Lexical Syntax Analysis for Hostile Behaviour Identification

Akshath Jain
North Allegheny Senior High School

90%

of all terrorist communication happens through social media

Background

Social media has 4 main purposes for terrorist groups:

1. Share operational and tactical information
2. Gateway to other online radical content
3. Media outlet for terrorist propaganda
4. Remote reconnaissance for targeting purposes

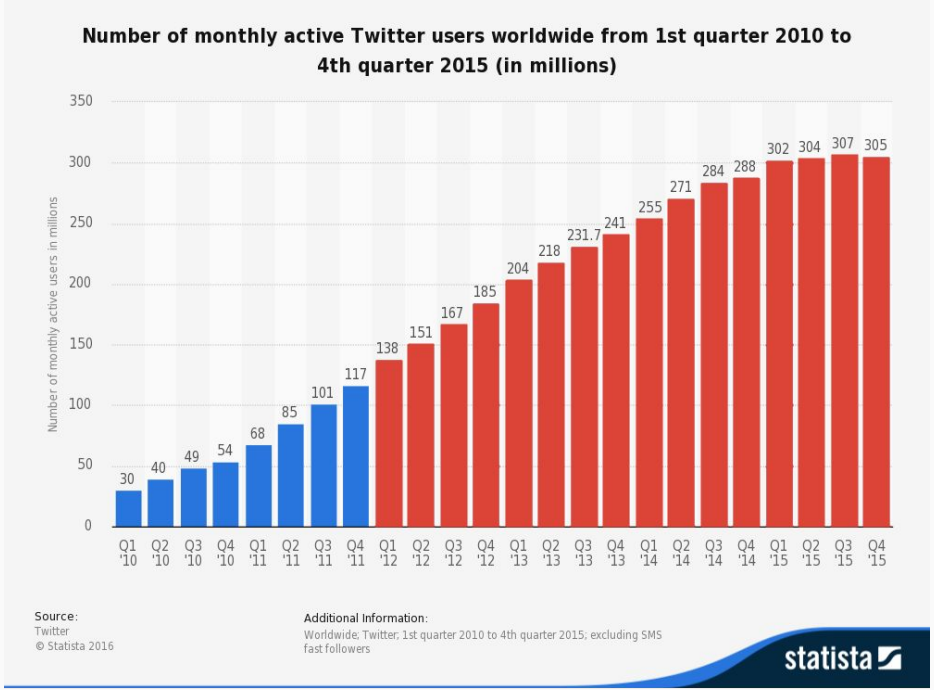
Twitter is the platform on which most of this occurs.

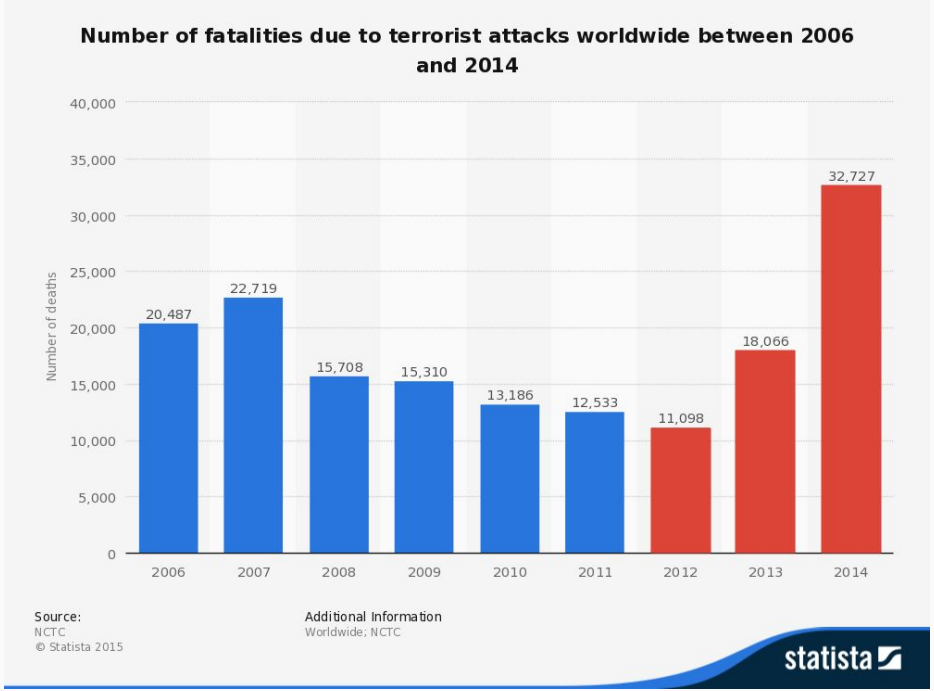
- Microblogging site
- Active user base of 300 million

There are an estimated 46,000 terrorist accounts on Twitter (0.01%).

Background

An increase in Twitter users is correlated with an increase in terrorist attacks.





Analyzing social media feeds with machine learning algorithms can identify accounts as hostile

Algorithm

A simple 3 step process ensures optimal efficiency within the program.

Twitter Account Data Collection

Data Parsing

Data Analysis & Prediction

Methodology

The parameters used to ensure minimal false-positives.

Diction

- Word choice/frequency
- Percent match
- Average match distribution

Affiliation to Known Accounts

- Friends
- Followers
- Retweets
- Content mentions

Miscellaneous

- Date
- Time/Time Zone
- Number of tweets per day
- Location
- Language

Code Sample

```
78  public double[] getDiction() {
79      dictionary = new ArrayList<>();
80      blacklistedWords = new ArrayList<>();
81      wordIndex = new ArrayList<>();
82      commonWords = new ArrayList<>();
83      dictionaryFrequency = new ArrayList<>();
84
85      //initialize dictionary
86      try {
87          //reads in the dictionary
88          BufferedReader fin = new BufferedReader(new FileReader(dictionaryFileName));
89          int dictionarySize = Integer.parseInt(fin.readLine());
90          dictionary = new ArrayList<>(dictionarySize);
91          for (int i = 0; i < dictionarySize; i++)
92              dictionary.add(fin.readLine());
93          fin.close();
94
95          //reads in the blacklisted words
96          fin = new BufferedReader(new FileReader("blacklistedWords.txt"));
97          int size = Integer.parseInt(fin.readLine());
98          blacklistedWords = new ArrayList<String>(size);
99          for (int i = 0; i < size; i++)
100             blacklistedWords.add(fin.readLine());
101         fin.close();
102
103         //reads in the frequency of words in the dictionary
104         fin = new BufferedReader(new FileReader("dictionaryFrequency.txt"));
105         size = Integer.parseInt(fin.readLine());
106         dictionaryFrequency = new ArrayList<>(size);
107         for (int i = 0; i < size; i++) {
108             dictionaryFrequency.add(new WordModePair(dictionary.get(i), Integer.parseInt(fin.readLine())));
109         }
110         fin.close();
111     } catch (FileNotFoundException e) {
112         e.printStackTrace();
113     } catch (IOException e) {
114         e.printStackTrace();
115     }
116
117     //parses the retrieved data
118     String[] formatedTweets = new String[accountTweets.size()]; //an array of all the formated tweets
119     for (int i = 0; i < accountTweets.size(); i++) {
120         String data = accountTweets.get(i).getText();
121
122         //takes out the https:// from the data (these are associated with pictures in the tweet)
123         while (data.contains("http")) {
124             int startPos = data.indexOf("http");
125             int endPos = data.indexOf(" ", startPos);
126             if (endPos == -1)
127                 endPos = data.length();
128             data = data.substring(0, startPos) + data.substring(endPos, data.length());
129         }
130
131         //takes out all unnecessary junk from the data (anything that isn't a letter or a number)
132         String dataFormated = "";
133         data = data.toLowerCase();
134         for (int j = 0; j < data.length(); j++) {
135             if (Character.isLetterOrDigit(data.charAt(j)) || Character.isSpaceChar(data.charAt(j)))
136                 dataFormated += data.charAt(j);
137             if (data.charAt(j) == '\'' && j != data.length() - 1 && data.charAt(j + 1) == 's')
138                 j += 2;
139         }
140
141         formatedTweets[i] = dataFormated;
142
143         //adds missing words to dictionary + indexes all the words
144         int startPos = 0;
145         for (int j = 0; j < dataFormated.length(); j++) {
146             if (Character.isSpaceChar(dataFormated.charAt(j)) || j == dataFormated.length() - 1) {
147                 String temp = dataFormated.substring(startPos, j + 1).trim();
148                 if (!temp.isEmpty()) {
149                     if (dictionary.contains(temp)) {
150                         wordIndex.add(dictionary.indexOf(temp));
```

Code Sample

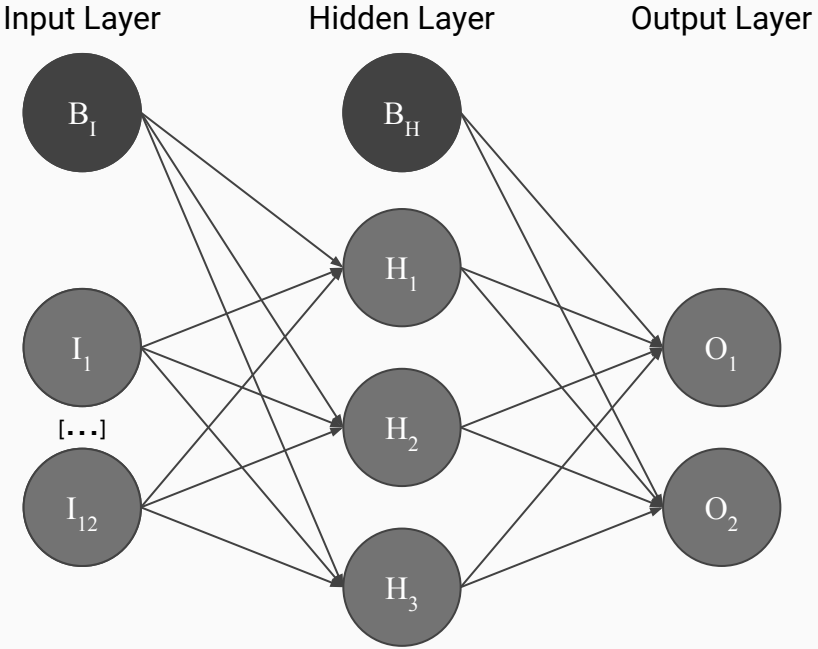
```
                    } else {
                        dictionary.add(temp);
                        dictionaryFrequency.add(new WordModePair(temp, 0));
                        wordIndex.add(dictionary.size() - 1);
                    }
                }
                startPos = j + 1;
            }
        }
    }
    writeDictionaryToFile();
    writeDictionaryFrequencyToFile();

    //finds the most common words in the twitter account and then finds how many of those words ar
    QuickSort q = new QuickSort("sort", null, wordIndex);
    q.start();
    wordIndex = q.getSortedListInt();

    //finds most common words by sorting all the words and then counting the number of times each
    int counter = 1;
    for (int i = 1; i < wordIndex.size(); i++) {
        if (wordIndex.get(i).equals(wordIndex.get(i - 1)))
            counter++;
        else {
            if (!blacklistedWords.contains(dictionary.get(wordIndex.get(i - 1))))
                commonWords.add(new WordModePair(dictionary.get(wordIndex.get(i - 1)), counter));
            counter = 1;
        }
    }

    //adjusts commonWords only to have the top topWordsSize words
    q = new QuickSort("sort", commonWords, null);
    q.start();
    commonWords = q.getSortedListObj();
    for (int i = 0; i < commonWords.size() - topWordsSize; i++) {
        commonWords.remove(i);
        i--;
    }

    //finds how many matches of common words in this account there are with known account
    ArrayList<WordModePair> tempDictFreq = (ArrayList) dictionaryFrequency.clone();
    q = new QuickSort("sort", tempDictFreq, null);
    q.start();
    tempDictFreq = q.getSortedListObj();

    //shortens tempDictFreq to its top topWordsSize words
    for (int i = 0; i < tempDictFreq.size() - topWordsSize; i++) {
        tempDictFreq.remove(i);
        i--;
    }

    //create a new arraylist that only has the strings in it so i can match the common wor
    ArrayList<String> tempDictFreqString = new ArrayList<>(topWordsSize);
    for (int i = 0; i < topWordsSize; i++) {
        tempDictFreqString.add(tempDictFreq.get(i).word);
    }

    /*//used for error catching the list of common words in the twitter account
    for (int i = 0; i < commonWords.size(); i++) {
        System.out.println(commonWords.get(i).mode + " " + commonWords.get(i).word);
        System.out.println(tempDictFreq.get(i).mode + " " + tempDictFreq.get(i).word + " "
    }*/

    //this finds how many words are shared between the account words and the overall dicti
    numSharedCommonWords = 0;
    double totalOccurrencesInList = 0; //this is needed to take the percentage of occurren
    double totalOccurrencesListCommonWords = 0;
    int matchPlacement = 0;

    for (int i = 0; i < commonWords.size(); i++) {
        totalOccurrencesInList += commonWords.get(i).mode;
        if (tempDictFreqString.contains(commonWords.get(i).word)) {
            listOfMatchWords.add(commonWords.get(i));
```

Code Sample

```
224                 numSharedCommonWords++;
225                 matchPlacement += 100 - i;
226                 totalOccurrencesListCommonWords += commonWords.get(i).mode;
227             }
228         }
229
230         double[] diction = new double[3];
231         diction[0] = totalOccurrencesListCommonWords / totalOccurrencesInList;  //higher is better
232         diction[1] = numSharedCommonWords; //higher is better
233
234         if(numSharedCommonWords == 0)
235             diction[2] = 0;
236         else
237             diction[2] = (double) matchPlacement / numSharedCommonWords; //lower is better
238
239         return diction;
240     }
241
242     public double getAffiliation() {...}
321
322     public ArrayList<WordModePair> getListofMatchWords() { return listOfMatchWords; }
325
326     public void mergeAccountWithMetaData() {
327         //this is going to take the most common words in the account, and combine them with the most c
328         //it will increment the frequency in the dictionaryFrequency.txt file
```

Prediction Algorithm

Neural Network Diagram

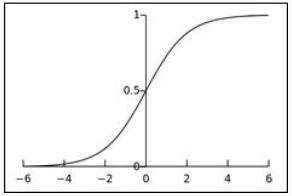


Regular Usage

$$H_j = \sigma\left(w_{B_I H_j} + \sum_{i=1}^{12} I_n w_{I_i H_j}\right)$$

$$O_k = \sigma\left(w_{B_H O_k} + \sum_{j=1}^{3} H_j w_{H_j O_k}\right)$$

$$\sigma(s) = \frac{1}{1+e^{-s}}$$



Network Training

$$\delta O_k = O_k(E)\left(1 - O_k(E)\right)\left(T_k(E) - O_k(E)\right)$$

$$\delta H_j = H_j(E)\left(1 - H_j(E)\right)\sum_{k=1}^{2} w_{H_j O_k}\delta O_k$$

$$\Delta w_{I_i H_j} = \eta I_i(E)\delta H_j$$

$$\Delta w_{H_j O_k} = \eta H_j(E)\delta O_k$$

Neural Network Code Sample

```
66      public int classify(TwitterAccount account) {
67          //initializes input layer
68          double[] temp = account.getDiction();
69          for (int i = 1; i <= 3; i++)
70              inputLayer[i] = temp[i - 1];
71          inputLayer[4] = account.getAffiliation()[0];
72          inputLayer[5] = account.getAffiliation()[1];
73          inputLayer[6] = account.getAffiliation()[2];
74          inputLayer[7] = account.getAffiliation()[3];
75          inputLayer[8] = account.getTime();
76          inputLayer[9] = account.getLocation();
77          inputLayer[10] = account.getNumAverageTweets();
78          inputLayer[11] = account.getLanguage();
79
80          //set hidden layer values
81          for (int j = 1; j < numHiddenNodes; j++) {
82              double sum = 0;
83              for (int i = 0; i < numInputNodes; i++)
84                  sum += inputLayer[i] * wIH[i][j];
85
86              hiddenLayer[j] = sigmoid(sum);
87          }
88
89          //set output layer values
90          for (int k = 0; k < numOutputNodes; k++) {
91              double sum = 0;
92              for (int j = 0; j < numHiddenNodes; j++)
93                  sum += hiddenLayer[j] * wHO[j][k];
94
95              outputLayer[k] = sigmoid(sum);
96          }
97
98          //determine account
99          if (outputLayer[0] > outputLayer[1]) //the first node: is a match; second node: not a match
100             return 1; //account is a match
101         else
102             return 0; //account isn't a match
103     }
```

Experimentation

Methodology

A two step process to train and test the prediction algorithm.

Neural Network Training

- Supervised learning with sample data
- Computer “learns” patterns

Testing

- Substitute data set used (NFL accounts)
  - Similar, but less radical, intent
  - Similar organizational and hierarchical structure

Overview

Training

- Training set (120 total)
  - 12 positive
  - 108 negative
- Validation set used to prevent overfitting (120 total)
  - 12 positive
  - 108 negative

Testing

- Trials 1 - 3: True accounts (64 total)
  - Core accounts (10%)
  - Fan accounts (40%)
  - Individual accounts (50%)
- Trials 4 - 12: False accounts from different categories (6,336 total)
  - Brands and Products (2.50%)
  - Companies and Organizations (2.50%)
  - Local Businesses (0.05%)
  - Movies (0.95%)
  - Music (4.73%)
  - People (85.44%)
  - Sports (2.03%)
  - Television (0.23%)
  - Websites (1.57%)

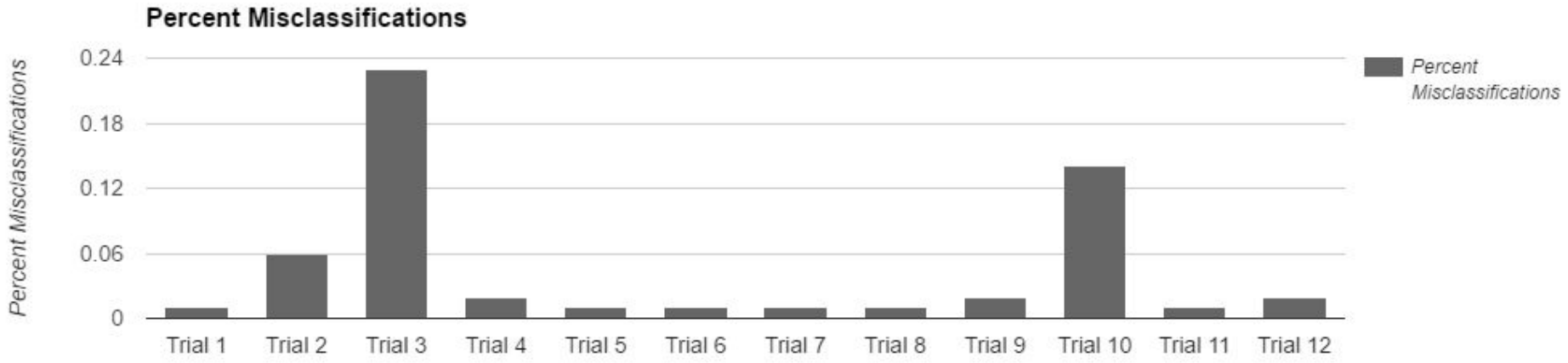
Results/Analysis

84%

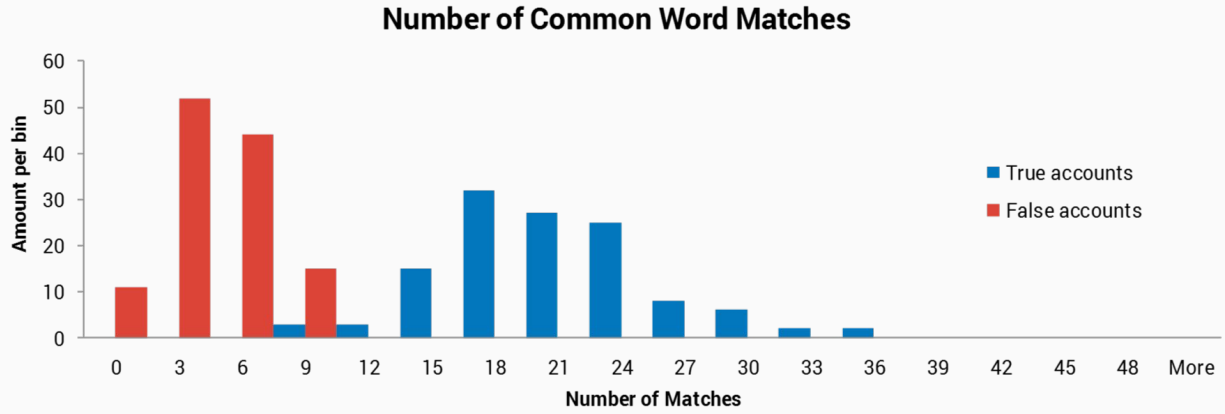
accurate in correctly classifying terrorist accounts

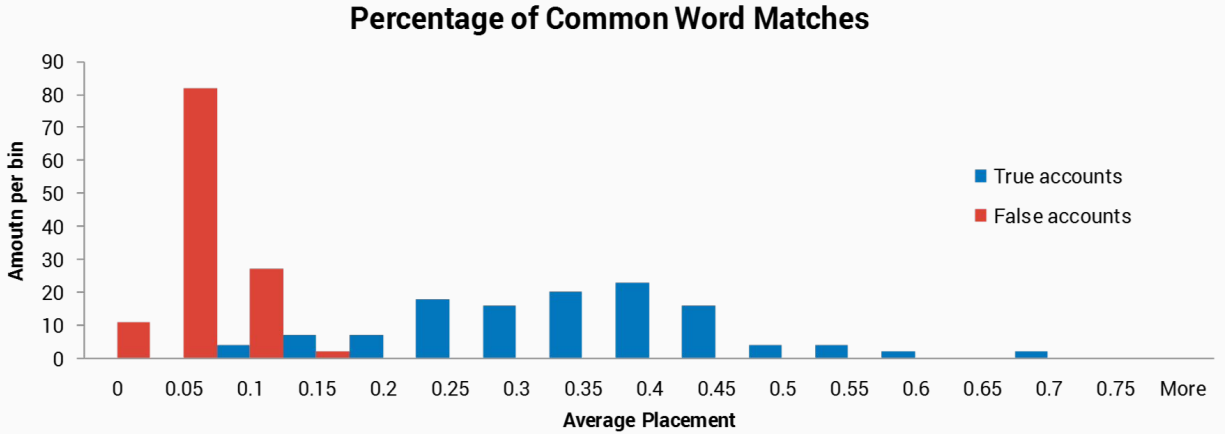
Analysis

	TRUE ACCOUNTS			FALSE ACCOUNTS								
	Trial 1: Core	Trial 2: Fan	Trial 3: Individual	Trial 4: Brands	Trial 5: Companies	Trial 6: Local Businesses	Trial 7: Movies	Trial 8: Music	Trial 9: People	Trial 10: Sports	Trial 11: Television	Trial 12: Websites
Percent Misclassifications	1%	12%	23%	2%	1%	1%	1%	1%	2%	14%	1%	2%

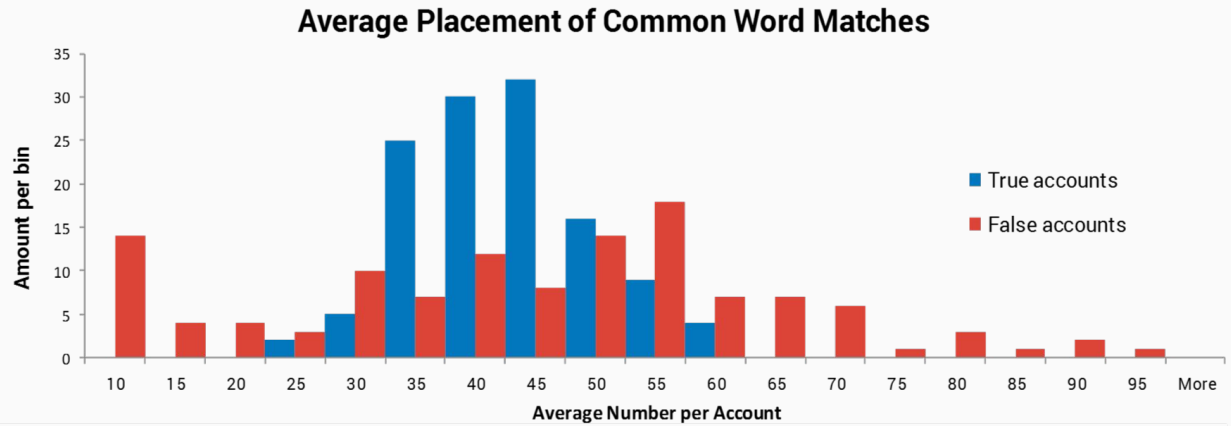


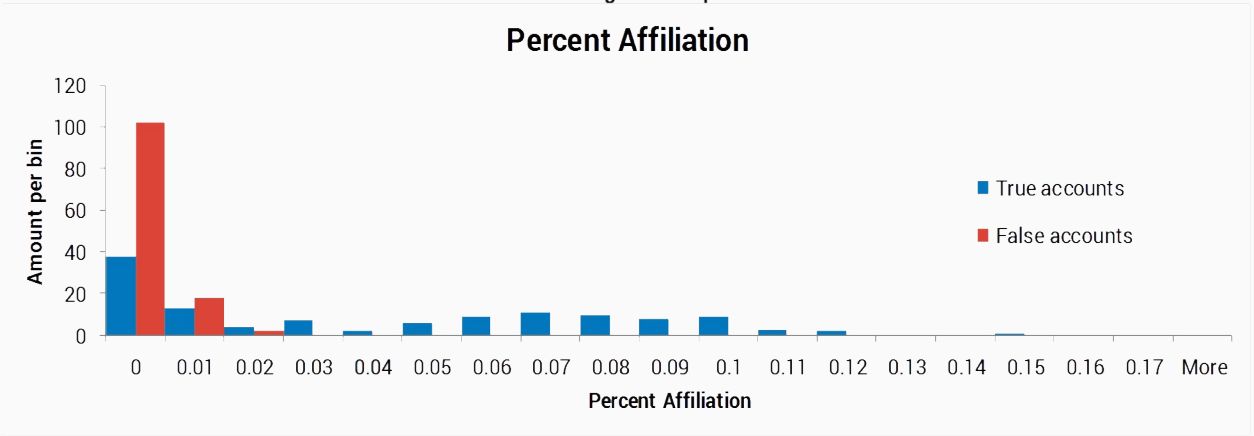
Analysis


Number of Common Word Matches
Amount per bin
60
50
40
30
20
10
0
0
3
6
9
12
15
18
21
24
27
30
33
36
39
42
45
48
More
Number of Matches
True accounts
False accounts


Percentage of Common Word Matches
Amoutn per bin
90
80
70
60
50
40
30
20
10
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65
0.7
0.75
More
Average Placement
True accounts
False accounts

Analysis


Average Placement of Common Word Matches
Amount per bin
0
5
10
15
20
25
30
35
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
More
Average Number per Account
True accounts
False accounts


Percent Affiliation
Amount per bin
0
20
40
60
80
100
120
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
0.11
0.12
0.13
0.14
0.15
0.16
0.17
More
Percent Affiliation
True accounts
False accounts

Conclusion

Overview

Hypothesis was correct:

- Analyzing social media feeds with machine learning algorithms can identify accounts as hostile

Experiment was successful

- National Institute of Justice: “Success rates are based on the consequences of errors”
- 84% accurate in identifying accounts
- False positive rates show areas for improvement

Plans for Improvement

Ameliorations to my project for its betterment.

Implement Additional Parameters

- Visual media analysis
- Connotation and tone analysis

Test Using a Larger Sample Size

Enhance Efficiency

Works Cited

Blakes, Jason A., ed. *#Terrorist: The Use of Social Media By Extremist Groups*. *Academia*. Academia, Oct. 2014. Web. 26 Jan. 2016. <https://www.academia.edu/9015149/_Terrorist_The_Use_of_Social_Media_By_Extremist_Groups>.

Brooking, E.T. "Anonymous vs the Islamic State." *Foreign Policy*. Foreign Policy, 13 Nov. 2015. Web. 24 Nov. 2015. <https://foreignpolicy.com/2015/11/13/anonymous-hackers-islamic-state-isis-chan-online-war/>.

Chemaly, Soraya. "Twitter's Safety and Free Speech Tightrope." *Time*. Time, 23 Apr. 2015. Web. 24 Nov. 2015. <http://time.com/3831595/twitter-free-speech-safety/>.

Coker, Margaret, Sam Schechner, and Alexis Flynn. "How the Islamic State Teaches Tech Savvy to Evade Detection." *Wall Street Journal*. Dow Jones & Company, 16 Nov. 2015. Web. 24 Nov. 2015. <http://www.wsj.com/articles/islamic-state-teaches-tech-savvy-1447720824>.

Colton, Simon. "Multi-Layer Artificial Neural Networks." *Imperial College London*. Imperial College London, 2004. Web. 6 Mar. 2016. <http://www.doc.ic.ac.uk/~sgc/teaching/pre2012/v231/lecture13.html>.

Works Cited

Gerber, Matthew S. *Predicting Crime Using Twitter and Kernel Density Estimation*. N.p.: n.p., 2014. *Predictive Technology Laboratory @ University of Virginia*. Web. 6 Mar. 2016. <http://ptl.sys.virginia.edu/ptl/sites/default/files/manuscript_gerber.pdf>.

Jungherr, Andreas, et al. *Digital Trace Data in the Study of Public Opinion An Indicator of Attention Toward Politics Rather Than Political Support*. N.p.: Sage Journals, 2014. *Social Science Computer Review*. Web. 6 Mar. 2016. <http://ssc.sagepub.com/content/early/2016/02/15/0894439316631043.abstract>.

Louis, Connie St, and Gozde Zorlu. *Can Twitter Predict Disease Outbreaks? The British Medical Journal*. BMJ, 17 May 2012. Web. 6 Mar. 2016. <http://www.bmj.com/content/344/bmj.e2353.abstract>.

Mastroianni, Brian. "Could Policing Social Media Help Prevent Terrorist Attacks?" *CBS News*. CBS News, 15 Dec. 2015. Web. 26 Jan. 2016. <http://www.cbsnews.com/news/could-policing-social-media-prevent-terrorist-attacks/>.

Morgan, Jonathan. *The ISIS Twitter Census*. N.p.: Brookings Project on US Relations with the Islamic World, n.d. Print.

Works Cited

"The Role of Social Media in the Work of Terrorist Groups. The Case of ISIS and Al-Qaeda." *Research and Science Today* 3 (2015): 77-83. Print.

Stergiou, Christos, and Dimitrios Siganos. "Neural Networks." *Imperial College London* 4 (1997): n. pag. Print.

Wiemann, Gabriel. *New Terrorism and New Media*. Research rept. no. 2. *Wilson Center*. Wilson Center, 2014. Web. 26 Jan. 2016. <https://www.wilsoncenter.org/sites/default/files/new_terrorism_v3_1.pdf>.

"Number of Fatalities Due to Terrorists Attacks Worldwide between 2006 and 2014." *Statista*. Statista, 2014. Web. 22 Mar. 2016. <http://www.statista.com/statistics/202871/number-of-fatalities-by-terrorist-attacks-worldwide/>.

"Number of Monthly Active Twitter Users World Wide from 1st Quarter 2010 to 4th Quarter 2015 (in millions)." *Statista*. Statista, 2015. Web. 6 Mar. 2016. <http://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/>.

Works Cited

Perlroth, Nicole, and Mike Isaac. “Terrorists Mock Bids to End Use of Social Media.” *New York Times*. New York Times, 7 Dec. 2015. Web. 26 Jan. 2016. <http://www.nytimes.com/2015/12/08/technology/terrorists-mock-bids-to-end-use-of-social-media.html>.

Perrin, Andrew. *Social Media Usage: 2005-2015*. *Pew Research Center*. Pew Research Center, 8 Oct. 2015. Web. 6 Mar. 2016. <http://www.pewinternet.org/files/2015/10/PI_2015-10-08_Social-Networking-Usage-2005-2015_FINAL.pdf>.

Ritter, Nancy. “Predicting Recidivism Risk: New Tool in Philadelphia Shows Great Promise.” *National Institute of Justice*. Office of Justice Programs, 27 Feb. 2013. Web. 27 Jan. 2016. <http://nij.gov/journals/271/Pages/predicting-recidivism.aspx>.

Secara, Diana. “The Role of Social Media in the Work of Terrorist Groups. The Case of ISIS and Al-Qaeda.” *Research and Science Today* 3 (2015): 77-83. Print.

Stergiou, Christos, and Dimitrios Siganos. “Neural Networks.” *Imperial College London* 4 (1997): n. pag. Print.

Works Cited

Twitter4J. 4th ed. Vers. 0. Rel. 4. *Twitter4J*. Twitter4J, n.d. Web. 6 Mar. 2016. <http://twitter4j.org/en/index.html>.

Wiemann, Gabriel. *New Terrorism and New Media*. Research rept. no. 2. *Wilson Center*. Wilson Center, 2014. Web. 26 Jan. 2016. <https://www.wilsoncenter.org/sites/default/files/new_terrorism_v3_1.pdf>.