Extracting Emotions (and more) from Faces with Face++ and Microsoft Azure

Theresa Küntzler

Social Science Data Lab | MZES & Online
February, 13 2020

A picture is worth a thousand words

Creator: Frank C. Müller
Cut out by myself
Licence: CC BY-SA 4.0
Link to Source: Wikipedia

A picture is worth a thousand words

Creator: Frank C. Müller
Cut out by myself
Licence: CC BY-SA 4.0
Link to Source: Wikipedia

Creator: Armin Linnartz
Ctut out by myself
Licence: CC BY-SA 3.0
Link to Source: Wikipedia

Facial Images as Data

Humans infer general information from faces...

- Age
- Gender
- Race
- Competence
- Attractiveness
- ...

... and use them in a political context.

- Inferred competence predicts election outcomes. (Todorov et al. 2005)
- Machines can do it, too. (Joo et al. 2015)
- Inference of party affiliation from faces,
- which is biased by candidate attractiveness and attributed competence. (Herrmann & Shikano 2016)

Computer Vision and APIs

Computer Vision

- Images as series of numbers
- Part of Machine Learning
- Object detection and classification
- Allows to scale image analysis

API

- Here: Web-API
- In simple terms: Set of functions to access an offered service (online)

Face++

- Address: `https://www.faceplusplus.com/`
- Face Detection API

Variables

- Gender
- Age
- Emotion (6+1)
- Smiling
- Headpose
- Facequality
- Blur
- Eyestatus
- Ethnicity
- Beauty
- Mouthstatus
- Eyegaze
- Skinstatus

Free Account includes:

- Unlimited usage
- Possible capacity restrictions
- Only one key per account
- Storage limits

Microsoft Azure Face API

- Address: `https://azure.microsoft.com/en-us/services/cognitive-services/face/`
- Face API

Variables

- Age
- Emotion (7+1)
- Blur
- Exposure
- Facial hair
- Gender
- Glasses
- Hair
- Head Pose
- Makeup
- Noise
- Occlusion
- Smile

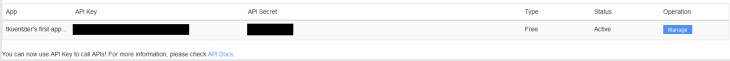
Free Account includes:

- 20 transactions per minute
- 30,000 transactions per month

Access credentials

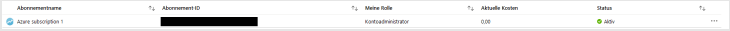
Face++

1. Register here: *Face++ Registration*
2. After log-in, go to Console > API Apps > API Key > + Get API Key
3. Fill out the form (choose “Free” for API Key Type) and Submit
4. You can then see your personal API Key and Secret



Azure Face API

1. Register here: *Azure Registration*
2. Follow *these instructions* to create an API key
3. You can then see your personal API Key in the Portal >



Structure of the Functions

Authentification Function (for Face++)

Merge API key and secret to one object

API Call Function

Input: Vector with filepaths to images, Athentification data
Result: Data table with filepaths and output data

```
create empty table to fill with API output: faces;
for each image do
    make API call;
    if at least on face is found then
        write output of first face in dataframe;
        if more than one face is found then
            add lines to data frame and write output;
        end
    end
    Wait 2 seconds before next call;
end
return faces
```

Code: API Authentification

```
## Function creates object with Face++ key and secret, pass object
     to faceEst function
## Input:
## 1. api_key: character, given from faceplusplus account
## 2. api_secret: character, given from faceplusplus account
## Output:
## 1. authentifiaction object to be used in faceEST function
## Note: Function written by Sascha Goebel

authFacepp <- function(api_key, api_secret){
  auth <- structure(list(api_key = api_key,
                         api_secret = api_secret),
                    class="FaceppProxy")
}
```

Code: Face++ Call Function: Prepare data.table

```
## Note: Earlier version of this function written by Sascha Goebel
faceEst <- function(fullpath, auth) {
  ## Initilize Object to store API output for single image
  face <- NULL

  ## create empty table to fill with API output
  faces <- data.table(
    emo_anger = as.numeric(NA), emo_disgust = as.numeric(NA),
    emo_fear = as.numeric(NA), emo_happiness = as.numeric(NA),
    emo_neutral = as.numeric(NA), emo_sadness = as.numeric(NA),
    emo_surprise = as.numeric(NA),

    gender = as.character(NA),

    facecount = as.numeric(NA),

    fullpath = fullpath)

  #[...]
```

Code: Face++ Call Function: Make the call

```
#[...]
## go over each fullpath and send to API
run <- 0
for (i in 1:length(fullpath)) {
  run <- run + 1
  cat(run, "\n") #count and print the paths sent, for user info
  while(is.null(face)) {
    try(
      face <- as.character(
        httr::RETRY(
          "POST",
          "https://api-us.faceplusplus.com/facepp/v3/detect",
          body = list(api_key  = auth$api_key,
            api_secret = auth$api_secret,
            image_file = upload_file(fullpath[i]),
            return_landmark = 0,
            return_attributes = "emotion,gender"),
          times = 2,
          encode = "multipart")),
    silent = FALSE
  )}
#[...]
```

Code: Face++ Call Function: Make the call

```
#[...]
## go over each fullpath and send to API
run <- 0
for (i in 1:length(fullpath)) {
  run <- run + 1
  cat(run, "\n") #count and print the paths sent, for user info
  while(is.null(face)) {
    try(
      face <- as.character(
       httr::RETRY(
          "POST",
          "https://api-us.faceplusplus.com/facepp/v3/detect",
          body = list(api_key = auth$api_key,
            api_secret = auth$api_secret,
            image_file = upload_file(fullpath[i]),
            return_landmark = 0,
            return_attributes = "emotion,gender"),
          times = 2,
          encode = "multipart")),
    silent = FALSE
  )}
#[...]
```

Code: Face++ Call Function: Make the call

The *Reference Manual* specifies the information for the function.

```
face <- as.character(
    httr::RETRY(
      "POST",
      "https://api-us.faceplusplus.com/facepp/v3/detect",

      body = list(api_key  = auth$api_key,
          api_secret = auth$api_secret,
          image_file = upload_file(fullpath[i]),
          return_landmark = 0,
          return_attributes = "emotion,gender"),
      times = 2,
      encode = "multipart")),
```

Request URL

https://api-us.faceplusplus.com/facepp/v3/detect

Request Method

POST

Code: Face++ Call Function: Make the call

The *Reference Manual* specifies the information for the function.

```
face <- as.character(
    httr::RETRY(
      "POST",
      "https://api-us.faceplusplus.com/facepp/v3
          /detect",
      body = list(api_key = auth$api_key,
          api_secret = auth$api_secret,
          image_file = upload_file(fullpath[i]),
          return_landmark = 0,
          return_attributes = "emotion,gender"),
      times = 2,
      encode = "multipart")),
```

Request Parameter

	Name	Type
Required	api_key	String
Required	api_secret	String
Required (choose any of three)	image_url	String
	image_file	File
	image_base64	String

Code: Face++ Call Function: Make the call

The *Reference Manual* specifies the information for the function.

```
face <- as.character(
    httr::RETRY(
      "POST",
      "https://api-us.faceplusplus.com/facepp/v3
          /detect",
      body = list(api_key  = auth$api_key,
          api_secret = auth$api_secret,
          image_file = upload_file(fullpath[i]),
          return_landmark = 0,
          return_attributes = "emotion,gender"),
      times = 2,
      encode = "multipart")),
```

Optional	return_landmark	Int	Whether or not dete 2 detect and ret 1 detect and ret 0 do not detect Note: default value i
Optional	return_attributes	String	Whether or not dete none • gender • age • smiling • headpose • facequality • blur • eyestatus • emotion • ethnicity

API Output

What does `face` look like for a test image?: Its a JSON notation.


```
> face
[1]"{\"request_id\":\"1588927013,dd00d1c1-
fff2-4050-8c46-9a2188486c19\",\"time_used
\":141,\"faces\":[{\"face_token\":\"ef9be2
97e8184963ddd2fd154e7c054e\",\"face_recta
ngle\":{\"top\":146,\"left\":300,\"width\
":195,\"height\":195},\"attributes\":{\"g
ender\":{\"value\":\"Male\"},\"emotion\":{
\"anger\":0.000,\"disgust\":0.000,\"fear\
":0.063,\"happiness\":99.937,\"neutral\":
0.000,\"sadness\":0.000,\"surprise\":0.00
0}}}],\"image_id\":\"KdgNW2IvGLbViBZ1ialuL
Q==\",\"face_num\":1}\n"
```


Link to the image (Wikipedia)

Code: Face++ Call Function: Save the output

```
#[...]
## if face is found, extract information and write into data.
    table
facecount <- length(fromJSON(face)$faces$face_token)

if (facecount != 0) {
  emotion <- fromJSON(face)$faces$attributes$emotion
  gender <- fromJSON(face)$faces$attributes$gender

  ## write info to data.table
  faces[faces$fullpath == fullpath[i],][,1:9] <- c(emotion
      [1,], gender[1,], facecount)

  #[...]
```

```
> emotion
  anger disgust  fear happiness neutral sadness surprise
1     0       0 0.063    99.937       0       0        0
```

Code: Face++ Call Function: Save the output

```
    #[...]
    ## if more than one face found, make df with all info and
        merge
      if (facecount > 1) {
      faces <- union(x = faces,
        y = data.table(emo_anger = emotion[,1],
        emo_disgust = emotion[,2],
        emo_fear = emotion[,3],
        emo_happiness = emotion[,4],
        emo_neutral = emotion[,5],
        emo_sadness = emotion[,6],
        emo_surprise = emotion[,7],

        gender = gender[,1],

        facecount = facecount,

        fullpath = fullpath[i]))
      } # end if(facecount > 1)
  #[...]
```

Code: Face++ Call Function: Bookkeeping

```
  #[...]
      face <- NULL
      Sys.sleep(2)
    } # end if(facecount != 0)
    else {
      face <- NULL
      Sys.sleep(2)
    }
  } # end for(i in 1:length(fullpath))
  return(faces)
} # end function
```

What changes for the Azure API?

To modify the function for a call zu the Azure API only a few changes are necessary:

1. Adjust the variables of the faces dataframe
   - Contempt
   - (Error Messages)
2. Adjust the saving procedure accordingly
3. Adjust the actual call

Demonstration

Live Demo in R

Words of caution...

... on the method

- These algorithms remain black boxes
- Is performance well on your data?
- Racial and other biases in computer vision (Zou & Schiebinger 2018)
- Subjectiveness of variables

... on law

- Read the Terms of Service
- Make sure you are allowed to send the images you want to send (copyright, personality rights, etc.)

Thoughts? Ideas? Questions?

Contact me!

Theresa Küntzler
theresa.kuentzler@uni-konstanz.de
@ TKuentzler

References I

- Herrmann, Michael & Susumu Shikano (2016). "Attractiveness and Facial Competence Bias Face-BasedInferences of Candidate Ideology". In: *Political Psychology* 37 (3), pp. 401–417. DOI: 10.1111/pops.12256.
- Joo, Jungseock, Francis F. Steen & Song-Chun Zhu (2015). "Automated Facial Trait Judgment and Election Outcome Prediction:Social Dimensions of Face". In: *IEEE*. DOI: 10.1109/ICCV.2015.423.
- Todorov, Alexander et al. (2005). "Inferences of Competence fromFaces Predict Election Outcomes". In: *Science* 308, pp. 1623–1626. DOI: 10.1126/science.1110589.

References II

Zou, James & Londa Schiebinger (2018). "AI can be sexist and racist — it's time to make it fair". In: *Nature (Comment)*. URL: https://www.nature.com/articles/d41586-018-05707-8?source=post_page-----817fa60d75e9----------&fbclid=IwAR0iajZ2WRp7mlrW4gy10X1jsmJLvRH-8blaSI0vGO_OMK05yJBW2X2FevI#ref-CR6.

Links to Image Sources

- Drawing of Aylan Kurdi:
  https://commons.wikimedia.org/w/index.php?curid=47487290
- Angela Merkel:
  https://commons.wikimedia.org/w/index.php?curid=16103982
- Barack Obama:
  https://en.wikipedia.org/wiki/Barack_Obama#/media/File:President_Barack_Obama.jpg