Logan Weir
8/14/13

# The Turing Classifier

Artificial Conversational Entities, commonly known as chatterbots, are slowly developing ubiquity on the internet. Initially developed in the mid 1950's, chatterbots reached a wider consumer audience with rise of instant messaging. Aside from entertainment, these programs can also be used for customer service, information gathering and therapy. As these programs become better imitators, so too do they become potentially better at fraud. Luckily, there is a way to improve these chatterbots while simultaneously improving fraud detection.

Many websites already use an image-based test such as CAPTCHA to confirm if a user is human. However after that test there is nothing to determine if the account is used by a program, and there is similarly no test to apply to the counterparty of a chat service. Thus there is room for an additional security step to confirm that a user of any text-based internet service is human. I hypothesize that a supervised learning algorithm can accurately classify humans and programs by analyzing the text that each produces, given a sufficient amount of text.

Many chatterbots use natural language processing, which is itself based on machine learning algorithms. Relying on a large corpus of previous conversations, chatterbot programs will attach a value to a word or string of words based on counterparty input, and respond with the highest scoring string. My proposed classifier will operate with many of the same tools, only in reverse; relying on a large corpus of conversations in which programs are identified, the classifier will take text output and score each word or string of

words based on previous examples, ultimately producing a probability that the producer of the text is human.

In order to find many examples of human and program dialogue, I turned to the transcripts from the Loebner Prize. The Loebner Prize is an annual competition of chatterbots in which judges engage in simultaneous, separate textual dialogue with a human and a program. The program that can fool the most judges is deemed the victor. If a program is ever able to fool all the judges, it's creator will be awarded $25,000 and the annual competition will cease.

The Loebner Prize began in 1990. Out of the 23 years of competitions, I was able to download 12 years of transcripts, and out of that set, I was able to utilize 8. Even with only 8 years of transcripts, there were 3 different types of transcripts within that sample. The first and most complicated included transcripts from 2012, 2010, 2009, 2007 and 2006. These transcripts recorded individual keystrokes and accompanying timestamps for all three participants in two simultaneous conversations. For these transcripts, writing mistakes and their accompanying 'Backspace' key strokes were recorded. This additional information was ultimately ignored in my classifier, the purpose of which was to analyse text output in it's final form.

The goal was to translate these 8 years of transcripts into one file with three columns: one for the identity of the speaker (human or program), one for what was written, called the utterance, and finally one for the word count of what was written. In addition, '<start>' and '<end>' were appended to the beginning and ending, respectively, of every utterance. This allowed for additional coefficients of starting and ending words, in the hope

of increasing the power of the classifier. This was especially important given the limited data available - in the end, 8 years of transcripts produced only 8,481 lines of dialogue on which to build the classifier.

Limited data shaped the classifier in many other ways. In using the Scikit-learn CountVectorizer, stopwords were not ignored, and no minimum on token sparsity was applied. In fact, token_patttern was adjusted so that the vectorizer would consider single characters such as ‘I’ and ‘a’ as tokens. A tf-idf vectorizer was applied to the data, producing very similar coefficients but a lower cross validation score. Were more data to become available, a tf-idf vectorizer would probably be more appropriate to control the weights of stopwords. Interestingly, stopwords received more spots in the top ten highest/lowest coefficient weights with a tf-idf vectorizer. “It” was the 2nd highest, with “you” and “me” as the two lowest (i.e. most robotic), respectively.

It was the nature of the data, as opposed to the size, that determined whether to use a logistic regression classifier or a naive bayes classifier. For starters, text is typically internally correlated, so that a feature like ‘it’ is usually associated with another feature like “is”. Furthermore, my hypothesis relies on the assumption that people and programs each employ a distinct dictionary of terms, terms which must be themselves correlated. Due to this dearth of feature independence, I chose Scikit-learn’s LogisticRegression classifier over their multinomial Naive Bayes classifier. It would be interesting to compare the weights and p-values of coefficients from these two classifiers.

The resulting classifier produced a cross validation score of 0.71. The resulting coefficients and their weights were interesting, fuel for speculation but indicative of little

without their accompanying p-values. “haha” and “<start> haha” consistently had two of the highest coefficient weights for determining humanity. Some coefficients were obvious anomalies due to the limited data - “yoga” became one of strongest classifiers against humanity, due to one program aggressively attempting to convince the judges it was a yoga instructor from Boulder, CO.

The results, and the resulting ‘humanity detector’ application, are at this point amusing, snacks for thought. Yet the results also show how much more can be done with this concept. Old data can be translated, and new data can be found or pulled. Coefficients can be selected based on their significance and minimum sparsity values can be applied along with a TfiDf vectorizer. Going beyond current methods, text output could be classified with the consideration of the prior counterparty text, enabling a ‘response value’. One additional, albeit speculative measurement, might give further significance to our model - if a program receives a high score during the Loebner prize, will it receive a relatively high score from this classifier?

There is much more to be done in order to make this classifier a legitimate and reliable security tool. Finding out what needs doing is an enlightening first step.