Aaron Schmitz (schmitza1@hawkmail.newpaltz.edu)

CPS493 – Introduction to Data Science

6 May 2019

Professor Min Chen

Final Report

THE NEWS HEADLINE SARCASM DETECTOR

Table of Contents

Principal Investigator

Task	Member 1	Member 2	Member 3	Total
Introduction				100%
Background				100%
Implementation				100%
Experiment Results and Discussion		Aaron Schmitz schmitz1@hawkmail.newpaltz.edu		100%
Conclusion				100%
Other contribution and explain				100%

Title of Project

THE NEWS HEADLINE SARCASM DETECTOR

Mentoring

Professor Min Chen, Department of Computer Science, SUNY-New Paltz

chenm@newpaltz.edu

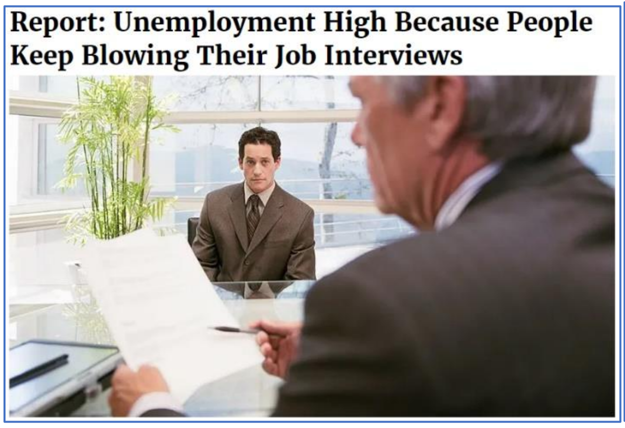
Contents of Zipped Folder

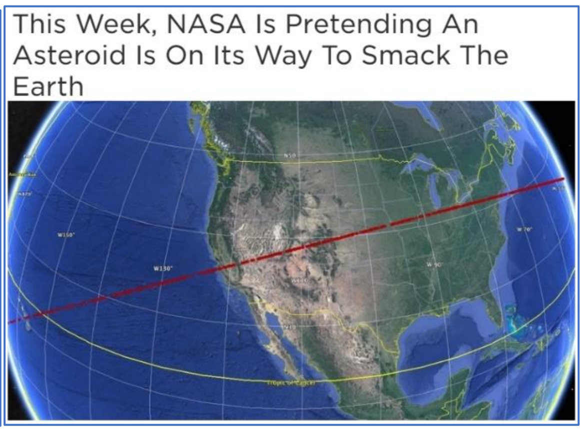
• HeadlineSplitter.java	-	MapReduce function
• Main.java	-	Main program
• MainAccuracyTest.java	-	For testing accuracy
• WordCount.java	-	Standard Hadoop WordCount function
• CountSort.java	-	Modified WordCount for sorting
• mappedData	-	Mapped dataset
• reducedData	-	Mapped-reduced dataset
• allWords	-	All words from all headlines counted
• seriousWords	-	Words from serious headlines counted
• sarcasticWords	-	Words from sarcastic headlines counted
• sd.jar	-	Runnable .jar file to run the program!

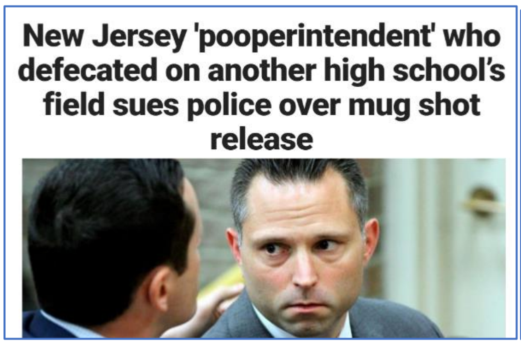
Introduction

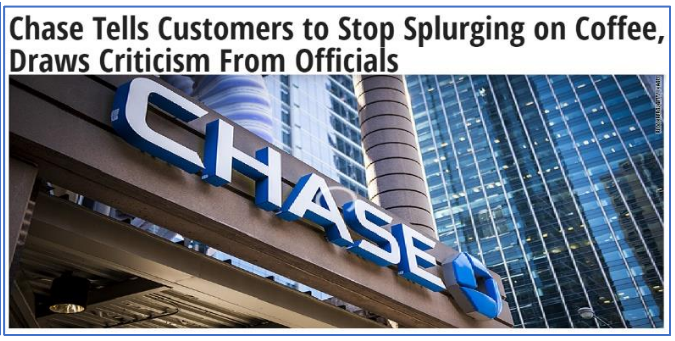
Project Motivation

I’ll let these images speak for themselves:


Report: Unemployment High Because People Keep Blowing Their Job Interviews

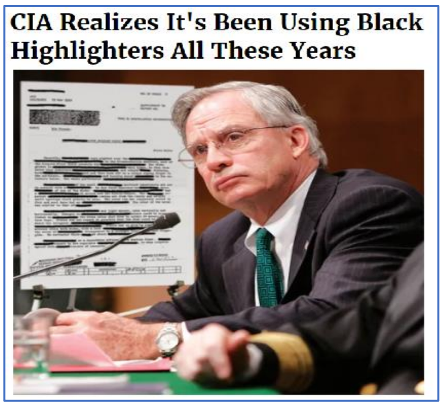

This Week, NASA Is Pretending An Asteroid Is On Its Way To Smack The Earth

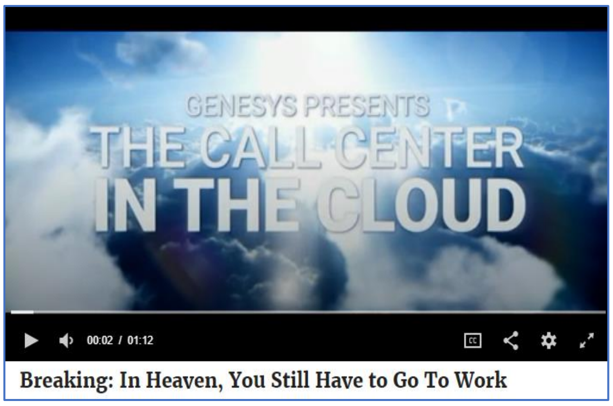

New Jersey 'pooperintendent' who defecated on another high school’s field sues police over mug shot release


Chase Tells Customers to Stop Splurging on Coffee, Draws Criticism From Officials

Every one of these four “news” headlines can be found on social media, but only *one* of the above images is “sarcastic” (i.e. a joke, written for comedic reasons – not *actual* news). If you ask me, most of those headlines each sound ridiculous (in their own way), but unfortunately, *three-out-of-the-four* are real! The least ridiculous headline—the one about unemployment—is the sarcastic one; written by *The Onion*, a satirical news website. All the others are real.

Sometimes the sarcastic/joke/satire news is easy to distinguish, like these, for example:


CIA Realizes It's Been Using Black Highlighters All These Years



Breaking: In Heaven, You Still Have to Go To Work

Both of these are jokes.

As seen in the first example, sometimes it's not so easy to tell them apart, and somebody, somewhere out there, is going to fall for them. If one were to come across any of these articles on social media, they may take it at face value. Most people don't have the time/knowledge/will to fact check everything they read on Facebook (and many will only read the headline), and the news is supposed to report true events. Currently, there's a lot of pressure on social media networks (mostly Facebook) about their (mis)handling of information.

Which is why all social media platforms should buy my project!

Aims and Objectives

Make a program that can differentiate serious news from sarcastic news with reasonable accuracy!

Report Structure

My report uses the terms "serious" and "sarcastic" to refer to the news headlines. *Serious* will mean any news article that is meant to be taken seriously, such as real breaking news stories. *Sarcastic* will mean any news that is *not* meant to be taken seriously, it's lighthearted, a joke.

I will also provide many examples of my project in action, predicting whether a news headline is likely or unlikely to be sarcastic.

Background

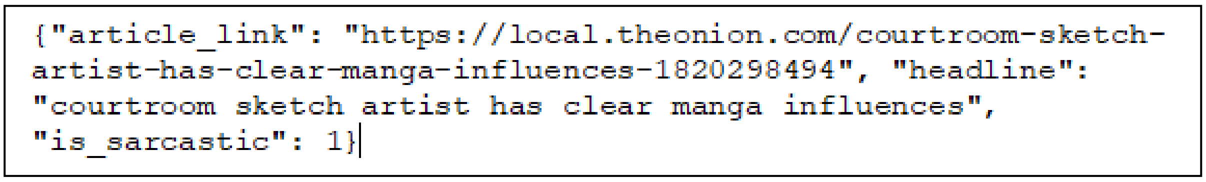
Recently, Facebook came under fire because of the *Cambridge Analytica* (CA) fiasco, when a political consulting firm used data science algorithms to target certain individuals with news recommendations to influence elections around the globe. There should probably be some fact-checking algorithms put in place to verify the reliability of news first, to prevent another *Cambridge Analytica*-esque conundrum in the future. Plus, a sarcastic text identifier would just be very useful in general.

Approach and Implementation

My program was written with the Eclipse IDE, coded in Java, with a Hadoop MapReduce cluster, and a Kaggle dataset.

First, the dataset is a JSON format document containing 26,708 news headlines, their links, and if they are sarcastic or not. It was obtained from Kaggle, described as a dataset specifically made for anyone willing to try and make a news headline sarcasm detector (i.e. me).

Dataset Format: `(article_link, headline, is_sarcastic)`

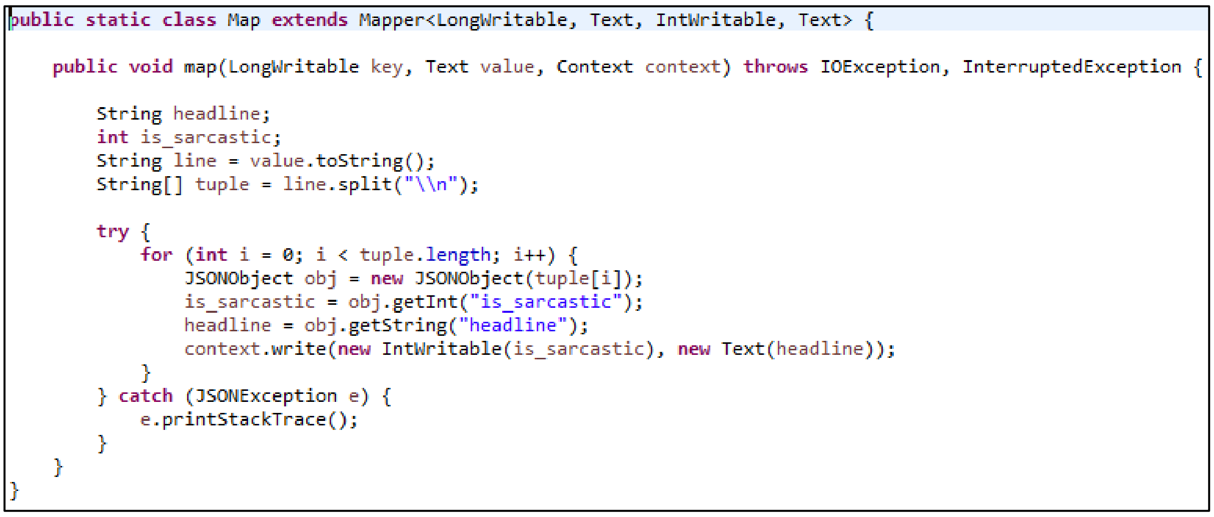


```
{"article_link": "https://local.theonion.com/courtroom-sketch-
artist-has-clear-manga-influences-1820298494", "headline":
"courtroom sketch artist has clear manga influences",
"is_sarcastic": 1}
```

(article_link is not used)

Second, the code is broken into two classes: *HeadineSplitter*, and *Main*. HeadlineSplitter contains the MapReduce function. As the name suggests, it splits the headlines and sorts them based on if they are sarcastic or not. HeadlineSplitter merely generates the data file for Main, and Main is where the actual “sarcasm detection” takes place.

Map Function: `(key Sarcasm_Headlines_Dataset.json, value Text)`

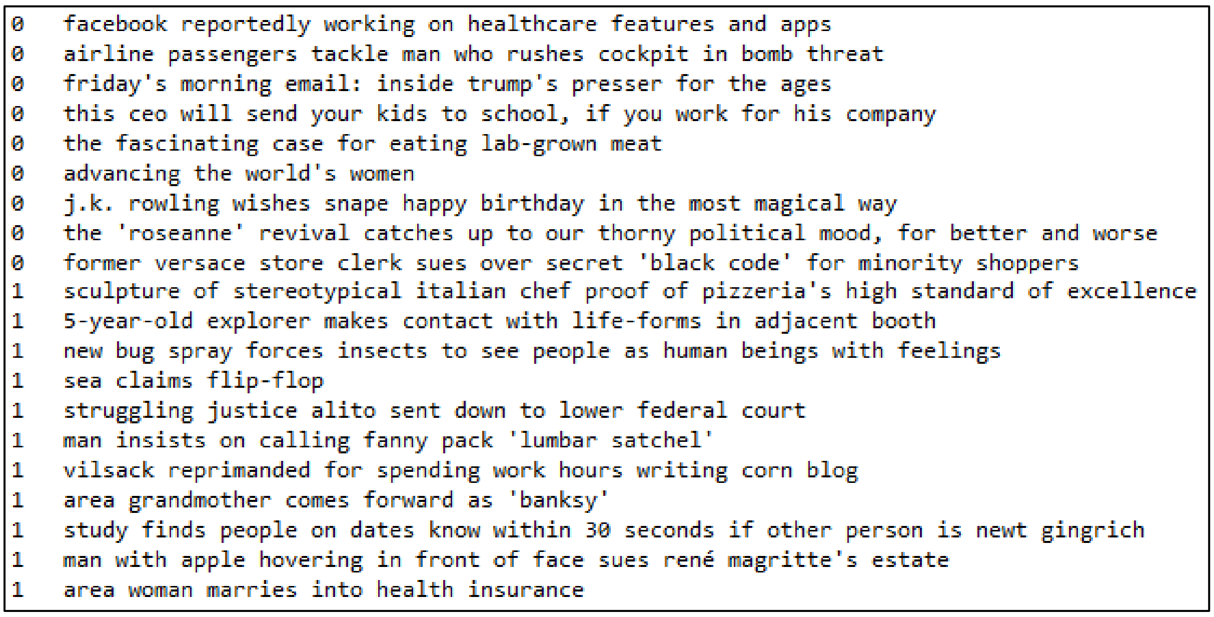


```java
public static class Map extends Mapper<LongWritable, Text, IntWritable, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        String headline;
        int is_sarcastic;
        String line = value.toString();
        String[] tuple = line.split("\\n");

        try {
            for (int i = 0; i < tuple.length; i++) {
                JSONObject obj = new JSONObject(tuple[i]);
                is_sarcastic = obj.getInt("is_sarcastic");
                headline = obj.getString("headline");
                context.write(new IntWritable(is_sarcastic), new Text(headline));
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

The Map function parses the .json file and separates the headlines into two groups: 0 and 1.

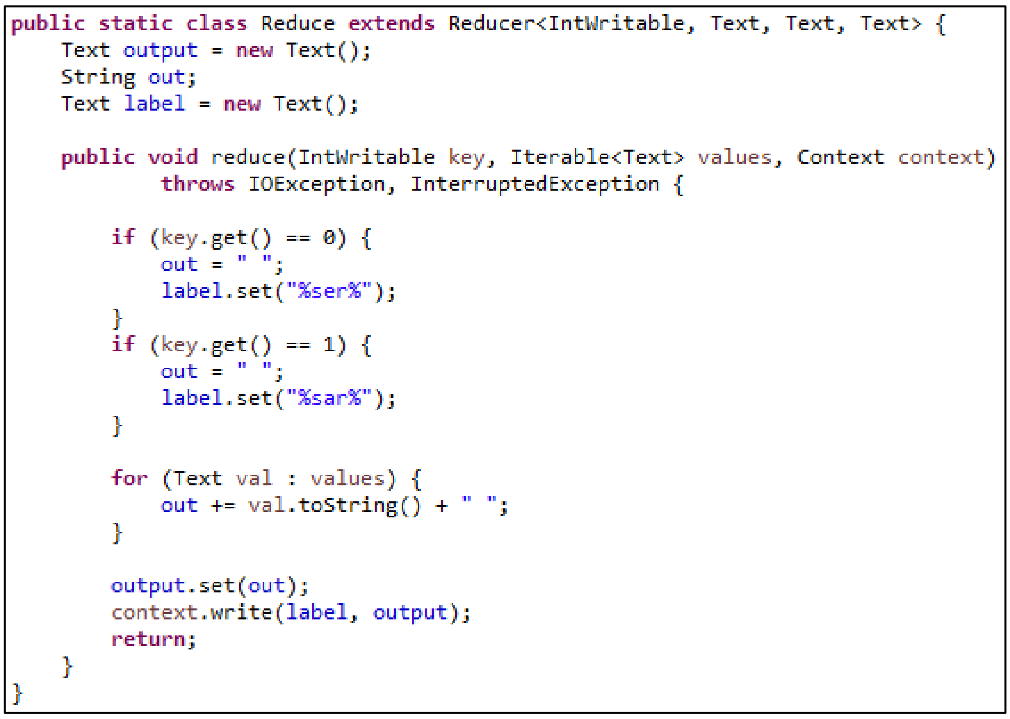
0 denotes serious headlines; 1 denotes sarcastic headlines.

Mapped .json: `(key is_sarcastic, value headline)`



```
0	facebook reportedly working on healthcare features and apps
0	airline passengers tackle man who rushes cockpit in bomb threat
0	friday's morning email: inside trump's presser for the ages
0	this ceo will send your kids to school, if you work for his company
0	the fascinating case for eating lab-grown meat
0	advancing the world's women
0	j.k. rowling wishes snape happy birthday in the most magical way
0	the 'roseanne' revival catches up to our thorny political mood, for better and worse
0	former versace store clerk sues over secret 'black code' for minority shoppers
1	sculpture of stereotypical italian chef proof of pizzeria's high standard of excellence
1	5-year-old explorer makes contact with life-forms in adjacent booth
1	new bug spray forces insects to see people as human beings with feelings
1	sea claims flip-flop
1	struggling justice alito sent down to lower federal court
1	man insists on calling fanny pack 'lumbar satchel'
1	vilsack reprimanded for spending work hours writing corn blog
1	area grandmother comes forward as 'banksy'
1	study finds people on dates know within 30 seconds if other person is newt gingrich
1	man with apple hovering in front of face sues rené magritte's estate
1	area woman marries into health insurance
```

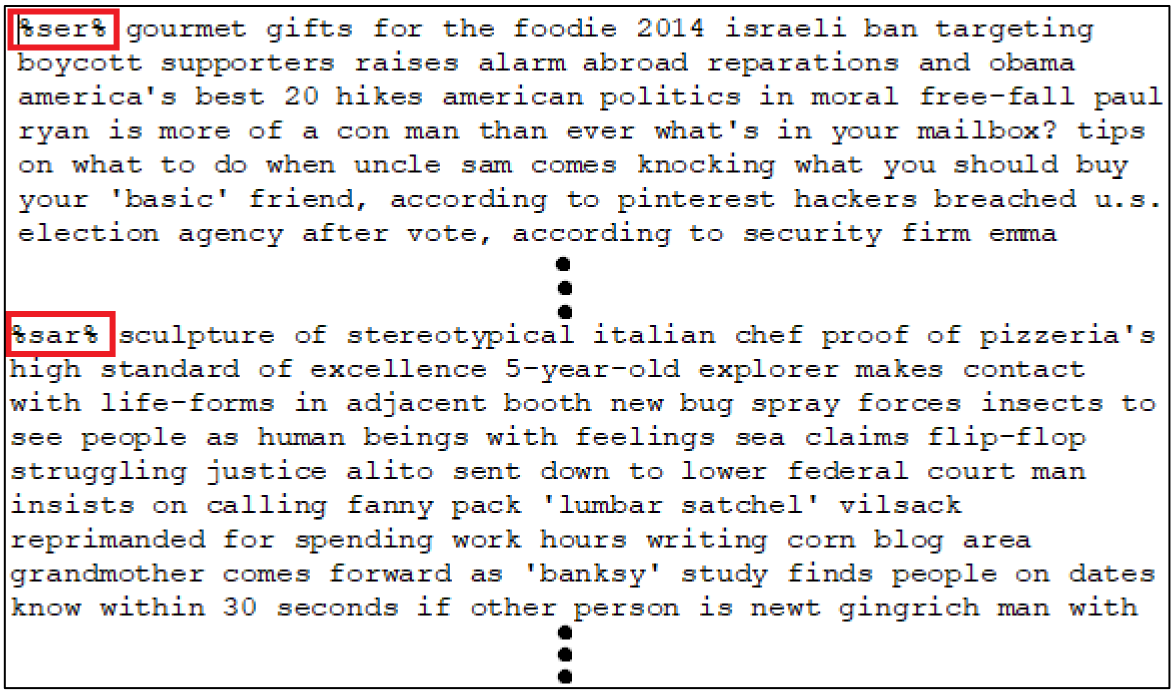
Reduce Function: `(key is_sarcastic, value headline)`



```java
public static class Reduce extends Reducer<IntWritable, Text, Text, Text> {
    Text output = new Text();
    String out;
    Text label = new Text();

    public void reduce(IntWritable key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {

        if (key.get() == 0) {
            out = " ";
            label.set("%ser%");
        }
        if (key.get() == 1) {
            out = " ";
            label.set("%sar%");
        }

        for (Text val : values) {
            out += val.toString() + " ";
        }

        output.set(out);
        context.write(label, output);
        return;
    }
}
```

The Reduce function parses every line in the mapped .json, then concatenates them all onto a single line depending on if they were serious or sarcastic. Each line starts with a unique token so the final project can recognize where the lines begin/end (%ser% = serious; %sar% = sarcastic).

Reduced .json: `(key is_sarcastic, value headlines_concatenated)`



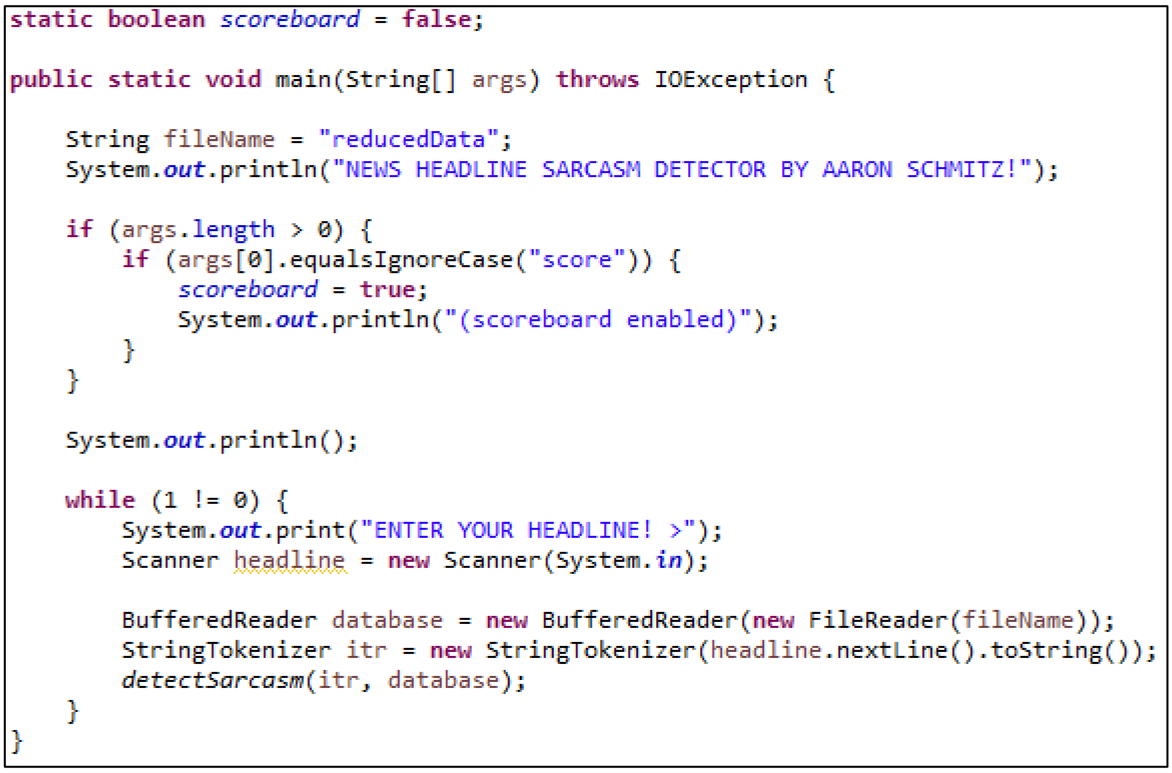
```
%ser% gourmet gifts for the foodie 2014 israeli ban targeting
boycott supporters raises alarm abroad reparations and obama
america's best 20 hikes american politics in moral free-fall paul
ryan is more of a con man than ever what's in your mailbox? tips
on what to do when uncle sam comes knocking what you should buy
your 'basic' friend, according to pinterest hackers breached u.s.
election agency after vote, according to security firm emma
                              ⋮
%sar% sculpture of stereotypical italian chef proof of pizzeria's
high standard of excellence 5-year-old explorer makes contact
with life-forms in adjacent booth new bug spray forces insects to
see people as human beings with feelings sea claims flip-flop
struggling justice alito sent down to lower federal court man
insists on calling fanny pack 'lumbar satchel' vilsack
reprimanded for spending work hours writing corn blog area
grandmother comes forward as 'banksy' study finds people on dates
know within 30 seconds if other person is newt gingrich man with
                              ⋮
```

It doesn't look like it, but the file is only two lines: %ser% and %sar% are where they start. These are only small snippets of the map-reduced json. The original file contains over 26,000 headlines!

By calculating the frequency of the words that appear in each group, the final product will be able to predict whether an inputted news headline is likely or unlikely to be sarcastic (ideally with a reasonable degree of accuracy).

Third, the Main code is where the sarcasm detection happens.

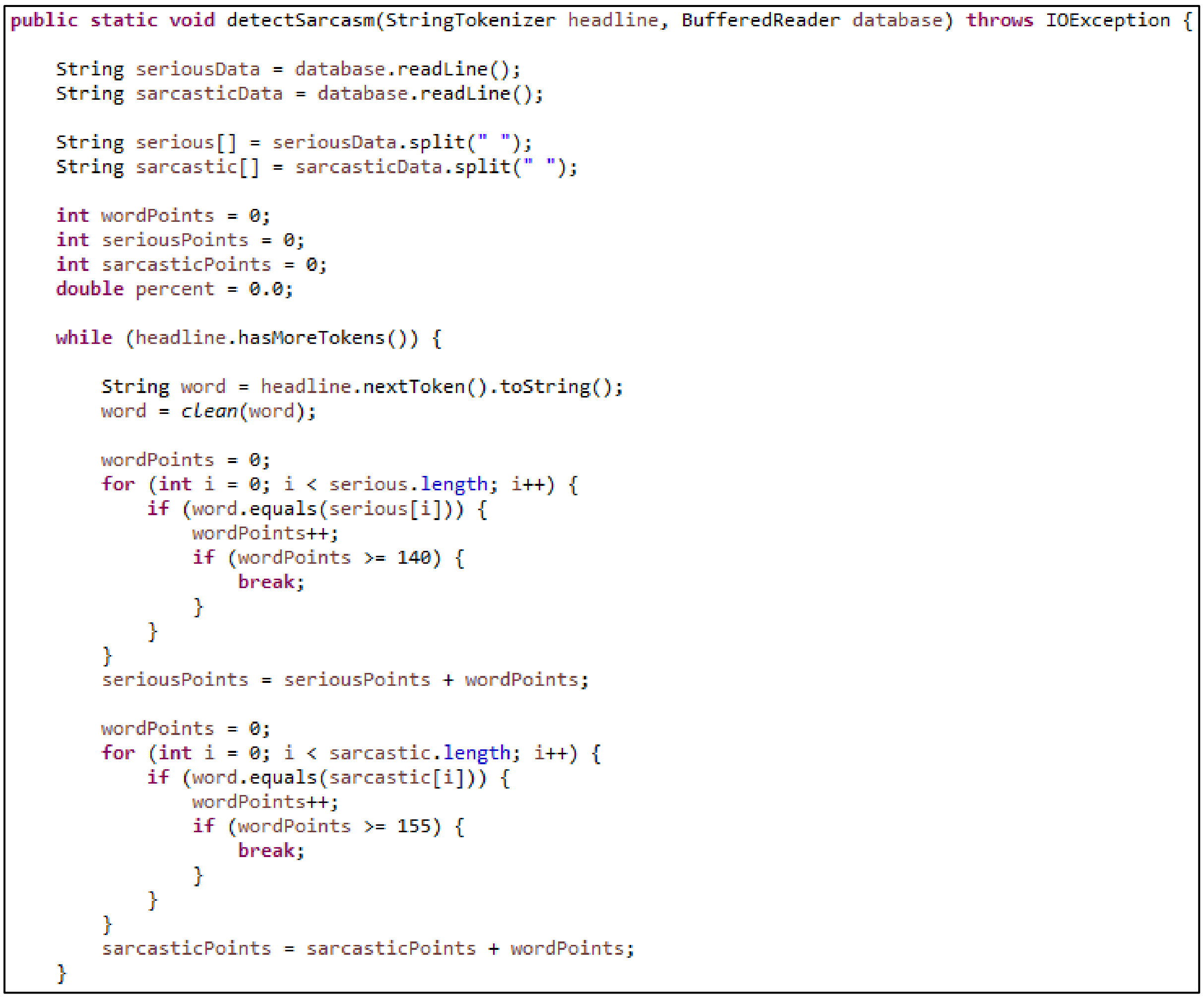
Main



```
static boolean scoreboard = false;

public static void main(String[] args) throws IOException {

    String fileName = "reducedData";
    System.out.println("NEWS HEADLINE SARCASM DETECTOR BY AARON SCHMITZ!");

    if (args.length > 0) {
        if (args[0].equalsIgnoreCase("score")) {
            scoreboard = true;
            System.out.println("(scoreboard enabled)");
        }
    }

    System.out.println();

    while (1 != 0) {
        System.out.print("ENTER YOUR HEADLINE! >");
        Scanner headline = new Scanner(System.in);

        BufferedReader database = new BufferedReader(new FileReader(fileName));
        StringTokenizer itr = new StringTokenizer(headline.nextLine().toString());
        detectSarcasm(itr, database);
    }
}
```

The Main block opens the map-reduced file and asks for the user input. Once entered, the `detectSarcasm` method is called. This loop will continue forever, allowing for multiple runs. There is also an optional scoreboard which'll display the total results at the end (off by default; can be enabled by typing "score" when running the program).

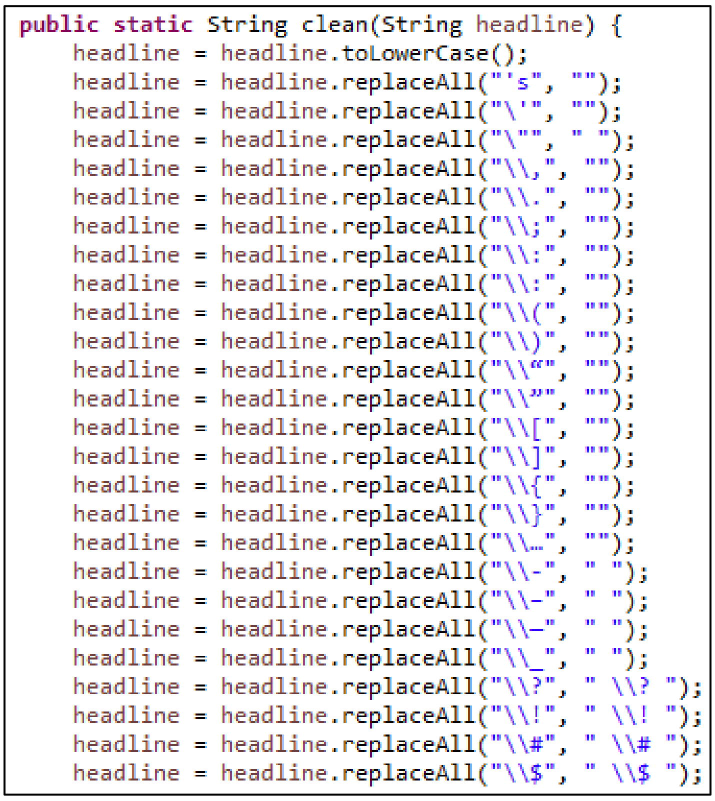
`detectSarcasm(headline, database)`



```java
public static void detectSarcasm(StringTokenizer headline, BufferedReader database) throws IOException {

    String seriousData = database.readLine();
    String sarcasticData = database.readLine();

    String serious[] = seriousData.split(" ");
    String sarcastic[] = sarcasticData.split(" ");

    int wordPoints = 0;
    int seriousPoints = 0;
    int sarcasticPoints = 0;
    double percent = 0.0;

    while (headline.hasMoreTokens()) {

        String word = headline.nextToken().toString();
        word = clean(word);

        wordPoints = 0;
        for (int i = 0; i < serious.length; i++) {
            if (word.equals(serious[i])) {
                wordPoints++;
                if (wordPoints >= 140) {
                    break;
                }
            }
        }
        seriousPoints = seriousPoints + wordPoints;

        wordPoints = 0;
        for (int i = 0; i < sarcastic.length; i++) {
            if (word.equals(sarcastic[i])) {
                wordPoints++;
                if (wordPoints >= 155) {
                    break;
                }
            }
        }
        sarcasticPoints = sarcasticPoints + wordPoints;
    }
```

The `detectSarcasm()` function parses the headline for each word, then counts up every occurrence of the word in the data file. Each time the parsed word(s) appears in the serious group, serious points go up by one, and each time the parsed word(s) appears in the sarcastic category, sarcastic points go up by one. **Note:** there's a limit on how many points each word can give. Since my algorithm relies on grouping, it's highly susceptible to outliers. In this case, any words that appear vastly more times in one category than the other (e.g. a word appears a thousand times in the serious group, but zero times in the sarcastic group), such a case would highly skew the results. **Also note:** each word passes through a `clean(word)` method. This method filters out characters and common words.

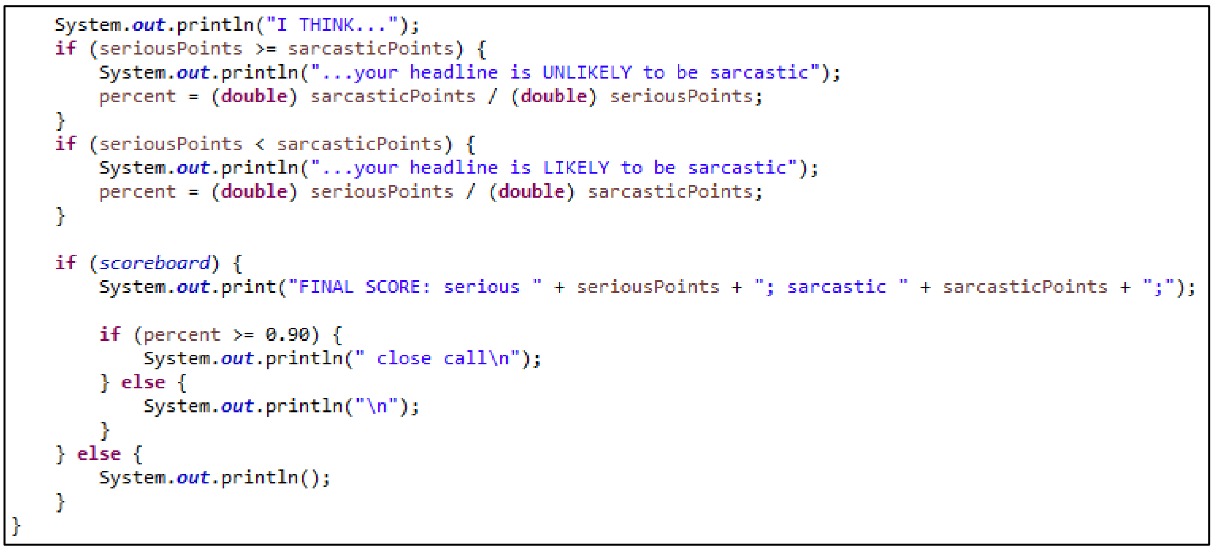
clean(word)



```java
public static String clean(String headline) {
    headline = headline.toLowerCase();
    headline = headline.replaceAll("'s", "");
    headline = headline.replaceAll("\'", "");
    headline = headline.replaceAll("\"", " ");
    headline = headline.replaceAll("\\,", "");
    headline = headline.replaceAll("\\.", "");
    headline = headline.replaceAll("\\;", "");
    headline = headline.replaceAll("\\:", "");
    headline = headline.replaceAll("\\:", "");
    headline = headline.replaceAll("\\(", "");
    headline = headline.replaceAll("\\)", "");
    headline = headline.replaceAll("\\“", "");
    headline = headline.replaceAll("\\”", "");
    headline = headline.replaceAll("\\[", "");
    headline = headline.replaceAll("\\]", "");
    headline = headline.replaceAll("\\{", "");
    headline = headline.replaceAll("\\}", "");
    headline = headline.replaceAll("\\…", "");
    headline = headline.replaceAll("\\-", " ");
    headline = headline.replaceAll("\\–", " ");
    headline = headline.replaceAll("\\—", " ");
    headline = headline.replaceAll("\\_", " ");
    headline = headline.replaceAll("\\?", " \\? ");
    headline = headline.replaceAll("\\!", " \\! ");
    headline = headline.replaceAll("\\#", " \\# ");
    headline = headline.replaceAll("\\$", " \\$ ");
```

Special characters like question marks and exclamation points aren't removed, but rather separated so the program reads them as separate words. I figured those symbols were important for understanding the context of the headline, so the algorithm would be more accurate. Single quotes are removed, for "word's" and "word" are the same. Also, all words are downshifted to lowercase.

detectSarcasm(headline, database) **PART 2**



```java
    System.out.println("I THINK...");
    if (seriousPoints >= sarcasticPoints) {
        System.out.println("...your headline is UNLIKELY to be sarcastic");
        percent = (double) sarcasticPoints / (double) seriousPoints;
    }
    if (seriousPoints < sarcasticPoints) {
        System.out.println("...your headline is LIKELY to be sarcastic");
        percent = (double) seriousPoints / (double) sarcasticPoints;
    }

    if (scoreboard) {
        System.out.print("FINAL SCORE: serious " + seriousPoints + "; sarcastic " + sarcasticPoints + ";");

        if (percent >= 0.90) {
            System.out.println(" close call\n");
        } else {
            System.out.println("\n");
        }
    } else {
        System.out.println();
    }
}
```

Once the program is complete, it'll make its prediction based on how many serious/sarcastic points were counted. If more serious points were counted, the program will print the message "…your headline is UNLIKELY to be sarcastic", and the reverse if more sarcastic points were counted. If the scoreboard is enabled, the program will also print out the total number of points.

Experiment Results and Discussion

```
ENTER YOUR HEADLINE! >Chase Tells Customers to Stop Splurging on Coffee, Draws Criticism From Officials
I THINK...
...your headline is UNLIKELY to be sarcastic
```

```
ENTER YOUR HEADLINE! >This Week, NASA Is Pretending An Asteroid Is On Its Way To Smack The Earth
I THINK...
...your headline is UNLIKELY to be sarcastic
```

```
ENTER YOUR HEADLINE! >New Jersey 'pooperintendent' who defecated on another high school's field sues police over mug shot release
I THINK...
...your headline is UNLIKELY to be sarcastic
```

```
ENTER YOUR HEADLINE! >Report: Unemployment High Because People Keep Blowing Their Job Interviews
I THINK...
...your headline is LIKELY to be sarcastic
```

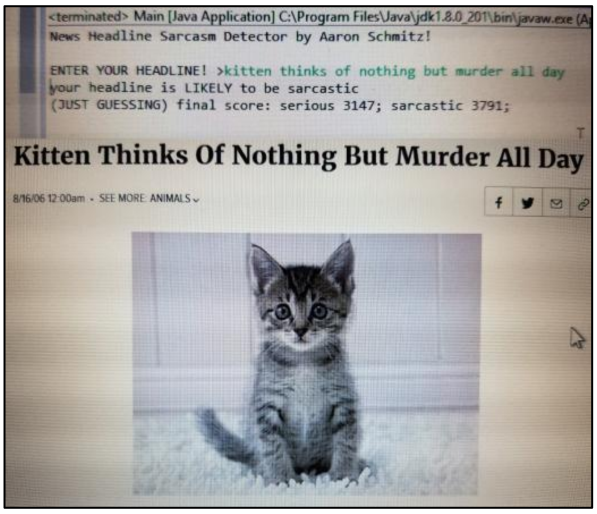
Despite being just a simple word counting algorithm, it's surprisingly accurate. It correctly predicts the four examples from the introduction, despite their ridiculousness at face value. I ran an accuracy test for the program multiple times, using every headline from the original dataset, and counting the amount correct predictions. At maximum, I was able to get about an 80% accuracy rate:

```
FINAL RESULTS: 21251 / 26708
```

(21,251 correct predictions out of 26,708)

Some more examples:

These two pulled off *The Onion*

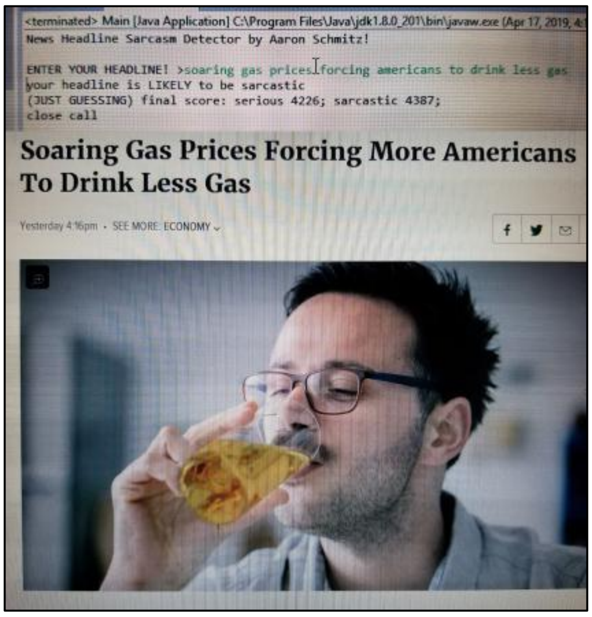


```
<terminated> Main [Java Application] C:\Program Files\Java\jdk1.8.0_201\bin\javaw.exe (A
News Headline Sarcasm Detector by Aaron Schmitz!

ENTER YOUR HEADLINE! >kitten thinks of nothing but murder all day
your headline is LIKELY to be sarcastic
(JUST GUESSING) final score: serious 3147; sarcastic 3791;
```

Kitten Thinks Of Nothing But Murder All Day

8/16/06 12:00am • SEE MORE: ANIMALS

(A headline from my project proposal)

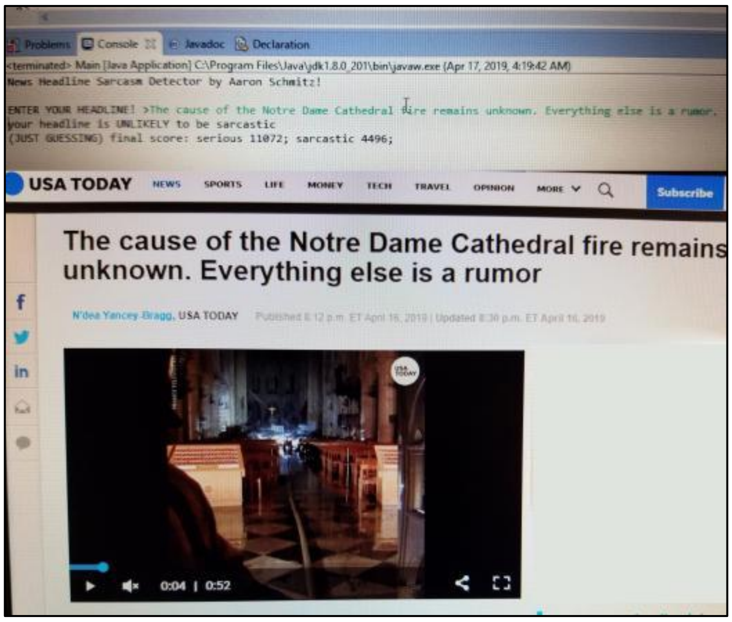


```
<terminated> Main [Java Application] C:\Program Files\Java\jdk1.8.0_201\bin\javaw.exe (Apr 17, 2019, 4:
News Headline Sarcasm Detector by Aaron Schmitz!

ENTER YOUR HEADLINE! >soaring gas prices forcing americans to drink less gas
your headline is LIKELY to be sarcastic
(JUST GUESSING) final score: serious 4226; sarcastic 4387;
close call
```

Soaring Gas Prices Forcing More Americans To Drink Less Gas

Yesterday 4:16pm • SEE MORE: ECONOMY

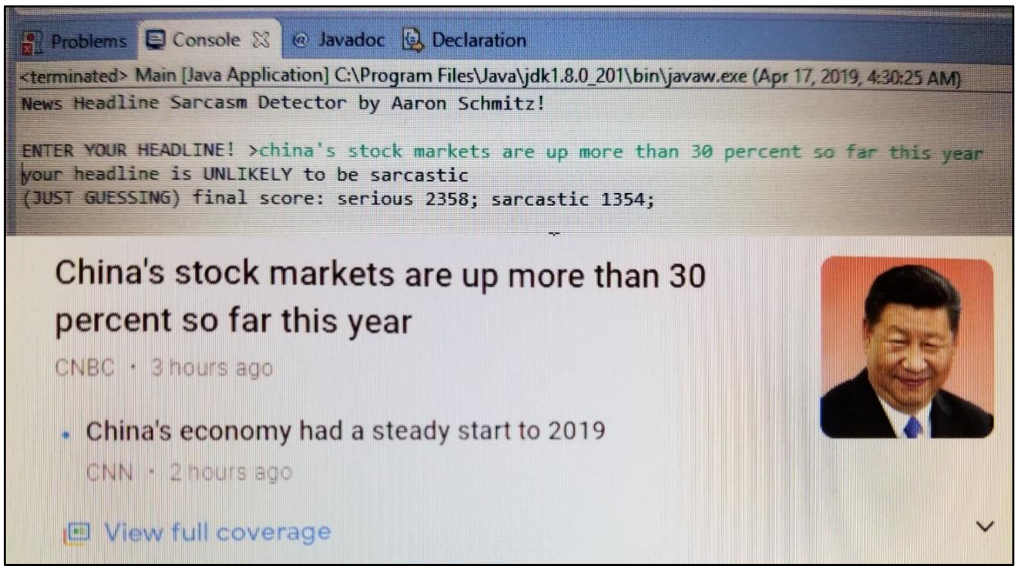
Three headlines pulled off Google News



```
<terminated> Main [Java Application] C:\Program Files\Java\jdk1.8.0_201\bin\javaw.exe (Apr 17, 2019, 4:19:42 AM)
News Headline Sarcasm Detector by Aaron Schmitz!

ENTER YOUR HEADLINE! >The cause of the Notre Dame Cathedral fire remains unknown. Everything else is a rumor.
your headline is UNLIKELY to be sarcastic
(JUST GUESSING) final score: serious 11072; sarcastic 4496;
```

The cause of the Notre Dame Cathedral fire remains unknown. Everything else is a rumor

N'dea Yancey-Bragg, USA TODAY Published 6:12 p.m. ET April 16, 2019 | Updated 8:30 p.m. ET April 16, 2019



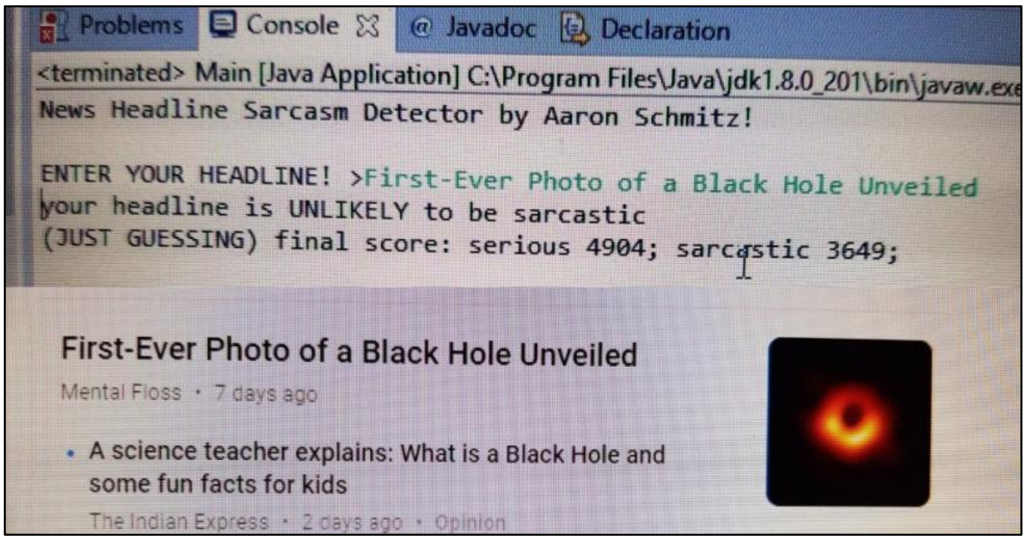
```
<terminated> Main [Java Application] C:\Program Files\Java\jdk1.8.0_201\bin\javaw.exe (Apr 17, 2019, 4:30:25 AM)
News Headline Sarcasm Detector by Aaron Schmitz!

ENTER YOUR HEADLINE! >china's stock markets are up more than 30 percent so far this year
your headline is UNLIKELY to be sarcastic
(JUST GUESSING) final score: serious 2358; sarcastic 1354;
```

China's stock markets are up more than 30 percent so far this year

CNBC · 3 hours ago

- China's economy had a steady start to 2019

  CNN · 2 hours ago



```
<terminated> Main [Java Application] C:\Program Files\Java\jdk1.8.0_201\bin\javaw.exe
News Headline Sarcasm Detector by Aaron Schmitz!

ENTER YOUR HEADLINE! >First-Ever Photo of a Black Hole Unveiled
your headline is UNLIKELY to be sarcastic
(JUST GUESSING) final score: serious 4904; sarcastic 3649;
```

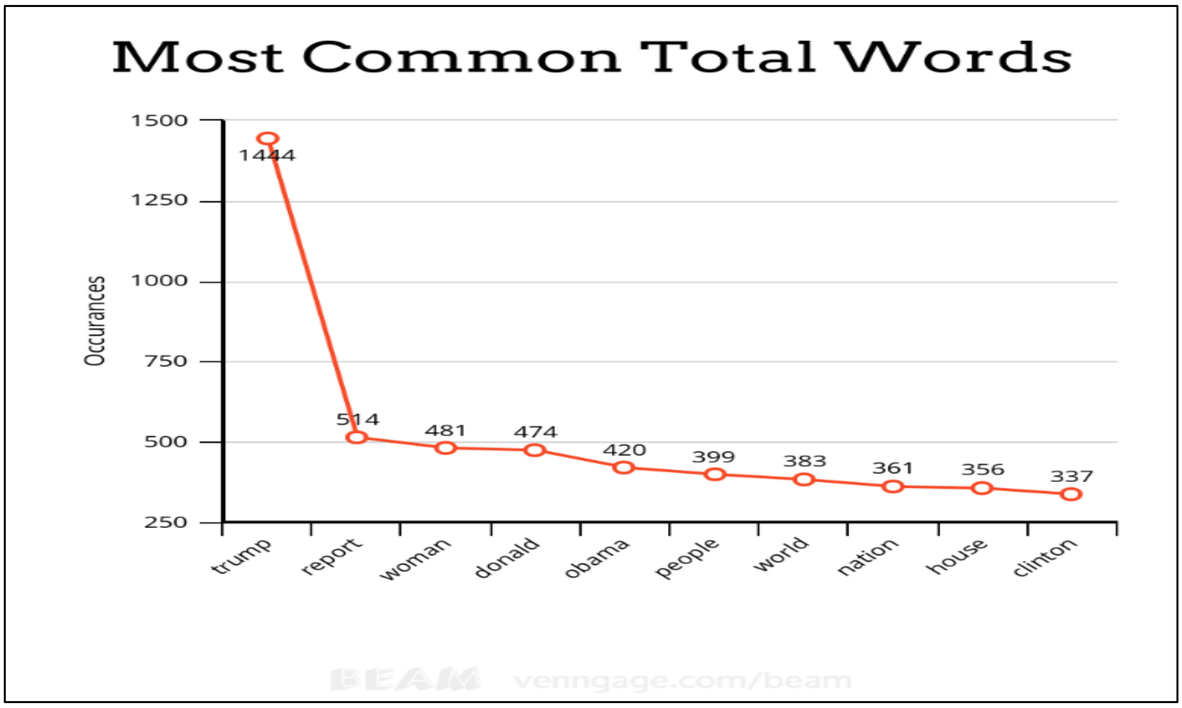
First-Ever Photo of a Black Hole Unveiled

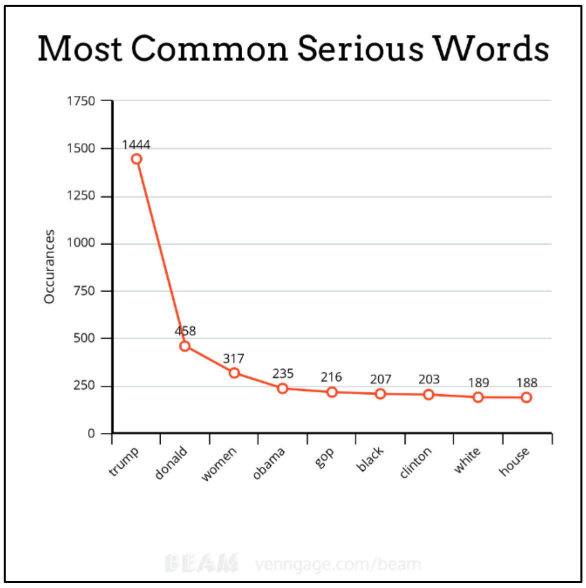
Mental Floss · 7 days ago

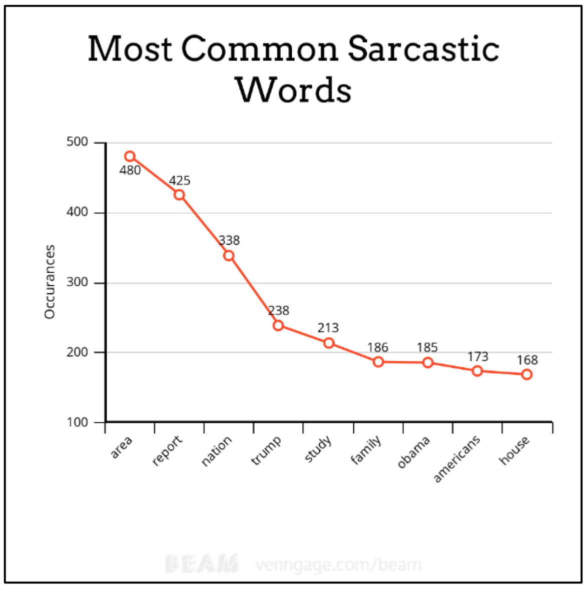
- A science teacher explains: What is a Black Hole and some fun facts for kids

  The Indian Express · 2 days ago · Opinion

Some charts (numbers gathered via word count):







“Trump” is the biggest outlier in the dataset, and the entire reason for putting a limit on how many points each word could give. Any headline with “Trump” in it was guaranteed to always have (much) more serious points, hence the need for a point limit. Through testing, I found the point limit of 140 for both categories yielded the most accurate results. I would’ve tested more, but it takes a long time to check all 26,708 headlines. Too much data and not enough computing power *was* a recurring theme throughout the semester, after all.

Conclusion

I was very surprised with the accuracy of my project. Initially, I was quite skeptical that merely counting the frequency of words in a headline could accurately predict the tone of an entire article, but I counted on the sheer size of the dataset to deliver accurate results. I never expected to get anywhere near an 80% accuracy rate from such a simple program. I would've been happy with just a 51% accuracy rate.

With some machine learning, and/or help from a professional linguist, a much better program with a much higher accuracy rate could definitely be made. Perhaps one could find a use for the article link as well. Ideally one would be able to make a sarcasm detector that gets it right *every, single,* time. And then there would never be any more debate about the credibility of modern-day journalism!

Overall, this was a fun project, and I'm happy with the results. This course made me *very* interested in data science, and just how far it can go. Data is the new oil.

References

Zipped Folder contains the dataset, map-reduced files, the code for `HeadlineSplitter`, `Main`, and `MainAccuracyTest` (for testing accuracy – takes a long time to do). The folder also contains lists of counted words used to make the graphs. Lastly, the folder also contains `sd.jar`, a runnable .jar file for running the program (it's user interactive!)

Sources

Beam (for making graphs): https://venngage.com/blog/beam/

Cambridge Analytica: https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html

Common Words: https://en.wikipedia.org/wiki/Most_common_words_in_English

Dataset: https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection

Hadoop MapReduce: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

Google News: https://news.google.com/?hl=en-US&gl=US&ceid=US:en

The Onion: https://www.theonion.com/