**Getting Black Metal Cats tweets**

It won’t come as a surprise for the loyal readers of this blog that I just had to use rtweet. I kept only original standalone tweets and removed the picture link from the tweet.

black\_tweets <- rtweet::get\_timeline("evilbmcats")

black\_tweets <- dplyr::filter(black\_tweets, is.na(reply\_to\_user\_id), !is\_retweet, !is\_quote)

black\_tweets <- dplyr::select(black\_tweets, text, created\_at, status\_id)

black\_tweets <- dplyr::mutate(black\_tweets, text = stringr::str\_replace(text, "https.\*", ""))

readr::write\_csv(black\_tweets, path = "data/2018-01-03-bubblegumpuppies\_cats.csv")

Now that the dark material is ready, let’s sweeten it…

**Modifying the tweet text**

Black Metal Cats tweet heavy metal lyrics so as you can imagine, they’re sad. How to make them happy while keeping the text similar enough to the original one? And this without too much effort? My simplistic strategy was to identify negative words via sentiment analysis and to replace them with positive words.

*Note, had I not wanted to stay close to the original tweet, I could have just chosen lyrics picked from* [*this dataset*](https://www.rdocumentation.org/packages/billboard/versions/0.1.0/topics/lyrics?tap_a=5644-dce66f&tap_s=10907-287229) *for instance, and filtered them by sentiment via the sentimentr package.*

**Finding negative words**

I computed sentiment using tidytext:

If I understand the above correctly, I’m allowed to scrape the titles of the columns, great!

I also noticed the crawl delay at the end of the robots.txt, of 1 second. Since I’ve decided to be a really nice scraper and also because I only have 29 pages to scrape in total, I’ll use a delay of 2 seconds between requests. In his post Bob says that if there is no indication, you should wait 5 seconds.

After these checks, I started working on the scraping itself.

library("rvest")

xtract\_titles <- function(node) {

css <- 'span[class="js-headline-text"]'

html\_nodes(node, css) %>% html\_text(trim = **TRUE**)

}

get\_titles\_from\_page <- function(page\_number){

Sys.sleep(2)

link <- paste0("https://www.theguardian.com/lifeandstyle/series/experience?page=", page\_number)

page\_content <- read\_html(link)

xtract\_titles(page\_content)

}

experience\_titles <- purrr::map(1:29, get\_titles\_from\_page) %>% unlist()

save(experience\_titles, file = "data/2017-10-02-guardian-experience.RData")

set.seed(1)

sample(experience\_titles, 10)

## [1] "Experience: pregnancy sickness nearly killed me"

## [2] "Experience: I was a sperm donor for my friends"

## [3] "Experience: I was attacked in my front garden"

## [4] "I was brought up in the exclusive brethren"

## [5] "Experience: I am Dancing Man"

## [6] "The boy who missed the mainstream"

## [7] "I still can't explain what I saw"

## [8] "Experience: My twin rewrote my childhood"

## [9] "Experience: I've renewed my wedding vows more than 50 times"

## [10] "Experience: I talk with my eyes"

See, these are really diverse topics! And I think this sample of 10 titles actually shows many heavy topics.

**Experience: I computed the most frequent words**

I’ll first remove the “Experience: " part of many titles, since it’s not exactly the most interesting word.

experience\_titles <- stringr::str\_replace(experience\_titles, "^Experience: ", "")

I then unnested words. Interestingly in order to remember how to do this I went and read [my Guardian blind dates post](https://masalmon.eu/2017/03/07/blinddates/) (the “So what did they talk about?” part).

library("tidytext")

library("rcorpora")

stopwords <- corpora("words/stopwords/en")$stopWords

words <- tibble::tibble(title = experience\_titles) %>%

unnest\_tokens(word, title) %>%

dplyr::filter(!word %in% stopwords) %>%

dplyr::count(word, sort = **TRUE**)

knitr::kable(words[1:20,])

| **word** | **n** |
| --- | --- |
| years | 23 |
| fell | 21 |
| lost | 20 |
| saved | 20 |
| life | 19 |
| man | 19 |
| baby | 15 |
| killed | 13 |
| survived | 13 |
| car | 12 |
| daughter | 12 |
| love | 12 |
| father | 11 |
| friend | 11 |
| husband | 11 |
| birth | 9 |
| dad | 8 |
| married | 8 |
| attacked | 7 |
| days | 7 |

In my opinion this list of the most common words support my feeling topics are often heavy, but I also think it might be because there are many, many different words that can describe a light topic while well death will be primarily described by “killed”. Could sentiment analysis of the titles help me?

**Experience: I computed the sentiment of titles**

afinn <- get\_sentiments("afinn")

sentiment <- tibble::tibble(title = experience\_titles) %>%

dplyr::mutate(saved\_title = title) %>%

unnest\_tokens(word, title) %>%

dplyr::inner\_join(afinn) %>%

dplyr::group\_by(saved\_title) %>%

dplyr::summarize(sentiment = sum(score)) %>%

dplyr::filter(!is.na(sentiment))

knitr::kable(sentiment[1:10,])

| **saved\_title** | **sentiment** |
| --- | --- |
| ‘I stopped a terrorist attack’ | -2 |
| a coup interrupted our wedding | -2 |
| A great white shark ate my leg | 3 |
| a head injury made me a musical prodigy | -2 |
| a ladybird nearly killed me | -3 |
| A machine keeps me alive | 1 |
| A six-metre wall collapsed on top of me | 0 |
| Becoming homeless helped me find love | 3 |
| Being obese made me feel like a social outcast | 2 |
| Blind date | -1 |

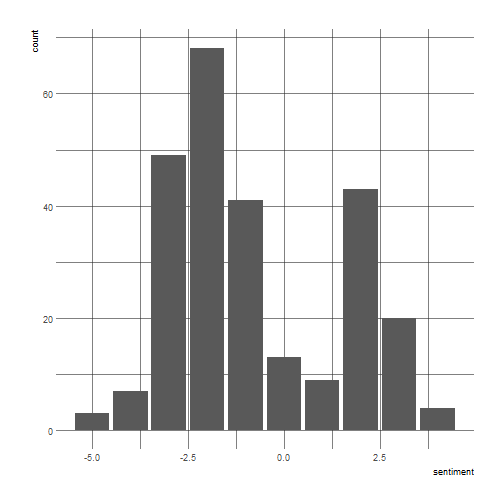
library("ggplot2")

library("hrbrthemes")

ggplot(sentiment) +

geom\_bar(aes(sentiment)) +

theme\_ipsum\_rc()



Honestly, I think sentiment analysis didn’t help much here: the titles are too short, and the sample presented above is not very convincing. Moreoever, would the sentiment reveal the dramatic intensity of light vs. heavy, anyway?

**Experience: I tried using machine learning to derive a topic from the title**

In the following I’ll use [my own monkeylearn package](https://github.com/ropensci/monkeylearn) and in particular [this topic classifier](https://app.monkeylearn.com/main/classifiers/cl_5icAVzKR/) without too much hope since I’m feeding it a title, not a whole article.

topics <- monkeylearn::monkeylearn\_classify(experience\_titles,

classifier\_id = "cl\_5icAVzKR")

titles <- tibble::tibble(title = experience\_titles,

text\_md5 = purrr::map\_chr(experience\_titles, digest::digest, algo = "md5"))

titles <- dplyr::inner\_join(titles, topics, by = "text\_md5")

Here’s a sample of the results after an arbitrary filtering based on probability:

titles <- dplyr::filter(titles, probability > 0.5)

set.seed(1)

dplyr::sample\_n(titles, size = 20) %>%

dplyr::select(title, label, probability) %>%

knitr::kable()

| **title** | **label** | **probability** |
| --- | --- | --- |
| my family was attacked by lions | Land Mammals | 0.680 |
| Muhammad Ali was my mentor | Religion & Spirituality | 0.681 |
| I’m a championship arm-wrestler | Entertainment & Recreation | 0.873 |
| I lit my father’s funeral pyre | Relationships | 0.603 |
| I have sudden death syndrome | Health & Medicine | 0.816 |
| One drink and I’m dead | Food & Drink | 0.511 |
| I was a compulsive gambler | Mental health | 0.805 |
| I flew the English Channel using a bunch of balloons | Aircraft | 0.828 |
| I crushed my £1m violin | Humanities | 0.625 |
| I crashed into the North Sea | Travel | 0.549 |
| I said yes to marriage the first time we met | Society | 0.775 |
| I can fly | Aircraft | 0.930 |
| We were told our son has cystic fibrosis – he hasn’t | Special Occasions | 0.511 |
| I found out I’m not my son’s father | Society | 0.521 |
| I became a famous artist at the age of 94 | Music | 0.673 |
| I was impaled while pregnant | Mental health | 0.548 |
| A great white shark ate my leg | Animals | 0.656 |
| The holiday capsule wardrobe | Accommodation | 0.761 |
| I don’t wear shoes | Beauty & Style | 0.798 |
| I became a famous artist at the age of 94 | Art | 0.531 |

Note that after this filtering I had at least one topic for 288 titles. I don’t think this classification is really useful either but at least it’s fun to look at the proposed topic. What are the most frequent ones?

titles %>%

dplyr::group\_by(label) %>%

dplyr::summarise(n = n(),

some = toString(title[1:3])) %>%

dplyr::arrange(dplyr::desc(n)) %>%

head(n = 10) %>%

knitr::kable()

| **label** | **n** | **some** |
| --- | --- | --- |
| Transportation | 45 | I pulled a man from a burning car, I was hit by a car doing 101mph, a car crashed into me in the bath |
| Relationships | 36 | I’m a divorce party planner, a coup interrupted our wedding, my husband didn’t meet our daughter until she was 27 |
| Society | 32 | my husband didn’t meet our daughter until she was 27, I first met my mother at a party, I was accused of having a sham marriage |
| Land Vehicles | 30 | I pulled a man from a burning car, I was hit by a car doing 101mph, a car crashed into me in the bath |
| Special Occasions | 29 | I fell in love through Airbnb, I made peace with my daughter’s killer, I’ve been protesting for more than 60 years |
| Animals | 26 | my dog rescues cats, I accidentally bought a giant pig, I was bitten by a shark |
| Parenting | 19 | I had a free birth, I saved a stranger’s life, We found a baby in a manger |
| Travel | 16 | a car crashed into me in the bath, I crashed into the North Sea, I saved my school bus from crashing |
| Land Mammals | 15 | my dog rescues cats, my cat saved me from a fire, I own the world’s ugliest dog |
| Health & Medicine | 13 | I have sudden death syndrome, I am afraid of pregnancy, my anti-malaria drugs made me psychotic |

That, in a way, makes me more okay with the classification. I’ve always had the impression (you have to believe me) that many of the columns dealt with accidents, which corresponds to the transportation category, and families and relationships, and well animals, the ones [that try to eat you](https://www.theguardian.com/lifeandstyle/2010/oct/16/experience-shark-attack-paralympian) or [that steal your tractor](https://www.theguardian.com/lifeandstyle/2015/jun/12/experience-my-dog-stole-my-tractor). But now does it help me judge whether the Experience columns deal with rather light or heavy topics? Hum, no.

**Experience: I could not really answer my initial question**

So, it was fun, but I can’t really tell Gabriella Paiella whether she was right or wrong. One thing is sure, these columns are quite varied… so everyone can find what they’re looking for, either a dramatic story or a funny one?

library("tidytext")

library("magrittr")

bing <- get\_sentiments("bing")

sentiment <- dplyr::select(black\_tweets, text) %>%

dplyr::mutate(saved\_text = text) %>%

unnest\_tokens(word, text) %>%

dplyr::inner\_join(bing, by = "word") %>%

dplyr::filter(sentiment == "negative")

It was a bit disappointing since out of the 97 only 63 were represented in that data.frame. But well, this shall do! I just looked rapidly at some non included tweets.

dplyr::filter(black\_tweets, !text %in% sentiment$saved\_text) %>%

head() %>%

knitr::kable()

| **text** | **created\_at** | **status\_id** |
| --- | --- | --- |
| Black covfefe cats. | 2017-12-29 21:01:06 | 946848575446663168 |
| Black coffee cats. | 2017-12-28 00:15:48 | 946172798203916288 |
| Say goodbye to the light. | 2017-12-26 15:50:00 | 945683121336344576 |
| The storm will never end. | 2017-12-23 15:49:00 | 944595705204678656 |
| Satan, your time has come. This world must end. | 2017-12-18 22:36:00 | 942886191107575808 |
| Timely. | 2017-12-13 00:43:30 | 940743951287504896 |

Ok so some of them are probably negative tweets that the [sentimentr package](https://cran.r-project.org/web/packages/sentimentr/README.html) would help detect but that do not contain negative words.

**Replacing words**

I seriously considered using the [wordnet package](https://cran.r-project.org/web/packages/wordnet/index.html) because of this [Stack Overflow question “Getting antonyms using the wordnet package”](https://stackoverflow.com/questions/19360107/getting-antonyms-using-the-r-wordnet-package) but I was not brave enough, my strength failed me in front of the Java needs of that package.

I decided to use praise to get positive words, and cleanNLP (as in [this post](http://www.masalmon.eu/2017/12/05/badderb/)) to try and identify correctly negative words as adjective or verbs for instance in order to be able to replace them. The right annotation for that is a *token*.

library("cleanNLP")

#init\_spaCy()

get\_token\_with\_text <- function(x){

obj <- run\_annotators(x, as\_strings = TRUE)

entities <- get\_token(obj)

entities$text <- x

entities

}

possibly\_get\_tokens <- purrr::possibly(get\_token\_with\_text,

otherwise = NULL)

tokens <- purrr::map\_df(sentiment$saved\_text, possibly\_get\_tokens)

head(tokens) %>%

knitr::kable()

|  |
| --- |
|  |
|  |
|  |
|  |

Once here, I joined sentiment and tokens.

tokens <- dplyr::mutate(tokens, word = enc2native(word))

tokens <- dplyr::mutate(tokens, word = tolower(word))

tokens <- dplyr::left\_join(sentiment, tokens, by = c("saved\_text" = "text", "word"))

head(tokens) %>% knitr::kable()

| **saved\_text** | **word** | **sentiment** | **id** | **sid** | **tid** | **lemma** | **upos** | **pos** | **cid** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| We are the creatures you fear. | fear | negative | 1 | 1 | 6 | fear | VERB | VBP | 25 |
| In darkness we walk. | darkness | negative | 1 | 1 | 2 | darkness | NOUN | NN | 3 |
| Standing tall, side by side, we shall build thy throne upon the burning ashes. | burning | negative | 1 | 1 | 15 | burn | VERB | VBG | 64 |
| This is the dawn of the infernal reign. | infernal | negative | 1 | 1 | 7 | infernal | ADJ | JJ | 24 |
| A breath of the past so distant and so unreal. A soul condemned to haunt a frozen burial ground. | condemned | negative | 1 | 2 | 3 | condemn | VERB | VBD | 54 |
| A breath of the past so distant and so unreal. A soul condemned to haunt a frozen burial ground. | condemned | negative | 1 | 2 | 3 | condemn | VERB | VBD | 54 |

The praise package provides adjectives that I’ll use to replace adjectives and to add an exclamation at the beginning of each text. Nouns, present and preterit verbs will be replaced with love/loves/loved because hey, this is pop music inspiration. I’ll lose capital letters and won’t bother too much, puppies probably do not care either. What I prepare below is what Hilary Parker called a dictionary table [in this tweet](https://twitter.com/hspter/status/948646331677118465). VBZ is for instance a verb in the present form like “haunts”.

modifiers <- tibble::tibble(pos = c( "NN", "VBG", "JJ",

"VBD", "VB", "NNS",

"VBP", "VBN", "NNP", "VBZ"),

modifier = c("love", "${adjective}", "${adjective}",

"loved", "love", "lovers",

"love", "${adjective}",

"love", "loves"))

knitr::kable(modifiers)

| **pos** | **modifier** |
| --- | --- |
| NN | love |
| VBG | ${adjective} |
| JJ | ${adjective} |
| VBD | loved |
| VB | love |
| NNS | lovers |
| VBP | love |
| VBN | ${adjective} |
| NNP | love |
| VBZ | loves |

Now let’s get to work on the sweetening of the texts at last! Some occurrences of “hate” and “evil” remained, and I removed them by hand.

# praise has some randomness, let's make it reproducible

set.seed(42)

modifiable\_tweets <- dplyr::left\_join(tokens, black\_tweets,

by = c("saved\_text" = "text"))

modifiable\_tweets <- dplyr::left\_join(modifiable\_tweets, modifiers,

by = "pos")

# for easier use with map

replace\_all <- function(pattern, replacement, x){

stringr::str\_replace\_all(x, pattern = pattern,

replacement = replacement)

}

modified\_tweets <- dplyr::group\_by(modifiable\_tweets, saved\_text) %>%

dplyr::summarize(praise\_template = purrr::map2\_chr(word, modifier, replace\_all,

x = tolower(saved\_text[1]))[1],

praise\_template = paste("${exclamation}!", praise\_template),

praise\_template = stringr::str\_replace\_all(praise\_template,

"hate", "love"),

praise\_template = stringr::str\_replace\_all(praise\_template,

"evil", "love"),

new\_text = praise::praise(praise\_template))

modified\_tweets <- dplyr::select(modified\_tweets, saved\_text, new\_text)

puppies\_and\_cats <- dplyr::left\_join(modified\_tweets, black\_tweets, by = c("saved\_text" = "text"))

head(puppies\_and\_cats) %>% knitr::kable()

| **saved\_text** | **new\_text** | **created\_at** | **status\_id** |
| --- | --- | --- | --- |
| A breath of the past so distant and so unreal. A soul condemned to haunt a frozen burial ground. | yikes! a breath of the past so distant and so unreal. a soul loved to haunt a frozen burial ground. | 2017-12-28 20:35:00 | 946479619020115968 |
| Be very afraid. | yippie! be very fine. | 2017-11-18 21:02:00 | 931990900523307008 |
| Beware of the cat. | yay! love of the cat. | 2017-12-11 14:54:02 | 940233218204172288 |
| Beware the legions of Satan, they’re ready for attack! | ole! love the legions of satan, they’re ready for attack! | 2017-12-05 22:48:00 | 938178169399422976 |
| Born by the nothingness into eternal blasphemy. | mm! born by the nothingness into eternal love. | 2017-12-03 04:15:01 | 937173299448221696 |
| Born on a day of curses and damnation, I was doomed to hate. | whoa! born on a day of lovers and damnation, i was doomed to love. | 2017-12-04 21:40:01 | 937798672347226113 |

So although some new lyrics do not look that cheerful, they’re at least grammatically correct.

**Modifying the picture**

I recently discovered [Pexels](https://www.pexels.com/), a website with CC-0 pictures, that I even learnt to scrape for an unpublished (yet) project. So many photographs you can use and modify for free without any attribution! Quite cool, really. To scrape the page I first had to scroll down to get enough pictures, which I did following [this Stack Overflow thread](https://stackoverflow.com/questions/29861117/r-rvest-scraping-a-dynamic-ecommerce-page) with RSelenium. I tried using seleniumPipes instead but had trouble setting up the server and not too much time to dwell on that.

Yes, you got it right, the code below automatically downloads pics of puppies into your laptop. Happy New Year.

library("rvest")

library("RSelenium")

library("magrittr")

# https://stackoverflow.com/questions/29861117/r-rvest-scraping-a-dynamic-ecommerce-page

rD <- rsDriver()

remDr <- rD[["client"]]

# open the webpage

remDr$navigate("https://www.pexels.com/search/puppies/")

# scroll down

for(i in 1:30){

remDr$executeScript(paste("scroll(0,",i\*10000,");"),

args = list("dummy"))

# be nice and wait

Sys.sleep(1)

}

page\_content <- remDr$getPageSource()

remDr$close()

# functiosn for getting the pic links

get\_link\_from\_src <- function(node){

xml2::xml\_attrs(node)["src"] %>%

as.character() %>%

stringr::str\_replace("\\?h.\*", "")

}

xtract\_pic\_links <- function(source) {

css <- '.photo-item\_\_img'

read\_html(source[[1]]) %>%

html\_nodes(css) %>%

purrr::map\_chr(get\_link\_from\_src)

}

links <- xtract\_pic\_links(page\_content)

links <- links[1:nrow(puppies\_and\_cats)]

# save

dir.create("data/puppies")

save\_pic <- function(url, name){

Sys.sleep(1)

name <- paste0("puppy", name, ".png")

try(magick::image\_read(url) %>%

magick::image\_write(paste0("data/puppies/", name)),

silent = TRUE)

}

purrr::walk2(links, 1:nrow(puppies\_and\_cats), save\_pic)

I took a moment to browse the 66 pics. Carpe diem, carpe canes (I learnt Latin for 6 years as a teen(ager) and had to check the plural of *canem*…).

The pics were a bit too big so I resized them.

resize <- function(pic){

magick::image\_read(pic) %>%

magick::image\_resize("300x300") %>%

magick::image\_write(pic)

}

purrr::walk(dir("data/puppies/", full.names = TRUE), resize)

**Tweeting the cheerful tweets – DON’T DO IT LIKE I DID!**

[a Twitter account for the Bubblegum Puppies](https://twitter.com/cheerful_doggos) so that they could interact with the Black Metal Cats in their natural environment. I was asked by Scott whether I’d make a bot out of my idea and I don’t intend to especially since my simplistic strategy can only answer tweets with negative words detected, but I’d be glad to let someone adopt the account, especially since I did not educate my puppies well to begin with as you see below! [Here](https://www.wjakethompson.com/post/tidyverse-tweets/) is a tutorial for making a Twitter bot with rtweet. In the meantime, I simply decided to tweet the answers I had created… without enough thinking.

*Note: It has been a long since I last obtained Twitter access tokens, so I refered to* [*the vignette*](http://rtweet.info/articles/auth.html)*.*

Then I wrote the code to send replies, an important point being that you need to mention the username you answer to in your tweet otherwise the in\_reply\_to\_status\_id argument is ignored. It was a dangerous code because it sent too many replies, which made it a spam bot… Don’t do that. So if you ever need an example of a very stupid spammer, think of me. Lesson learnt, I’ll never be that reckless again, because I do not want to spam anyone.

post\_a\_reply <- function(df, pic){

rtweet::post\_tweet(paste("@evilbmcats", df$new\_text), in\_reply\_to\_status\_id = df$status\_id,

media = pic)

Sys.sleep(1)

}

pics <- dir("data/puppies/", full.names = TRUE)

purrr::walk2(split(puppies\_and\_cats, puppies\_and\_cats$status\_id), pics,

post\_a\_reply)

So how would I use the tweets if I could do it again? Well, I’d post them with a faaar bigger delay. And I think the account would be fun as a bot which’d reply only when the cats account tweets again. That way the puppies would be cute, not cute and annoying. Live and learn! Thanks to Bob Rudis to encourage me to post this!

**Ending with some cuteness**

And now, because I do not want you to think I’m now as depressed as a Black Metal Cat, I’ll end this post by showing you a few replies thanks to the brand new rtweet::tweet\_shot function added in the dev version of rtweet by [Bob Rudis](https://twitter.com/hrbrmstr?lang=en) after he saw my [#best9of2017 post](http://www.masalmon.eu/2017/12/30/best9of2017/). I resorted to saving the files and add the Markdown code to show them by hand but in a normal Rmd, not in a website, the images (magick objects) actually render very well. [Head to Twitter](https://twitter.com/cheerful_doggos/with_replies) to see the rest!

save\_one\_tweet <- function(status\_id){

rtweet::tweet\_shot(status\_id) %>%

magick::image\_write(paste0("data/2018-01-04-",

status\_id, ".png"))

}

c("948942337912328192",

"948942293272334336",

"948942240713531392") %>%

purrr::walk(save\_one\_tweet)