Research Internship

Developing a Valid Political Attitude Measurement in the Context of Twitter Analytics

Néstor Narbona Chulvi

June, 2022

1 What the thesis was about and what issues it had

The bachelor thesis was entitled Using *Social Media Analytics to Explore Political Involvement and Polarization: Challenges and New Insights*. The goal was to test a particular prediction steaming from the Cusp Catastrophe model of attitudes, and to do so in the context of political attitudes on Twitter. More specifically, we wanted to test if involvement in an issues made moderate attitudes towards that issue inaccessible. At the population level, that means that the political attitude distribution becomes more extremely bimodal as a function of political involvement (see Figure 1). In order to test this an R package called *cusp* and developed by Han van der Maas, Raoul Grasman, and Eric-Jan Wagenmakers was going to be used. Nevertheless, before data could be used to test this hypothesis, the data itself had to be collected. This is were most of the thesis efforts went in.

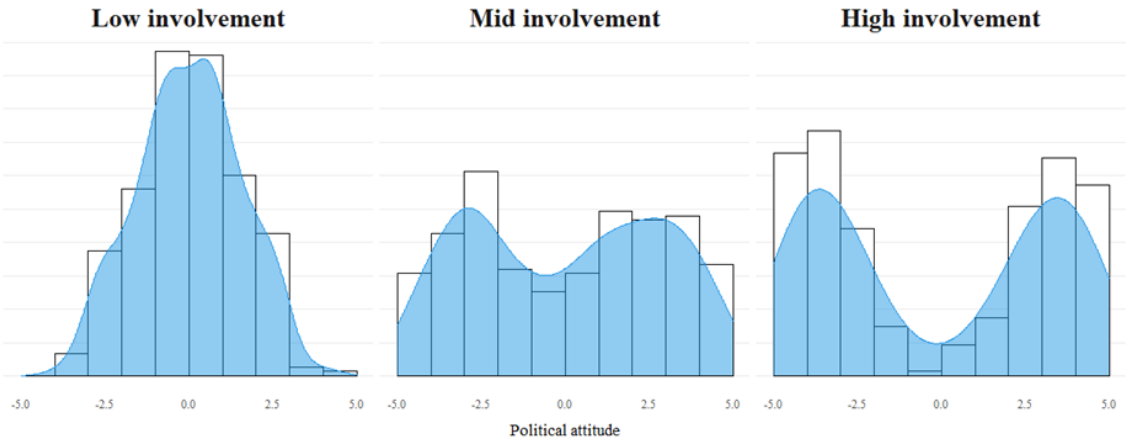

Figure 1: Expected political attitudes distribution at different levels of political involvement

The data included two variables: the user's political involvement and the user's political attitude. The former was measured by assessing how many of the a user's last 100 posts were political (i.e. included a word that Oxford Dictionary considered political). The latter was measured by performing a dictionary based sentiment analysis on users' posts that mentioned only Joe Biden and no other political figure. Each user had ¿3 posts of this kind, thus the political attitude score was computed by calculating the average sentiment of their posts.

It was with the political attitude measurement that issues arise. The sentiment scores ranged

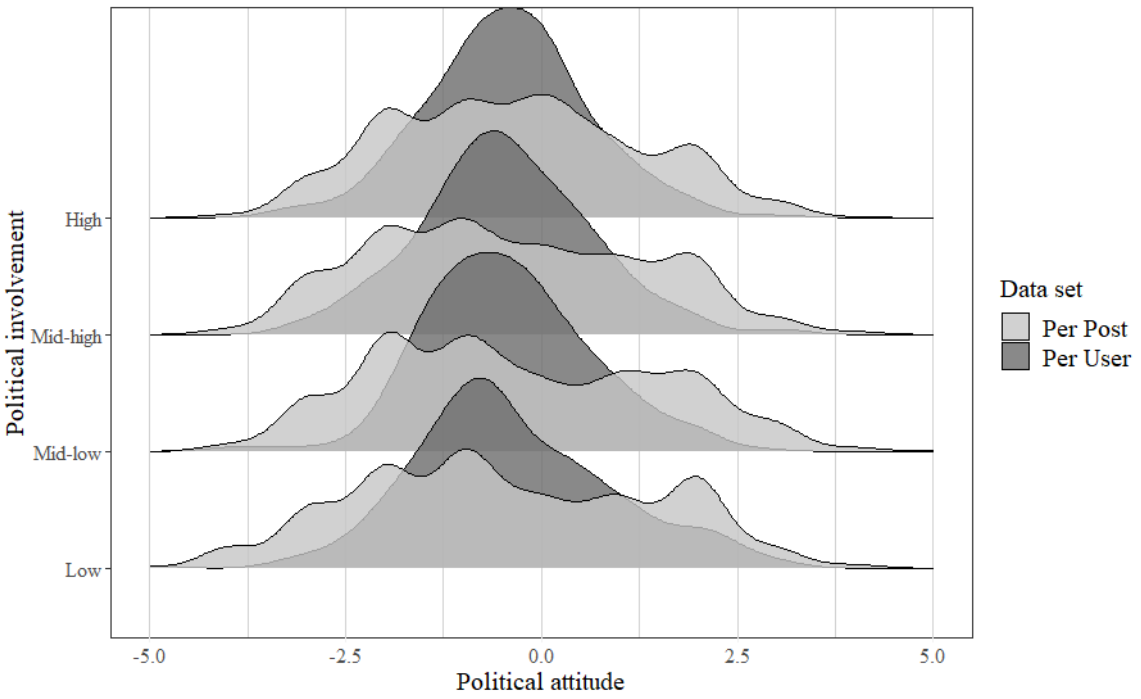

Figure 2: Political attitudes distribution per user and per post at different political involvement levels

from -5 to 5. Because the analysis was only performed on posts that mentioned only Joe Biden, it was assumed that whatever sentiment the user expressed in the post it would be directed at Joe Biden. Given that we did not expect people to have both positive and negative attitudes towards the same political figure, we also did not expect one same user to have both positive and negative sentiment scores in his posts. This assumptions were proven wrong. As it can be seen in Figure 2, the political attitudes distribution shows a great spread when one plots posts independently (i.e. lighter curve), and this spread disappears one the plotted data is per user (i.e. darker curve). It seems to be the case that many users had bod extremely positive and extremely negative posts. Thus when the political attitude score was computed based on the mean of this values, they would average out and output a moderate value. This clearly cast doubts on the validity of the measurement. How can we consider someone to have a moderate political attitude, if he is only expressing extremely negative or positive sentiments?

2 How we planned to fix this issue

There where other issues and possible improvements with the design (e.g. filtering out bots and institutional accounts would improve the analysis). Nevertheless, given the available time we focused on the averaging out of attitudes that was affecting the political attitudes measurement. After careful consideration of different possible solutions, we decided that the best approach was separating the polarity and strength measurements of the attitude. This is, using one independent method to assess the direction of a user's attitude (e.g. pro-Biden or anti-Biden) and then use a different method to asses the strength of this attitude. The reason why this fixed the problem is that now sentiment

scores could be used in their absolute value (i.e. from 0 to 5), and then averaged per user to obtain a measure of the strength of their attitude. For instance, if a user had two extremely positive and one extremely negative posts the measure of his attitude strength would be *extremely positive* (i.e. ≈ 5). In short, there is no averaging out of extreme attitudes using this method.

The second aspect of this new methodology was finding an appropriate measure for the polarity of the attitude. We found the answer to this in the Twitter Analytics literature involving stance detecting (i.e. different methods aimed at assessing whether a person is in favor or against a certain topic). These procedures usually assume that a person has one and only one stance, and use different indicators to predict it. After some evaluation, we chose a machine learning approach that used followers and followings of a user as indicators. More specifically we trained a Support Vector Machine (SVM) with a linear kernel to predict whether an account was a Hillary Clinton supporter or critic. This politician was chosen because the only available annotated data set had annotated ≈ 700 users' attitude towards her. The nature of this method lead to a predictive algorithm with ≈ 10000 predictors which would clearly lead to over-fitting. To attenuate this problem a Recurrent Feature Elimination was used in order to select the best performing **N** predictors, and these were later used.

Overall, the new methodology only changed the political attitudes measure and thus the general design of the thesis could be used again. First, a new sample of posts would be collected, this time referring to Hillary Clinton instead of Joe Biden. Then, their political involvement would be measured using the same method as had been used before. Thirdly, the strength of their attitudes would be measured using sentiment analysis and the polarity would be assessed using the newly trained SVM algorithm. This would result in two variables (i.e. political attitude and political involvement) that would be used to fit the Cusp model again in order to test our original prediction.

3 What the results were

On one hand the trained SVM performed well. Using the best 150 performing predictors, it showed a sensitivity of 1 and an specificity of 0.82. Although there may be some over-fitting, the recurrent feature elimination selected these 150 predictors based on their performance on a 5-fold cross-validation. Given that this model seemed to be effective, we moved on to gathering 11216 posts including the word 'Clinton'. There where 9468 unique users and of these only those with three or more posts were selected, leaving a sample of 299 unique users. Of these Twitter accounts the political involvement and emotionality was measured in a similar fashion as in the original thesis project. Then, their

Model	AIC	BIC
Linear	1127.74	1138.47
Restricted	439.63	457. 51
Unrestricted	440.49	461.94

Table 1: Fit measures for the linear, restricted and unrestrictied models

followers and followings were accessed to be used as predictors for their political stance through the trained SVM model. For 268 users it was possible to measure all political involvement, emotionality and political stance variables. After this had been done, the emotionality measure was multiplied

by the political stance prediction (i.e. -1 or 1), to obtain the variables political attitude. Finally the cusp catastrophe model was fitted using the R package *cusp*. As it can be seen in Table 1, the restricted cusp model outperformed the linear model in both the AIC and BIC measures considerably. Nevertheless, it barely showed any improvement when compared to the unrestricted model. As it can be seen in Figure 3, this mediocre fit of the Cusp Catastrophe model is due to an equally extremely bimodal political attitude distribution at different political involvement levels. This could be because even the least involved users were in fact very involved as they had posted about Hillary Clinton at least three times in the two days data collection took place. It could also be the case that the Cusp Catastrophe prediction does not apply in this context.

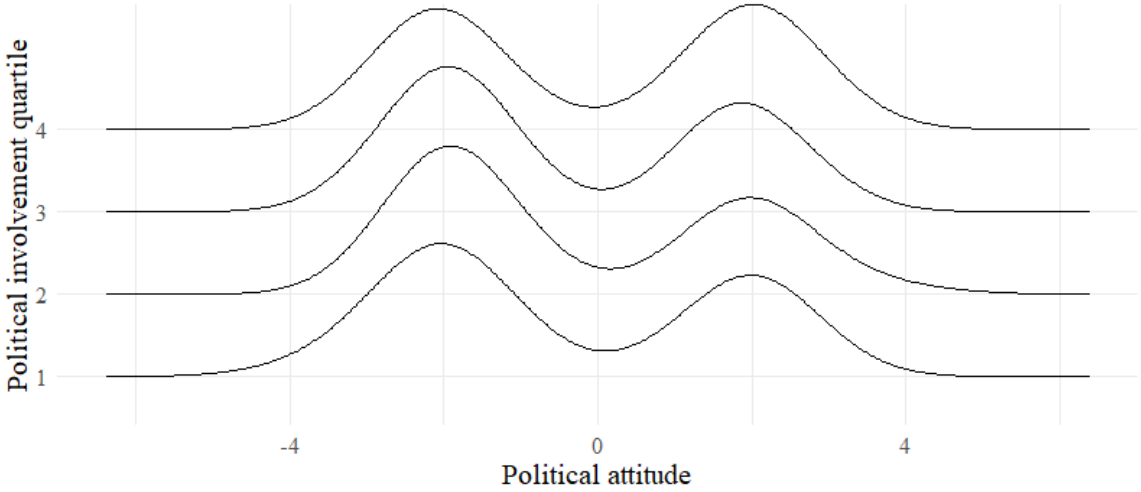

Figure 3: Political attitudes distribution per user and per post at different political involvement levels

Overall, the research internship has meant an important improvement over the original bachelor thesis research design. The main concern we had (i.e. averaging out extreme attitudes in our political attitudes measurement) was indeed fixed with the newly designed measure. Furthermore, I learned: how to do an effective literature review in the field of social media analytics and machine learning, how to apply a Support Vector Machine to my data and how to reduce the problem of over-fitting through Recurrent Feature Elimination. Although we did not find the pattern we expected (i.e. bimodality extremity did not increase as a function of political involvement), the improvements in the design as well as everything we learned during the process made the project fruitful and enjoyable.

Appendix 1: Support Vector Machine code

```
library(rtweet); library(tidyverse); library(tidyft);library(caret)
#semval data set


testset <- read.csv('testset.csv')
trainingset <- read.csv('trainingset.csv')

#select Hillary Clinton tweets and combine data set
hillary <- testset %>%
  dplyr::add_row(trainingset) %>%
  dplyr::filter(Target == 'Hillary Clinton')

#delete 'NONE' cases
#because default SVM predicts 2 categories only
hillary <- hillary%>%
  filter(Stance!='NONE')

#solve repeated users problem

#list with only users that have 1 post
hillary_unique <- hillary%>%
  group_by(UserID)%>%
  mutate(number = n())%>%
  ungroup()%>%
  filter(number == 1)

#keep those users that have one stance only
duplicates <- hillary%>%
  group_by(UserID)%>%
  mutate(number = n())%>%
  ungroup()%>%
  filter(number>2)

dup_IDs <- unique(duplicates$UserID)

dups_OK <- c()
for (i in dup_IDs){
  stances <- duplicates%>%
    filter(UserID == i)%>%
    select(Stance)
  if(length(unique(stances$Stance)) == 1){
    dups_OK <- append(dups_OK,i)
  }
}

to_add <- duplicates%>%
  filter(UserID %in% dups_OK)%>%
  distinct(UserID,.keep_all = TRUE)

hillary_final <- hillary_unique%>%
  add_case(to_add)
```

```r
#Connection Network: obtain users' followers and friends(followings)
IDs <- hillary_final$UserID

#get followers and friends
FL <- list()
FR <- list()
for (i in 1:length(IDs)){
  rate_fl <- rate_limit(token = NULL, query = get_followers, parse = TRUE)$remaining
  rate_fr <- rate_limit(token = NULL, query = get_friends, parse = TRUE)$remaining

  while(rate_fl == 0 | rate_fr == 0){
    Sys.sleep(120) #if rate limit is reached for followers or friends stop for 2 min
    rate_fl <- rate_limit(token = NULL, query = get_followers, parse = TRUE)$
    remaining
    rate_fr <- rate_limit(token = NULL, query = get_friends, parse = TRUE)$remaining
  }

  FL[[i]] <- get_followers(IDs[i])
  FR[[i]] <- get_friends(IDs[i])
}

#Create list of all unique accounts a user may have as follower and another for
    friends

#delete counts that are not followed/follow many people:irrelevant and problematic
    computationally
#this also leaves a unique instance of each count

freq_FR <- as.data.frame(table(unlist(FR)))
pos_FR <- freq_FR%>%
  dplyr::filter(Freq > 10)
pos_FR <- pos_FR$Var1
pos_FR <- as.character(pos_FR)

freq_FL <- as.data.frame(table(unlist(FL)))
pos_FL <- freq_FL%>%
  dplyr::filter(Freq > 10)
pos_FL <- pos_FL$Var1
pos_FL <- as.character(pos_FL)


#Training data set

  #Generate empty data set with ID and label
train_df <-  hillary_final %>%
  dplyr::select(ID = UserID,Stance)

  #Fill in info about friends
pos_FR_df <- as.data.frame(matrix(IDs,length(IDs),length(pos_FR)+1))
colnames(pos_FR_df) <- c('ID',pos_FR)

for (i in 1:length(IDs)){
  friends <- FR[[i]]$user_id
  for (fr in pos_FR){
```

```
    if (fr %in% friends){
      pos_FR_df[i,fr] = T
    } else {
      pos_FR_df[i,fr] = F
    }
  }
}
pos_FR_names <- c()   #new column names to differentiate following and being followed
     by a user

for (i in 1:length(pos_FR)){
  pos_FR_names[i] <- paste('FR_',pos_FR[i],sep = '')
}
colnames(pos_FR_df) <- c('ID',pos_FR_names)

train_df<- inner_join(train_df,pos_FR_df,by = 'ID')

  #Fill in info about followers
pos_FL_df <- as.data.frame(matrix(IDs,length(IDs),length(pos_FL)+1))
colnames(pos_FL_df) <- c('ID',pos_FL)

for (i in 1:length(IDs)){
  followers <- ifelse(!is.na(FL[[i]]$user_id), FL[[i]]$user_id, 0)
  for (fl in pos_FL){
    if (fl %in% followers){
      pos_FL_df[i,fl] = T
    } else {
      pos_FL_df[i,fl] = F
    }
  }
}

pos_FL_names <- c()   #new column names to differentiate following and being followed
     by a user
for (i in 1:length(pos_FL)){
  pos_FL_names[i] <- paste('FL_',pos_FL[i],sep = '')
}
colnames(pos_FL_df) <- c('ID',pos_FL_names)

train_df <- inner_join(train_df,pos_FL_df,by = 'ID')


###SVM##

#get rid of ID to make code easier
train_df_bin <- train_df%>%
  dplyr::select(-ID)

#ceck for NA values
anyNA(train_df_bin)

#make predicted variables numerical
train_df_bin$Stance <- ifelse(train_df_bin$Stance == 'AGAINST',0,1)
```

```r
#transform categorical variables into factors
for (i in 1:ncol(train_df_bin)){
  train_df_bin[,i] <- factor(train_df_bin[,i])
}

#delete variables with only one levelv (otherwise error appears)
delete = c()
for (i in 1:ncol(train_df_bin)){
  if(length(levels(train_df_bin[,i])) == 1){
    delete <- append(delete,i)
  }
}
train_df_bin <- train_df_bin[,-delete]


#do SVM using caret package
trctrl <- caret::trainControl(method = 'repeatedcv',number = 10,repeats = 3)
svm_linear <- caret::train(Stance ~., data = train_df_bin,method = 'svmLinear',
                           trControl = trctrl, preProcess = c('center','scale'),
                           tuneLength = 10)

preds <- predict(svm_linear,newdata = train_df_bin)

confusionMatrix(table(preds,train_df_bin$Stance))


#do RFE using caret package

#i use random forest because i cannot find the SVM function
library(randomForest)
rfecntrl <- rfeControl(functions = rfFuncs,method = 'repeatedcv', repeats = 5)

rfe_trial <- rfe(train_df_bin[,-1],train_df_bin[,1],size = 30, rfeControl = rfecntrl)

#run SVM again with best 30 predictors
best_pred <- rfe_trial$optVariables[1:150]

new_preds <- train_df_bin%>%
  dplyr::select('Stance',all_of(best_pred))

trctrl <- caret::trainControl(method = 'repeatedcv',number = 10,repeats = 3)
new_svm_linear <- caret::train(Stance ~., data = new_preds,method = 'svmLinear',
                           trControl = trctrl, preProcess = c('center','scale'),
                           tuneLength = 10)
new_predidictions <- predict(new_svm_linear,newdata = new_preds)

confusionMatrix(table(new_predidictions,new_preds$Stance))

roc(response = train)
```

Appendix 2: Overall Design code

```
library(rtweet);library(tidyverse);library(tidytext)
#1. Collect some posts including Hillary Clinton

rate_limit(token = NULL, query = search_tweets, parse = TRUE)
new_tweets2 <- search_tweets("Clinton",
                             n = 10000,
                             since ="2022-06-8", #can only go 7 days back!
                             until = "2022-06-9",
                             include_rts = FALSE,
                            retryonratelimit = TRUE)

HCtweets <- rbind(new_tweets,new_tweets2)
HCtweets2 <- HCtweets%>%
  dplyr::select(user_id,text)

#cleaning links
HCtweets2$text <- gsub('http?\\S+','',HCtweets2$text) #delete links
HCtweets2$text<- gsub('<\\S+>','',HCtweets2$text) #delete weird code
HCtweets2$text <- gsub('@\\S+','',HCtweets2$text) #delete users

#delete tweets with questions
questions <- grep('[?]',HCtweets2$text)
HCtweets2 <- HCtweets2[-(questions),]

#keep only posts that still metnion Clinton
HCtweets2 <- HCtweets2[grep('Clinton',HCtweets2$text),]

#keep only active users
active_users <- HCtweets2 %>%
  dplyr::group_by(user_id) %>%
  dplyr::mutate(activity = n())%>%
  dplyr::filter(activity > 2)

length(unique(active_users$user_id))

####2. Measure emotionality####

#Lexicon
sent_dic <- get_sentiments("afinn")

#tokenize
X = 1:length(active_users$user_id)
active_users <- active_users%>%
  add_column(X)

bigrams <- active_users%>% #we need to group by X (i.e. post), otherwise considers
    all posts of user 1 same text
  dplyr::group_by(X)

bigrams <- bigrams %>%
  unnest_tokens(word, text, token = "ngrams", n = 2)%>%
  tidyr::separate(word,c('word1','word2'), sep = " ")%>%
  dplyr::inner_join(bigrams)
```

```
#negation words

neg = c('hardly','lack','neither','nor','never','no','nobody','none','nothing','
    nowhere','not','cannot','without')

bigrams <- bigrams%>%
  dplyr::mutate(neg = ifelse(word1 %in% neg,-1,1))

#apply lexicon taking into account negation words
bigrams_sent <- bigrams %>%
  dplyr::inner_join(sent_dic,by = c(word2 = 'word'))%>%
  dplyr::mutate(value = value * neg)

#take into account sentiment words that are never in word2 (i.e. the first words in
    each tweet)
frst_wrd <- data.frame(user_id = character(),word = character())

for(i in 1:length(active_users$text)){                      #we find the first word
    in each tweet
  words <- unnest_tokens(active_users[i,],word,text)
  frst_wrd <- frst_wrd%>%
    add_row(words[1,])
}

frst_wrd_sent <- frst_wrd%>%
  inner_join(sent_dic)%>%
  dplyr::select(-word)

final_sent <- bigrams_sent%>%        #add first words with sentiment value to the
    data set
  dplyr::group_by()%>%
  dplyr::select(user_id,value)%>%
  dplyr::add_row(frst_wrd_sent)%>%
  dplyr::mutate(value = abs(value))

sentim <- final_sent %>%
  dplyr::mutate(user_id = as.factor(user_id)) %>%
  dplyr::group_by(user_id) %>%
  dplyr::summarise(sent = mean(value))

####3. Measure political involvement####
#this code requires regex_pol and entities_list to be loaded

IDs <- unique(sentim$user_id)

#matrix to store involvement
pol_inv <- matrix(c(IDs,rep(0,2*length(IDs))),length(IDs),3)


#the loop itself

for(u in 186:length(IDs)){
```

```
  #to stop when limit is reached
  rate <- rate_limit(token = NULL, query = get_timeline, parse = TRUE)$remaining

  while(is.null(rate)){
    Sys.sleep(120) #if rate limit is reached for followers or friends stop for 2 min
    rate <- rate_limit(token = NULL, query = get_timeline, parse = TRUE)$remaining
  }

  tl_try <- get_timeline(IDs[u])
  pol_tweets_n <- 0 #counter of political tweets for this particular user
  ind <- sample(1:length(tl_try$text),1) #to store a random tweet for evaluation
  figures <- 0 #a counter to see how relevant the figures and organization detection
    is

  if(length(tl_try$text)<1){#to ignore users that doesn't give access to their TL
    pol_inv[u,2] <- NA
    pol_inv[u,3] <- NA
  } else {

    for(i in 1:length(tl_try$text)){

      wrds <- grep(regex_pol, tolower(tl_try$text[i])) #see if there is any pol word
    in tweet
      if(length(wrds) > 0){
        pol_tweets_n <- pol_tweets_n + 1 #if POLITICAL WORDS > 0, add to counter
      } else {
        figs <- grep(entities_list,tolower(tl_try$text[i])) #see if political fig/
    orgs in tweets
        if(length(figs)>0){
          pol_tweets_n <- pol_tweets_n + 1 #if POLITICAL FITURES found != 0, add to
    counter
          figures <- figures+1
        }
      }
    }

    pol_inv[u,2] <- pol_tweets_n #number of political tweets for this user
    pol_inv[u,3] <- length(tl_try$text) #keep track of how many total tweets we
    obtained from this user

  }
}


pol_inv_df <- data.frame(user_id =IDs,pol_tw =  as.numeric(pol_inv[,2]),pol_tot=as.
    numeric(pol_inv[,3])) #transformed into a dataframe

pol_inv_df <- pol_inv_df%>%
  dplyr::filter(pol_tot >= 98)

####4. Predict political stance####

IDs <- unique(active_users$user_id)
```

```
#get followers and friends
FL <- list()
FR <- list()
for (i in 1:length(IDs)){
  rate_fl <- rate_limit(token = NULL, query = get_followers, parse = TRUE)$remaining
  rate_fr <- rate_limit(token = NULL, query = get_friends, parse = TRUE)$remaining

  while(rate_fl == 0 | rate_fr == 0){
    Sys.sleep(120) #if rate limit is reached for followers or friends stop for 2 min
    rate_fl <- rate_limit(token = NULL, query = get_followers, parse = TRUE)$
    remaining
    rate_fr <- rate_limit(token = NULL, query = get_friends, parse = TRUE)$remaining
  }

  FL[[i]] <- get_followers(IDs[i])
  FR[[i]] <- get_friends(IDs[i])
}

#create vector of followers and friends per user
Fr_and_Fl <- list()

for(i in 1:length(IDs)){
  if(length(FL[[i]]) != 0){
    FL_vec <- paste('FL_',FL[[i]] %>% pull(user_id),sep = '')
  } else {
    FL_vec <- 'NONE'
  }
  if(length(FR[[i]]) != 0){
    FR_vec <- paste('FR_',FR[[i]] %>% pull(user_id),sep = '')
  } else {
    FR_vec <- 'NONE'
  }
  Fr_and_Fl[[i]] <- c(FL_vec,FR_vec)
}

#create predictors table
data <- as.data.frame(matrix(NA,length(IDs),length(best_pred)+1))
colnames(data) <- c('ID',best_pred)
data$ID <- IDs

for(u in 2:(length(best_pred)+1)){
  pred <- best_pred[u]
  for(i in 1:length(IDs)){
    data[i,u] <- ifelse(pred %in% Fr_and_Fl[[i]],1,0)
  }
  data[,u] <- factor(data[,u])
}

means <- data%>%
  summarise(across(everything(),mean))
histogram(as.numeric(means))


#generate predictions
```

```
pol_stance <- predict(new_svm_linear,newdata = data[,-1])

pol_stance_df <- data.frame(user_id = IDs, Stance = ifelse(pol_stance == 1,1,-1))


####5. Create final data frame####

data2 <- inner_join(sentim,pol_stance_df)
data2 <- inner_join(data2,pol_inv_df)

data2 <- data2%>%
  dplyr::mutate(pol_att = Stance * sent)

####6. Analysis####

#cusp fit
fit <- cusp(y ~ pol_att, alpha ~ 1, beta ~ pol_tw, data2)
summary(fit)

#linear
linear <- lm(sent ~ invol,data)
summary(linear)

#unrestricted
fit2 <- cusp(y ~ pol_att, alpha ~ pol_tw, beta ~ pol_tw, data2)  #unrestricted
summary(fit2)

plotCuspBifurcation(fit)

#graph
data_gra <- data2%>%
  mutate(quantilegroup = ntile(pol_tw, 4))%>%
  mutate(quantilegroup = factor(quantilegroup))

ggplot(data_gra,aes(x = pol_att, y = quantilegroup))+
  geom_density_ridges(fill = rgb(0,0,0,0))+
  theme_minimal()+
  theme(text = element_text(family = 'serif',size = 15))+
  ylab('Political involvement quartile')+
  xlab('Political attitude')
```