Identifying Clickbait: Multi-Layer Regular Expression Filter

University of Minnesota
Jack Bartels
LING 5801
30 June 2019

Executive Summary

While the definition of clickbait is hard to pin down, there is at least a general agreement that it is content designed to make you click on it. As the internet grows through constant uploading of new content, the ability for computers to be able to identify clickbait is becoming increasingly relevant.

Because of this, I created a program in python that uses regular expressions to attempt to identify clickbait titles. The input data is pulled from YouTube's Trending feed, which is a collection of its most popular videos. The titles are then ran through a three-layer filter. The layers filter based on keywords, phrases, and lastly formats. YouTube seemed like a good platform choice due to the variety of cultural influences and the frequent new content, which is ideal for maximizing testing data.

This project is a perfect opportunity to study a few key areas of linguistics. The primary areas pertaining to clickbait are socio-linguistics, lexical variety, and syntax. The intersection of culture and linguistics makes this a particularly interesting and difficult task.

The goal of the program is to identify as many clickbait titles as possible while minimizing false positives. In other words, it is important for this program (if it is to be at all useful) to have a high recall and precision, which are then combined into an overall F1 score.

When testing, I found that formats were by far the most productive filter with a good precision of 80%. The keywords filter had a low recall and lowest precision due to its proneness to overcapture. The phrases filter had the lowest recall but it was still very useful in that it had perfect precision, raising the F1 score. Overall, the program was decently precise, but definitely fell short in terms of recall.

There are a number of reasons why the program may have been unable to identify many of the clickbait titles, but most likely it comes down to two main factors: (1) there is a lot of subjectivity involved in the categorization of clickbait or not; and (2) the program was only exposed to a few hundreds titles of input data, which could be way fewer than is necessary to arrive at a more accurate result.

Overall, it seems that this project would be better suited to using machine learning and large input datasets in order to achieve better results. Indeed many researchers and computational linguists are doing just that.

Introduction

Clickbait; click here to find out what it is! We hear this term on a regular basis, typically when it comes to browsing the internet. Simply put: clickbait is any content that is formatted in a way that is primarily meant to draw the attention of the user. This phenomenon has become more and more common as an increasing number of platforms offer the ability to monetize content through advertising, i.e. clicks. A lot of content creators avoid utilizing clickbait because of the stigma around exploiting their audiences. Others, though, are shameless in their boisterous, cliff-hanging titles and colorful, coming-right-at-you video previews.

While clickbait, on its surface, seems to be a fairly harmless and unavoidable byproduct of the present, view-driven state of the internet, there are a number of reasons the capability to identify it can be really useful. One example is from the perspective of a business or marketing organization that wants to determine how effective clickbait is by comparing viewership of clickbait content versus non-clickbait content. Another example is a user's ability to filter out such content if they so choose. Whether it perpetuates it or stifles it, effective clickbait identification would change the current landscape of the internet.

The Program

As a user of the internet myself, I am frequently bothered by the amount of clickbait content that I have to scroll past. This seemed like the perfect opportunity to explore how feasible it would be to use computational linguistics to capture patterns typical in clickbait titles with decent accuracy. My program is written in Python and it strips titles from YouTube videos. The HTML scraping to strip the titles was achieved by using the lxml and requests libraries. YouTube button titles and URLs were filtered out of the results manually because regular expressions had difficulties isolating the two different kinds of titles solely from HTML tags.

The program has a three-layered filter. If any of the titles to match any of the filter items, the title is flagged as clickbait. The three components of the identifier are keywords, phrases, and what I have generally labeled as 'formats', which are other patterns captured through the use of regular expressions. The filter items can be freely added and removed until testing performance is deemed adequate. Then, the program compares its performance to the human judgement data provided. Typically, a gold standard dataset is generated and then split into a testing and evaluation subset. For this program, I decided to instead opt for generating testing data by stripping titles directly from YouTube's Trending feed and then refining my filters manually, until satisfied with their performance. This has a principle drawback: it introduces more subjectivity. However, I still decided to adopt this method because of its massive advantage: the ability to test live, dynamic data at any time over the course of the project.

Being exposed to so much more testing data, I believe that the performance of the filters will naturally have a greater ability to generalize effectively beyond testing data. For creating the evaluation dataset, I reserved a day each week in May to not draw testing data from (or around to prevent overlap), but to instead collect into a list for other people to categorize, given only the definition of clickbait provided above, and then aggregated those responses.

The Platform

YouTube is the internet's single largest and most popular user-created video platform. It has content of all varieties and provides, I think, a good representation of the internet as a whole. I chose to build my program using this platform because of the diversity of topics, the frequency of updates to its trending feed that displays popular videos, and the ease of dynamically scraping and stripping its HTML using Python and its libraries. YouTube also provides a challenge when it comes to regular expressions due to the fact that individuals decide what the titles are. The result is that titles have many different capitalization mixes, punctuation formats, and even typos.

There was always the option to pull titles from multiple platforms, but I thought it would serve an additional purpose to focus on only one. That way, it can be tested if the program that I created using YouTube input data is able to generalize well to other platforms. Whether it can or not, it reveals something about how we use language the same or differently depending on which platform we are publishing to. Another interesting aspect of YouTube is that clickbait often has to do with the preview image provided for videos. When given the titles, respondents were not given the accompanying images, evaluating the titles based on the text alone. A possible future project could be to test the effectiveness of the clickbait title identifier, in the context of the image and title presented as a whole.

Linguistics

This project relates to linguistics in a number of ways. Primarily, it deals with socio-linguistics, lexical variations, and syntax. Socio-linguistics deals with how linguistics and socio-cultural factors interact. Many of the titles encountered on YouTube trending have to do with music, food, sports, gaming, events, and other cultural topics. As a result, successfully identifying the ways in which these cultural factors affect whether or not a title is considered clickbait needs to be examined and included in the filters. Even within the same language, there are numerous lexical variations depending on location, age, education, personality, etc. The program needs to be able to capture these variations otherwise it will suffer in its recall abilities. Additionally, this project is heavily influenced by syntax.

A large part of this class has been focused on ambiguity, regular expressions, and thinking beyond the word level when it comes to computational analysis of language. It is for this exact reason that I decided to make my filter multi-layered. There are some keywords that exist almost exclusively in clickbait titles, likewise with phrases. But targeting specific words and groups of words in this way is very prone to leading to false positives, over-capturing. Because of this, the third layer is focused solely on common formats used in clickbait, which are captured using regular expressions. As a result, the program takes into consideration each of the primary linguistic factors at play.

Approaching the Problem

In order to begin approaching the problem of accurately identifying clickbait, the manner of evaluation needs to be considered. The program will be given an F1 score based on its recall, the ability to identify clickbait titles, and precision, the proportion of identified titles that were indeed clickbait. With this in mind, the goal is to select filter items that successfully capture the majority of clickbait titles without being so broad that they misidentify non-clickbait titles. The importance of accuracy is relevant in the context of the application. For instance, if the identifier is being used to filter out clickbait content that is displayed to the user, the accuracy needs to be very high, especially the precision moreso than recall. The reason is that it would be unfair to content creators who don’t use clickbait to get captured by the filter by mistake. For our purposes, an F1 performance above 80% would be sufficient as a proof of concept.

Here are the final filters that were utilized:

Keywords	**Phrases**	**Formats**
biggest	here’s why	‘^Guess .*’
largest	you never knew	‘^[0-9]+ (?!in).*’
mind-blowing	won’t believe	‘^(?<!I)[A-Z]+ [(A-Z|0-9)]+ .*’
worst	weird trick	‘.***.*’
amazing	see this	‘.*\?!+.*’
never	watch this	‘.*!\?+.*’
click	find out	‘.*\?{3,}.*’
woah	life changing	‘.*\!{3,}.*’
wow	need to see	‘.*[A-Z]+ [A-Z]+ [A-Z]+.*’
surprising	will make you	
omg	this is what happens	
crazy	oh my god	
	this is nuts	

The filter items themselves give some linguistic insights. As can be seen, most of the keywords are exclamatory and superlative. When it comes to phrases, they tend to be directed at the user, with a common theme of expressing that they ought to click in order to gain some previously unknown information. Another theme common in the phrase filter items is the imperative. There are many phrases that plainly instruct the user to click the video. The format filter items clearly had three major themes: Numbers, punctuation, and capital letters. Because the format filters are the most productive in terms of recall, that means that those themes are therefore the primary tools used by people when crafting clickbait titles.

Results

The gold standard list was never consulted in order for evaluation to reflect the program’s ability to generalize to other data. In order to get a clear picture of the extent to which each filter layer contributes to the overall performance of the identifier, results are split up into four categories: keywords filter only, phrases filter only, formats filter only, and all filters combined.

Each result is broken down into the recall, precision, and F1 score for each test, which are displayed in a percentage form rounded to the nearest tenth of a percent:

Keywords Only:

Recall: 10.6%

Precision: 69.2%

F1 Score: 18.4%

Phrases Only:

Recall: 5.9%

Precision: 100.0%

F1 Score: 11.1%

Formats Only:

Recall: 42.4%

Precision: 80.0%

F1 Score: 55.4%

Combined:

Recall: 52.9%

Precision: 77.6%

F1 Score: 62.9%

What It Reveals

What the results reveal about human use of language when it comes to clickbait is that we tend to utilize specific styles of formatting rather than particular words or phrases in order to create clickbait titles. While the program fell short in identifying all of the clickbait titles, it had decent precision. Most revealing was the fact that the formats filter captured almost all of the clickbait titles identified, with a relatively high precision. The phrases filter had the lowest recall, in other words it was the least productive in helping to find clickbait titles, but it did so with perfect precision. These results show that the primary focuses for filtering should be on phrases, for its precision, and on formats, for its recall. Given that the filters were refined after going through hundreds of input titles and making adjustments and it still had low overall performance when tested on the gold standard means that either my judgement differed greatly from that of the respondents or that there are just so many potential keywords, phrases, and formats for clickbait that a lot more input data is needed—or both. This is where the limitations of subjectivity and lack of machine learning becoming most apparent.

Limitations

Right away, studying clickbait has a large limiting factor: its definition. While I came up with a standard definition to provide to respondents, what actually constitutes clickbait is not so clear cut and varies from person to person. In other words, there is a lot of subjectivity involved which limits the effectiveness of using a single standard filter. As mentioned before, using my own subjectivity in place of a gold standard subset for testing is also an opportunity for more divergence in the agreed upon categorization of clickbait.

On a similar note, the assumption that a title is either clickbait or not could itself be flawed. One could argue that the determination of whether or not something fits the criteria of clickbait is not a black and white issue but rather a sliding scale. For titles that are in a gray area, respondents may be forced to make a rather random decision. In other words, there is ambiguity involved. Such titles could decrease the effectiveness of the identifier by introducing overly general filter items or by not containing any attributes characteristic of typical clickbait.

Another limitation is the need to manually evaluate test data and adjust filters. Not only does this mean that I need to make decisions about each title, I also need to determine from small datasets what tends to be present in clickbait that isn't present in other titles. There is no way for the program to dynamically improve, only manually.

Potential Improvements

Given the time, technological, and knowledge constraints of this project, there are quite a few ways that it could be improved, including even entirely different approaches. Some improvements include sliding scale data, implementing a means of testing frequency and particular filter items individually, and using deep learning.

Sliding scale data rather than boolean decisions about whether a title is clickbait or not. Having that additional data could assist in choosing filter items that better balance recall and precision. It would be useful to be able to compare the frequency of keywords, phrases, and formats in clickbait vs non-clickbait titles. Items that are common in clickbait titles and uncommon in non-clickbait titles could easily be selected as new filter items. With a large enough processed dataset, individual filter items could be evaluated for their recall and precision to determine whether or not they are worth implementing. Finally, the most natural next step for a project like this is to have a program that can analyze large quantities of data and self improve.

Deep learning could do a much better job of creating a clickbait identifier than a human doing it manually, it would only be a matter of how much human processed data it could be fed.

In order to implement these improvements, more resources would be required; namely: time, expertise, input data, and computational power. However, even the current implementation could likely be further improved with more manual adjustment after testing on more data.

Related Projects

In one similar project, Lopez-Sanchez et al. identified one of the primary challenges of making a clickbait identifier as, "the problem of adaptibility. Due to the subjective nature of clickbaits, a single headline may be perceived differently by users of different interests or criteria" (2968). In order to overcome this challenge, they suggest using "Case-Based Reasoning". The basic idea behind this method is to identify and utilize the uniqueness of each user. Any data gathered about the user can be used to predict what they may or may not consider to be clickbait. They achieve this by using frequency analysis and neural networks.

Another related project is by Phillipe Thomas. Thomas used a fairly typical separation of testing and evaluation corpora. Each set was labeled as clickbait or not and then that data was fed into various kinds of neural networks. What is interesting about his work is that he also chose to explore the effect of incorporating the accompanying image. "As the annotation process was supported by the image informa-tion, we assume that the teaser images might be helpful to predictthe clickbait relevance of a given message" (Thomas 2). This prediction is in line with my hypothesis stated earlier in the 'Potential Improvements' section. However, Thomas was unable to get any helpful performance out of this addition, at least using his implementation.

Conclusion

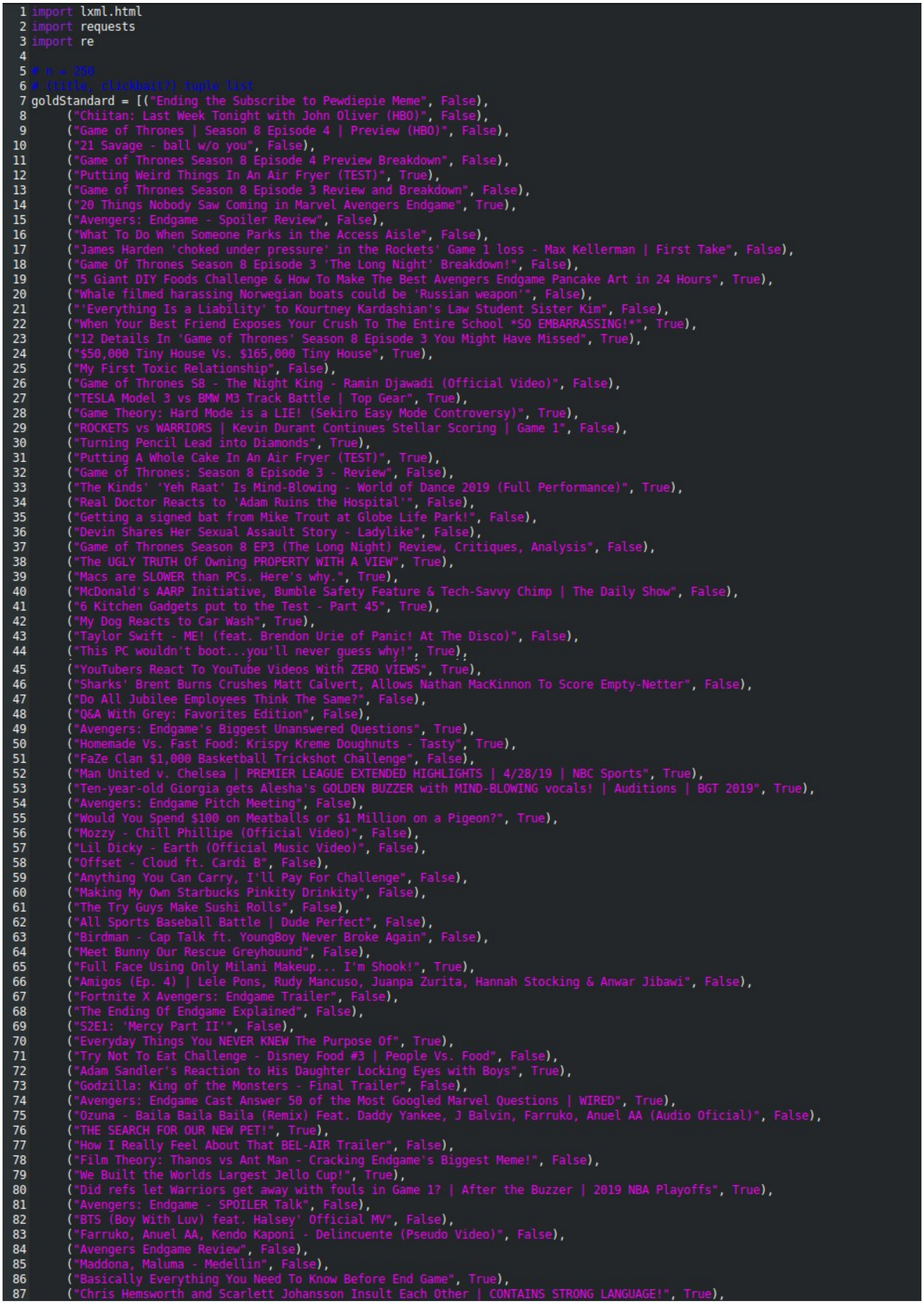
Identifying patterns in human language use certainly seems to be a task better suited to modern techniques that can utilize artificial intelligence and hefty computing power. However, this project still revealed a lot about what clickbait consists of linguistically. Primarily, it showed that clickbait is mostly a matter of formatting, reinforcing the semester-long message that resolving ambiguity and identifying patterns in real world language use requires looking beyond the word and phrase level and instead considering how we can analyze how all of the components of language interact as various scales. Even though I find the results to be disappointing in terms of performance, there does seem to be a lot of room to improve this program, even in its current implementation. There are a lot of other related projects and research being conducted out there on this and similar topics. Most of that work is being done using neural networks that analyze the frequency of patterns with massive input datasets. Even though there are a lot of challenges when it comes to something as subjective as whether or not a title is trying to exploit your attention, there does seem to be at least a certain extent to which computing can predict—or at least model—human language use in a beneficial way. Being able to identify clickbait could prove to be an extremely valuable ability that could lead to many tools and applications to be used by individuals, companies, and governments. I am excited to see the future progress that is made in this field.

Works Cited

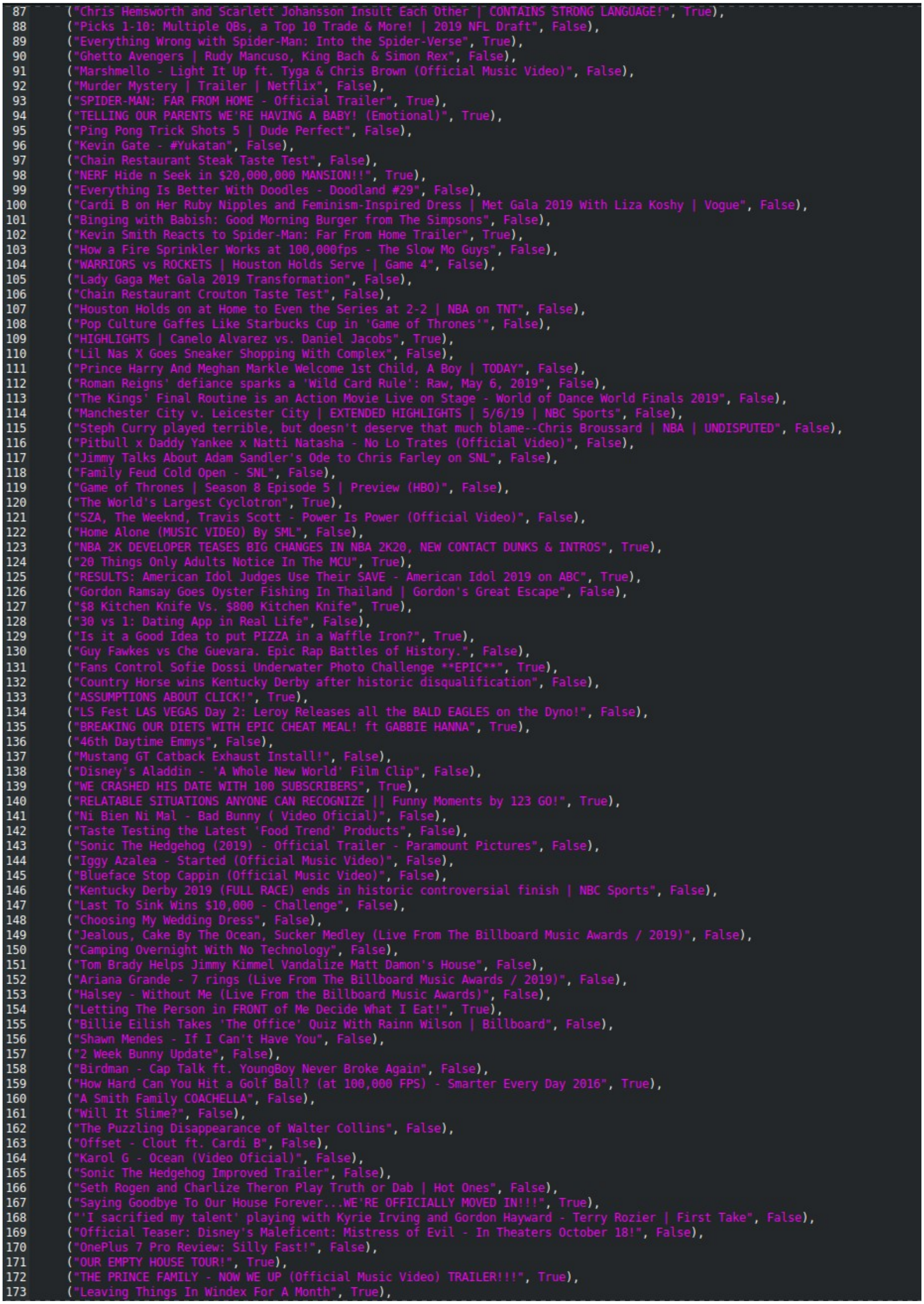
López-Sánchez, Daniel, et al. "Hybridizing Metric Learning and Case-Based Reasoning for Adaptable Clickbait Detection." *Applied Intelligence*, vol. 48, no. 9, 2018, pp. 2967–2982.

Thomas, Philippe. "Clickbait Identification Using Neural Networks." 2017.

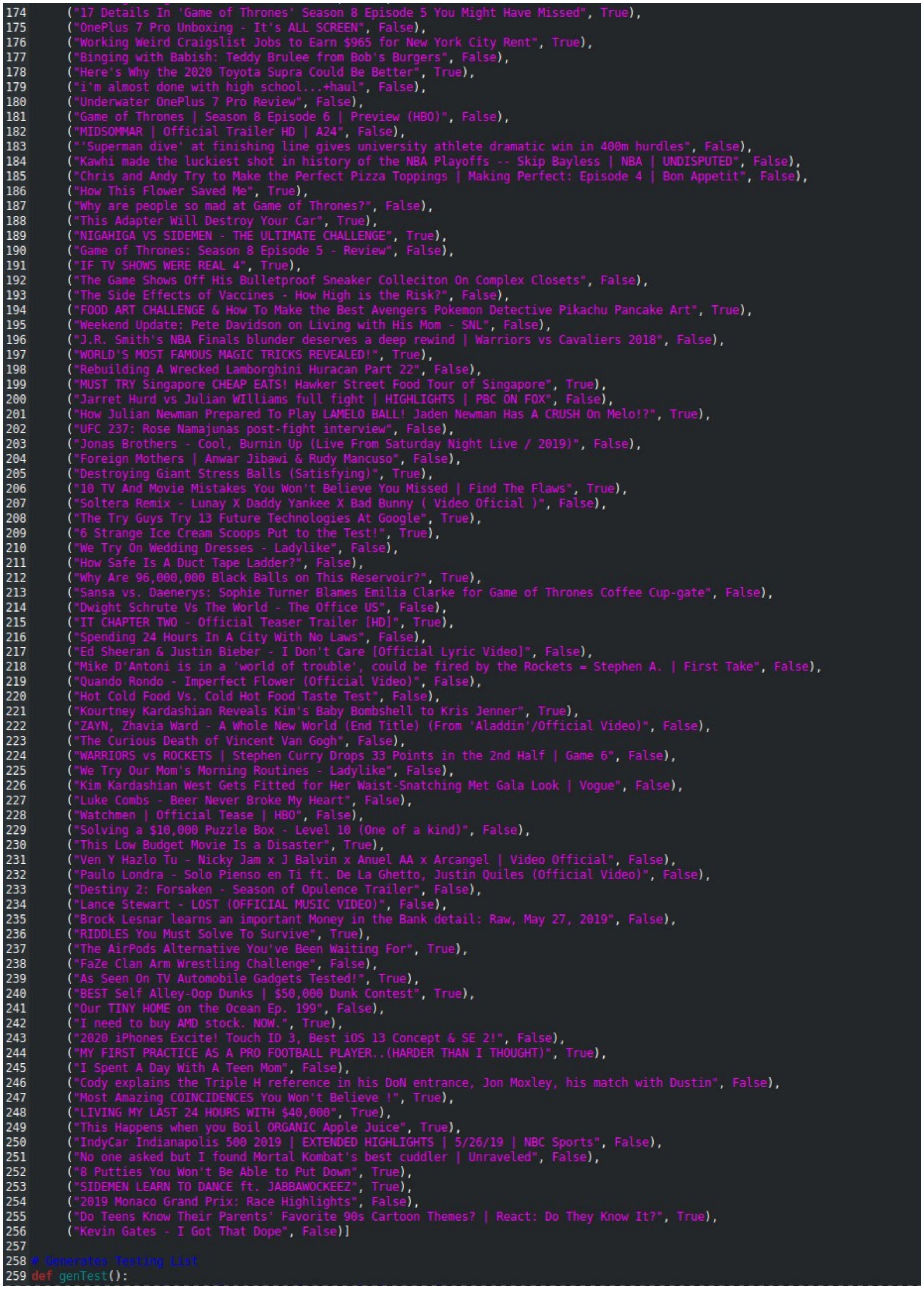
Appendix (Code)



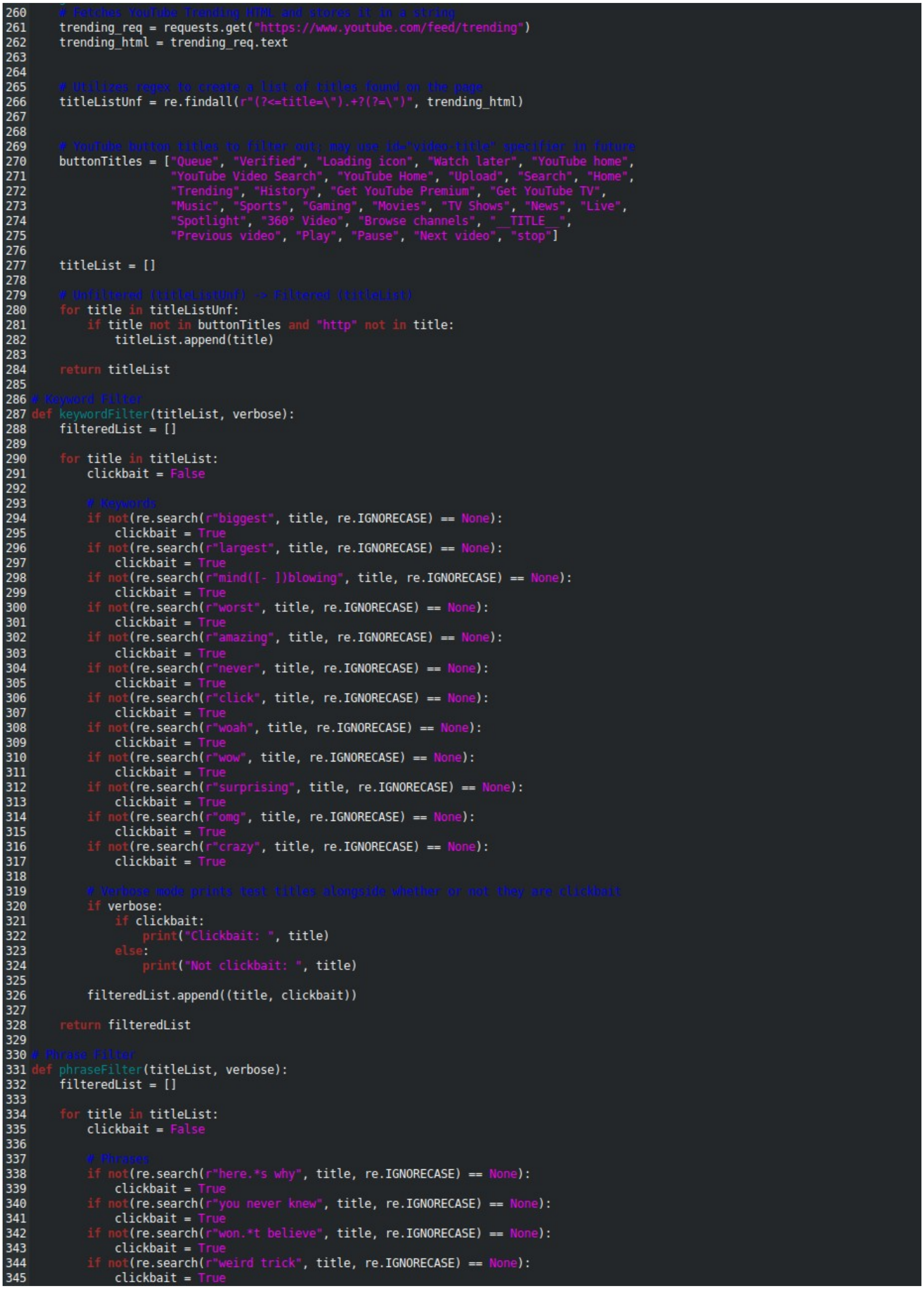
```python
import lxml.html
import requests
import re

# n = 250
# (title, clickbait?) tuple list
goldStandard = [("Ending the Subscribe to Pewdiepie Meme", False),
     ("Chiitan: Last Week Tonight with John Oliver (HBO)", False),
     ("Game of Thrones | Season 8 Episode 4 | Preview (HBO)", False),
     ("21 Savage - ball w/o you", False),
     ("Game of Thrones Season 8 Episode 4 Preview Breakdown", False),
     ("Putting Weird Things In An Air Fryer (TEST)", True),
     ("Game of Thrones Season 8 Episode 3 Review and Breakdown", False),
     ("20 Things Nobody Saw Coming in Marvel Avengers Endgame", True),
     ("Avengers: Endgame - Spoiler Review", False),
     ("What To Do When Someone Parks in the Access Aisle", False),
     ("James Harden 'choked under pressure' in the Rockets' Game 1 loss - Max Kellerman | First Take", False),
     ("Game Of Thrones Season 8 Episode 3 'The Long Night' Breakdown!", False),
     ("5 Giant DIY Foods Challenge & How To Make The Best Avengers Endgame Pancake Art in 24 Hours", True),
     ("Whale filmed harassing Norwegian boats could be 'Russian weapon'", False),
     ("'Everything Is a Liability' to Kourtney Kardashian's Law Student Sister Kim", False),
     ("When Your Best Friend Exposes Your Crush To The Entire School *SO EMBARRASSING!*", True),
     ("12 Details In 'Game of Thrones' Season 8 Episode 3 You Might Have Missed", True),
     ("$50,000 Tiny House Vs. $165,000 Tiny House", True),
     ("My First Toxic Relationship", False),
     ("Game of Thrones S8 - The Night King - Ramin Djawadi (Official Video)", False),
     ("TESLA Model 3 vs BMW M3 Track Battle | Top Gear", True),
     ("Game Theory: Hard Mode is a LIE! (Sekiro Easy Mode Controversy)", True),
     ("ROCKETS vs WARRIORS | Kevin Durant Continues Stellar Scoring | Game 1", False),
     ("Turning Pencil Lead into Diamonds", True),
     ("Putting A Whole Cake In An Air Fryer (TEST)", True),
     ("Game of Thrones: Season 8 Episode 3 - Review", False),
     ("The Kinds' 'Yeh Raat' Is Mind-Blowing - World of Dance 2019 (Full Performance)", True),
     ("Real Doctor Reacts to 'Adam Ruins the Hospital'", False),
     ("Getting a signed bat from Mike Trout at Globe Life Park!", False),
     ("Devin Shares Her Sexual Assault Story - Ladylike", False),
     ("Game of Thrones Season 8 EP3 (The Long Night) Review, Critiques, Analysis", False),
     ("The UGLY TRUTH Of Owning PROPERTY WITH A VIEW", True),
     ("Macs are SLOWER than PCs. Here's why.", True),
     ("McDonald's AARP Initiative, Bumble Safety Feature & Tech-Savvy Chimp | The Daily Show", False),
     ("6 Kitchen Gadgets put to the Test - Part 45", True),
     ("My Dog Reacts to Car Wash", True),
     ("Taylor Swift - ME! (feat. Brendon Urie of Panic! At The Disco)", False),
     ("This PC wouldn't boot...you'll never guess why!", True),
     ("YouTubers React To YouTube Videos With ZERO VIEWS", True),
     ("Sharks' Brent Burns Crushes Matt Calvert, Allows Nathan MacKinnon To Score Empty-Netter", False),
     ("Do All Jubilee Employees Think The Same?", False),
     ("Q&A With Grey: Favorites Edition", False),
     ("Avengers: Endgame's Biggest Unanswered Questions", True),
     ("Homemade Vs. Fast Food: Krispy Kreme Doughnuts - Tasty", True),
     ("FaZe Clan $1,000 Basketball Trickshot Challenge", False),
     ("Man United v. Chelsea | PREMIER LEAGUE EXTENDED HIGHLIGHTS | 4/28/19 | NBC Sports", True),
     ("Ten-year-old Giorgia gets Alesha's GOLDEN BUZZER with MIND-BLOWING vocals! | Auditions | BGT 2019", True),
     ("Avengers: Endgame Pitch Meeting", False),
     ("Would You Spend $100 on Meatballs or $1 Million on a Pigeon?", True),
     ("Mozzy - Chill Phillipe (Official Video)", False),
     ("Lil Dicky - Earth (Official Music Video)", False),
     ("Offset - Cloud ft. Cardi B", False),
     ("Anything You Can Carry, I'll Pay For Challenge", False),
     ("Making My Own Starbucks Pinkity Drinkity", False),
     ("The Try Guys Make Sushi Rolls", False),
     ("All Sports Baseball Battle | Dude Perfect", False),
     ("Birdman - Cap Talk ft. YoungBoy Never Broke Again", False),
     ("Meet Bunny Our Rescue Greyhouund", False),
     ("Full Face Using Only Milani Makeup... I'm Shook!", True),
     ("Amigos (Ep. 4) | Lele Pons, Rudy Mancuso, Juanpa Zurita, Hannah Stocking & Anwar Jibawi", False),
     ("Fortnite X Avengers: Endgame Trailer", False),
     ("The Ending Of Endgame Explained", False),
     ("S2E1: 'Mercy Part II'", False),
     ("Everyday Things You NEVER KNEW The Purpose Of", True),
     ("Try Not To Eat Challenge - Disney Food #3 | People Vs. Food", False),
     ("Adam Sandler's Reaction to His Daughter Locking Eyes with Boys", True),
     ("Godzilla: King of the Monsters - Final Trailer", False),
     ("Avengers: Endgame Cast Answer 50 of the Most Googled Marvel Questions | WIRED", True),
     ("Ozuna - Baila Baila Baila (Remix) Feat. Daddy Yankee, J Balvin, Farruko, Anuel AA (Audio Oficial)", False),
     ("THE SEARCH FOR OUR NEW PET!", True),
     ("How I Really Feel About That BEL-AIR Trailer", False),
     ("Film Theory: Thanos vs Ant Man - Cracking Endgame's Biggest Meme!", False),
     ("We Built the Worlds Largest Jello Cup!", True),
     ("Did refs let Warriors get away with fouls in Game 1? | After the Buzzer | 2019 NBA Playoffs", True),
     ("Avengers: Endgame - SPOILER Talk", False),
     ("BTS (Boy With Luv) feat. Halsey' Official MV", False),
     ("Farruko, Anuel AA, Kendo Kaponi - Delincuente (Pseudo Video)", False),
     ("Avengers Endgame Review", False),
     ("Maddona, Maluma - Medellin", False),
     ("Basically Everything You Need To Know Before End Game", True),
     ("Chris Hemsworth and Scarlett Johansson Insult Each Other | CONTAINS STRONG LANGUAGE!", True),
```



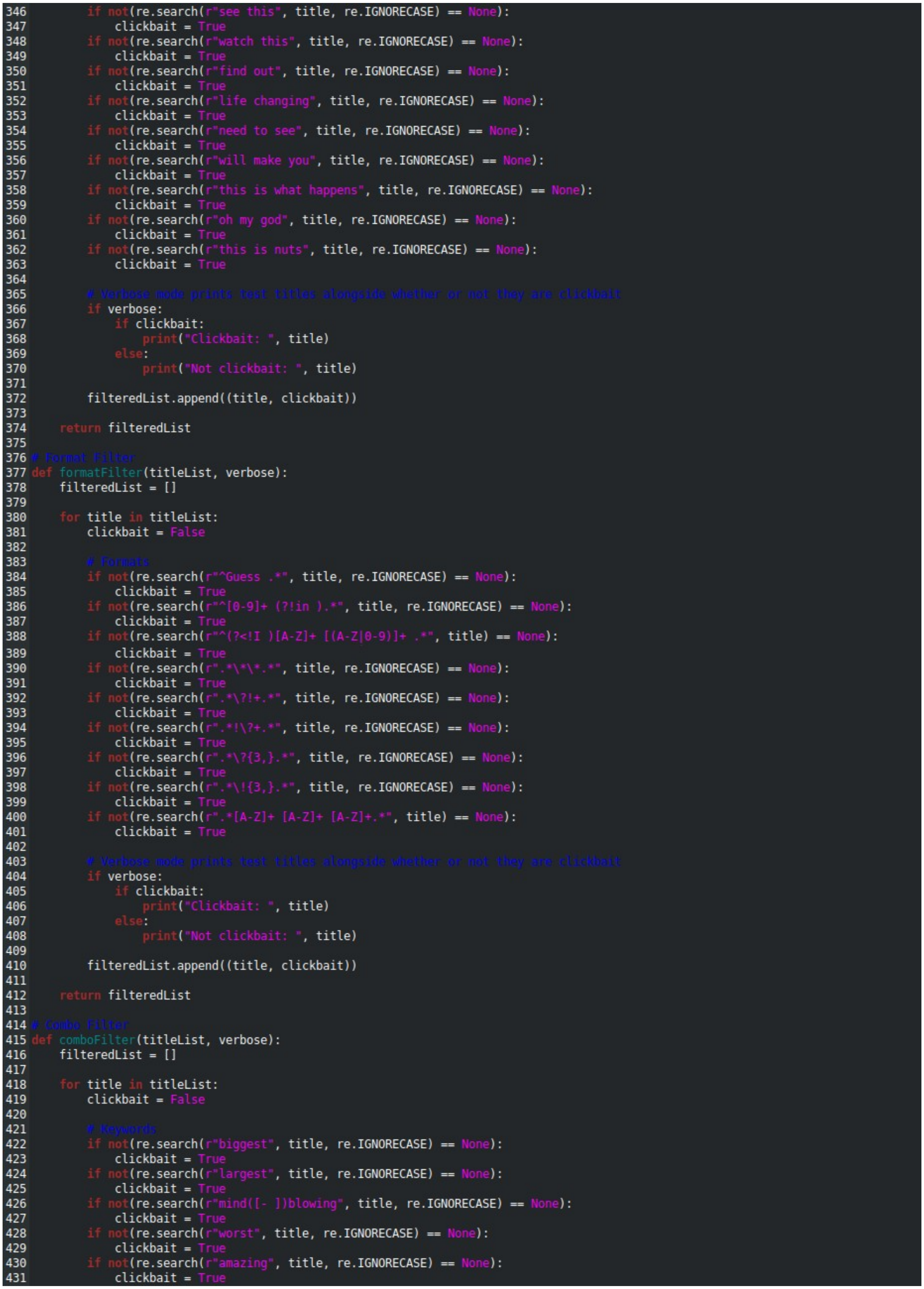
```
        ("Chris Hemsworth and Scarlett Johansson Insult Each Other | CONTAINS STRONG LANGUAGE!", True),
        ("Picks 1-10: Multiple QBs, a Top 10 Trade & More! | 2019 NFL Draft", False),
        ("Everything Wrong with Spider-Man: Into the Spider-Verse", True),
        ("Ghetto Avengers | Rudy Mancuso, King Bach & Simon Rex", False),
        ("Marshmello - Light It Up ft. Tyga & Chris Brown (Official Music Video)", False),
        ("Murder Mystery | Trailer | Netflix", False),
        ("SPIDER-MAN: FAR FROM HOME - Official Trailer", True),
        ("TELLING OUR PARENTS WE'RE HAVING A BABY! (Emotional)", True),
        ("Ping Pong Trick Shots 5 | Dude Perfect", False),
        ("Kevin Gate - #Yukatan", False),
        ("Chain Restaurant Steak Taste Test", False),
        ("NERF Hide n Seek in $20,000,000 MANSION!!", True),
        ("Everything Is Better With Doodles - Doodland #29", False),
        ("Cardi B on Her Ruby Nipples and Feminism-Inspired Dress | Met Gala 2019 With Liza Koshy | Vogue", False),
        ("Binging with Babish: Good Morning Burger from The Simpsons", False),
        ("Kevin Smith Reacts to Spider-Man: Far From Home Trailer", True),
        ("How a Fire Sprinkler Works at 100,000fps - The Slow Mo Guys", False),
        ("WARRIORS vs ROCKETS | Houston Holds Serve | Game 4", False),
        ("Lady Gaga Met Gala 2019 Transformation", False),
        ("Chain Restaurant Crouton Taste Test", False),
        ("Houston Holds on at Home to Even the Series at 2-2 | NBA on TNT", False),
        ("Pop Culture Gaffes Like Starbucks Cup in 'Game of Thrones'", False),
        ("HIGHLIGHTS | Canelo Alvarez vs. Daniel Jacobs", True),
        ("Lil Nas X Goes Sneaker Shopping With Complex", False),
        ("Prince Harry And Meghan Markle Welcome 1st Child, A Boy | TODAY", False),
        ("Roman Reigns' defiance sparks a 'Wild Card Rule': Raw, May 6, 2019", False),
        ("The Kings' Final Routine is an Action Movie Live on Stage - World of Dance World Finals 2019", False),
        ("Manchester City v. Leicester City | EXTENDED HIGHLIGHTS | 5/6/19 | NBC Sports", False),
        ("Steph Curry played terrible, but doesn't deserve that much blame--Chris Broussard | NBA | UNDISPUTED", False),
        ("Pitbull x Daddy Yankee x Natti Natasha - No Lo Trates (Official Video)", False),
        ("Jimmy Talks About Adam Sandler's Ode to Chris Farley on SNL", False),
        ("Family Feud Cold Open - SNL", False),
        ("Game of Thrones | Season 8 Episode 5 | Preview (HBO)", False),
        ("The World's Largest Cyclotron", True),
        ("SZA, The Weeknd, Travis Scott - Power Is Power (Official Video)", False),
        ("Home Alone (MUSIC VIDEO) By SML", False),
        ("NBA 2K DEVELOPER TEASES BIG CHANGES IN NBA 2K20, NEW CONTACT DUNKS & INTROS", True),
        ("20 Things Only Adults Notice In The MCU", True),
        ("RESULTS: American Idol Judges Use Their SAVE - American Idol 2019 on ABC", True),
        ("Gordon Ramsay Goes Oyster Fishing In Thailand | Gordon's Great Escape", False),
        ("$8 Kitchen Knife Vs. $800 Kitchen Knife", True),
        ("30 vs 1: Dating App in Real Life", False),
        ("Is it a Good Idea to put PIZZA in a Waffle Iron?", True),
        ("Guy Fawkes vs Che Guevara. Epic Rap Battles of History.", False),
        ("Fans Control Sofie Dossi Underwater Photo Challenge **EPIC**", True),
        ("Country Horse wins Kentucky Derby after historic disqualification", False),
        ("ASSUMPTIONS ABOUT CLICK!", True),
        ("LS Fest LAS VEGAS Day 2: Leroy Releases all the BALD EAGLES on the Dyno!", False),
        ("BREAKING OUR DIETS WITH EPIC CHEAT MEAL! ft GABBIE HANNA", True),
        ("46th Daytime Emmys", False),
        ("Mustang GT Catback Exhaust Install!", False),
        ("Disney's Aladdin - 'A Whole New World' Film Clip", False),
        ("WE CRASHED HIS DATE WITH 100 SUBSCRIBERS", True),
        ("RELATABLE SITUATIONS ANYONE CAN RECOGNIZE || Funny Moments by 123 GO!", True),
        ("Ni Bien Ni Mal - Bad Bunny ( Video Oficial)", False),
        ("Taste Testing the Latest 'Food Trend' Products", False),
        ("Sonic The Hedgehog (2019) - Official Trailer - Paramount Pictures", False),
        ("Iggy Azalea - Started (Official Music Video)", False),
        ("Blueface Stop Cappin (Official Music Video)", False),
        ("Kentucky Derby 2019 (FULL RACE) ends in historic controversial finish | NBC Sports", False),
        ("Last To Sink Wins $10,000 - Challenge", False),
        ("Choosing My Wedding Dress", False),
        ("Jealous, Cake By The Ocean, Sucker Medley (Live From The Billboard Music Awards / 2019)", False),
        ("Camping Overnight With No Technology", False),
        ("Tom Brady Helps Jimmy Kimmel Vandalize Matt Damon's House", False),
        ("Ariana Grande - 7 rings (Live From The Billboard Music Awards / 2019)", False),
        ("Halsey - Without Me (Live From the Billboard Music Awards)", False),
        ("Letting The Person in FRONT of Me Decide What I Eat!", True),
        ("Billie Eilish Takes 'The Office' Quiz With Rainn Wilson | Billboard", False),
        ("Shawn Mendes - If I Can't Have You", False),
        ("2 Week Bunny Update", False),
        ("Birdman - Cap Talk ft. YoungBoy Never Broke Again", False),
        ("How Hard Can You Hit a Golf Ball? (at 100,000 FPS) - Smarter Every Day 2016", True),
        ("A Smith Family COACHELLA", False),
        ("Will It Slime?", False),
        ("The Puzzling Disappearance of Walter Collins", False),
        ("Offset - Clout ft. Cardi B", False),
        ("Karol G - Ocean (Video Oficial)", False),
        ("Sonic The Hedgehog Improved Trailer", False),
        ("Seth Rogen and Charlize Theron Play Truth or Dab | Hot Ones", False),
        ("Saying Goodbye To Our House Forever...WE'RE OFFICIALLY MOVED IN!!!", True),
        ("'I sacrified my talent' playing with Kyrie Irving and Gordon Hayward - Terry Rozier | First Take", False),
        ("Official Teaser: Disney's Maleficent: Mistress of Evil - In Theaters October 18!", False),
        ("OnePlus 7 Pro Review: Silly Fast!", False),
        ("OUR EMPTY HOUSE TOUR!", True),
        ("THE PRINCE FAMILY - NOW WE UP (Official Music Video) TRAILER!!!", True),
        ("Leaving Things In Windex For A Month", True),
```



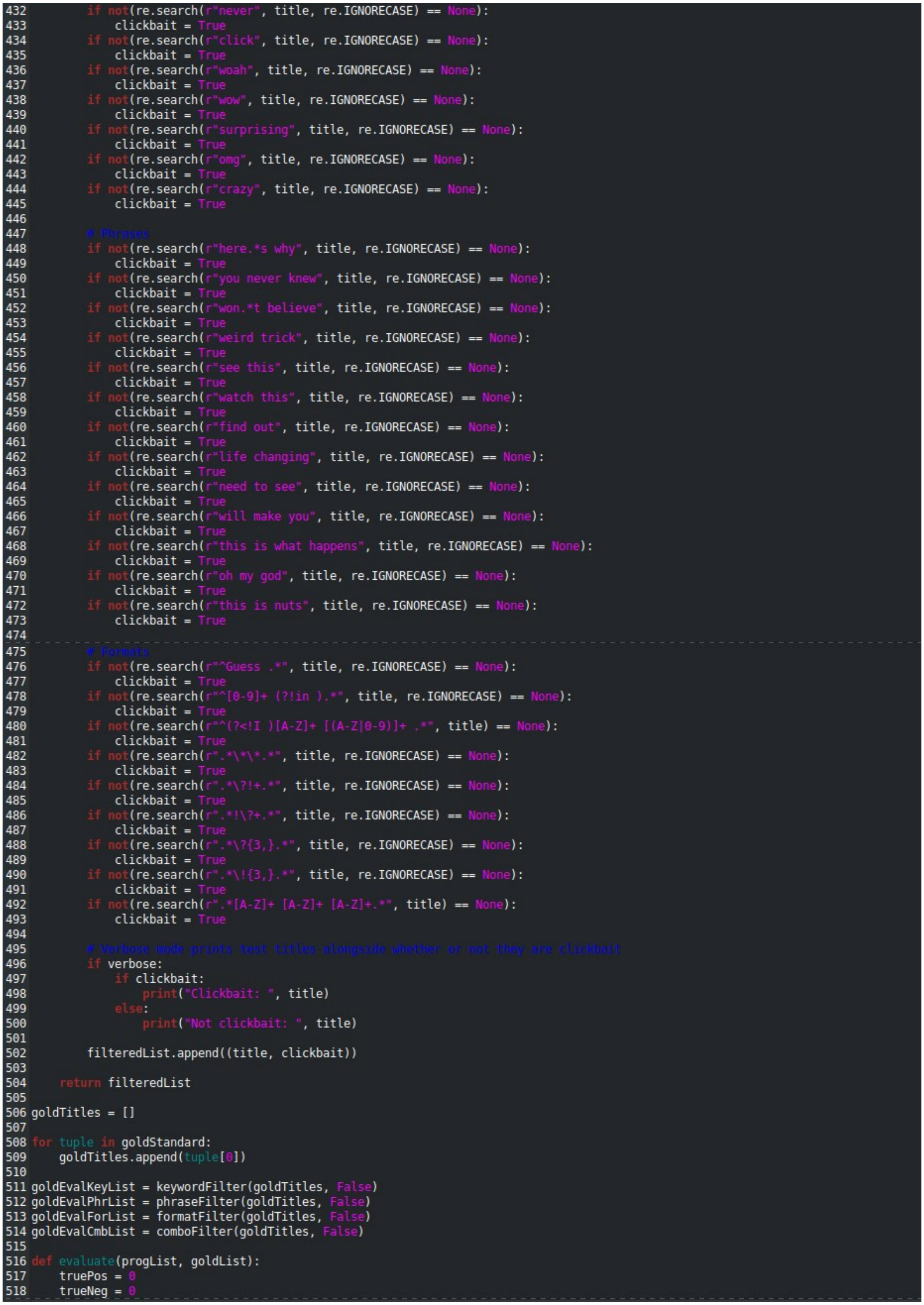
```
    ("17 Details In 'Game of Thrones' Season 8 Episode 5 You Might Have Missed", True),
    ("OnePlus 7 Pro Unboxing - It's ALL SCREEN", False),
    ("Working Weird Craigslist Jobs to Earn $965 for New York City Rent", True),
    ("Binging with Babish: Teddy Brulee from Bob's Burgers", False),
    ("Here's Why the 2020 Toyota Supra Could Be Better", True),
    ("i'm almost done with high school...+haul", False),
    ("Underwater OnePlus 7 Pro Review", False),
    ("Game of Thrones | Season 8 Episode 6 | Preview (HBO)", False),
    ("MIDSOMMAR | Official Trailer HD | A24", False),
    ("'Superman dive' at finishing line gives university athlete dramatic win in 400m hurdles", False),
    ("Kawhi made the luckiest shot in history of the NBA Playoffs -- Skip Bayless | NBA | UNDISPUTED", False),
    ("Chris and Andy Try to Make the Perfect Pizza Toppings | Making Perfect: Episode 4 | Bon Appetit", False),
    ("How This Flower Saved Me", True),
    ("Why are people so mad at Game of Thrones?", False),
    ("This Adapter Will Destroy Your Car", True),
    ("NIGAHIGA VS SIDEMEN - THE ULTIMATE CHALLENGE", True),
    ("Game of Thrones: Season 8 Episode 5 - Review", False),
    ("IF TV SHOWS WERE REAL 4", True),
    ("The Game Shows Off His Bulletproof Sneaker Colleciton On Complex Closets", False),
    ("The Side Effects of Vaccines - How High is the Risk?", False),
    ("FOOD ART CHALLENGE & How To Make the Best Avengers Pokemon Detective Pikachu Pancake Art", True),
    ("Weekend Update: Pete Davidson on Living with His Mom - SNL", False),
    ("J.R. Smith's NBA Finals blunder deserves a deep rewind | Warriors vs Cavaliers 2018", False),
    ("WORLD'S MOST FAMOUS MAGIC TRICKS REVEALED!", True),
    ("Rebuilding A Wrecked Lamborghini Huracan Part 22", False),
    ("MUST TRY Singapore CHEAP EATS! Hawker Street Food Tour of Singapore", True),
    ("Jarret Hurd vs Julian WIlliams full fight | HIGHLIGHTS | PBC ON FOX", False),
    ("How Julian Newman Prepared To Play LAMELO BALL! Jaden Newman Has A CRUSH On Melo!?", True),
    ("UFC 237: Rose Namajunas post-fight interview", False),
    ("Jonas Brothers - Cool, Burnin Up (Live From Saturday Night Live / 2019)", False),
    ("Foreign Mothers | Anwar Jibawi & Rudy Mancuso", False),
    ("Destroying Giant Stress Balls (Satisfying)", True),
    ("10 TV And Movie Mistakes You Won't Believe You Missed | Find The Flaws", True),
    ("Soltera Remix - Lunay X Daddy Yankee X Bad Bunny ( Video Oficial )", False),
    ("The Try Guys Try 13 Future Technologies At Google", True),
    ("6 Strange Ice Cream Scoops Put to the Test!", True),
    ("We Try On Wedding Dresses - Ladylike", False),
    ("How Safe Is A Duct Tape Ladder?", False),
    ("Why Are 96,000,000 Black Balls on This Reservoir?", True),
    ("Sansa vs. Daenerys: Sophie Turner Blames Emilia Clarke for Game of Thrones Coffee Cup-gate", False),
    ("Dwight Schrute Vs The World - The Office US", False),
    ("IT CHAPTER TWO - Official Teaser Trailer [HD]", True),
    ("Spending 24 Hours In A City With No Laws", False),
    ("Ed Sheeran & Justin Bieber - I Don't Care [Official Lyric Video]", False),
    ("Mike D'Antoni is in a 'world of trouble', could be fired by the Rockets = Stephen A. | First Take", False),
    ("Quando Rondo - Imperfect Flower (Official Video)", False),
    ("Hot Cold Food Vs. Cold Hot Food Taste Test", False),
    ("Kourtney Kardashian Reveals Kim's Baby Bombshell to Kris Jenner", True),
    ("ZAYN, Zhavia Ward - A Whole New World (End Title) (From 'Aladdin'/Official Video)", False),
    ("The Curious Death of Vincent Van Gogh", False),
    ("WARRIORS vs ROCKETS | Stephen Curry Drops 33 Points in the 2nd Half | Game 6", False),
    ("We Try Our Mom's Morning Routines - Ladylike", False),
    ("Kim Kardashian West Gets Fitted for Her Waist-Snatching Met Gala Look | Vogue", False),
    ("Luke Combs - Beer Never Broke My Heart", False),
    ("Watchmen | Official Tease | HBO", False),
    ("Solving a $10,000 Puzzle Box - Level 10 (One of a kind)", False),
    ("This Low Budget Movie Is a Disaster", True),
    ("Ven Y Hazlo Tu - Nicky Jam x J Balvin x Anuel AA x Arcangel | Video Official", False),
    ("Paulo Londra - Solo Pienso en Ti ft. De La Ghetto, Justin Quiles (Official Video)", False),
    ("Destiny 2: Forsaken - Season of Opulence Trailer", False),
    ("Lance Stewart - LOST (OFFICIAL MUSIC VIDEO)", False),
    ("Brock Lesnar learns an important Money in the Bank detail: Raw, May 27, 2019", False),
    ("RIDDLES You Must Solve To Survive", True),
    ("The AirPods Alternative You've Been Waiting For", True),
    ("FaZe Clan Arm Wrestling Challenge", False),
    ("As Seen On TV Automobile Gadgets Tested!", True),
    ("BEST Self Alley-Oop Dunks | $50,000 Dunk Contest", True),
    ("Our TINY HOME on the Ocean Ep. 199", False),
    ("I need to buy AMD stock. NOW.", True),
    ("2020 iPhones Excite! Touch ID 3, Best iOS 13 Concept & SE 2!", False),
    ("MY FIRST PRACTICE AS A PRO FOOTBALL PLAYER..(HARDER THAN I THOUGHT)", True),
    ("I Spent A Day With A Teen Mom", False),
    ("Cody explains the Triple H reference in his DoN entrance, Jon Moxley, his match with Dustin", False),
    ("Most Amazing COINCIDENCES You Won't Believe !", True),
    ("LIVING MY LAST 24 HOURS WITH $40,000", True),
    ("This Happens when you Boil ORGANIC Apple Juice", True),
    ("IndyCar Indianapolis 500 2019 | EXTENDED HIGHLIGHTS | 5/26/19 | NBC Sports", False),
    ("No one asked but I found Mortal Kombat's best cuddler | Unraveled", False),
    ("8 Putties You Won't Be Able to Put Down", True),
    ("SIDEMEN LEARN TO DANCE ft. JABBAWOCKEEZ", True),
    ("2019 Monaco Grand Prix: Race Highlights", False),
    ("Do Teens Know Their Parents' Favorite 90s Cartoon Themes? | React: Do They Know It?", True),
    ("Kevin Gates - I Got That Dope", False)]

# Generates Testing List
def genTest():
```



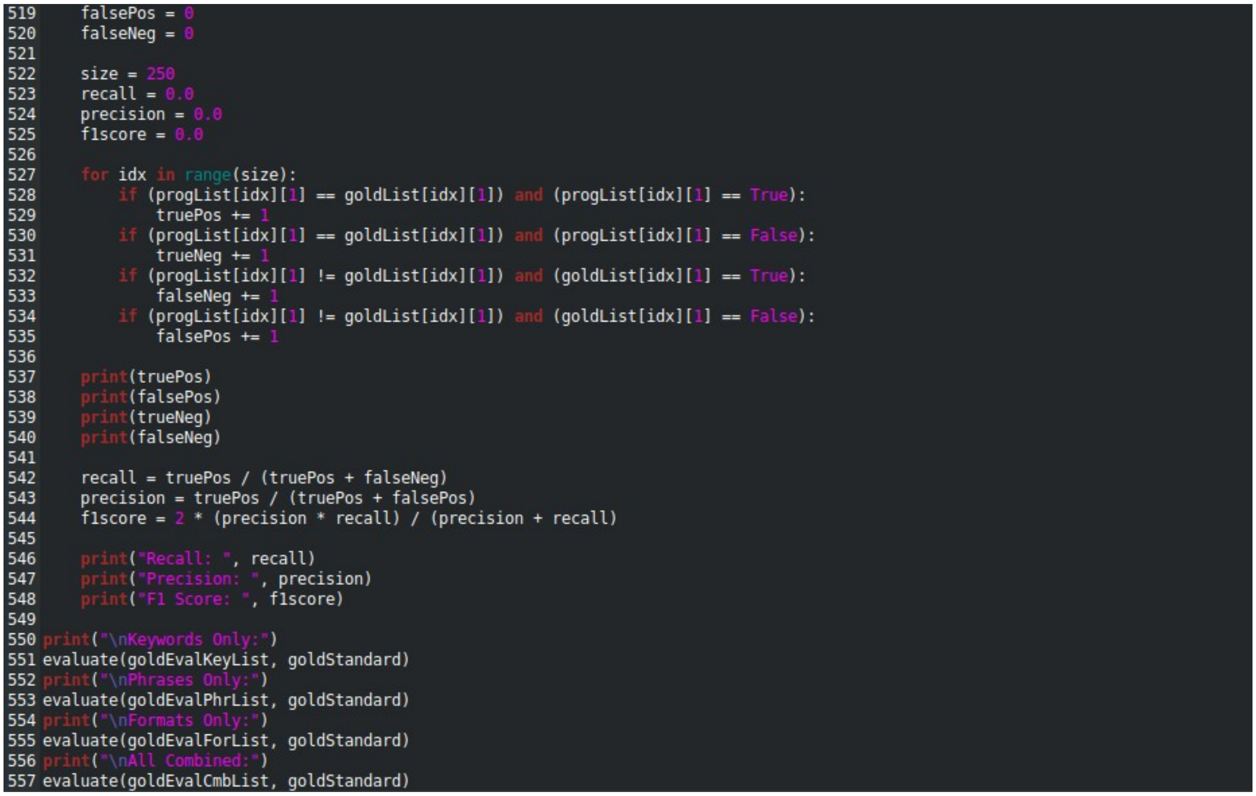
```
    # Fetches YouTube Trending HTML and stores it in a string
    trending_req = requests.get("https://www.youtube.com/feed/trending")
    trending_html = trending_req.text


    # Utilizes regex to create a list of titles found on the page
    titleListUnf = re.findall(r"(?<=title=\").+?(?=\")", trending_html)


    # YouTube button titles to filter out; may use id="video-title" specifier in future
    buttonTitles = ["Queue", "Verified", "Loading icon", "Watch later", "YouTube home",
                    "YouTube Video Search", "YouTube Home", "Upload", "Search", "Home",
                    "Trending", "History", "Get YouTube Premium", "Get YouTube TV",
                    "Music", "Sports", "Gaming", "Movies", "TV Shows", "News", "Live",
                    "Spotlight", "360° Video", "Browse channels", "__TITLE__",
                    "Previous video", "Play", "Pause", "Next video", "stop"]

    titleList = []

    # Unfiltered (titleListUnf) -> Filtered (titleList)
    for title in titleListUnf:
        if title not in buttonTitles and "http" not in title:
            titleList.append(title)

    return titleList

# Keyword Filter
def keywordFilter(titleList, verbose):
    filteredList = []

    for title in titleList:
        clickbait = False

        # Keywords
        if not(re.search(r"biggest", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"largest", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"mind([- ])blowing", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"worst", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"amazing", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"never", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"click", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"woah", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"wow", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"surprising", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"omg", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"crazy", title, re.IGNORECASE) == None):
            clickbait = True

        # Verbose mode prints test titles alongside whether or not they are clickbait
        if verbose:
            if clickbait:
                print("Clickbait: ", title)
            else:
                print("Not clickbait: ", title)

        filteredList.append((title, clickbait))

    return filteredList

# Phrase Filter
def phraseFilter(titleList, verbose):
    filteredList = []

    for title in titleList:
        clickbait = False

        # Phrases
        if not(re.search(r"here.*s why", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"you never knew", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"won.*t believe", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"weird trick", title, re.IGNORECASE) == None):
            clickbait = True
```



```
        if not(re.search(r"see this", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"watch this", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"find out", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"life changing", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"need to see", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"will make you", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"this is what happens", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"oh my god", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"this is nuts", title, re.IGNORECASE) == None):
            clickbait = True

        # Verbose mode prints test titles alongside whether or not they are clickbait
        if verbose:
            if clickbait:
                print("Clickbait: ", title)
            else:
                print("Not clickbait: ", title)

        filteredList.append((title, clickbait))

    return filteredList

# Format Filter
def formatFilter(titleList, verbose):
    filteredList = []

    for title in titleList:
        clickbait = False

        # Formats
        if not(re.search(r"^Guess .*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"^[0-9]+ (?!in ).*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"^(?<!I )[A-Z]+ [(A-Z|0-9)]+ .*", title) == None):
            clickbait = True
        if not(re.search(r".*\*\*.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*\?!+.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*!\?+.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*\?{3,}.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*\!{3,}.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*[A-Z]+ [A-Z]+ [A-Z]+.*", title) == None):
            clickbait = True

        # Verbose mode prints test titles alongside whether or not they are clickbait
        if verbose:
            if clickbait:
                print("Clickbait: ", title)
            else:
                print("Not clickbait: ", title)

        filteredList.append((title, clickbait))

    return filteredList

# Combo Filter
def comboFilter(titleList, verbose):
    filteredList = []

    for title in titleList:
        clickbait = False

        # Keywords
        if not(re.search(r"biggest", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"largest", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"mind([- ])blowing", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"worst", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"amazing", title, re.IGNORECASE) == None):
            clickbait = True
```



```
        if not(re.search(r"never", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"click", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"woah", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"wow", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"surprising", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"omg", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"crazy", title, re.IGNORECASE) == None):
            clickbait = True

        # Phrases
        if not(re.search(r"here.*s why", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"you never knew", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"won.*t believe", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"weird trick", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"see this", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"watch this", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"find out", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"life changing", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"need to see", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"will make you", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"this is what happens", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"oh my god", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"this is nuts", title, re.IGNORECASE) == None):
            clickbait = True

        # Formats
        if not(re.search(r"^Guess .*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"^[0-9]+ (?!in ).*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r"^(?<!I )[A-Z]+ [(A-Z|0-9)]+ .*", title) == None):
            clickbait = True
        if not(re.search(r".*\*\*.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*\?!+.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*!\?+.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*\?{3,}.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*\!{3,}.*", title, re.IGNORECASE) == None):
            clickbait = True
        if not(re.search(r".*[A-Z]+ [A-Z]+ [A-Z]+.*", title) == None):
            clickbait = True

        # Verbose mode prints test titles alongside whether or not they are clickbait
        if verbose:
            if clickbait:
                print("Clickbait: ", title)
            else:
                print("Not clickbait: ", title)

        filteredList.append((title, clickbait))

    return filteredList

goldTitles = []

for tuple in goldStandard:
    goldTitles.append(tuple[0])

goldEvalKeyList = keywordFilter(goldTitles, False)
goldEvalPhrList = phraseFilter(goldTitles, False)
goldEvalForList = formatFilter(goldTitles, False)
goldEvalCmbList = comboFilter(goldTitles, False)

def evaluate(progList, goldList):
    truePos = 0
    trueNeg = 0
```



```
    falsePos = 0
    falseNeg = 0

    size = 250
    recall = 0.0
    precision = 0.0
    f1score = 0.0

    for idx in range(size):
        if (progList[idx][1] == goldList[idx][1]) and (progList[idx][1] == True):
            truePos += 1
        if (progList[idx][1] == goldList[idx][1]) and (progList[idx][1] == False):
            trueNeg += 1
        if (progList[idx][1] != goldList[idx][1]) and (goldList[idx][1] == True):
            falseNeg += 1
        if (progList[idx][1] != goldList[idx][1]) and (goldList[idx][1] == False):
            falsePos += 1

    print(truePos)
    print(falsePos)
    print(trueNeg)
    print(falseNeg)

    recall = truePos / (truePos + falseNeg)
    precision = truePos / (truePos + falsePos)
    f1score = 2 * (precision * recall) / (precision + recall)

    print("Recall: ", recall)
    print("Precision: ", precision)
    print("F1 Score: ", f1score)

print("\nKeywords Only:")
evaluate(goldEvalKeyList, goldStandard)
print("\nPhrases Only:")
evaluate(goldEvalPhrList, goldStandard)
print("\nFormats Only:")
evaluate(goldEvalForList, goldStandard)
print("\nAll Combined:")
evaluate(goldEvalCmbList, goldStandard)
```