Using Sentiment Analysis to Improve YouTube Recommended Video Rankings

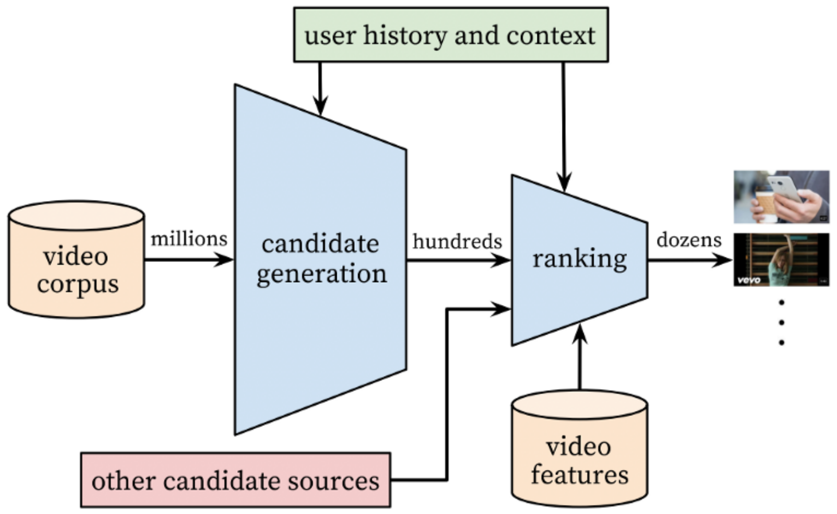
Mark Richards

5/10/2019

GitHub Repository

Introduction

This report will cover a method for improving YouTube recommendations based on natural language processing and sentiment analysis. The goal is to improve the ranking of recommended videos by evaluating the emotions that a video evokes in those that comment on it against the emotions expressed in the comments of a list of recommended videos. It is not meant to be a replacement for the deep neural network that Google uses to build its recommendations. It is simply meant to demonstrate the a potential for incorporating sentiment analysis as part of the ranking system for serving recommendations to users.
This project requires some background knowledge of YouTube and how it generates its recommendations. In 2016 Google released a paper titled "Deep Neural Networks for YouTube Recommendations" where it outlines its use of two deep neural nets to build and rank recommendations. This paper covers the processes they use in great detail unfortunately 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 Google 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 neural net incorporates additional features such as preview image and peer interest and uses that to order the list of recommendations (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 engagement 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 engagement. It may be that adding sentiment analysis could help improve engagement again by connecting users with content that matches emotional tones.

Analysis

Note: To properly run the code in this analysis an API key is required. Instructions on how to acquire an API key can be found here: "YouTube Data API Overview". This file will not be provided on the GitHub Repository for this paper. A copy of this script for those who wish to set up their own API key is available on my GitHub as SentimentRanker_wKey.R All of the API calls rely on the TubeR package to handle the Oauth token.
There is also an issue of reproducibility. The suggestions served to me in this report will almost certainly not be reproducible by anyone else. They are generated by the aforementioned deep neural nets and as such are tailored to what YouTube believes would best maximize my watch time. To counter this issue for the report I have created RDS files of any data that needs to be pulled from YouTube. The code for this report and the accompanying R script will call these RDS files rather than making the API calls. To examine how the API calls work see the script provided for those with keys.

Before running the script make sure you have the following RDS files in your working directory.

1. RelatedVids.rds
2. BaseVid.rds
3. BaseDetail.rds
4. RelatedComm.rds

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

```r
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 would check to make sure you have an API key installed. If you did not it would display a URL where you could learn how to set one up. If it did find a key RDS it would load the file and setup the Oauth token. It would then direct you to a website to authorize the app to access your account. Since you are not expected to set up an API key this code is only shown as an example.

```r
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 glossed over, it's time to pick a video that we want recommendations for. For this report I will be using a video from the cooking channel Binging With Babish where he attempts to recreate Jake's Perfect Sandwich from Adventure Time. The following code is where the URL for the video we like is entered. This is referred to as the "Base" video from this point on.

```
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 expression 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
```

With the Video ID isolated we can begin making calls to the YouTube API and pulling data down for analysis. Again, the code is shown as an example but the RelatedVids.rds will be loaded with the results of the API calls.

```
RelatedVids<- get_related_videos(video_id =Baselink, max_results = 11) #pull recomended
```

```
RelatedVids<- readRDS("RelatedVids.rds") #Load RDS with API results
```

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
hzAgFNh4vRY	Binging with Babish: Good Morning Burger from The Simpsons
npC5RTV7uBw	Chain Restaurant Steak Taste Test
KxPgrGdSHh8	Binging with Babish: South Park Special
wjoeVwJTUTA	Binging with Babish: The Garbage Plate from The Place Beyond The Pines (sort of)
sJtl5OFAUAA	How to cook a GOKU FEAST
Mf4wwXM2o_M	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show
MP_nWuLYpJw	Binging with Babish: Parks & Rec Burger Cookoff
KUu2gJn1dzc	How to Make New-York-Style Pizza - TMNT II: Secret of the Ooze

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 `5446` 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.

```r
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
4798	Muselk	So happy to see this channel finally taking off.	5980
2204	Barry Smith	Now you gotta let Magic Man steal it and waste 15 minutes trying to get it back	4379
4308	Geral Ferald	Now do the everything burrito!	3776
3289	IanIsTheHero	You forgot to wash a carrot and then not use it, but a+ 10/10 as usual	3590
3119	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...	3148
5402	Thund3rman80	is it possible to make a krabby pattie, from spongebob SquarePants	2955
2186	Brendan	Jesus you could feed a whole family with one of those	2698
5421	Kevin Lewis	Babish, your kitchen is the stuff of dreams.	2326
2125	yotam lichtig	I want to sleep in a bed made of your voice	2226
5428	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.	1628

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.

```r
#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	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	2
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	2	1	2
0	1	0	0	1	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0
0	0	1	0	1	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1

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.

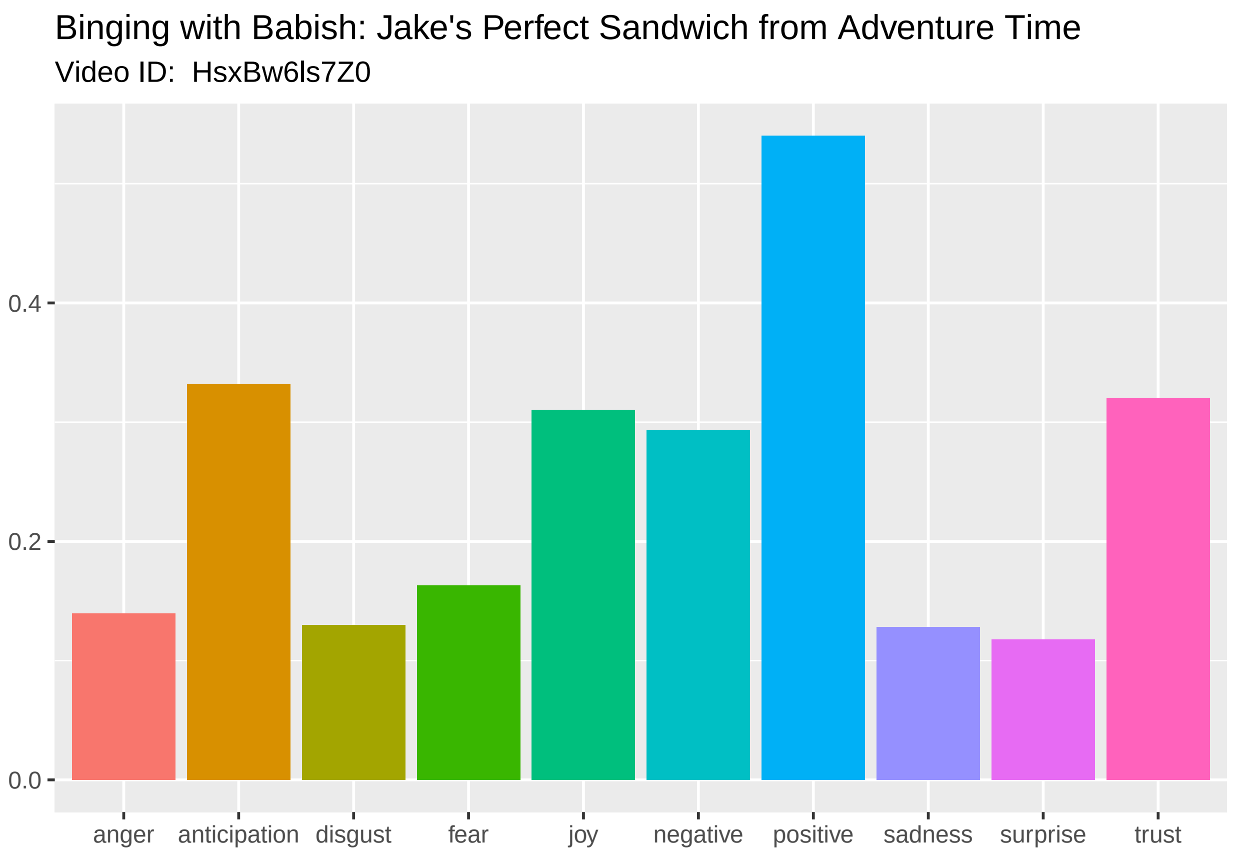
```r
#create averages of sentiment and emotion for base video
Baseavg<-apply(BaseEmo, 2, mean)
#convert averages to data frame for plotting
```

```r
Baseavg<-data.frame(name=Baseavg)
kable(Baseavg)
```

	name
anger	0.1395520
anticipation	0.3316195
disgust	0.1298201
fear	0.1628718
joy	0.3103195
sadness	0.1281675
surprise	0.1175174
trust	0.3198678
negative	0.2936100
positive	0.5402130

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.

```r
#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.

```r
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)
}
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	3571
basFyoMSjds	Binging with Babish: Bob's Burgers	4789
hzAgFNh4vRY	Binging with Babish: Good Morning Burger from The Simpsons	3292
npC5RTV7uBw	Chain Restaurant Steak Taste Test	3520
KxPgrGdSHh8	Binging with Babish: South Park Special	4606
wjoeVwJTUTA	Binging with Babish: The Garbage Plate from The Place Beyond The Pines (sort of)	5627
sJtl5OFAUAA	How to cook a GOKU FEAST	17476
Mf4wwXM2o_M	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show	15025
MP_nWuLYpJw	Binging with Babish: Parks & Rec Burger Cookoff	1953
KUu2gJn1dzc	How to Make New-York-Style Pizza - TMNT II: Secret of the Ooze	2617

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.

```r
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.

```r
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.

```r
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	0.1484178	0.3052366	0.1472977	0.1596192	0.4096892	0.1358163	0.1330160	0.4066088	0.3113974	0.6561187
basFyoMSjds	0.1110879	0.2699937	0.1144289	0.1342660	0.3624974	0.1632909	0.1179787	0.3102944	0.2904573	0.5464606
hzAgFNh4vRY	0.1436817	0.3174362	0.1427704	0.1755772	0.3456865	0.1567436	0.2065614	0.3544957	0.3411300	0.5880923
npC5RTV7uBw	0.1053977	0.2741477	0.1019886	0.1093750	0.2761364	0.1068182	0.1636364	0.3048295	0.2426136	0.4750000
KxPgrGdSHh8	0.1183239	0.3052540	0.1265740	0.1597916	0.3775510	0.1382979	0.1372123	0.3805905	0.2854972	0.5959618
wjoeVwJTUTA	0.1547894	0.3065577	0.2269415	0.1487471	0.4039453	0.1183579	0.1663409	0.3328594	0.3609383	0.6326639
sJtl5OFAUAA	0.1742962	0.2496567	0.1681735	0.1684024	0.2989242	0.1112955	0.1112383	0.2886816	0.3582055	0.5574502
Mf4wwXM2o_M	0.1774376	0.3242596	0.1759068	0.2039268	0.3389018	0.1399667	0.1502829	0.3599334	0.3393012	0.6015308
MP_nWuLYpJw	0.1039427	0.2524322	0.0947261	0.1213518	0.3277010	0.1080389	0.1121352	0.2974910	0.2232463	0.5217614
KUu2gJn1dzc	0.1669851	0.2839129	0.1532289	0.1474971	0.3297669	0.1559037	0.1402369	0.3381735	0.3695071	0.5831104

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	0.0088658	0.0263829	0.0174776	0.0032527	0.0993697	0.0076488	0.0154985	0.0867410	0.0177874	0.1159057
basFyoMSjds	0.0284641	0.0616258	0.0153912	0.0286058	0.0521779	0.0351234	0.0004613	0.0095734	0.0031527	0.0062476
hzAgFNh4vRY	0.0041297	0.0141833	0.0129503	0.0127053	0.0353670	0.0285762	0.0890439	0.0346280	0.0475200	0.0478793
npC5RTV7uBw	0.0341542	0.0574718	0.0278314	0.0534968	0.0341831	0.0213493	0.0461189	0.0150382	0.0509964	0.0652130
KxPgrGdSHh8	0.0212280	0.0263655	0.0032460	0.0030803	0.0672315	0.0101304	0.0196949	0.0607227	0.0081128	0.0557488
wjoeVwJTUTA	0.0152374	0.0250619	0.0971215	0.0141247	0.0936258	0.0098095	0.0488234	0.0129916	0.0673283	0.0924509
sJtl5OFAUAA	0.0347442	0.0819629	0.0383534	0.0055305	0.0113953	0.0168720	0.0062792	0.0311862	0.0645956	0.0172372
Mf4wwXM2o_M	0.0378856	0.0073600	0.0460868	0.0410550	0.0285823	0.0117993	0.0327654	0.0400657	0.0456912	0.0613178
MP_nWuLYpJw	0.0356093	0.0791874	0.0350940	0.0415201	0.0173815	0.0201285	0.0053823	0.0223768	0.0703637	0.0184516
KUu2gJn1dzc	0.0274331	0.0477067	0.0234088	0.0153747	0.0194474	0.0277362	0.0227195	0.0183057	0.0758971	0.0428974

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	hzAgFNh4vRY	npC5RTV7uBw	KxPgrGdSHh8	wjoeVwJTUTA	sJtl5OFAUAA	Mf4wwXM2o_M	MP_nWuLYpJw	KUu2gJn1dzc
anger	3	6	1	4	6	4	7	5	7	6
anticipation	7	10	4	9	7	5	10	1	10	9
disgust	5	5	3	3	2	10	8	9	6	5
fear	1	7	2	8	1	3	1	7	8	1
joy	9	9	7	5	10	9	3	3	2	3
sadness	2	8	5	2	4	1	4	2	4	7
surprise	4	1	10	6	5	6	2	4	1	4
trust	8	4	6	1	9	2	6	6	5	2
negative	6	2	8	7	3	7	9	8	9	10
positive	10	3	9	10	8	8	5	10	3	8

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

```r
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	5	1	9	2	4	8	6	10
basFyoMSjds	6	10	5	7	9	8	1	4	2	3
hzAgFNh4vRY	1	4	3	2	7	5	10	6	8	9
npC5RTV7uBw	4	9	3	8	5	2	6	1	7	10
KxPgrGdSHh8	6	7	2	1	10	4	5	9	3	8
wjoeVwJTUTA	4	5	10	3	9	1	6	2	7	8
sJtl5OFAUAA	7	10	8	1	3	4	2	6	9	5
Mf4wwXM2o_M	5	1	9	7	3	2	4	6	8	10
MP_nWuLYpJw	7	10	6	8	2	4	1	5	9	3
KUu2gJn1dzc	6	9	5	1	3	7	4	2	10	8

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

```r
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	2.0	4.5	5.0	2.5	8	3	5.0	9.0	3.0	9.0
basFyoMSjds	7.0	9.0	5.0	7.5	8	10	1.5	4.0	1.0	1.5
hzAgFNh4vRY	1.0	2.0	2.5	5.0	6	8	10.0	7.0	6.5	7.0
npC5RTV7uBw	3.5	6.5	2.5	9.5	5	3	8.5	1.0	4.5	9.0
KxPgrGdSHh8	7.0	4.5	1.0	2.5	10	6	7.0	10.0	2.0	5.0
wjoeVwJTUTA	3.5	3.0	10.0	6.0	8	1	8.5	2.5	4.5	5.0
sJtl5OFAUAA	9.5	9.0	8.0	2.5	3	6	3.0	7.0	8.5	3.0
Mf4wwXM2o_M	5.0	1.0	9.0	7.5	3	3	5.0	7.0	6.5	9.0
MP_nWuLYpJw	9.5	9.0	7.0	9.5	1	6	1.5	5.0	8.5	1.5
KUu2gJn1dzc	7.0	6.5	5.0	2.5	3	9	5.0	2.5	10.0	5.0

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

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

51	54.5	55	53	55	52	59.5	56	58.5	55.5

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

```r
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
```

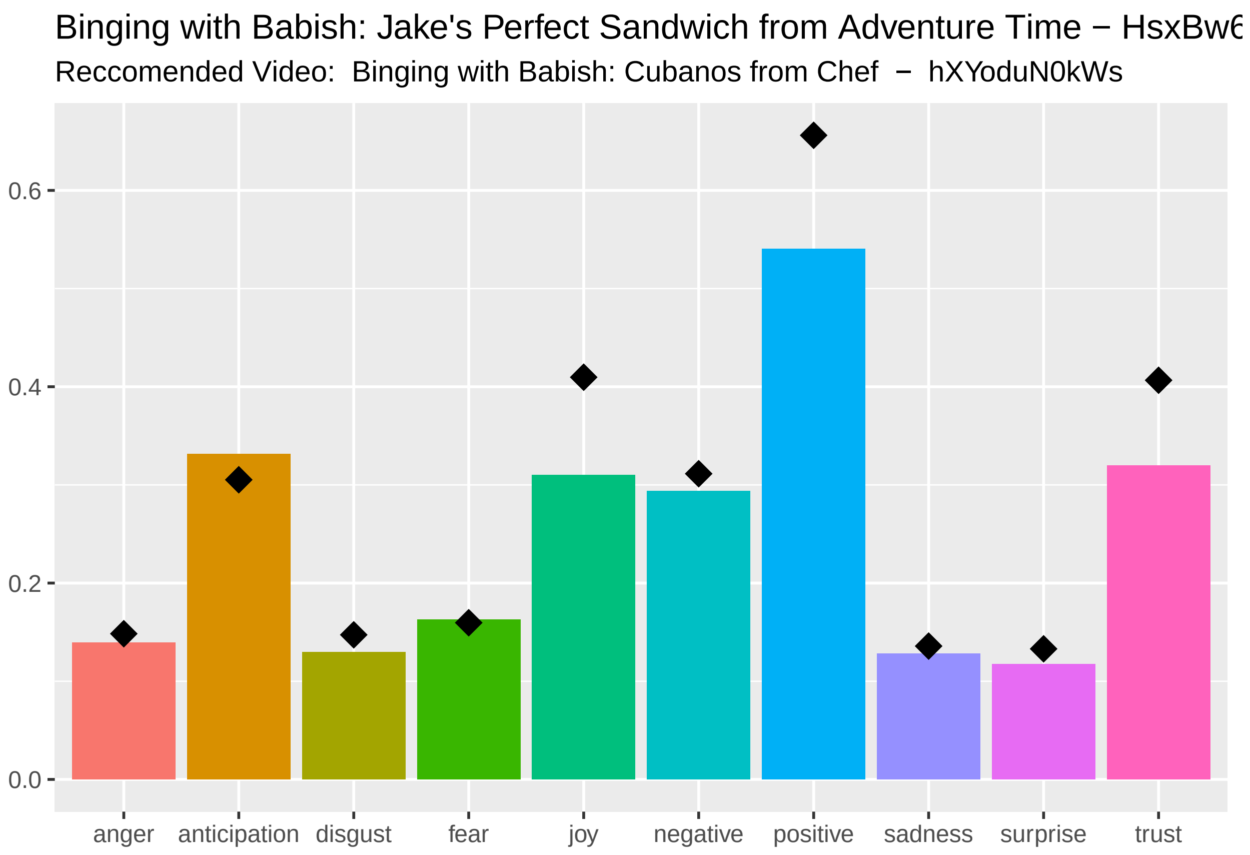
Finally we rank the row sums to build our final list.

```r
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
4	Binging with Babish: Bob's Burgers	https://www.youtube.com/watch?v=basFyoMSjds
5	Binging with Babish: Good Morning Burger from The Simpsons	https://www.youtube.com/watch?v=hzAgFNh4vRY
3	Chain Restaurant Steak Taste Test	https://www.youtube.com/watch?v=npC5RTV7uBw
6	Binging with Babish: South Park Special	https://www.youtube.com/watch?v=KxPgrGdSHh8
2	Binging with Babish: The Garbage Plate from The Place Beyond The Pines (sort of)	https://www.youtube.com/watch?v=wjoeVwJTUTA
10	How to cook a GOKU FEAST	https://www.youtube.com/watch?v=sJtl5OFAUAA
8	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show	https://www.youtube.com/watch?v=Mf4wwXM2o_M
9	Binging with Babish: Parks & Rec Burger Cookoff	https://www.youtube.com/watch?v=MP_nWuLYpJw
7	How to Make New-York-Style Pizza - TMNT II: Secret of the Ooze	https://www.youtube.com/watch?v=KUu2gJn1dzc

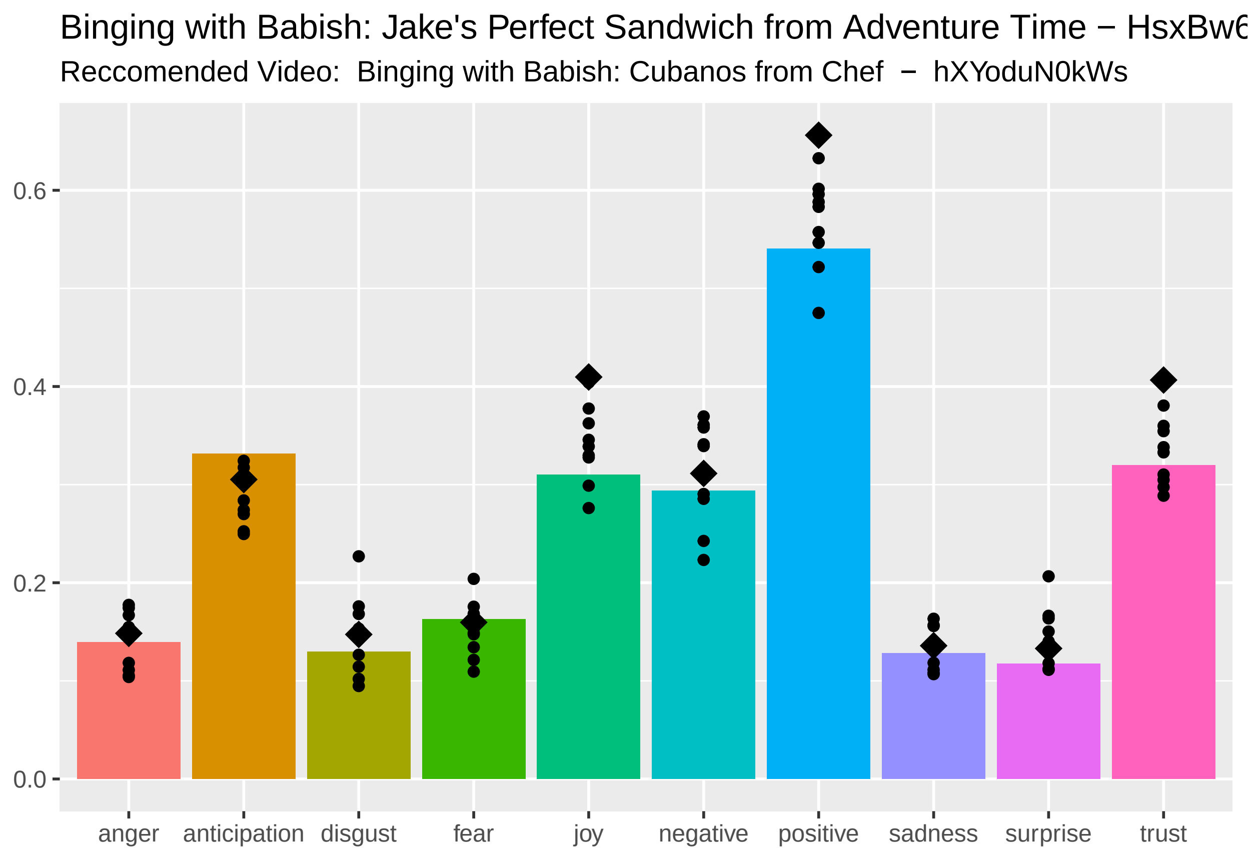
From this ranked list we may want to see how close the top reccomedation video is to our Base emmotional profile. The following code plots the best ranked reccomended vdeio emotion profile against our Base videos profile.

```r
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.

```r
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.

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

Results

In the table below we can see that we were able to adjust the rankings that YouTube provided (first column). This means it could be a possible avenue for inclusing in the ranknig system that YouTube uses to reccomend videos to users.

```r
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
6	2	Binging with Babish: The Garbage Plate from The Place Beyond The Pines (sort of)	https://www.youtube.com/watch?v=wjoeVwJTUTA
4	3	Chain Restaurant Steak Taste Test	https://www.youtube.com/watch?v=npC5RTV7uBw
2	4	Binging with Babish: Bob's Burgers	https://www.youtube.com/watch?v=basFyoMSjds
3	5	Binging with Babish: Good Morning Burger from The Simpsons	https://www.youtube.com/watch?v=hzAgFNh4vRY
5	6	Binging with Babish: South Park Special	https://www.youtube.com/watch?v=KxPgrGdSHh8
10	7	How to Make New-York-Style Pizza - TMNT II: Secret of the Ooze	https://www.youtube.com/watch?v=KUu2gJn1dzc
8	8	Binging with Babish 2 Million Subscriber Special: The Every-Meat Burrito from Regular Show	https://www.youtube.com/watch?v=Mf4wwXM2o_M
9	9	Binging with Babish: Parks & Rec Burger Cookoff	https://www.youtube.com/watch?v=MP_nWuLYpJw
7	10	How to cook a GOKU FEAST	https://www.youtube.com/watch?v=sJtl5OFAUAA

Conclusion

We have found that there is room to improve the reccomendations for videos served by YouTube with the additon of natural language processing to match emotional tones across suggestions. It also raises the possibiltiy of using these methods to build sentiment profiles of users. Combine the sentinments of every

comment they leave. As well as for every video they like. Find what emotions they are drawn to and feed that back to them.
There is a consideration that shold be made for perfomance. This paper is a proof of concept and woudl nto be the most efficent way to impliment this type fo ranking. It would make more sense for YouTube to build its own lexicon that is specific to its comments. This would allow them to get more accurate results reguarding expression of emotions and valance of comments.
They shoudl also consider running the sentiment analysis across all comments currently available and going forward when a user submits a comment or edits one it does the sentiment processign then and simpyl saves the numeric results. These logicals could then be quickly processed when building reccomendations.
As a further step when auto translation becomes more robust sentiment analysis could be run across the video transcripts to better match content emotion to a users preference. All of these steps may help improve YouTubes core metric of watch time. But barring prooper A/B testing which is outside of the scope of this paper the exact impact is unknonwn.