AI Based Opto-Lexical Pattern Analysis for Behavior Classification

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

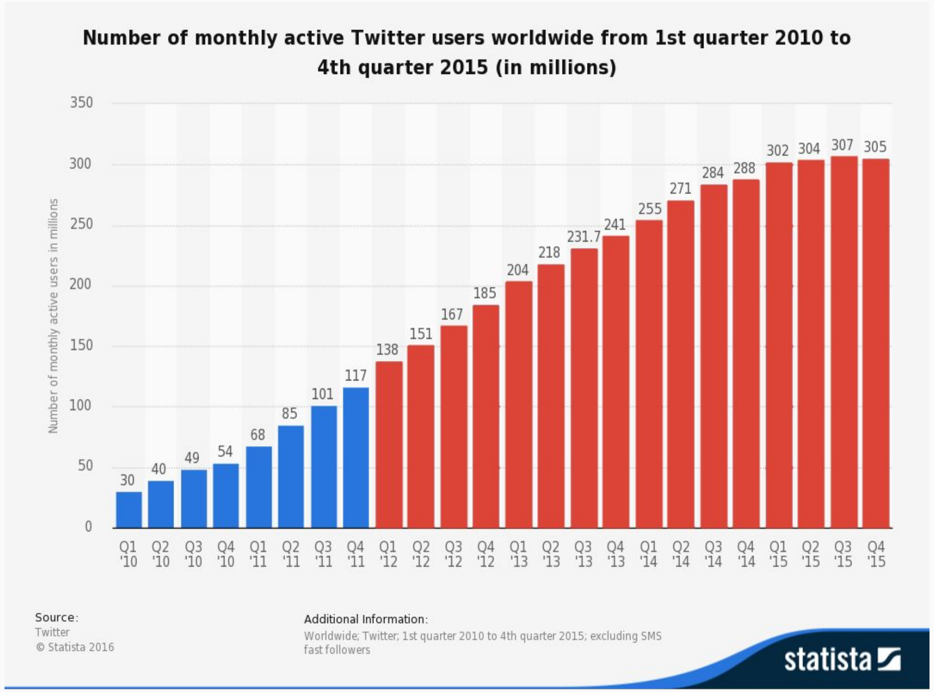
Twitter is the platform on which most of this occurs.

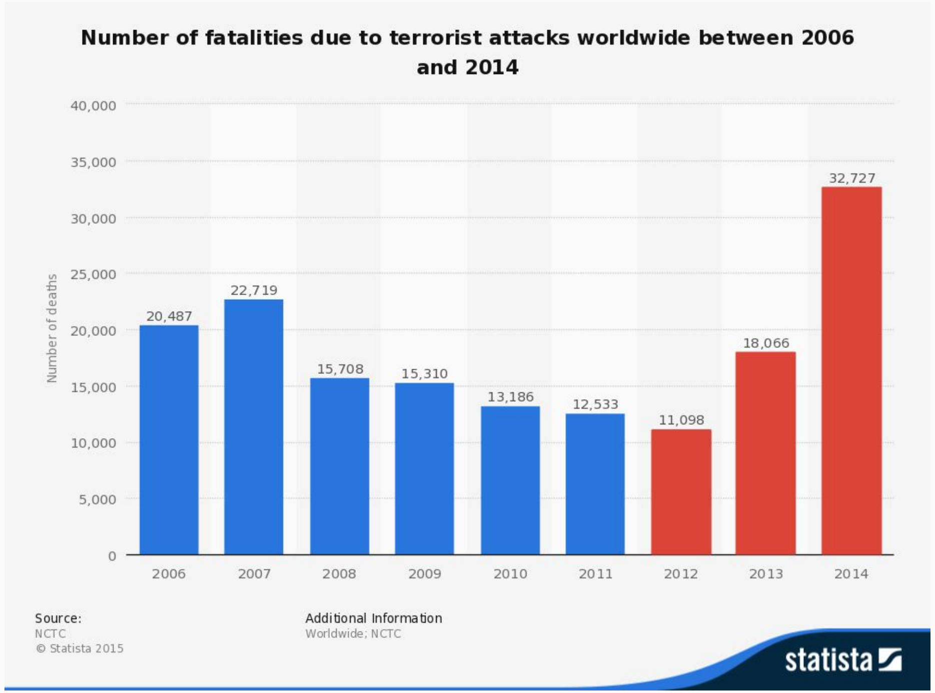
- Microblogging site
- Active user base of 300 million

There are an estimated 90,000 terrorist accounts on Twitter (0.03%).

Background

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





Analyzing social media feeds with machine learning algorithms can classify behavioral patterns

Algorithm

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

Twitter Account Data Collection

Data Parsing

Data Analysis & Prediction

Methodology

The 27 parameters used to ensure minimal false-positives.

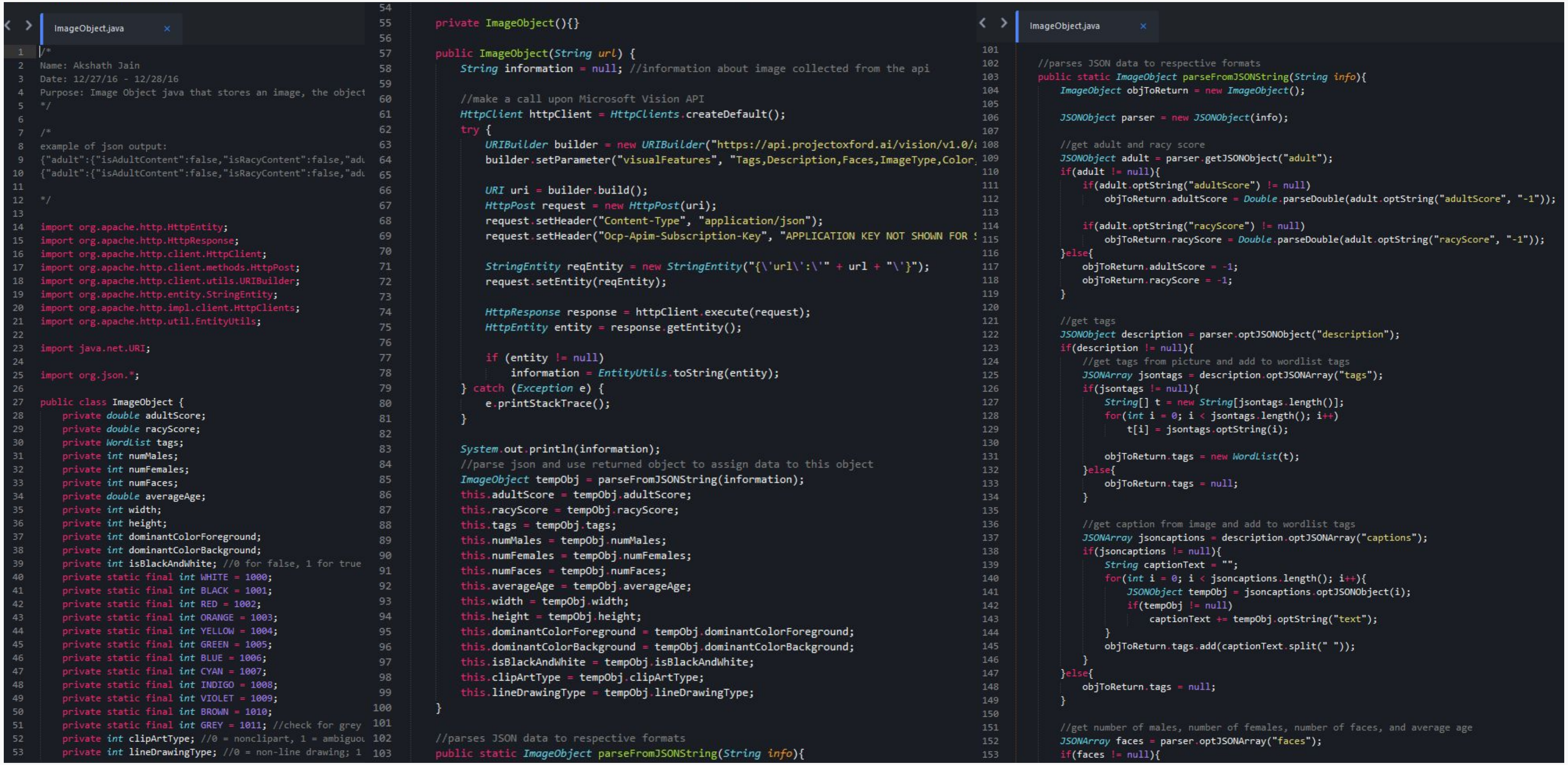
Diction
- Word choice/frequency
- Percent match
- Average match distribution
- Hashtags

Affiliation to Known Accounts
- Friends
- Followers
- Retweets
- Content mentions

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

Visual Media
- Adult Score
- Racy Score
- Autogenerated image caption
- Number of males
- Number of females
- Number of faces
- Average age
- Width
- Height
- Foreground color
- Background color
- Blaw and white status
- Clipart status
- Vector-style status

Code Sample



ImageObject.java

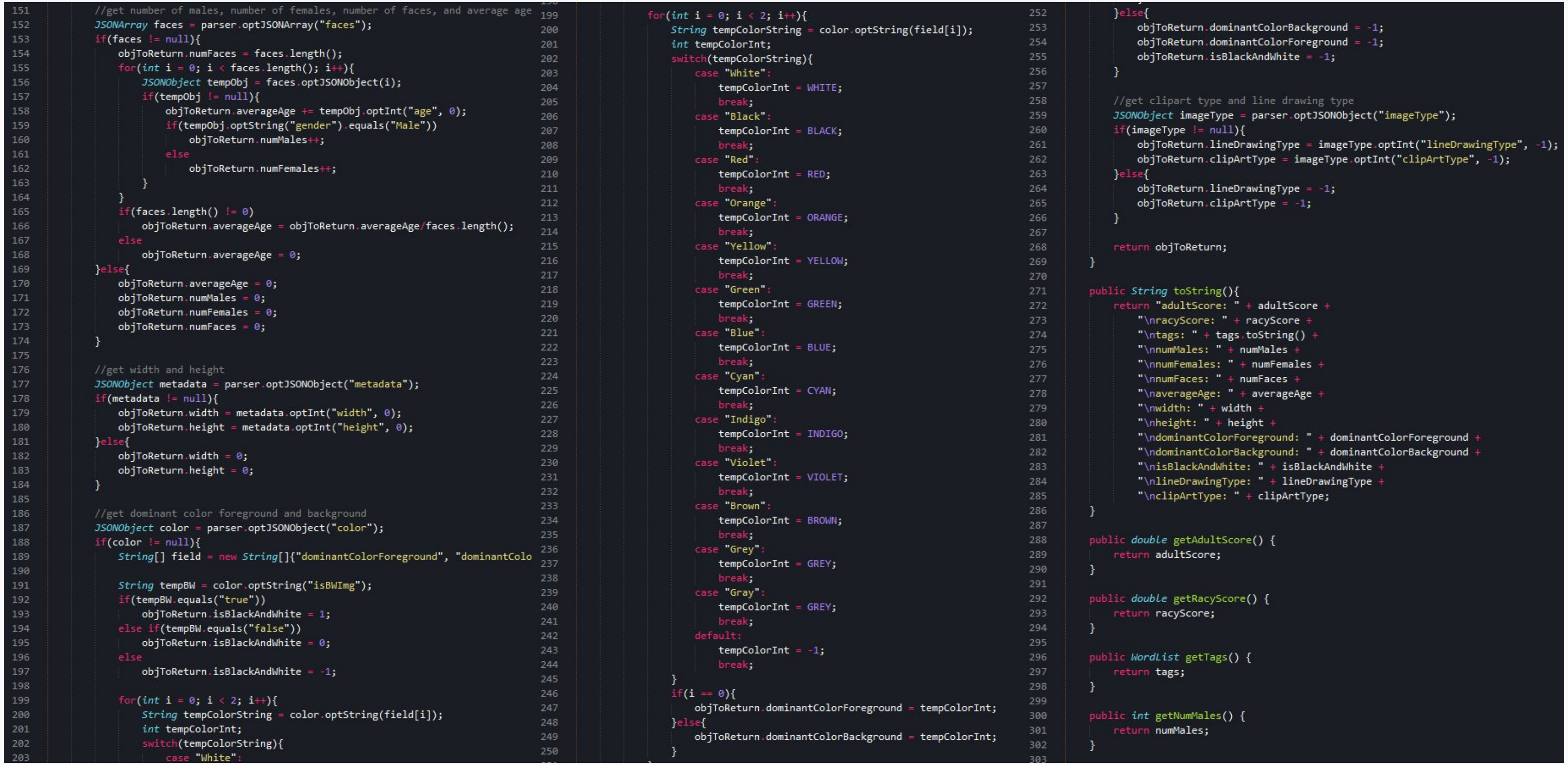
```java
/*
Name: Akshath Jain
Date: 12/27/16 - 12/28/16
Purpose: Image Object java that stores an image, the object
*/

/*
example of json output:
{"adult":{"isAdultContent":false,"isRacyContent":false,"adu
{"adult":{"isAdultContent":false,"isRacyContent":false,"adu

*/

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URIBuilder;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import java.net.URI;

import org.json.*;

public class ImageObject {
    private double adultScore;
    private double racyScore;
    private WordList tags;
    private int numMales;
    private int numFemales;
    private int numFaces;
    private double averageAge;
    private int width;
    private int height;
    private int dominantColorForeground;
    private int dominantColorBackground;
    private int isBlackAndWhite; //0 for false, 1 for true
    private static final int WHITE = 1000;
    private static final int BLACK = 1001;
    private static final int RED = 1002;
    private static final int ORANGE = 1003;
    private static final int YELLOW = 1004;
    private static final int GREEN = 1005;
    private static final int BLUE = 1006;
    private static final int CYAN = 1007;
    private static final int INDIGO = 1008;
    private static final int VIOLET = 1009;
    private static final int BROWN = 1010;
    private static final int GREY = 1011; //check for grey
    private int clipArtType; //0 = nonclipart, 1 = ambiguou
    private int lineDrawingType; //0 = non-line drawing; 1
```

```java

    private ImageObject(){}

    public ImageObject(String url) {
        String information = null; //information about image collected from the api

        //make a call upon Microsoft Vision API
        HttpClient httpClient = HttpClients.createDefault();
        try {
            URIBuilder builder = new URIBuilder("https://api.projectoxford.ai/vision/v1.0/
            builder.setParameter("visualFeatures", "Tags,Description,Faces,ImageType,Color

            URI uri = builder.build();
            HttpPost request = new HttpPost(uri);
            request.setHeader("Content-Type", "application/json");
            request.setHeader("Ocp-Apim-Subscription-Key", "APPLICATION KEY NOT SHOWN FOR 

            StringEntity reqEntity = new StringEntity("{\'url\':\'" + url + "\'}");
            request.setEntity(reqEntity);

            HttpResponse response = httpClient.execute(request);
            HttpEntity entity = response.getEntity();

            if (entity != null)
                information = EntityUtils.toString(entity);
        } catch (Exception e) {
            e.printStackTrace();
        }

        System.out.println(information);
        //parse json and use returned object to assign data to this object
        ImageObject tempObj = parseFromJSONString(information);
        this.adultScore = tempObj.adultScore;
        this.racyScore = tempObj.racyScore;
        this.tags = tempObj.tags;
        this.numMales = tempObj.numMales;
        this.numFemales = tempObj.numFemales;
        this.numFaces = tempObj.numFaces;
        this.averageAge = tempObj.averageAge;
        this.width = tempObj.width;
        this.height = tempObj.height;
        this.dominantColorForeground = tempObj.dominantColorForeground;
        this.dominantColorBackground = tempObj.dominantColorBackground;
        this.isBlackAndWhite = tempObj.isBlackAndWhite;
        this.clipArtType = tempObj.clipArtType;
        this.lineDrawingType = tempObj.lineDrawingType;
    }

    //parses JSON data to respective formats
    public static ImageObject parseFromJSONString(String info){
```

ImageObject.java

```java

    //parses JSON data to respective formats
    public static ImageObject parseFromJSONString(String info){
        ImageObject objToReturn = new ImageObject();

        JSONObject parser = new JSONObject(info);

        //get adult and racy score
        JSONObject adult = parser.getJSONObject("adult");
        if(adult != null){
            if(adult.optString("adultScore") != null)
                objToReturn.adultScore = Double.parseDouble(adult.optString("adultScore", "-1"));

            if(adult.optString("racyScore") != null)
                objToReturn.racyScore = Double.parseDouble(adult.optString("racyScore", "-1"));
        }else{
            objToReturn.adultScore = -1;
            objToReturn.racyScore = -1;
        }

        //get tags
        JSONObject description = parser.optJSONObject("description");
        if(description != null){
            //get tags from picture and add to wordlist tags
            JSONArray jsontags = description.optJSONArray("tags");
            if(jsontags != null){
                String[] t = new String[jsontags.length()];
                for(int i = 0; i < jsontags.length(); i++)
                    t[i] = jsontags.optString(i);

                objToReturn.tags = new WordList(t);
            }else{
                objToReturn.tags = null;
            }

            //get caption from image and add to wordlist tags
            JSONArray jsoncaptions = description.optJSONArray("captions");
            if(jsoncaptions != null){
                String captionText = "";
                for(int i = 0; i < jsoncaptions.length(); i++){
                    JSONObject tempObj = jsoncaptions.optJSONObject(i);
                    if(tempObj != null)
                        captionText += tempObj.optString("text");
                }
                objToReturn.tags.add(captionText.split(" "));
            }
        }else{
            objToReturn.tags = null;
        }

        //get number of males, number of females, number of faces, and average age
        JSONArray faces = parser.optJSONArray("faces");
        if(faces != null){
```

Code Sample

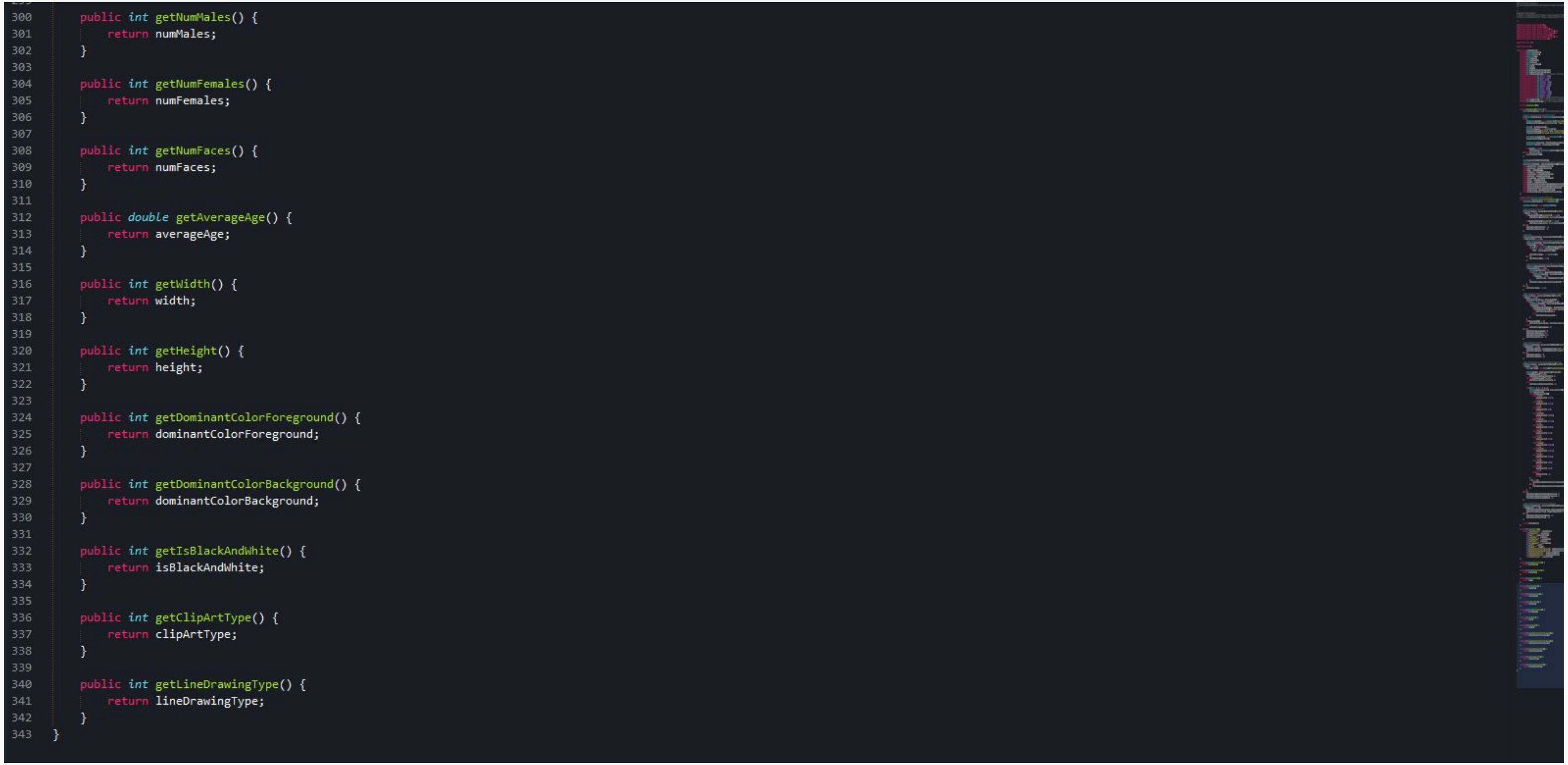


```java
        //get number of males, number of females, number of faces, and average age
        JSONArray faces = parser.optJSONArray("faces");
        if(faces != null){
            objToReturn.numFaces = faces.length();
            for(int i = 0; i < faces.length(); i++){
                JSONObject tempObj = faces.optJSONObject(i);
                if(tempObj != null){
                    objToReturn.averageAge += tempObj.optInt("age", 0);
                    if(tempObj.optString("gender").equals("Male"))
                        objToReturn.numMales++;
                    else
                        objToReturn.numFemales++;
                }
            }
            if(faces.length() != 0)
                objToReturn.averageAge = objToReturn.averageAge/faces.length();
            else
                objToReturn.averageAge = 0;
        }else{
            objToReturn.averageAge = 0;
            objToReturn.numMales = 0;
            objToReturn.numFemales = 0;
            objToReturn.numFaces = 0;
        }

        //get width and height
        JSONObject metadata = parser.optJSONObject("metadata");
        if(metadata != null){
            objToReturn.width = metadata.optInt("width", 0);
            objToReturn.height = metadata.optInt("height", 0);
        }else{
            objToReturn.width = 0;
            objToReturn.height = 0;
        }

        //get dominant color foreground and background
        JSONObject color = parser.optJSONObject("color");
        if(color != null){
            String[] field = new String[]{"dominantColorForeground", "dominantColo

            String tempBW = color.optString("isBWImg");
            if(tempBW.equals("true"))
                objToReturn.isBlackAndWhite = 1;
            else if(tempBW.equals("false"))
                objToReturn.isBlackAndWhite = 0;
            else
                objToReturn.isBlackAndWhite = -1;

            for(int i = 0; i < 2; i++){
                String tempColorString = color.optString(field[i]);
                int tempColorInt;
                switch(tempColorString){
                    case "White":
```

```java
            for(int i = 0; i < 2; i++){
                String tempColorString = color.optString(field[i]);
                int tempColorInt;
                switch(tempColorString){
                    case "White":
                        tempColorInt = WHITE;
                        break;
                    case "Black":
                        tempColorInt = BLACK;
                        break;
                    case "Red":
                        tempColorInt = RED;
                        break;
                    case "Orange":
                        tempColorInt = ORANGE;
                        break;
                    case "Yellow":
                        tempColorInt = YELLOW;
                        break;
                    case "Green":
                        tempColorInt = GREEN;
                        break;
                    case "Blue":
                        tempColorInt = BLUE;
                        break;
                    case "Cyan":
                        tempColorInt = CYAN;
                        break;
                    case "Indigo":
                        tempColorInt = INDIGO;
                        break;
                    case "Violet":
                        tempColorInt = VIOLET;
                        break;
                    case "Brown":
                        tempColorInt = BROWN;
                        break;
                    case "Grey":
                        tempColorInt = GREY;
                        break;
                    case "Gray":
                        tempColorInt = GREY;
                        break;
                    default:
                        tempColorInt = -1;
                        break;
                }
                if(i == 0){
                    objToReturn.dominantColorForeground = tempColorInt;
                }else{
                    objToReturn.dominantColorBackground = tempColorInt;
                }
```

```java
        }else{
            objToReturn.dominantColorBackground = -1;
            objToReturn.dominantColorForeground = -1;
            objToReturn.isBlackAndWhite = -1;
        }

        //get clipart type and line drawing type
        JSONObject imageType = parser.optJSONObject("imageType");
        if(imageType != null){
            objToReturn.lineDrawingType = imageType.optInt("lineDrawingType", -1);
            objToReturn.clipArtType = imageType.optInt("clipArtType", -1);
        }else{
            objToReturn.lineDrawingType = -1;
            objToReturn.clipArtType = -1;
        }

        return objToReturn;
    }

    public String toString(){
        return "adultScore: " + adultScore +
            "\nracyScore: " + racyScore +
            "\ntags: " + tags.toString() +
            "\nnumMales: " + numMales +
            "\nnumFemales: " + numFemales +
            "\nnumFaces: " + numFaces +
            "\naverageAge: " + averageAge +
            "\nwidth: " + width +
            "\nheight: " + height +
            "\ndominantColorForeground: " + dominantColorForeground +
            "\ndominantColorBackground: " + dominantColorBackground +
            "\nisBlackAndWhite: " + isBlackAndWhite +
            "\nlineDrawingType: " + lineDrawingType +
            "\nclipArtType: " + clipArtType;
    }

    public double getAdultScore() {
        return adultScore;
    }

    public double getRacyScore() {
        return racyScore;
    }

    public WordList getTags() {
        return tags;
    }

    public int getNumMales() {
        return numMales;
    }
```

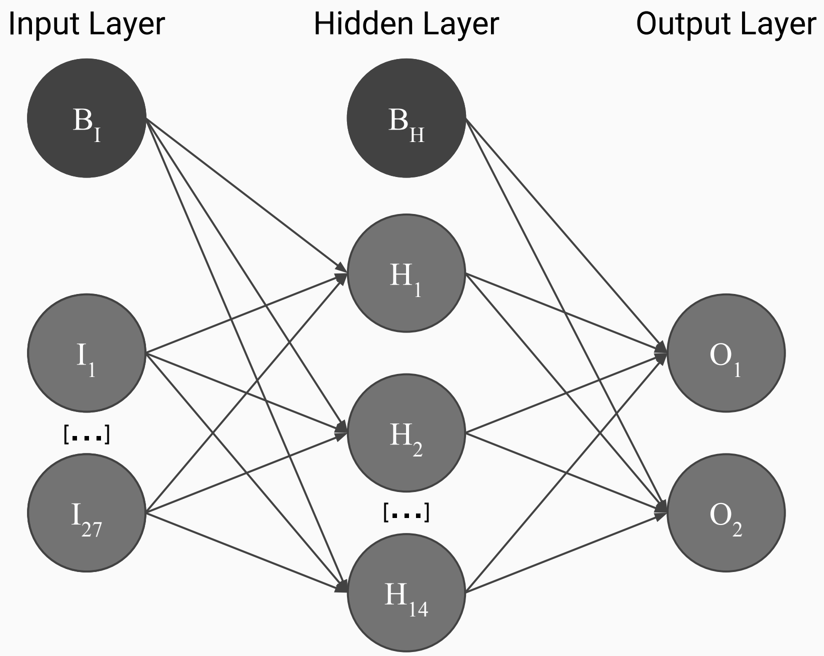
Code Sample



```java
    public int getNumMales() {
        return numMales;
    }

    public int getNumFemales() {
        return numFemales;
    }

    public int getNumFaces() {
        return numFaces;
    }

    public double getAverageAge() {
        return averageAge;
    }

    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }

    public int getDominantColorForeground() {
        return dominantColorForeground;
    }

    public int getDominantColorBackground() {
        return dominantColorBackground;
    }

    public int getIsBlackAndWhite() {
        return isBlackAndWhite;
    }

    public int getClipArtType() {
        return clipArtType;
    }

    public int getLineDrawingType() {
        return lineDrawingType;
    }
}
```

Prediction Algorithm

Neural Network Diagram

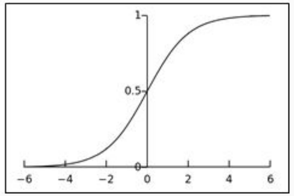


Regular Usage

$$H_j = \sigma\left(w_{B_I H_j} + \sum_{i=1}^{14} I_n w_{I_i H_j}\right)$$

$$O_k = \sigma\left(w_{B_H O_k} + \sum_{j=1}^{2} H_j w_{H_j O_k}\right)$$

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



Network Training

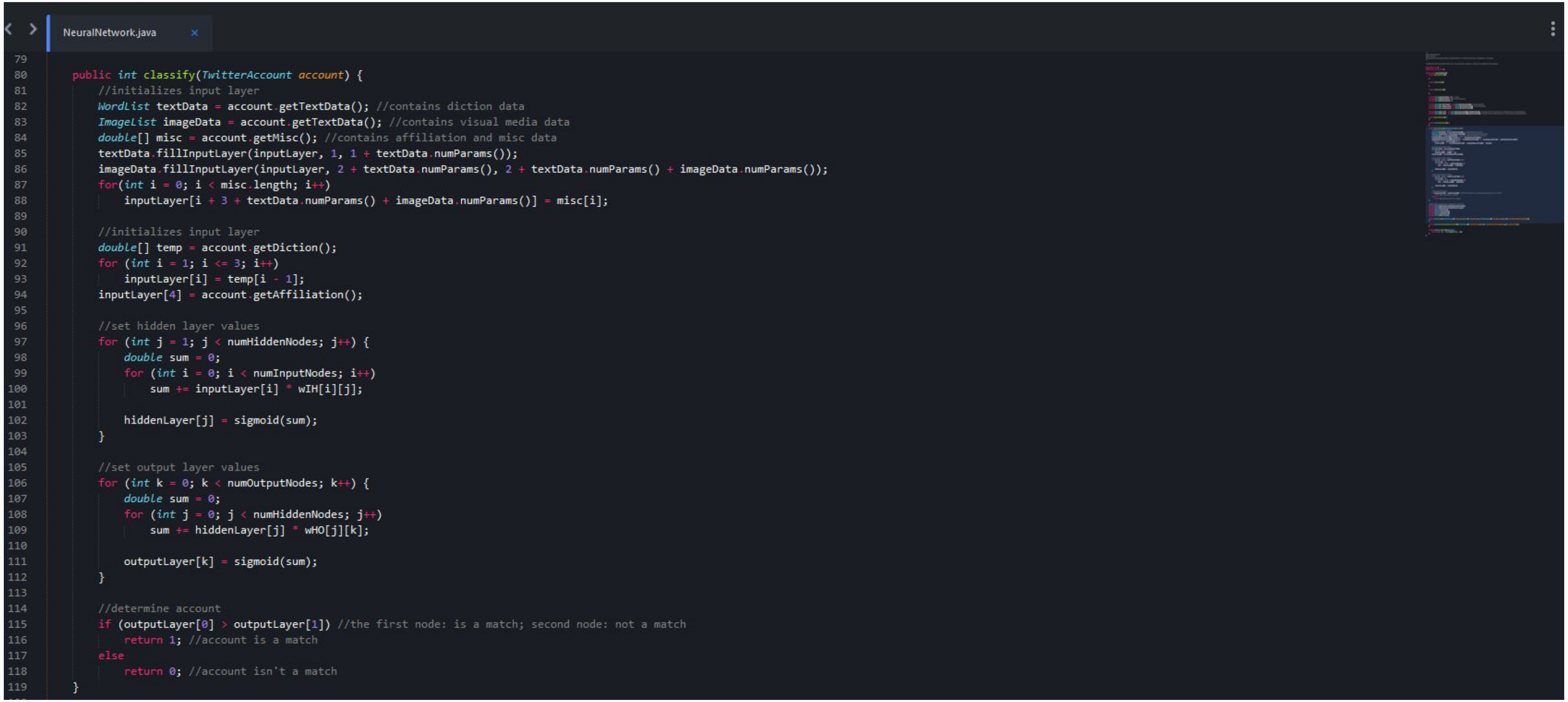
$$\delta O_k = O_k(E)\left(1 - O_k(E)\right)\left(T_k(E) - O_k(E)\right)$$

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

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

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

Neural Network Code Sample



```java
public int classify(TwitterAccount account) {
    //initializes input layer
    WordList textData = account.getTextData(); //contains diction data
    ImageList imageData = account.getTextData(); //contains visual media data
    double[] misc = account.getMisc(); //contains affiliation and misc data
    textData.fillInputLayer(inputLayer, 1, 1 + textData.numParams());
    imageData.fillInputLayer(inputLayer, 2 + textData.numParams(), 2 + textData.numParams() + imageData.numParams());
    for(int i = 0; i < misc.length; i++)
        inputLayer[i + 3 + textData.numParams() + imageData.numParams()] = misc[i];

    //initializes input layer
    double[] temp = account.getDiction();
    for (int i = 1; i <= 3; i++)
        inputLayer[i] = temp[i - 1];
    inputLayer[4] = account.getAffiliation();

    //set hidden layer values
    for (int j = 1; j < numHiddenNodes; j++) {
        double sum = 0;
        for (int i = 0; i < numInputNodes; i++)
            sum += inputLayer[i] * wIH[i][j];

        hiddenLayer[j] = sigmoid(sum);
    }

    //set output layer values
    for (int k = 0; k < numOutputNodes; k++) {
        double sum = 0;
        for (int j = 0; j < numHiddenNodes; j++)
            sum += hiddenLayer[j] * wHO[j][k];

        outputLayer[k] = sigmoid(sum);
    }

    //determine account
    if (outputLayer[0] > outputLayer[1]) //the first node: is a match; second node: not a match
        return 1; //account is a match
    else
        return 0; //account isn't a match
}
```

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

- Sample data set used
  - Precompiled data set of known hostile accounts
  - Contains images, text, media, etc

Overview

Training

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

Testing

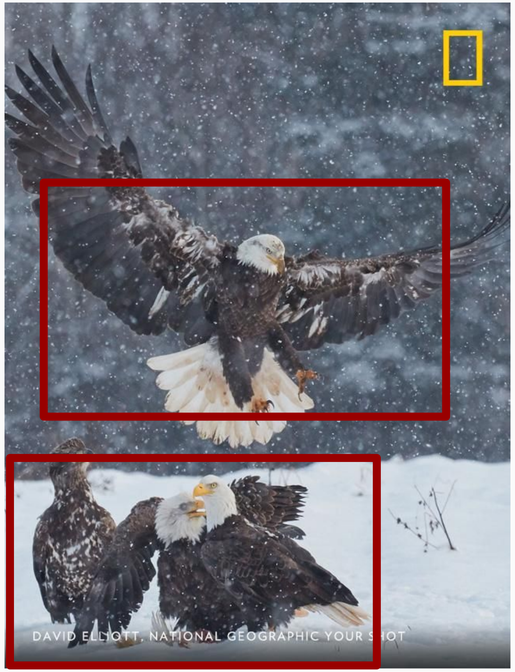
- Trials 1 - 3: Positive accounts (64 total)
  - Core accounts (10%)
  - Imitation accounts (40%)
  - Individual accounts (50%)
- Trials 4 - 12: Negative accounts from different categories (6,336 total)
  - Brands and Products (2.50%)
  - Companies and Organizations (2.50%)
  - Local Businesses (0.05%)
  - 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

Visual Media Analysis Sample #1



Auto-generated caption: *"a group of birds sitting in the snow"*

Source: @NatGeoPhotos

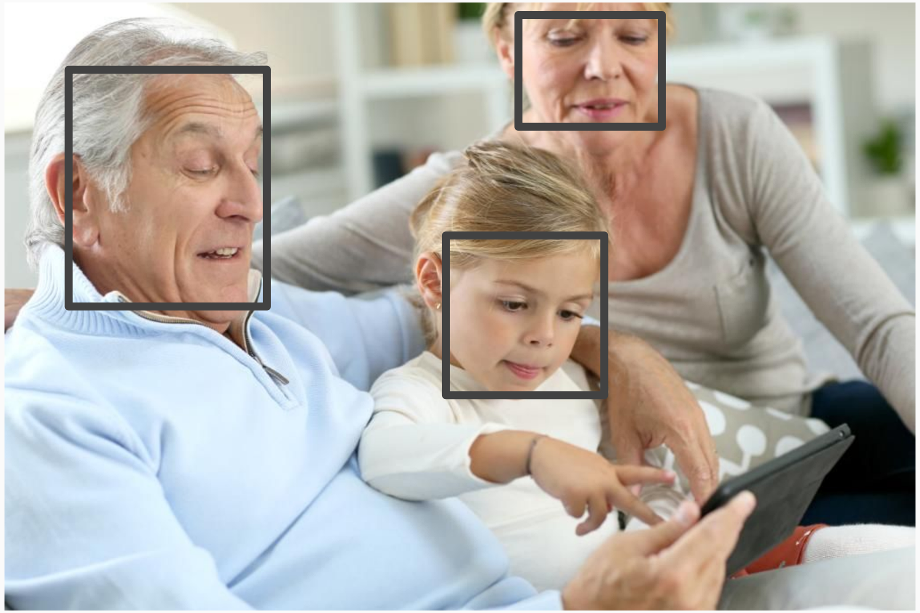
Metadata

- Height: *864*
- Width: *648*
- Prominent foreground color: *grey*
- Prominent background color: *grey*

Description

"fish", "animal", "water", "snow", "sitting", "table", "man", "top", "boat", "bird", "large", "standing", "blue", "parked", "air", "skiing", "ocean", "white", "laying", "group", "people", "riding", "playing", "cat", "beach"

Visual Media Analysis Sample #2



Auto-generated caption: *"a man with a computer holding a little girl"*

Metadata

- Height: *683*
- Width: *1024*
- Male: *1, age 70*
- Females: *2, ages 6 and 55*
- Prominent foreground color: *white*
- Prominent background color: *white*

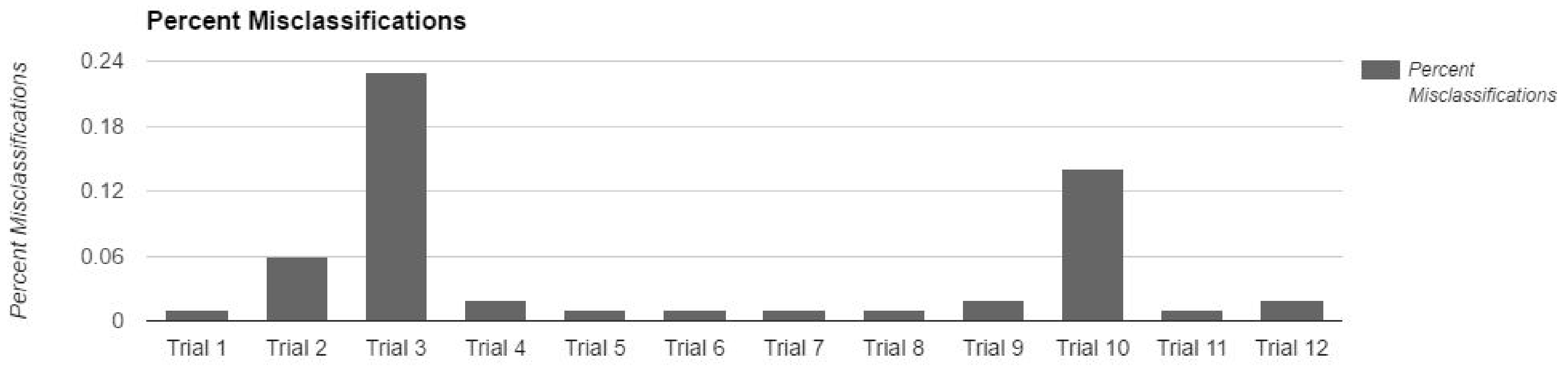
Description

"Person", "sitting", "man", "table", "looking", "holding", "laptop", "older", "people", "using", "computer", "woman", "baby", "food", "playing", "young", "white", "player", "room", "group", "phone"

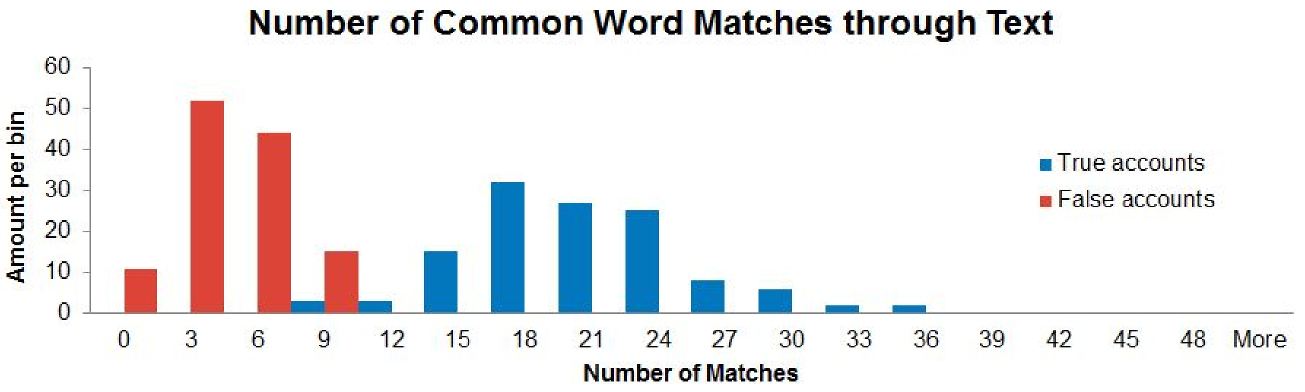
Source: @Forbes

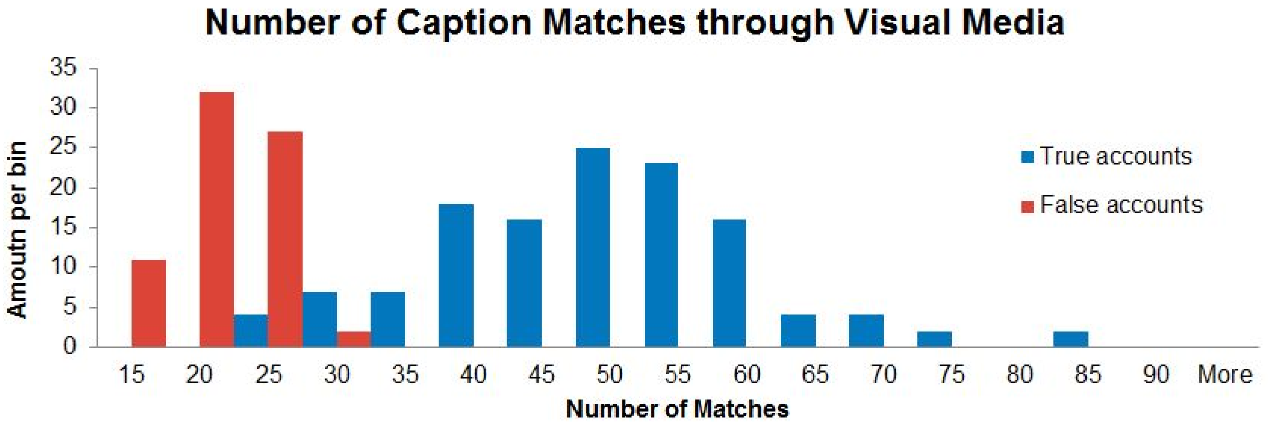
Analysis

	TRUE ACCOUNTS			FALSE ACCOUNTS								
	Trial 1: Core	Trial 2: Imitation	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%

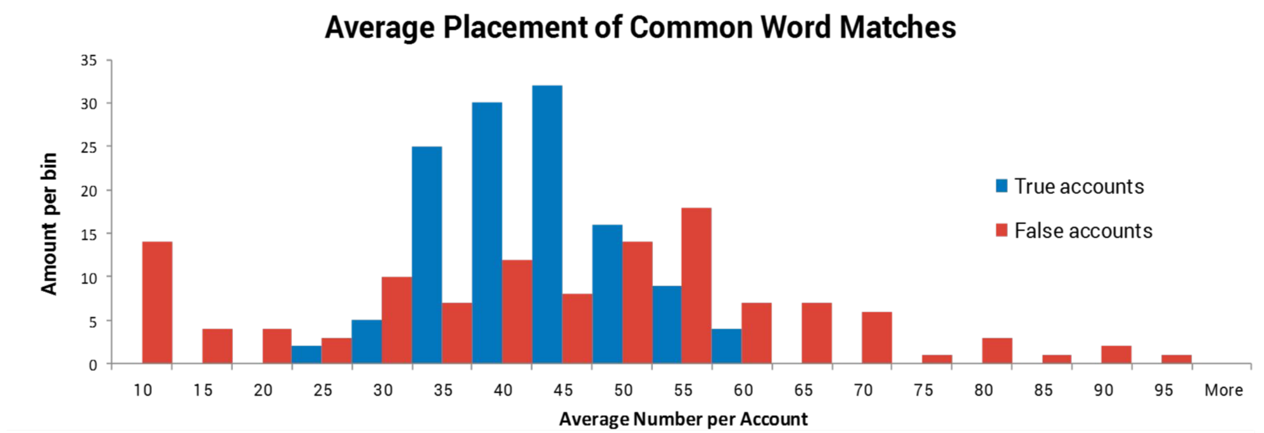


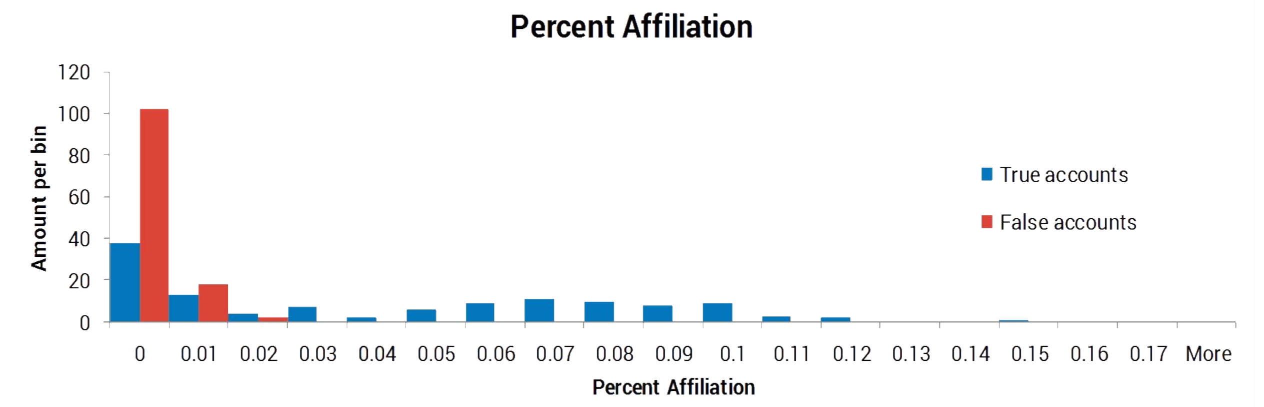
Analysis


Number of Common Word Matches through Text
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


Number of Caption Matches through Visual Media
Amoutn per bin
35
30
25
20
15
10
5
0
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
More
Number of Matches
True accounts
False accounts

Analysis


Average Placement of Common Word Matches
Amount per bin
35
30
25
20
15
10
5
0
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
120
100
80
60
40
20
0
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 classify behavioral patterns

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.

Expand Analysis Platforms

- Other types of social media
- More analysis parameters

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. 2017.

Brooking, E.T. “Anonymous vs the Islamic State.” *Foreign Policy*. Foreign Policy, 13 Nov. 2015. Web. 24 Nov. 2016.

Chemaly, Soraya. “Twitter’s Safety and Free Speech Tightrope.” *Time*. Time, 23 Apr. 2015. Web. 24 Nov. 2016.

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. 2016.

Colton, Simon. “Multi-Layer Artificial Neural Networks.” *Imperial College London*. Imperial College London, 2004. Web. 6 Nov. 2016.

Gerber, Matthew S. *Predicting Crime Using Twitter and Kernel Density Estimation*. N.p.: n.p., 2014. *Predictive Technology Laboratory @ University of Virginia*. Web. 6 Nov. 2016.

Works Cited

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 Nov. 2016.

Louis, Connie St, and Gozde Zorlu. *Can Twitter Predict Disease Outbreaks? The British Medical Journal*. BMJ, 17 May 2012. Web. 6 Dec. 2016.

Mastroianni, Brian. “Could Policing Social Media Help Prevent Terrorist Attacks?” *CBS News*. CBS News, 15 Dec. 2015. Web. 26 Jan. 2017.

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

“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

Wiemann, Gabriel. *New Terrorism and New Media*. Research rept. no. 2. *Wilson Center*. Wilson Center, 2014. Web. 26 Jan. 2017.

"Number of Fatalities Due to Terrorists Attacks Worldwide between 2006 and 2014." *Statista*. Statista, 2014. Web. 27 Jan. 2017.

"Number of Monthly Active Twitter Users Worldwide from 1st Quarter 2010 to 4th Quarter 2015 (in millions)." *Statista*. Statista, 2015. Web. 26 Mar. 2017.

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. 2017.

Perrin, Andrew. *Social Media Usage: 2005-2015*. *Pew Research Center*. Pew Research Center, 8 Oct. 2015. Web. 26 Jan. 2017.

Ritter, Nancy. "Predicting Recidivism Risk: New Tool in Philadelphia Shows Great Promise." *National Institute of Justice*. Office of Justice Programs, 27 Feb. 2013. Web. 26 Jan. 2017.

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.

Works Cited

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

Twitter4J. 4th ed. Vers. 0. Rel. 4. *Twitter4J*. Twitter4J, n.d. Web. 6 Mar. 2017.

Wassner, Hubert. "Twitter Friends." Twitter Friends. Kaggle, Aug. 2016. Web. 26 Jan. 2017.

Wiemann, Gabriel. *New Terrorism and New Media*. Research rept. no. 2. *Wilson Center*. Wilson Center, 2014. Web. 26 Jan. 2017.

Zaman, Khuram. "How ISIS Uses Twitter." How ISIS Uses Twitter. Kaggle, May 2016. Web. 26 Jan. 2017.