings sought.
22 public void extractFromFiles(){
23    File files[];
24    File directory;
25    String path = "/Users/Bill/Documents/UH/ICS 661/project/
         Ass1_2005_to_2008/";
26
27    int iCount = 0;
28    int i;
29    String compareTo1 = "i";
30    String compareTo2 = "i'm";
31    String compareTo3 = "i'll";
32    String compareTo4 = "i'd";
33    String compareTo5 = "i've";
34    String temp;
35    File ftemp=null;
```

```
36
37     try{
38         directory = new File(path);
39         files = directory.listFiles();
40
41         System.out.println("filename, word_count, feature_count");
42
43         for (File f : files) {
44             ftemp = f;
45                 String text = new Scanner( new File(path+f.getName())
                       , "ISO-8859-1" ).useDelimiter("\\A").next();
46             //Splits text into an array, dividing the text into
                   elements wherever common punctuation (but not
                   apostrophe) or spaces are found.
47                 String [] str = text.split("([.,!?:;\" -]|\\s)+");
48                 iCount = 0;
49                 for(i = 0; i < str.length; i++){
50                     temp = str[i].toLowerCase();
51                     if(temp.equals(compareTo1) || temp.equals(
                           compareTo2) || temp.equals(compareTo3)
52                                 || temp.equals(compareTo4) || temp.equals
                                       (compareTo5)){
53                         ++iCount;
54                     }
55                 }
56             // Prints the file name of the current, the number of
                   words in the file, and the final count of all the
                   strings we are interested in.
57                 System.out.println(f.getName() + ", " + str.length +
                       ", " + iCount);
58         }
59
60     }catch(Exception e){
61         e.printStackTrace(System.err);
62         System.err.println("Error: " + e.getMessage() + " filename:
               " + ftemp.getName());
63     }
64 }
65
66 public static void main(String[] args) {
```

```
67
68      ExtractFeatures ef = new ExtractFeatures();
69      ef.extractFromFiles();
70  }
71
72  }
```

References

[Chang and Lin, 2011] Chang, C. and Lin, C. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

[Estival et al., 2007] Estival, D., Gaustad, T., Pham, S., Radford, W., and Hutchinson, B. (2007). Tat: an author profiling tool with application to arabic emails. In *Proceedings of the Australasian Language Technology Workshop*, pages 21–30.

[Jurafsky and Martin, 2000] Jurafsky, D. and Martin, J. (2000). *Speech & Language Processing*. Pearson Education India.

[Luyckx and Daelemans, 2008] Luyckx, K. and Daelemans, W. (2008). Using syntactic features to predict author personality from text. In *Proceedings of Digital Humanities 2008 (DH 2008)*, pages 146–149.

[Oberlander and Gill, 2006] Oberlander, J. and Gill, A. (2006). Language with character: A stratified corpus comparison of individual differences in e-mail communication. *Discourse Processes*, 42(3):239–270.

[Pedersen, 2012] Pedersen, T. (2012). Retrieved oct. 2012.

[Pennebaker and King, 1999] Pennebaker, J. and King, L. (1999). Linguistic styles: language use as an individual difference. *Journal of personality and social psychology*, 77(6):1296.

[Roshchina et al., 2011] Roshchina, A., Cardiff, J., and Rosso, P. (2011). A comparative evaluation of personality estimation algorithms for the twin recommender system. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, pages 11–18. ACM.

[Sumner et al., 2012] Sumner, C., Byers, A., Boochever, R., and Park, G. (2012). Predicting dark triad personality traits from twitter usage and a linguistic analysis of tweets. *Proceedings of the IEEE 11th International Conference on Machine Learning and Applications ICMLA 2012*. To appear in December 2012.