Using Sentiment Analysis to Improve Youtube Reccomended Video Rankings

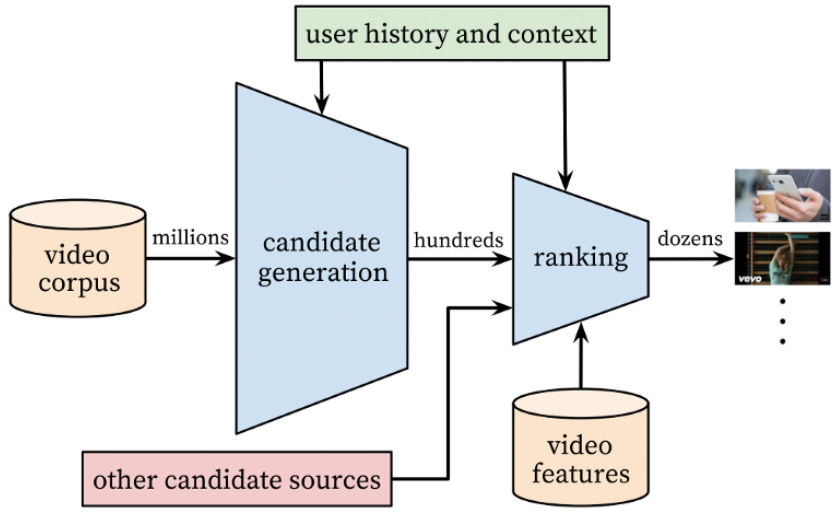
Mark Richards

5/4/2019

GitHub Repository

Introduction

This report will cover a method for improving YouTube reccomendations based on the emotions that videos evoke in those that watch and comment on them. It is not meant to be a replacement for the deep neural network that Google uses to build its reccomendations. It is meant to demonstrate the usefulness of incorperating sentiment analysis as part of the ranking network for serving reccmendations to users.
This project requires some background knowledge of youTube and how it generates its reccomendations. In 2016 Google relased a paper titled "Deep Neural Networks for YouTube Recommendations" where it outlines its use of two deep neural nets to build and rank reccomendations. This paper covers the processes they use in great detail unfortuantly however it reads like an academic journal article. In 2018 Moin Nadeem a then Junior at the Massachusetts Institute of Technology (MIT) wrote a more human friendly overview of the topics covered in the paper for the website Towards Data Science titled simply "How YouTube Recommends Videos".
A brief overview is that Google uses two deep neural networks to create suggestions. The first looks at things like videos previously watched, search history, and demographics to narrow down the pool of videos to be ranked. The second deep neural net takes more features such as preview image and peer interest and uses that to order the list of reccomendations (Nadeem 2018). The goal of these networks is to maximize watch time. A diagram of this process is included below.




```
## [1] "Diagram of YouTube Deep Neural Networks (Nadeem 2018)"
```

At the end of april 2019 news reports such as "Alphabet had more than $70 billion in market cap wiped out, and it says YouTube is one of the problems" began to appear. These articles blame a dip in YouTube engagment and ad revenue for the drop in market value. They also indicate that YouTube's recent efforts to curb fake news and conspiracy theories may be to blame for the drop in engagment.

It may be that adding sentiment analysis can help improve engament again by connecting users with content that matches emotional tones. This also opens the door to building emotional profiles of users but that discussion is beyond this paper.

Analysis

Note: To run the code in this analysis an API key is required. Instructons on how to aquire an API key can be found here: "YouTube Data API Overview". This API key should be saved to your working directory as an RDS file with a data.frame in the following format. This file will **not** be provided on the GitHub Repository for this paper.

```
APIKey<-data.frame(
  app_id='Your App ID Here',
  app_secret = 'Your App Secret Here')
```

There are several packages we will require to start our analysis. This code will install and load them.

```
if (!require(tuber)) install.packages('tuber')
if (!require(syuzhet)) install.packages('syuzhet')
if (!require(ggplot2)) install.packages('ggplot2')
if (!require(tidyverse)) install.packages('tidyverse')
if (!require(knitr)) install.packages('knitr')
if (!require(kableExtra)) install.packages('kableExtra')
library(tuber)
library(syuzhet)
library(ggplot2)
library(tidyverse)
library(knitr)
library(kableExtra)
```

Once the packages are installed and loaded the script will check to make sure you have an API key installed. If you do not it will display the URL for how to get one. If it is able to find a key RDS is will load the file and setup the Oauth token. You will be directed to a website to authorize the app the first time you run this.

```
APIKey<-file_test("-f","APIKey.rds") #test if apikey exists
if (APIKey == FALSE) { #run code block if apikey does not exist
  print("No youtube API Key")
  print("See https://developers.google.com/youtube/v3/getting-started")
  print("expects key as RDS data.frame in form
        APIKey<-data.frame(app_id='Your App ID Here',
        app_secret = 'Your App Secret Here')")
  rm(APIKey) #removes file test
}else { #run code block if api key exists
  APIKey<-readRDS("APIKey.rds") #read API key
  yt_oauth(app_id=APIKey$app_id,
           app_secret = APIKey$app_secret, token = "") #load key to memory
}
```

Now that the API key is in place its time to pick a video that we want reccomendations for. for this paper I will be using a video from the cooking channel Binging With Babish where he attemts to recreate Jake's Perfect Sandwich from Adventure Time. The following code sets the "Base" video URL.

```
Baselink<-"https://www.youtube.com/watch?v=HsxBw6ls7Z0" #link to a liked youtube video
```

Once the base video URL is entered we need to separate out the Video ID. This is the string of alphanumeric characters after the "=" in the URL. A regular expresion is used to select everything past the "=" sign and

save it for use with the API calls.

```
Baselink<-str_match(Baselink,"[^=]+$") #regex to extract video ID
```

When the Video ID is isolated we can begin making calls to the YouTube API and pulling data down for analysis. To start lets retieve what YouTube considers to be the top ten reccomendations for me.

```
yt_oauth(app_id=APIKey$app_id,
         app_secret = APIKey$app_secret, token = "")
#Markdown has issues with chunk execution order so the key may need to be re entered
RelatedVids<- get_related_videos(video_id =Baselink, max_results = 11) #Get recomended
```

This call returns a list of videos with 17 variables. To view the list we will only display the two most releveant variables Video ID and Title.

```
RelatedTable<-RelatedVids[,c("rel_video_id","title")]
kable(RelatedTable) %>%
  kable_styling(latex_options="scale_down")
```

rel_video_id	title
hXYoduN0kWs	Binging with Babish: Cubanos from Chef
basFyoMSjds	Binging with Babish: Bob's Burgers
wBhhlE92mIQ	Binging with Babish: Rick & Morty Szechuan Sauce
KxPgrGdSHh8	Binging with Babish: South Park Special
Rzd0mLf366I	Binging with Babish: Krabby Supreme from Spongebob Squarepants
27mUWs2wjPs	Binging with Babish: Homer Simpson's Patented Space Age Out-Of-This-World Moon Waffles
Mf4wwXM2o_M	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show
KUu2gJn1dzc	How to Make New-York-Style Pizza - TMNT II: Secret of the Ooze
uEjMyHccX8U	Binging with Babish: Goodfellas Prison Sauce
OmwE0aJqkdA	Binging with Babish: Big Kahuna Burger from Pulp Fiction

Note: Reproducability of this list will almost certianly be impossible. This is the list of reccomendations generated specifically for me based on Googles deep neural networks being fed my personal preferances. So consider everything past this point an illustarative example.
Now that the list of reccomendations is stored we can begin analyzing our Base video. We are going to want to grab all of the comments for the Base video. We will also get the deatils for the Base video and extract the Title of the video for use later.

```
yt_oauth(app_id=APIKey$app_id,
         app_secret = APIKey$app_secret, token = "")
#Markdown has issues with chunk execution order so the key may need to be re entered
baseVid<-get_all_comments(video_id = Baselink) #get the comments from that video
BaseDetail<-get_video_details(video_id = Baselink)#pull details of the base video
BaseTitle<-BaseDetail$items[[1]]$snippet$title #extract the title of the base video
```

The API call to get all comments tells us that the Base video has `5436` comments. The Video Details call gives us basic information like the video description, thumbnail urls, video tags, and title. A quick view of the relevant parts of the comments section shows us things like who made the comment, what the comment said, and how many poeple liked the comment. We can take a look at the top rated comments. After sanitizing the unicode characters since LaTex does not like them.

```
BaseCommentTable<-baseVid[,c("authorDisplayName" ,"textOriginal","likeCount")]
BaseCommentTable$textOriginal<-iconv(BaseCommentTable$textOriginal, to="ASCII")
BaseCommentTable$authorDisplayName<-iconv(BaseCommentTable$authorDisplayName, to="ASCII")
BaseCommentTable$likeCount<-as.numeric(BaseCommentTable$likeCount)
BaseCommentTable <- BaseCommentTable[order(BaseCommentTable$likeCount, decreasing = TRUE),]
kable(head(BaseCommentTable,10)) %>%  kable_styling(latex_options="scale_down")
```

	authorDisplayName	textOriginal	likeCount
4788	Muselk	So happy to see this channel finally taking off.	5977
2194	Barry Smith	Now you gotta let Magic Man steal it and waste 15 minutes trying to get it back	4378
4296	Geral Ferald	Now do the everything burrito!	3743
3278	IanIsTheHero	You forgot to wash a carrot and then not use it, but a+ 10/10 as usual	3588
3108	Ricardo Arcila	"...not totally necessary but i'm a completist..." "...it's not exactly physically possible, but i'm no quitter..." awesome attitude, best thing about this videos...	3144
5395	Thund3rman80	is it possible to make a krabby pattie, from spongebob SquarePants	2929
2176	Brendan	Jesus you could feed a whole family with one of those	2696
2119	yotam lichtig	I want to sleep in a bed made of your voice	2225
5414	Kevin Lewis	Babish, your kitchen is the stuff of dreams.	2208
5421	JT Turner	I love the music. Glad that you can finally make some money off these videos, you deserve it man! I do miss the intro, but hey, it's not the end of the world.	1627

Now that we have the comments for the base video ready to go we can start getting them ready to feed n our sentiment analysis engine. In this case we will be using the syuzhet package. and its nrc_sentiment() function. This fucntion loads a dictionary of words that have been tagged with a set of emotions and Calls calcualts the presenace and the valance either positive or negative. The NRC Word-Emotion Association Lexicon was built by Saif Mohammad and the National Research Council of Canada. More information about it can be found here: NRC Emotion Lexicon. We can look at the first few rows to see what the results are.

```
#sanitize the text by converting to utf-8
baseVid$sanitary<-iconv(baseVid$textOriginal, to="UTF-8")
#get emotional levels from each comment
BaseEmo<-get_nrc_sentiment(baseVid$sanitary)
kable(head(BaseEmo,10)) %>%  kable_styling(latex_options="scale_down")
```

anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	0	0	1	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0

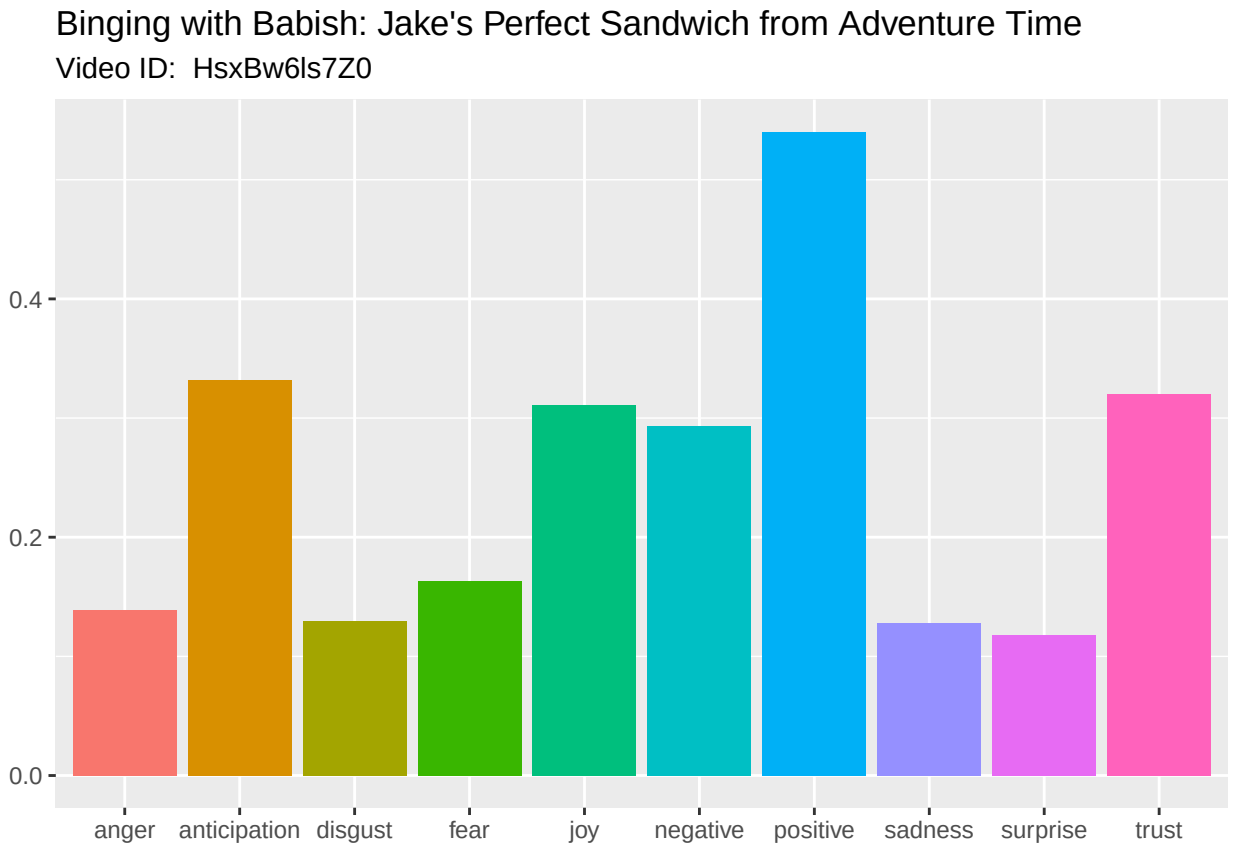
These logicals are intereting but not very useful on theor own. We are more interested in the emotions people are expressing overall. To find that we can take the average of each of these columns and get a sort of emotional profile.

```
#create averages of sentiment and emotion for base video
Baseavg<-apply(BaseEmo, 2, mean)
#convert averages to data frame for plotting
Baseavg<-data.frame(name=Baseavg)
kable(Baseavg)
```

	name
anger	0.1392568
anticipation	0.3318617
disgust	0.1293230
fear	0.1629875
joy	0.3107064
sadness	0.1284032
surprise	0.1177336
trust	0.3202723
negative	0.2934143
positive	0.5402870

Now we want to make a graph showing these averages so we can understand what we are looking at. We will also include the Title of the video and the Video ID.

```
#plot the emotional sentiment averages
Avg<-ggplot(Baseavg, aes_(x=row.names(Baseavg), y=Baseavg$name,
fill=row.names(Baseavg)))+geom_bar(stat="identity")+theme(legend.position="none",
                        axis.title.x=element_blank(),axis.title.y=element_blank())
Avg+ggtitle(BaseTitle,paste("Video ID: ",baseVid$videoId[1]))
```



Here we can see that the comments are overwhelimigly positive. While Anticipation, Joy, Negativity, and Trust were all about equally represented. This graphs represents the general level of different emotions expressed by people who commented on the video. This gives us the emotional profile we will be attempting to match across the set of videos YouTube reccomends to us.

the next step is to use the API to pull the comments for our list of reccomended videos. The following code will build a list of data frames that contain all of the comments for the reccomended videos. For this report it will also display the number of comments on each of the related videos.

```
yt_oauth(app_id=APIKey$app_id,
         app_secret = APIKey$app_secret, token = "")
#Markdown has issues with chunk execution order so the key may need to be re entered
seq<-1:length(RelatedVids$rel_video_id) #set iteration length
RelatedComm<-list() #initalize a list for the related video comments
for (n in seq){ #get comments,sanitize by converting to UTF-8 format
  RelatedComm[[n]]<-get_all_comments(video_id = as.character(RelatedVids$rel_video_id[n]))
  RelatedComm[[n]]$sanitary<-iconv(RelatedComm[[n]]$textOriginal, to="UTF-8")
  RelatedComm[[n]]$likeCount<-as.numeric(RelatedComm[[n]]$likeCount)
  RelatedTable$Comments[n]<-length(RelatedComm[[n]]$id)
}
```

```
## Auto-refreshing stale OAuth token.
```

```
rm(n,seq) #remove n and sequence used to itterate over list
kable(RelatedTable) %>%
  kable_styling(latex_options="scale_down")
```

rel_video_id	title	Comments
hXYoduN0kWs	Binging with Babish: Cubanos from Chef	3569
basFyoMSjds	Binging with Babish: Bob's Burgers	4779
wBhhlE92mIQ	Binging with Babish: Rick & Morty Szechuan Sauce	4450
KxPgrGdSHh8	Binging with Babish: South Park Special	4606
Rzd0mLf366I	Binging with Babish: Krabby Supreme from Spongebob Squarepants	9384
27mUWs2wjPs	Binging with Babish: Homer Simpson's Patented Space Age Out-Of-This-World Moon Waffles	3844
Mf4wwXM2o_M	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show	14987
KUu2gJn1dzc	How to Make New-York-Style Pizza - TMNT II: Secret of the Ooze	871
uEjMyHccX8U	Binging with Babish: Goodfellas Prison Sauce	1591
OmwE0aJqkdA	Binging with Babish: Big Kahuna Burger from Pulp Fiction	1468

Now we need to go entry by entry on the list and get the NRC sentiments. This code will perform that function. A word of caution this process can take some some if there is a large numebr of caomments.

```
seq<-1:length(RelatedVids$rel_video_id) #set iteration length
RelatedEmo<-list()
for (n in seq){ #itterate over all comments and get emotional sentiment
  RelatedEmo[[n]]<-get_nrc_sentiment(RelatedComm[[n]]$sanitary)
}
rm(n,seq)
```

Once the sentiments are calucalted we need to build the averages of those sentiments.

```
seq<-1:length(RelatedEmo) #set iteration length
RelatedEmoAvg<-list()
for (n in seq){ #calulate averages of emotional senitment
  RelatedEmoAvg[[n]]<-apply(RelatedEmo[[n]][1:10], 2, mean)
}
rm(n,seq)
```

Here there is a bit of data reshaoing going on. The list of related video emotional averages needs to be converted from a list to a data frame. We also want to add the video IDs to the list so we can keep track of which video is which.

```
RelatedEmoAvg<-data.frame(RelatedEmoAvg) #reshape the dataframe
RelatedEmoAvg<-data.frame(t(RelatedEmoAvg))
rownames(RelatedEmoAvg)<-RelatedVids$rel_video_id #set row names
kable(RelatedEmoAvg) %>%
```

```
  kable_styling(latex_options="scale_down")
```

	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
hXYoduN0kWs	0.1487812	0.3048473	0.1482208	0.1597086	0.4099187	0.1358924	0.1330905	0.4071168	0.3121322	0.6564864
basFyoMSjds	0.1111111	0.2690940	0.1146683	0.1339192	0.3615819	0.1630048	0.1171793	0.3092697	0.2904373	0.5461394
wBhhlE92mIQ	0.1253933	0.2611236	0.1148315	0.1287640	0.2932584	0.0957303	0.1175281	0.3067416	0.3557303	0.4988764
KxPgrGdSHh8	0.1189752	0.3056882	0.1276596	0.1610942	0.3775510	0.1387321	0.1369952	0.3814590	0.2865827	0.5963960
Rzd0mLf366I	0.2054561	0.2524510	0.1603794	0.2231458	0.2507460	0.1722080	0.1643223	0.2686488	0.3593350	0.4408568
27mUWs2wjPs	0.2476587	0.2544225	0.1318939	0.1560874	0.2825182	0.2104579	0.1212279	0.3480749	0.4232570	0.5416233
Mf4wwXM2o_M	0.1774323	0.3245029	0.1760977	0.2039904	0.3389163	0.1401975	0.1502736	0.3596023	0.3391832	0.6016949
7UEIBdMyj9g	0.1974742	0.2445465	0.1400689	0.3467279	0.3042480	0.1423651	0.1917336	0.3088404	0.3191734	0.5430540
uEjMyHccX8U	0.1665619	0.3544940	0.1175361	0.1797612	0.4066625	0.1565053	0.1546197	0.4116908	0.3331238	0.6838466
OmwE0aJqkdA	0.1505450	0.3092643	0.1212534	0.1689373	0.3569482	0.1294278	0.1505450	0.3399183	0.3133515	0.6348774

With all of the comments now converted to averages we need to rank the videos. We do this by calcualting the absolute differences then ranking each emotion and each video. This again will rely on a considerable amount of data reshaping to properly rank all of the elements. This next bit of code starts the process by clauclating the absolute differences of each emotion from the Base video average.

```
Baseavg<-data.frame(t(Baseavg)) #transpose the baseavg df
AbsoluteEmo<-data.frame() #initialize ABS difference dataframe
seq<-1:length(RelatedEmo)
for (n in seq){ #calculate difference between base emotions and related video emotions
  AbsoluteEmoTemp<-abs(Baseavg[1,]-RelatedEmoAvg[n,])
  AbsoluteEmo<-rbind(AbsoluteEmo,AbsoluteEmoTemp)
}
rm(n,seq,AbsoluteEmoTemp)
rownames(AbsoluteEmo)<-RelatedVids$rel_video_id #set row names
kable(AbsoluteEmo) %>%
  kable_styling(latex_options="scale_down")
```

	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
hXYoduN0kWs	0.0095244	0.0270144	0.0188978	0.0032789	0.0992123	0.0074892	0.0153569	0.0868446	0.0187180	0.1161994
basFyoMSjds	0.0281457	0.0627677	0.0146547	0.0290683	0.0508755	0.0346016	0.0005543	0.0110025	0.0029769	0.0058524
wBhhlE92mIQ	0.0138635	0.0707381	0.0144916	0.0342234	0.0174480	0.0326729	0.0002055	0.0135307	0.0623161	0.0414106
KxPgrGdSHh8	0.0202816	0.0261734	0.0016635	0.0018933	0.0668446	0.0103289	0.0192616	0.0611867	0.0068316	0.0561090
Rzd0mLf366I	0.0661993	0.0794107	0.0310563	0.0601583	0.0599605	0.0438048	0.0465886	0.0516235	0.0659208	0.0994302
27mUWs2wjPs	0.1084019	0.0774392	0.0025708	0.0069001	0.0281882	0.0820546	0.0034943	0.0278027	0.1298427	0.0013363
Mf4wwXM2o_M	0.0381755	0.0073588	0.0467747	0.0410029	0.0282099	0.0117943	0.0325400	0.0393300	0.0457690	0.0614079
7UEIBdMyj9g	0.0582174	0.0873152	0.0107459	0.1837404	0.0064584	0.0139619	0.0740000	0.0114318	0.0257591	0.0027670
uEjMyHccX8U	0.0273051	0.0226324	0.0117869	0.0167737	0.0959561	0.0281021	0.0368861	0.0914185	0.0397095	0.1435597
OmwE0aJqkdA	0.0112882	0.0225974	0.0080696	0.0059498	0.0462418	0.0010246	0.0328113	0.0196460	0.0199372	0.0945904

With the table of absolute differences we can now begin reshaping and ranking the data. The idea here is that we want to rotate the data frame above so that each emotion is a row and each video is a column. Then we want to rank each columns values. The larger the value the higher its rank, becuase ranking happens in accending order.

For example if a video had the exact same level of trust as the base video it would rank that videos trust as a 1. If that same video have a very different level of anger it would rank it a 10.

```
AbsoluteEmo<-data.frame(t(AbsoluteEmo)) #transpose dataframe
AbsoluteEmo<-apply(AbsoluteEmo,2,rank) #get rank of abs diff for each emotion by video
kable(AbsoluteEmo) %>%
  kable_styling(latex_options="scale_down")
```

	hXYoduN0kWs	basFyoMSjds	wBhhlE92mIQ	KxPgrGdSHh8	Rzd0mLf366I	X27mUWs2wjPs	Mf4wwXM2o_M	X7UEIBdMyj9g	uEjMyHccX8U	OmwE0aJqkdA
anger	3	6	3	6	8	9	5	7	4	4
anticipation	7	10	10	7	9	7	1	9	3	7
disgust	6	5	4	1	1	2	9	3	1	3
fear	1	7	7	2	6	4	7	10	2	2
joy	9	9	5	10	5	6	3	2	9	9
sadness	2	8	6	4	2	8	2	5	5	1
surprise	4	1	1	5	3	3	4	8	6	8
trust	8	4	2	9	4	5	6	4	8	5
negative	5	2	9	3	7	10	8	6	7	6
positive	10	3	8	8	10	1	10	1	10	10

Now we want to rotate it back to the original shape so each video is a row and each emotion is a column.

```
AbsoluteEmo<-t(AbsoluteEmo) #transose dataframe
kable(AbsoluteEmo) %>%
  kable_styling(latex_options="scale_down")
```

	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
hXYoduN0kWs	3	7	6	1	9	2	4	8	5	10
basFyoMSjds	6	10	5	7	9	8	1	4	2	3
wBhhlE92mIQ	3	10	4	7	5	6	1	2	9	8
KxPgrGdSHh8	6	7	1	2	10	4	5	9	3	8
Rzd0mLf366I	8	9	1	6	5	2	3	4	7	10
X27mUWs2wjPs	9	7	2	4	6	8	3	5	10	1
Mf4wwXM2o_M	5	1	9	7	3	2	4	6	8	10
X7UEIBdMyj9g	7	9	3	10	2	5	8	4	6	1
uEjMyHccX8U	4	3	1	2	9	5	6	8	7	10
OmwE0aJqkdA	4	7	3	2	9	1	8	5	6	10

We then rank across the columns again to get an index of which videos most closely matched the Base videos emotion

```
AbsoluteEmo<-apply(AbsoluteEmo,2,rank) #rank columns to get rank of each emotion for all
AbsoluteEmo<-data.frame(AbsoluteEmo) #convert back to dataframe
kable(AbsoluteEmo) %>%
  kable_styling(latex_options="scale_down")
```

	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
hXYoduN0kWs	1.5	4.5	9.0	1	7.5	3.0	5.5	8.5	3.0	8.0
basFyoMSjds	6.5	9.5	8.0	8	7.5	9.5	1.5	3.0	1.0	3.0
wBhhlE92mIQ	1.5	9.5	7.0	8	3.5	8.0	1.5	1.0	9.0	4.5
KxPgrGdSHh8	6.5	4.5	2.0	3	10.0	5.0	7.0	10.0	2.0	4.5
Rzd0mLf366I	9.0	7.5	2.0	6	3.5	3.0	3.5	3.0	6.5	8.0
X27mUWs2wjPs	10.0	4.5	4.0	5	5.0	9.5	3.5	5.5	10.0	1.5
Mf4wwXM2o_M	5.0	1.0	10.0	8	2.0	3.0	5.5	7.0	8.0	8.0
X7UEIBdMyj9g	8.0	7.5	5.5	10	1.0	6.5	9.5	3.0	4.5	1.5
uEjMyHccX8U	3.5	2.0	2.0	3	7.5	6.5	8.0	8.5	6.5	8.0
OmwE0aJqkdA	3.5	4.5	5.5	3	7.5	1.0	9.5	5.5	4.5	8.0

We then want to to add the sum of each row as a new column.

```
AbsoluteEmo$sum<-rowSums(AbsoluteEmo) #add sum of rows
kable(t(AbsoluteEmo$sum))
```

51.5	57.5	53.5	54.5	52	58.5	57.5	57	55.5	52.5

next we add some more columns like the video titles and build a link to each item on the list.

```
AbsoluteEmo$title<-RelatedVids$title #add titles to emo df
AbsoluteEmo$link<-paste("https://www.youtube.com/watch?v=",
RelatedVids$rel_video_id, sep="") #build links to videos
```

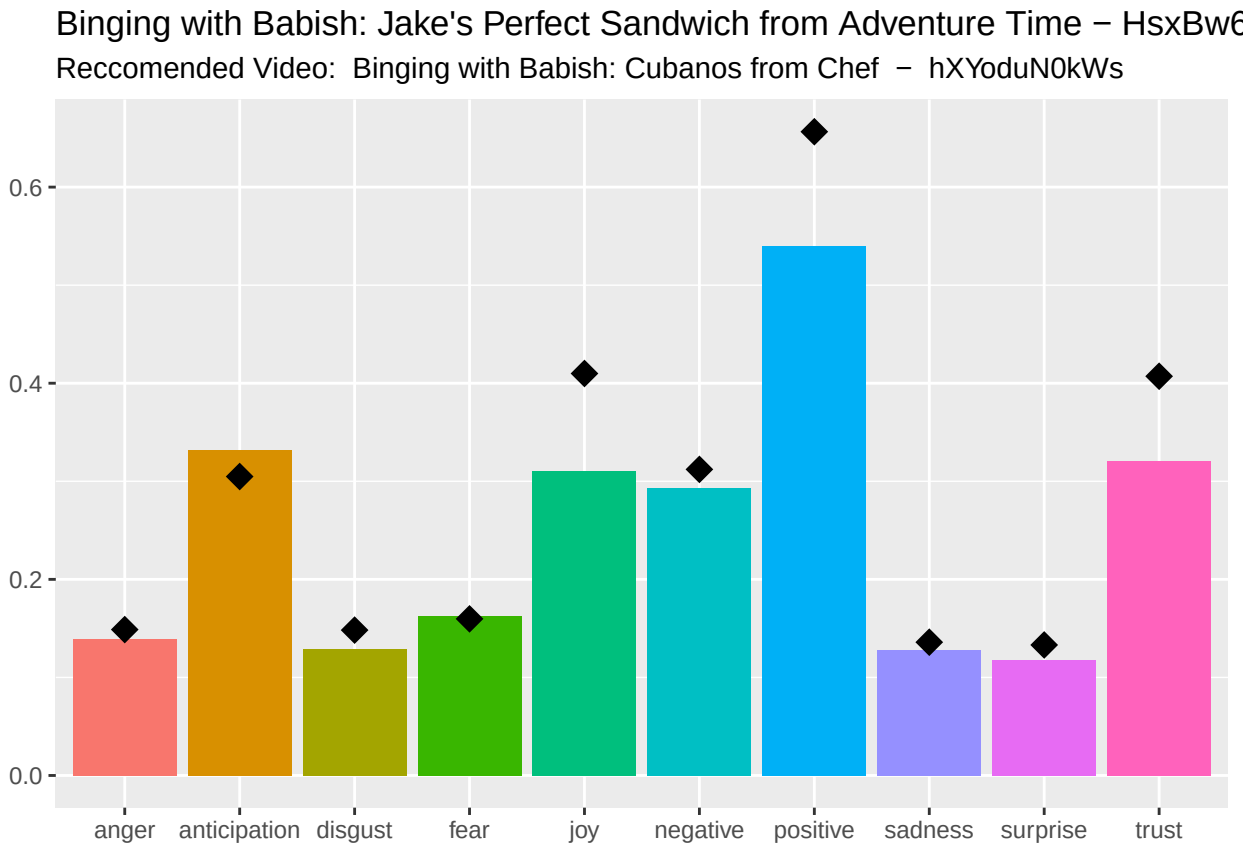
Finally we rank the row sums to build our final list.

```
RankedList<-data.frame(rank=rank(AbsoluteEmo$sum, ties.method ="first"),
title=AbsoluteEmo$title,link=AbsoluteEmo$link)
#generate a list of recommended videos and rank of best emotional fit
kable(RankedList) %>%
  kable_styling(latex_options="scale_down")
```

rank	title	link
1	Binging with Babish: Cubanos from Chef	https://www.youtube.com/watch?v=hXYoduN0kWs
8	Binging with Babish: Bob's Burgers	https://www.youtube.com/watch?v=basFyoMSjds
4	Binging with Babish: Rick & Morty Szechuan Sauce	https://www.youtube.com/watch?v=wBhhlE92mIQ
5	Binging with Babish: South Park Special	https://www.youtube.com/watch?v=KxPgrGdSHh8
2	Binging with Babish: Krabby Supreme from Spongebob Squarepants	https://www.youtube.com/watch?v=Rzd0mLf366I
10	Binging with Babish: Homer Simpson's Patented Space Age Out-Of-This-World Moon Waffles	https://www.youtube.com/watch?v=27mUWs2wjPs
9	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show	https://www.youtube.com/watch?v=Mf4wwXM2o_M
7	Binging with Babish: Louis C.K.'s Potluck Fried Chicken	https://www.youtube.com/watch?v=7UEIBdMyj9g
6	Binging with Babish: Goodfellas Prison Sauce	https://www.youtube.com/watch?v=uEjMyHccX8U
3	Binging with Babish: Big Kahuna Burger from Pulp Fiction	https://www.youtube.com/watch?v=OmwE0aJqkdA

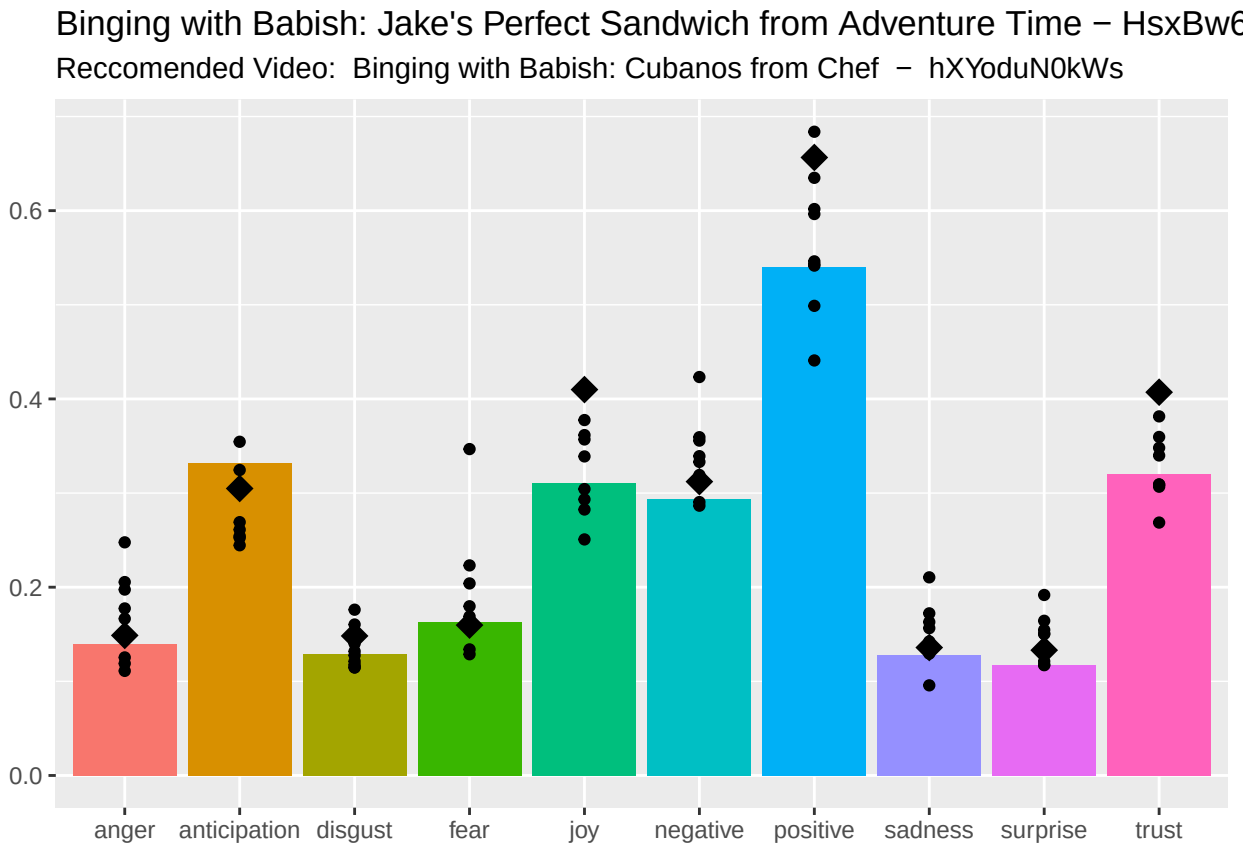
From this ranked list we may want to see how close the top reccomedation video is to our Base emmotional profile. The following code plots the best ranked reccomended vdeio emotion profile against our Base videos profile.

```
TopRank<-which.min(RankedList$rank)# get the topranked video index value
ClosestMatch<-RelatedEmoAvg[which.min(RankedList$rank),]
#pull emotional avg values from closest match
ClosestMatch<-data.frame(t(ClosestMatch)) #transpose df
colnames(ClosestMatch)<-"1" #set column name to set value for graphing
Recc<-Avg+geom_point(data=ClosestMatch,aes(shape=18, size=5, y=ClosestMatch$'1',
x=colnames(Baseavg)))+scale_shape_identity()+ggtitle(paste(BaseTitle,"-",
baseVid$videoId[1]),paste("Reccomended Video: ",
RelatedVids$title[TopRank]," - ",
RelatedVids$rel_video_id[TopRank]))
#add to ggplot to show how close best match video is
Recc
```



from this graph we can see that the reccomendation is quite close in several of the emotional categories. As a final grpahical display it might be interesting to plot all of the reccomendations to see what kind of variability there was.

```
RelatedEmoAvgTrans<-data.frame(t(RelatedEmoAvg)) # reshape df for graphing
seq<-c(1:length(RelatedVids$rel_video_id)) #set number of itterations
for (n in seq){ #add data points for each video to the graph to show spread.
  Recc<-Recc+geom_point(data=RelatedEmoAvgTrans,aes_(y=RelatedEmoAvgTrans[,n]))
}
rm(n,seq)
Recc #display final graph
```



Now the list is built it is saved as CSV in the working directory.

```
write.csv(RankedList,"RankedList.csv", row.names = FALSE) #output ranked list to csv
```

Results

We can see from the table below that our

```
kable(RankedList[order(RankedList$rank),]) %>%
  kable_styling(latex_options="scale_down")
```

	rank	title	link
1	1	Binging with Babish: Cubanos from Chef	https://www.youtube.com/watch?v=hXYoduN0kWs
5	2	Binging with Babish: Krabby Supreme from Spongebob Squarepants	https://www.youtube.com/watch?v=Rzd0mLf366I
10	3	Binging with Babish: Big Kahuna Burger from Pulp Fiction	https://www.youtube.com/watch?v=OmwE0aJqkdA
3	4	Binging with Babish: Rick & Morty Szechuan Sauce	https://www.youtube.com/watch?v=wBhhlE92mIQ
4	5	Binging with Babish: South Park Special	https://www.youtube.com/watch?v=KxPgrGdSHh8
9	6	Binging with Babish: Goodfellas Prison Sauce	https://www.youtube.com/watch?v=uEjMyHccX8U
8	7	Binging with Babish: Louis C.K.'s Potluck Fried Chicken	https://www.youtube.com/watch?v=7UEIBdMyj9g
2	8	Binging with Babish: Bob's Burgers	https://www.youtube.com/watch?v=basFyoMSjds
7	9	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show	https://www.youtube.com/watch?v=Mf4wwXM2o_M
6	10	Binging with Babish: Homer Simpson's Patented Space Age Out-Of-This-World Moon Waffles	https://www.youtube.com/watch?v=27mUWs2wjPs

```
kable(RelatedTable) %>%
  kable_styling(latex_options="scale_down")
```

rel_video_id	title	Comments
hXYoduN0kWs	Binging with Babish: Cubanos from Chef	3569
basFyoMSjds	Binging with Babish: Bob's Burgers	4779
wBhhlE92mIQ	Binging with Babish: Rick & Morty Szechuan Sauce	4450
KxPgrGdSHh8	Binging with Babish: South Park Special	4606
Rzd0mLf366I	Binging with Babish: Krabby Supreme from Spongebob Squarepants	9384
27mUWs2wjPs	Binging with Babish: Homer Simpson's Patented Space Age Out-Of-This-World Moon Waffles	3844
Mf4wwXM2o_M	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show	14987
KUu2gJn1dzc	How to Make New-York-Style Pizza - TMNT II: Secret of the Ooze	871
uEjMyHccX8U	Binging with Babish: Goodfellas Prison Sauce	1591
OmwE0aJqkdA	Binging with Babish: Big Kahuna Burger from Pulp Fiction	1468

Conclusion