Escola Universitària Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Grau en Enginyeria Informàtica

Text Mining Reddit's Top Posts: The Potential Information In Internet-Based Communities

Report

ALEX FERNÁNDEZ PAWLUKOJC – Researcher
XAVIER FONT ARAGONES – Tutor

SPRING 2017

TecnoCampus Mataró-Maresme

This work is dedicated to the professors, tutors, monitors and educators that brought me to where I have arrived, as well as to my parents. Without any of them, and without their care and patience, I probably would not have ever made it to where I am.

I would also like to dedicate this work to my friends and acquaintances for the support and discussion they gave me, especially Jonathan N. from Cornell. Your humour and good spirit helped me to push through with this!

Abstract

As of late, there has been a growing interest in the field of "data-mining" which involves, among other things, the processment of massive amounts of data for a higher purpose. The result of such processes usually allows for the assessment of interesting variables concerning huge pools of population, like trending fads or the development of political currents. One such pool is Reddit, a web content aggregator service which has been gaining popularity around the world over the last few years. A statistical analysis has been performed, focused on the titles that people gave to the most popular content published in Reddit within two specific timeframes: 2013 and 2016, both in August. The results show interesting patterns regarding the most popular words and combinations of words, which points towards promising results should further investigation be undertaken.

Resum

Últimament, hi ha hagut un augment en l'interès en el camp del "data-mining", el qual implica, entre altres coses, el processament de quantitats massives de dades. El resultat d'anàlisis d'aquest tipus normalment permeten observar dades interessants al respecte de grans mostres de població, com temes de moda o el desenvolupament de corrents polítiques. Una d'aquelles és Reddit, un servei web d'agregació de continguts que ha guanyat bastant popularitat en els últims anys. S'han realitzat anàlisis estadístiques sobre els títols del contingut més popular publicat en dos períodes específics: Agost de 2013 i 2016. Els resultats mostren uns patrons interessants en quant a les paraules i combinacions de paraules més freqüents, el qual apunta cap a un resultats prometedors en el cas de realitzar-se investigacions més extensives.

Resumen

Últimamente ha habido un aumento en el interés por el "data-mining", el cual implica, entre otras cosas, el processamiento de cantidades masivas de datos. El resultado de análisis de este tipo normalmente permite observar datos interesantes respecto a grandes muestras de población, como temas de moda o el desarrollo de corrients políticas. Una de ellas es Reddit, un servicio web de agregación de contenidos que ha ganado bastante popularidad en los últimos años. Se han realizado análisis estadísticos sobre los títulos del contenido más popular publicado en dos periodos específicos: Agosto de 2013 i 2016. Los resultados muestran unos patrones interesantes respecto a las palabras y combinaciones de palabras más frecuentes, lo cual apunta a unos resultados prometedores si se realizaran investigaciones más extensivas.

Index

Index of figures.

Index of tables.

Glossary of terms

API: Application user interface.

Bigram: Pair of elements associated in a certain context, such as contiguous characters, words or sentences.

CSV: Comma-separated value(s).

Dataset: A collection of data grouped within a single online location or a single local file.

GB: Giga-byte, approximately a thousand kilobytes or a million of bytes.

GNU: A free operative system with its associated collection of open software and legal texts.

NSFW: Not Safe For Work; common expression used to signal content that shouldn't be openly displayed without the viewer's consent.

RFC: Request For Comments; type of publication used by the Internet Engineering Task Force and the Internet Society to officially document and discuss technical advances in network-related technologies.

Reddit: America-based website which functions as a news aggregator, content rater and discussion hoster, with their active users numbering in the hundreds of millions.

SQL: Structured Query Language, a programming language used in most database-related environments.

Subreddit: Basic unit of structured community in Reddit.

TDM/DTM: Term-document/document-term matrix. In text analytics, matricial structure where the x-axis represents the words and the y-axis the document, or vice-versa. Each intersection contains the amount of times the specific term appears in the specific document.

Upvote/Downvote: Reddit jargon corresponding to the act of voting certain content, either favourably ("up") or disfavourably ("down") thus repercuting in said content's popularity.

1. Introduction/Objectives

This project's objective is to analyse the titles of the top 2.5 million most popular posts from Reddit from August of 2013 and 2016, and derive information of interest relative to the language used in them such as:

- the most commonly used words
- the most common combinations of words
- general shifts in the vocabulary
- general shifts in the theme of the posts

with the usage of statistical tools on two datasets collected from external sources.

All the work was performed on a modified ASUS F556U with an Intel Core i5-6200U processor, and 8 GBs of RAM capacity.

The intent of this word is purely demonstrative, with the intention of showing the potential utility of analyses on such sets of data. Execution of more complex analyses, or studying the actual use that could be given to the results, remains out of the scope of this project.

2. Contextualisation of the work

2.1 Big Data / Data Mining

While the idea of compounding large amounts of data to get interesting conclusions isn't totally new, only with the advent of the so-called "Information Age" has it become relevant and profitable for companies to develop. The sheer amount of information collected by every terminal connected to the internet is generating ever-bigger repositories of data that offer a great potential market to anyone who manages to deal with it effectively, efficiently and with the proper equipment. It may allow for companies to make profit by allowing to optimise gains, minimise costs, reduce risks, optimise feedback from Customer Relationship Management tools, etc.

This is where (and when) the fields of "Big Data" and "Data Mining" enter the stage. Many models and solutions have already been developed by major tech companies to accommodate for the growing analytical needs of the greater companies. One of the main issues with Big Data in its modern form, though, is that it is still a pretty new field, and thus counts with little to no universally agreed definitions that could help us define what it exactly is, and how is it supposed to be done. The current consesus lies around the fact that Big Data projects involve three common questions:

- **Volume,** literally the amount of data to be processed, which nowdays goes around the scale of Peta-, Exa- and Zettabytes.
- **Variety,** the source of the data to be processed, which can be anything, including but not limited to: sensors of all kinds, smartphones, social media APIs, structured databases, unstructured databases, rich text formats, etc.
- **Velocity,** or what is the target speed that is required for the retrieval, storage and processment of all the desired data.

Regarding data mining, it is of note that over time there have been initiatives to standardise projects involving this field of analytics, most notably the **CRISP-DM** model, which is structured into six major steps:

1. **Business Understanding:** Or else, comprehending the goals and requirements of the project from a business perspective, making this knowledge up into a defined data mining problem, and make up a preliminary plan to achieve the objectives.

2. **Data Understanding:** This phase starts off with an initial data collection, and proceeds with familiarisation with the data, identification of data quality problems, and identifying subsets of this raw dataset that could hint for interesting information.

3. **Data Preparation:** This covers the start of the design and construction of the final dataset from the initial raw data, which will be onwards optimised for the tools used in the process.

4. **Modeling:** Here the dataset approaches its final form as modelling techniques get applied to it. Should any issues arise, it may be advisable stepping back to 3 until the result it satisfactory.

5. **Evaluation:** In this phase it is determined if the model built achieves the project's goals, whether the previous steps have been well executed, and the usage of the project's results is discussed and determined.

6. **Deployment:** Publication and usage of the results by the company.

As you may have noticed, this chapter is equivalent to step 1, the following chapter will be equivalent to steps 2 through 4, and the chapter after that will be equivalent to step 5. Step 6 lies beyond this project's scope and will not be further delved into.

2.2 Text Mining

2.1.1 Definition and history

Also known as Text Analytics, it is essentially the process of analysing a set of target text(s) with the purpose of abstracting valuable information from it for the user. From the initial explanations at the introductions of [1,2,4] we can conclude that, broadly speaking, the whole process follows four steps: data collection, manipulation and preparation of the input data, analysis of the formatted data, and publication of the results. Nowadays, the most common way of achieving this goal is through tools of statistical analysis and text formatting that automate fully or partially each of the mentioned steps.

Following the historical contextualisation at [3], text mining as a process started to be applied in the 80s as companies started to realise the grand amount of interesting information that could be extracted from textual sources, but initially wasn't very successful due to the technology not allowing yet for executing the analysis at a cost-effective pace. Thus, during the 80s and 90s, the focus shifted towards analysis of numerical data stored in relational databases. Nonetheless, as the technology progressed, so did the field, and by 1999 professor M.A. Hearst stated in the summary of [5]:

> *«For almost a decade the computational linguistics community has viewed large text collections as a resource to be tapped in order to produce better text analysis algorithms. In this paper, I have attempted to suggest a new emphasis: the use of large online text collections to discover new facts and trends about the world itself. I suggest that to make progress we do not need fully artificial intelligent text analysis; rather, a mixture of computationally-driven and user-guided analysis may open the door to exciting new results.»*

While the former statement did not constitute by any means a prediction, the evolution of text mining in the 2000s and and 2010s has shown that this semi-automated approach to analysis of textual data has proven fruitful, and with a host of potential uses that will surely change how companies, sociologists and psychologists analyse the behaviour of society and make conclusions about it. Nowadays, the field is advancing at an ever-faster pace, as both the theory and practice of data mining keeps being developed by academics and engineers alike.

Many applications involving text analytics have appeared, such as https://databasic.io/, which lets you input files or webpages and then get the most frequent words and combinations of words; though in many cases they have an upper limit to the size of the analysed texts. For bigger requests (like the ones that will be explained in this report) a tailored environment will have to be used.

2.1.2 The R language and the Rstudio Environment

R is, as defined in the opening of [6], an «open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing» which nowadays is regularly used and improved by a growing community of statisticians, developers and data miners.[7] It is a dynamically typed, multi-paradigm, and interpreted (scripted) language that allows branching and looping.[6,9]

As described in [6,8,9] the language was developed by Ross Ihaka and Robert Gentleman, from the University of Auckland, during the early 90s, as a refined implementation of the earlier S language, which in turn had been developed by Bell Labs' employee John Chambers during the 70s. Ever since, it has been improved and expanded by the «R Core Team» as well as an online community that has contributed with internal code and bug reports.

The core of the language includes lots of pre-made statistical functions readily available for the user, including but not limited to «linear and generalized linear models, nonlinear regression models, time series analysis, classical parametric and nonparametric tests, clustering and smoothing» among others. Graphical utilities are included. Furthermore, it includes functions for loading additional pre-loaded functionality through modules or «addon packages» for a great variety of purposes.[9] Most packages are distributed free of charge. R itself operates as free software distributed under a GNU General Public License, and is an official part of the GNU project.[9] Its official website is http://www.r-project.org. Most packages are distributed free of charge as well.

Rstudio is the most used interface for R developers, which includes a fully-flegded integrated development environment (IDE) for R. It counts with both free and commercial releases, the former of the which will be used for the development of this project. Its official website is: https://www.rstudio.com/

2.1.3 The comma-separated value format (CSV)

The CSV is a open data storage format. Its usage was officially documented for the first time around 2005 in an informational RFC memo [10] and has spread ever since, despite never having been fully standardised. Its basic structure consists of representing tabular structures in a textual, simple and easily transferable format. Different rows of the table translate directly to different lines of text, while columns are translated as strings separated from each other by commas (“ , ”) or semicolons (“ ; ”) depending on the designer's discretion. This will be the preferred import/export format throughout the present work.

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

Fig.1. Example of a CSV file, extracted from the corresponding Wikipedia article.

2.2 Reddit

Reddit is a website which serves as an internet content aggregator and rater, as well as a place of discussion through its comment system. Created in 2005 by two University of Virginia students, it has grown to become the top 4th most visited website in the United States, as well as the 9th most visited in the world, counting with over 250 million unique users, most of them from the US, but with an increasing chunk of people from other countries such as Germany, France, the UK or Australia.[11] Signing up is free and doesn't require an email address. Upon registry, each user gets assigned a private personal page following the pattern «www.reddit.com/user/*username*», where all the posts and comments he makes are listed in a public or private manner, at the user's discretion.

Reddit is primarily organised in sections called «subreddits» (also known as «sub-reddits» or just «subs») which are usually thematic in nature (e.g. rocket science, videogames, geopolitical comics, music interests, jokes, etc.) and each one allows users to post content into it – with varying degrees of restriction on *who* can post and comment, depending on moderation policy – and other users to comment to these posts, comments which may in turn be answered to, deriving in «threads» or «trees» of comments that may get large depending on the size of each community. Both posts and threads can be voted favourably («upvoted») or disfavourably («donwvoted») which are added up to make a score, which affects the respective post/comment's popularity.

Users register for the whole of Reddit, and then they can subscribe to any subreddit at will, which will appear in their daily feed. Conversely, they can unsubscribe («unsub») to remove said subreddit from the feed.

Additionally, registered users may pay through a medium called «Reddit Gold» to sign up for a period of «premium membership». This activates some special temporary features for the user, and the money paid is used to finance the administrators' maintenance budget.[12] Also, registered users may «give gold» (or just «gild») other people's posts or comments, which means they consider it so valuable or entertaining that they are paying for it. This adds an icon at the side of the post/comment's score, and gives its author access to the premium features for a period of time proportional to the total amount deposited into it.

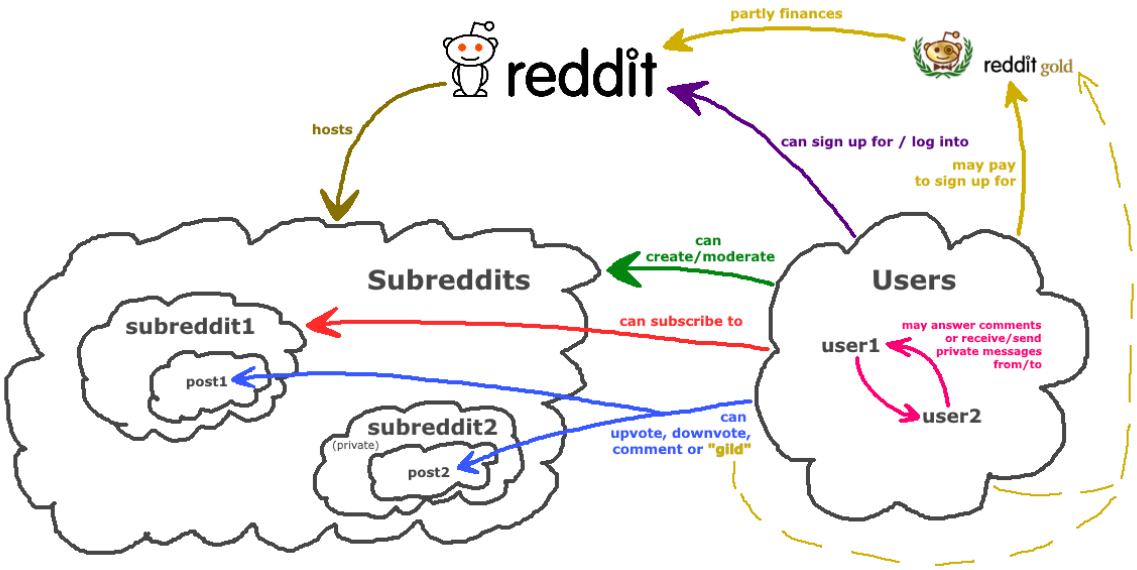

Fig. 2. Simple visualisation of Reddit's basic structure

Subreddits can be usually accessed with a URL that follows the pattern «www.reddit.com/r/*subredditname*», which also serves as a base to locate the URLs of posts and comments published within that specific community. Subs can be freely created by any user at will, and can be either public (set by default) or private, where in the latter case the content will not be visible to anyone but those approved by the moderators of the sub itself. In theory there's no restriction to the theme of any given subreddit, though the official policy of Reddit's administrators is to remove by force any subs involving criminal activity.

Given its sheer size and thematic organisation, Reddit can act as a huge sample of information from the part of human society that is able to operate in the Internet. This could potentially allow for the tracking of trends that are invisible to the human eye due to their long-term, large-group effects, such as general analyses of shifts in the interests or vocabulary of the general population. Those could be then used to conduct sociological surveys, take business decisions or build marketing campaigns.

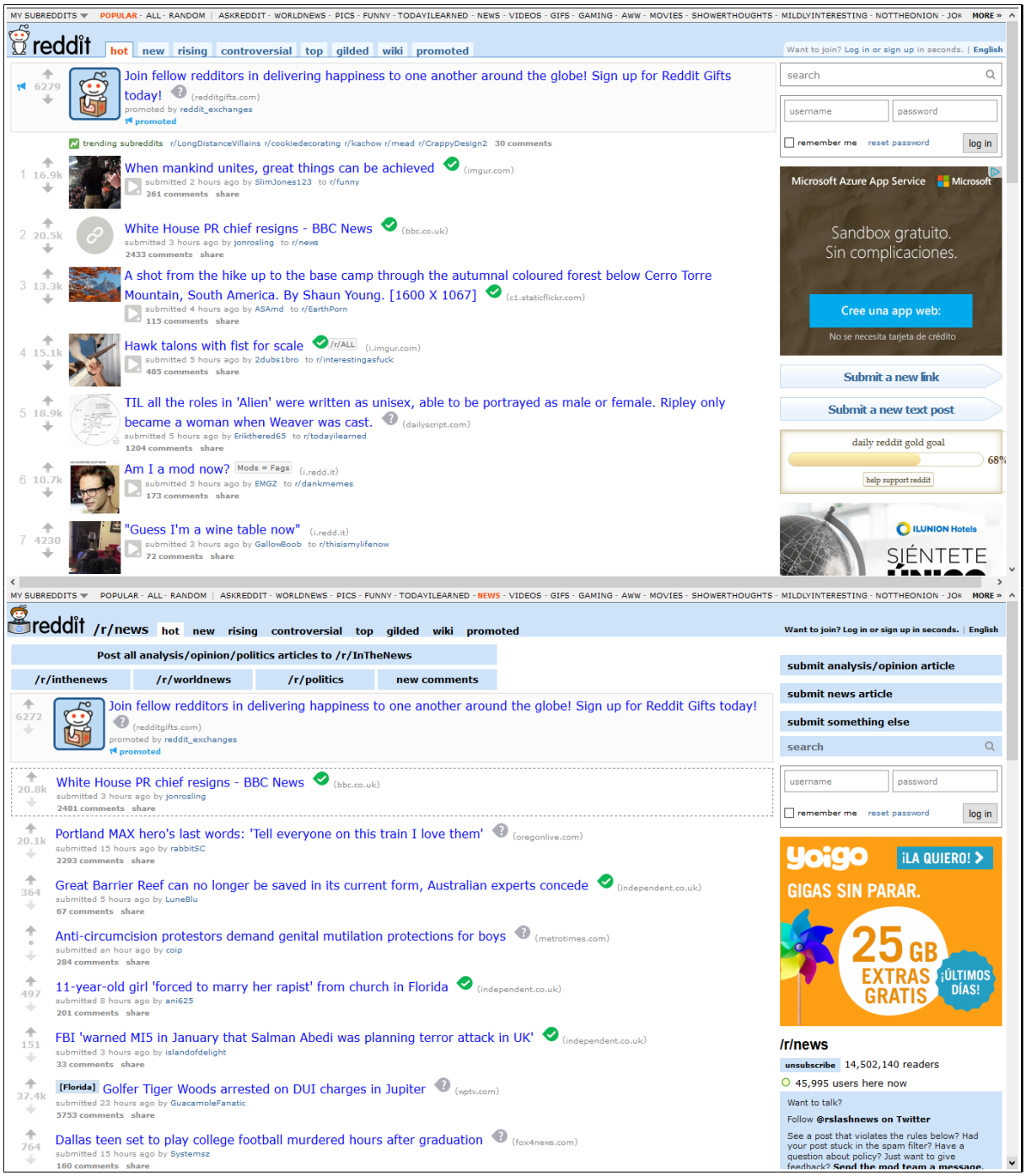

Fig. 3, 4. Respectively, examples of the default reddit.com *page and of* reddit.com/r/news.

3.Work on the project.

The work started by getting, upon the tutor's suggestion, the book "Automated Data Collection with R" by S. Munzert et al., which discusses ways of extracting and processing information with R from the various formats of data used in the internet. Over the next weeks, we discussed ways of extracting the desired information from Reddit, in order to get a similar set of information to the static one we started with, as well as many aspects of its sampling and analysis. This chapter will describe the process by which the data were gathered, organised, formatted and analysed.

3.1 Gathering of the information

In short, the data have been obtained from two sources: first through a CSV-based dataset from 2013, and then a sample from the Google BigQuery service downloaded during April and May of 2017. The data of interest are the titles of the Reddit posts. The output is all in CSV, easily importable to Rstudio via the read.csv() command from the core packages of R.

3.1.1 The 2013 CSV dataset

This set of information has been extracted from a Github repository, released under a Public Domain Dedication and License, which allows for its free usage.[13] All of the data files can be found at: https://github.com/umbrae/reddit-top-2.5-million/tree/master/data

According to the author, the data have been sampled from reddit's 2500 top subreddits, ordered by total amount of subscribers, and then the top 1000 posts from each subreddit, ordered by total score, which is the sum of its upvotes and downvotes. The result is 2500 CSV files, each containing 1000 rows with basic data from each selected post, including the title, time of creation, total score, domain of the post's content and author.

3.1.2 Google BigQuery's dataset

Google BigQuery [14] is one of the many services offered within the Google Cloud Platform, which include solutions for computing, storage and databases, bigdata and networking, among other functionalities.[15] Specifically, BigQuery offers a service able to process grand SQL databases in relatively short time, thanks to its massive computing capabilities. Our dataset of interest is one which contains databases with information regarding Reddit's posts, grouped by month of publication; ranging, at the time of this report's writing, from December 2012 to March 2017. This dataset is maintained by Felipe Hoffa, BigQuery Developer Advocate at Google, in collaboration with Jason Michael Baumgartner (also known as stuck_in_the_matrix in reddit) who each month polls the information off Reddit. [16]

In our case, the data for August 2016 have been downloaded for further analysis and comparison with the older dataset. This particular month was chosen to align the polling conditions of both datasets. The raw data can be found at:

https://bigquery.cloud.google.com/table/fh-bigquery:reddit_posts.2016_08

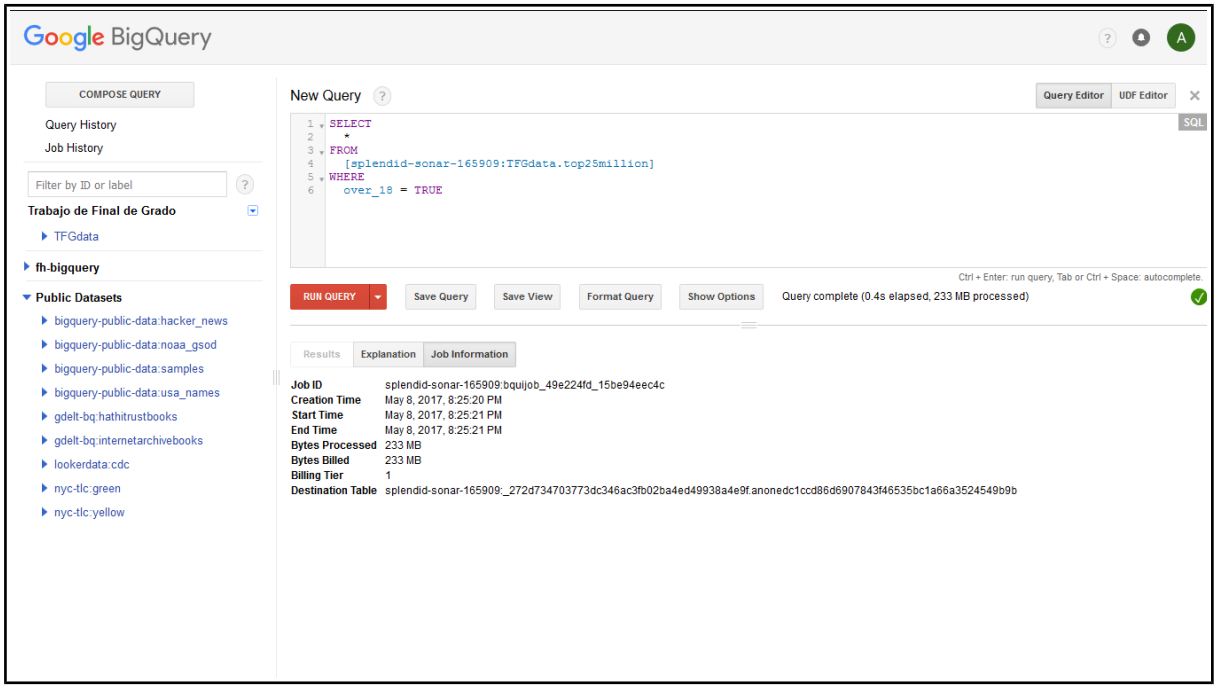

Fig. 5. Example of the BigQuery interface.

3.2 Organisation and formatting of the information

3.2.1 Consolidating the datasets into R dataframes

The sheer amount of data that both datasets contained (~15 GB) was too big to be processed by any home computer. The original idea was to directly translate each of the datasets into a single R dataframe to ease the automation of the analysis, via R scripts that would read and import the data. This immediately proved difficult because the size of the sets exceeded the working laptop's capacities (8GB) by far.

The solution was simple: trim away all the unnecessary data off the datasets. With a SQL statement in BigQuery (fig.1) The columns of the set that could be of interest were able to be isolated, which reduced the size of the data to be handled by an order of magnitude. During the process, it was discovered that old dataset itself had also been uploaded to BigQuery by Hoffa with the same data. This in turn allowed to run another SQL statement in BigQuery (fig. 2) which reduced the load of data to ~20% of its original size. All in all, the new dataset had a size (~1,74 GB) much more manageable for the working laptop's capacity.

```
SELECT created_utc,subreddit,author,score,title,over_18
FROM  [fh-bigquery:reddit_posts.2016_08]

SELECT subr, created_utc, score, title, author, over_18
FROM [fh-bigquery:reddit.top25million]
```

Fig 6. SQL queries to filter unnecessary data off the datasets.

3.2.2 Loading the data into the Rstudio Environment

With the data gathered and the files downloaded, we can proceed to load them into the a preexisting Rstudio project. Given that we're working with CSV's, the read.csv() command from the core R package should suffice.

```
top25million201308 <- read.csv("…/top25million_reduced.csv")

redditposts201608 <- read.csv("…/redditposts_201608.csv")
```

Fig. 7. Examples of commands used to load the data from the CSV files.
(the suspensive dots represent the rest of the file's full address)

After this, all of the desired data should be loaded and ready to be processed however we want them within the environment.

3.2.3 The *tm*, *snowball* and *dplyr* packages

The tm package, developed by I. Feinerer, K. Hornik and D. Meyer from the Wirtschaftsuniversität at Wien, offers basic text-mining functionality for R. It is capable of managing text documents, abstracting document manipulation, and supports multiple text and file formats.[1] To minimize memory consumption, the package has integrated database backend support. It incorporates advanced metadata management for collections of text documents, to ease the usage of large metadata-enriched documents.[1]

The base unit in tm are the Corpuses, which are generated upon importation of external files. Those corpuses are composite values consisting of meta-information about the document(s) imported to the environment, as well as the actual text that they contained.

The tm package also provides preprocessing and manipulation mechanisms, such as removal of white spaces, stemming, or conversion between file formats (e.g., PDF to plain text). Furthermore, it has a generic filter architecture in place to filter documents according to certain criteria, or perform searches within its full text.

Another relevant feature is the converstion of Corpuses into Term-Document Matrices (or TDMs/TDMs for short) which are used to perform more complex text mining tasks, and in turn allows for the straightforward integration of pre-existing classification and clustering methods.

In association to this, the Snowball package, created by Milan Bouchet-Valan, offers additional functionality for stemming: this is, altering the text so that declensions of the same word are added to the same count later on. And the dplyr package, developed by H.Wickham, R. Francois and maintained by the Rstudio team, adds further functionality to manipulate R data-frames.

It is worth of mention that all these packages are distributed free of charge. From now on, it will be assumed that all the tasks are performed in an environment that has all of these packages loaded.

3.2.4 Polling the data

Another potential issue is the way the data has been polled in each database. While in the 2013 dataset the posts are grouped by subreddit and then by popularity measured by the total score (sum of upvotes minus downvotes) in the BigQuery dataset the posts are grouped by month of publication, with no regard of their popularity – since it is a compilation of all posts published for each month. Furthermore, the old dataset filtered out mature («NSFW») content from its own polling of from Reddit, while the BigQuery dataset doesn't account for such restrictions – though its tables do include a «over_18» column which allows to mark individual rows as including mature content.

We can compensate for these circumstances by generating from the downloaded BigQuery dataset a set of data statistically equivalent to the 2013 one. This could be achieved by:

- taking a dataset from the same time of the year (as explained in the previous section)
- ordering the posts by total score
- taking the top 2.5 million posts of it and filtering the rest
- all while ignoring rows with the over_18 variable set to true.

The script to achieve this is shown below.

```
#Import the titles of the 2013 posts
top2500K <- Corpus(VectorSource(top25million201308$title))

#get the top 2.5 million posts of the 2016 dataset
#while ignoring mature (over_18) posts
redditposts201608 <- top_n(
                        redditposts201608[
                              redditposts201608$over_18=='false',],
                              2500000,score)

#Import the titles of the 2016 posts
posts201608 <- Corpus(VectorSource(redditposts201608$title))
```

Fig. 8. Script used to fine-tune both datasets into Corpuses as similar as possible.

With these measures taken, the newer dataset can be granted to be as similar as possible to the older, and the analyses performed on them to be statistically equivalent. The two generated variables, top2500K and posts2016, will constitute the basis for all forthcoming analyses.

3.3 Analysis of the data

Note: all the precise results are stored for further interpretation in section 3.4.

3.3.1 Text Frequency Analysis with *tm* (First Attempt)

Before performing the actual analyses, some preparations must be made to ease the task to the tools of the tm package. First, we perform a stemming of the text, eliminate needless white spaces, convert all the text to lower case, and remove stop words (i.e. "the", "at", "it") that hold no meaning in themselves. Finally, the TDMs are generated to obtain our targets for analysis. The script that was used to perform it is shown below. Notice that some additional languages have been configured to avoid the parser to get confused with non-English words.

```
top2500K_TDM <- TermDocumentMatrix(top2500K,
                        control=list(
                              tolower=TRUE, #convert to lower case
language = parse_IETF_language_tag(c("en","es","fr","pt","ru","de")),
#tag some of the major western languages as present in the text
                              removePunctuation=TRUE,
                              stopwords=TRUE, #remove stop-words
                              stemming=TRUE #perform a stemming
                                    ))


posts201608_TDM <- TermDocumentMatrix(posts201608,
                        control=list(
                              tolower=TRUE, #convert to lower case
language = parse_IETF_language_tag(c("en","es","fr","pt","ru","de")),
#tag some of the major western languages as present in the text
                              removePunctuation=TRUE,
                              stopwords=TRUE, #remove stop-words
                              stemming=TRUE #perform a stemming
                                    ))
```

Fig. 9. Script to generate the target environment for analysis.

```
#Convert all the text to lower-case letters
tm_map(top250OK,tolower)
tm_map(posts201608,tolower)

#Remove all punctuation marks
tm_map(top250OK,removePunctuation)
tm_map(posts201608,removePunctuation)

#Stem the document
tm_map(top250OK,stemDocument)
tm_map(posts201608,stemDocument)

#Reduce any set of consecutive spaces to just one space
tm_map(top250OK,stripWhitespace)
tm_map(posts201608,stripWhitespace)

#Defining a custom set of stop words to be removed from the documents
{
myStopwords <- c(stopwords(kind = 'en'),
                 stopwords(kind = 'es'),
                 stopwords(kind = 'fr'),
                 stopwords(kind = 'pt'),
                 stopwords(kind = 'ru'),
                 stopwords(kind = 'de'))
}

#Remove the words from the documents
tm_map(top250OK, removeWords, myStopwords)
tm_map(posts201608, removeWords, myStopwords)

#Generate the Term-Document Matrices
top250OK_TDM <- TermDocumentMatrix(top250OK)
posts201608_TDM <- TermDocumentMatrix(posts201608)
```

Fig. 10. Alternative script to generate the target environment for analysis. Notice this one performs the preprocessing on the corpus itself, rather than operating on the output file, as the other script does. This may or may not be preferred depending on the planned posterior use of the environment files.

Now, the following script allows to derive sets of words that appear at least in 10% of the titles from the term-document matrices:

```
top250OK_MostFrequentWords <- findFreqTerms(top250OK_TDM,22671,Inf)

posts201608_MostFrequentWords <- findFreqTerms(posts201608_TDM,25000,Inf)

commonFrequentWords <- intersect(top250OK_MostFrequentWords,
                                 posts201608_MostFrequentWords)

#Print results on screen
print(top250OK_MostFrequentWords)
print(posts201608_MostFrequentWords)
print(commonFrequentWords)

#Print unique words for each set
print(setdiff(top250OK_MostFrequentWords,commonFrequentWords))
print(setdiff(posts201608_MostFrequentWords,commonFrequentWords))
```

Fig. 11. Script to find the most frequent words, their similarities/differences, and print the result on screen.

This yields 34 words for the 2013 corpus and 39 for the 2016 one, with 28 being shared.

For the 2013 corpus, the words are, in alphabetical order:

(Words unique to this set in **underlined bold**)

amp, anyon, best, can, day, don, ever, feel, first, found, ***free****, friend, game, get, good, got, guy, help, just, know,* ***last****, like, live, look, love, made, make, man, need, new, now, old, one, people, play, post,* ***reddit****, say, see, show, start, take, thing, think,* ***thought****, time, today, tri, use, video, want, week, will, work, year.*

And for the 2016 corpus, the words are, in alphabetical order:

(Words unique to this set in **underlined bold**)

2016*, amp, anyon,* ***back****, best, can,* ***clinton****,* ***come****, day, don, ever, feel,* ***find****, first, found, friend, game, get, good, got, guy, help,* ***hillary****, just, know,* ***life****, like, live, look, love, made, make, man, need, new, now, old, one, people, play, post,* ***question****,* ***right****, say, see, show,* ***someon****, start, take, thing, think, time, today, tri,* ***trump****, use, video, want,* ***way****, week, will, work,* ***world****, year.*

The two sets share these words:

amp, anyon, best, can, day, don, first, game, get, good, got, help, just, know, like, made, make, need, new, now, one, people, post, think, time, today, video, will.

Furthermore, a script analysing the most common characters was elaborated. Both the generator script and its results are shown below.

```
#Generate the tables with absolute frequencies
AbsCharFreqs2013 <- data.frame(
                              table(
                              strsplit(
                              paste(top2500K
                                          $content, collapse = ''),"")))

AbsCharFreqs2016 <- data.frame(
                              table(
                              strsplit(
                              paste(posts201608
                                          $content, collapse = ''),"")))

#Arrange the elements of the tables by frequency
AbsCharFreqs2013_Ordered <- arrange(AbsCharFreqs2013,desc(Freq))[2:206,]
AbsCharFreqs2016_Ordered <- arrange(AbsCharFreqs2016,desc(Freq))[2:211,]

#It is taken from the second character onwards
#because the first one is the space (“ “) character,
#which is a trivial expected result, and thus can be ignored.

#Calculating the total amount of characters in each set
TotalChars2013 <- sum(AbsCharFreqs2013_Ordered$Freq)
TotalChars2016 <- sum(AbsCharFreqs2016_Ordered$Freq)

#Calculating the relative frequencies of each character
RelCharFreqs2013 <- (AbsCharFreqs2013_Ordered$Freq)*100/TotalChars2013
RelCharFreqs2016 <- (AbsCharFreqs2016_Ordered$Freq)*100/TotalChars2016

#Drawing plots of relative frequencies
barplot(RelCharFreqs2013[1:20],
            las = 2, names.arg = AbsCharFreqs2013_Ordered[1:20,]$Var1,
            col ="lightblue",
            main ="20 most frequent non-space characters 2013",
            ylab = "Relative frequency, percentage")

barplot(RelCharFreqs2016[1:20],
            las = 2, names.arg = AbsCharFreqs2016_Ordered[1:20,]$Var1,
            col ="lightblue",
            main ="20 most frequent non-space characters 2016",
            ylab = "Relative frequency, percentage")
```

Fig. 12. Script used to calculate character frequencies and represent them with bar plots.

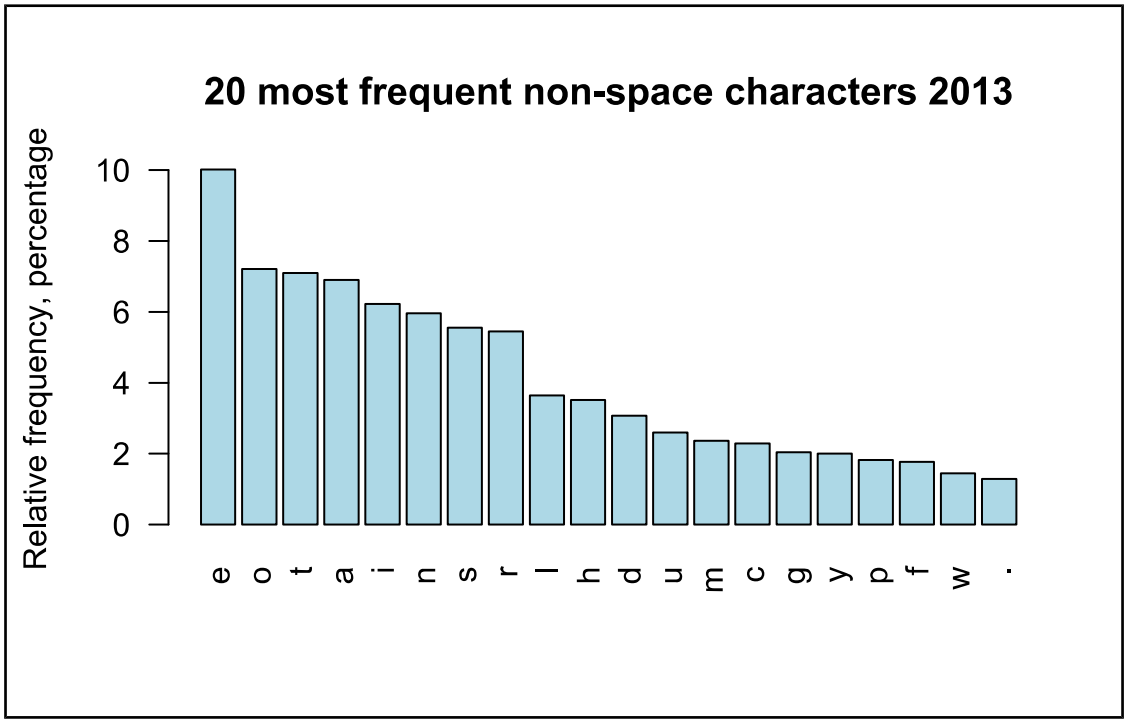

Fig. 13. Character analysis figures for 2013.

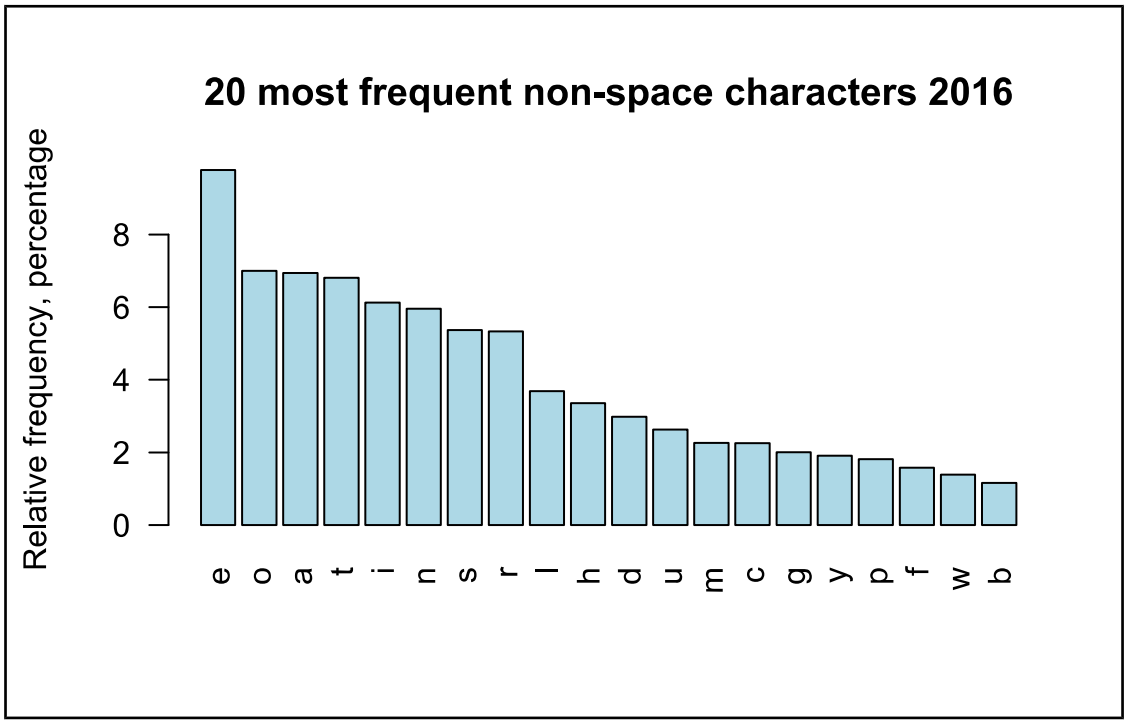

Fig. 14. Character analysis figures for 2016.

3.3.2 Text Frequency Analysis with *tm* (Second Attempt)

While the previous procedure yielded some interesting information, no data regarding the absolute or relative frequency of certain words could be derived. Furthermore, the scripts did not yield the same amount of "top" words. A second attempt was undertaken to devise a script to derive the absolute and relative frequencies of the most frequent words from both data sets. The attempt was successful, and the plots thereafter generated are shown below, as well as the script that calculates the frequencies and generates the plots themselves.

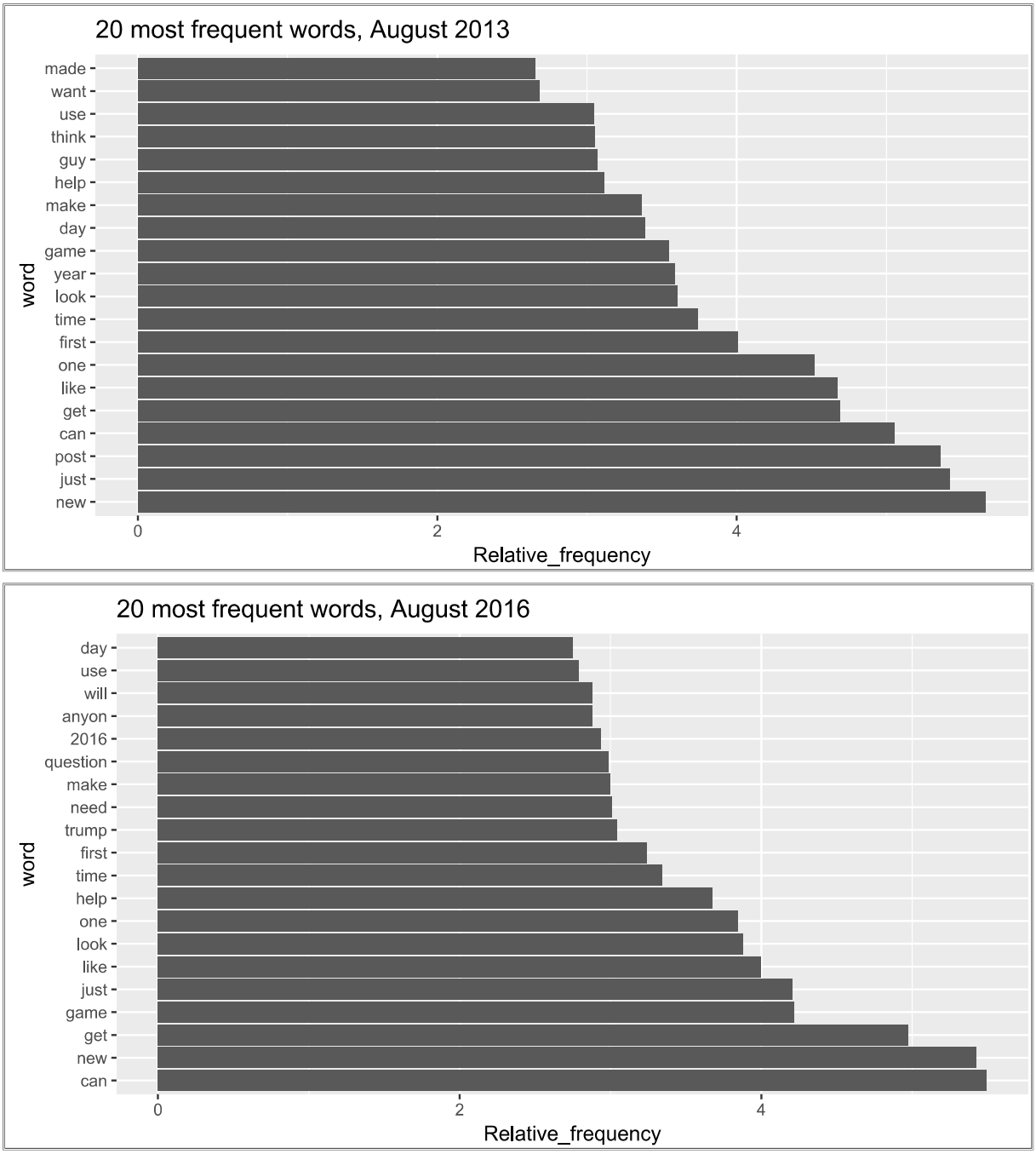

Fig. 15, 16. Generated plots. Notice the frequencies are per-thousand, not absolute.

```
#Generating the clean word sets
top2500K_TDM <- TermDocumentMatrix(top2500K,
                                   control = list(tolower = TRUE,
                                   language = parse_IETF_language_tag(c("en")),
                                   removePunctuation = TRUE,
                                   stopwords = TRUE,
                                   stemming = TRUE))

posts201608_TDM <- TermDocumentMatrix(posts201608,
                                   control = list(tolower = TRUE,
                                   language = parse_IETF_language_tag(c("en")),
                                   removePunctuation = TRUE,
                                   stopwords = TRUE,
                                   stemming = TRUE))

#Getting the absolute frequencies of the words
AbsWordFreqs2013 <- data.frame(word = top2500K_TDM$dimnames$Terms,
          Absolute_frequency = as.vector(slam::row_sums(top2500K_TDM)))

AbsWordFreqs2016 <- data.frame(word = posts201608_TDM$dimnames$Terms,
          Absolute_frequency = as.vector(slam::row_sums(posts201608_TDM)))

#Preparing the dataframes to be properly processed by the ggplot method
AbsWordFreqs2013 <- arrange(AbsWordFreqs2013, desc(Absolute_frequency))
AbsWordFreqs2016 <- arrange(AbsWordFreqs2016, desc(Absolute_frequency))

AbsWordFreqs2013$word <- factor(AbsWordFreqs2013$word,
                                   levels = AbsWordFreqs2013$word[
                                   order(AbsWordFreqs2013$Absolute_frequency,
                                         decreasing = TRUE)])

AbsWordFreqs2016$word <- factor(AbsWordFreqs2016$word,
                                   levels = AbsWordFreqs2016$word[
                                   order(AbsWordFreqs2016$Absolute_frequency,
                                         decreasing = TRUE)])

#Getting the relative frequencies, per thousand
RelWordFreqs2013 <- data.frame(word = AbsWordFreqs2013$word,
                                Relative_frequency =
                                  (AbsWordFreqs2013$Absolute_frequency
                                     /sum(AbsWordFreqs2013
                                           $Absolute_frequency)*1000)

RelWordFreqs2016 <- data.frame(word = AbsWordFreqs2016$word,
                                Relative_frequency =
                                  (AbsWordFreqs2016$Absolute_frequency
                                     /sum(AbsWordFreqs2016
                                           $Absolute_frequency)*1000)

#Generating the plots
plot2013 <- ggplot(data = RelWordFreqs2013[1:20,], mapping = aes(y =
Relative_frequency, x = word))
plot2013 + geom_bar(stat="identity") + coord_flip()

plot2016 <- ggplot(data = RelWordFreqs2016[1:20,], mapping = aes(y =
Relative_frequency, x = word))
plot2016 + geom_bar(stat="identity") + coord_flip()
```

Fig. 17. Script to calculate word frequencies and generate the corresponding graphs.

3.3.3 Analysis of bigrams with the *quanteda* package

Another bit of information that may be interesting is the most frequent combinations of words that appear in the titles, such as "am big", "one pole", "push through", and others. While the tm package supports analysis of such properties to a certain degree, its performance grinds down to a halt once the target texts grow bigger and bigger; this is the so-called "curse of dimensionality". Thus, an alternative tool has been used for this specific task, namely the quanteda package, created, maintained and distributed free of charge by Kenneth Benoit. It offers an optimised functionality for this type of analyses, reducing the computation time significantly. The graphs and scripts used to generate the results are shown below.

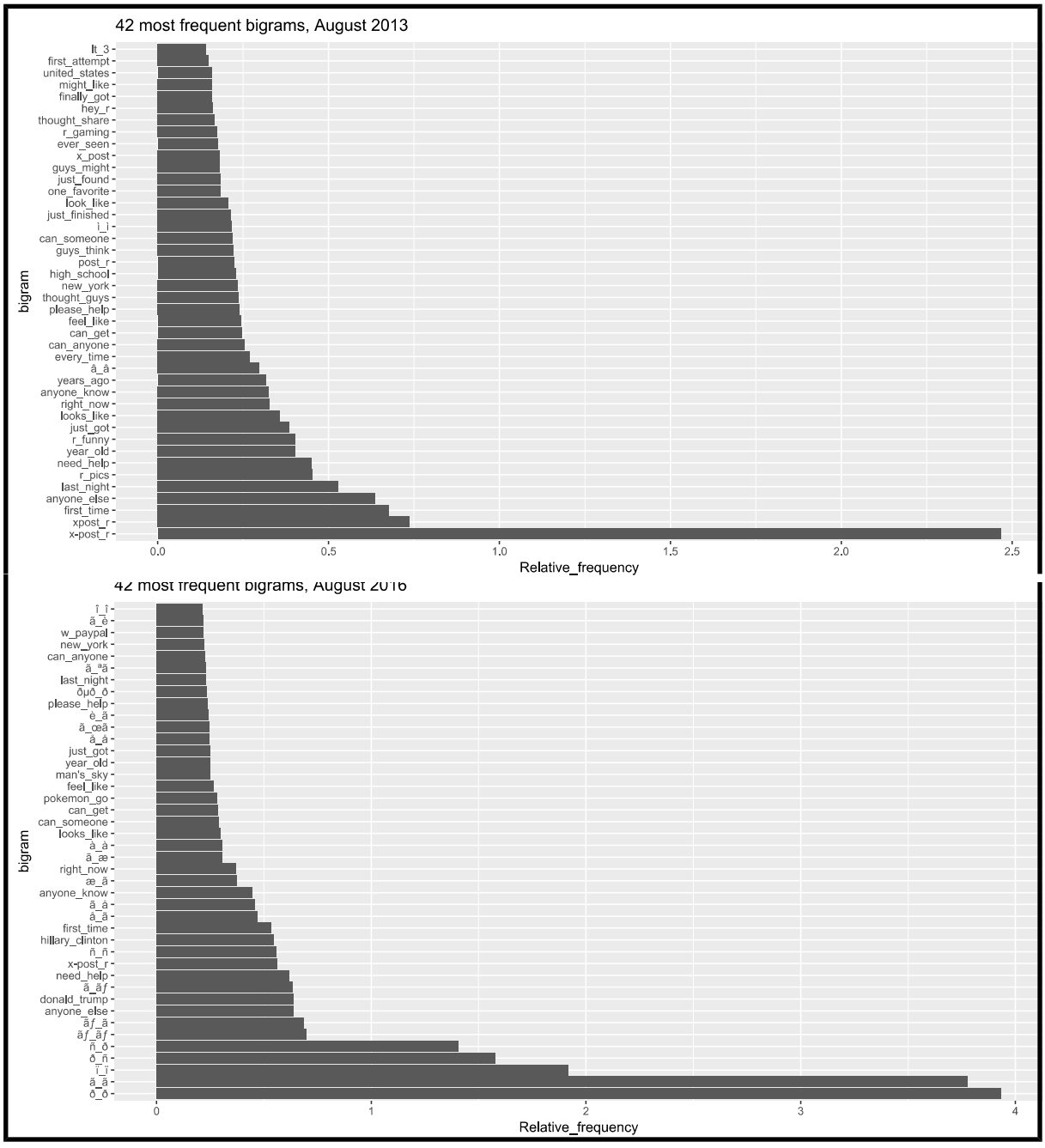

Fig. 18, 19. Graphical output. Note that the frequencies are relative, per thousand.

```
# isolate word tokens
top2500K_tokens <- tokens(char_tolower(top2500K$content), remove_punct = TRUE)

posts201608_tokens <- tokens(char_tolower(posts201608$content), remove_punct = TRUE)

#filter out stop-words
top2500K_tokens <- removeFeatures(top2500K_tokens, stopwords("english"))

posts201608_tokens <- removeFeatures(posts201608_tokens, stopwords("english"))

#compute the bigrams
Bigrams2013 <- tokens_ngrams(top2500K_tokens, 2)

Bigrams2016 <- tokens_ngrams(posts201608_tokens, 2)

#turn the output sets into document-feature matrices
AbsBigramFreqs201308 <- dfm(Bigrams2013, verbose = FALSE)

AbsBigramFreqs201608 <- dfm(Bigrams2016, verbose = FALSE)

#compute the ocurrences of all the bigrams and save the resulting table
AbsBigramFreqs201308 <- data.frame(topfeatures(AbsBigramFreqs201308,
                                    n = length(AbsBigramFreqs201308@Dimnames$features)))

AbsBigramFreqs201608 <- data.frame(topfeatures(AbsBigramFreqs201608,
                                    n = length(AbsBigramFreqs201608@Dimnames$features)))

AbsBigramFreqs201308 <- data.frame(bigram = row.names(AbsBigramFreqs201308),
                        Absolute_frequency = unlist(AbsBigramFreqs201308), row.names = NULL)

AbsBigramFreqs201608 <- data.frame(bigram = row.names(AbsBigramFreqs201608),
                        Absolute_frequency = unlist(AbsBigramFreqs201608), row.names = NULL)

#Preparing the dataframes to be properly processed by the ggplot method
AbsBigramFreqs201308$bigram <- factor(AbsBigramFreqs201308$bigram,
                                    levels = AbsBigramFreqs201308$bigram[
                                            order(AbsBigramFreqs201308
                                            $Absolute_frequency, decreasing = TRUE)])

AbsBigramFreqs201608$bigram <- factor(AbsBigramFreqs201608$bigram,
                                    levels = AbsBigramFreqs201608$bigram[
                                            order(AbsBigramFreqs201608
                                            $Absolute_frequency, decreasing = TRUE)])

#calculate the relative ocurrences of the bigrams
RelBigramFreqs201308 <- data.frame(
                                bigram = AbsBigramFreqs201308$bigram,
                                Relative_frequency = 1000 *
                                            (AbsBigramFreqs201308$Absolute_frequency
                                            /sum(AbsBigramFreqs201308$Absolute_frequency)))

RelBigramFreqs201608 <- data.frame(
                                bigram = AbsBigramFreqs201608$bigram,
                                Relative_frequency = 1000 *
                                            (AbsBigramFreqs201608$Absolute_frequency
                                            /sum(AbsBigramFreqs201608$Absolute_frequency)))

#generate the plots
BigramPlot2013 <- ggplot(
                        data = RelBigramFreqs201308[1:42,],
                        mapping = aes(y = Relative_frequency, x = bigram))

BigramPlot2016 <- ggplot(
                        data = RelBigramFreqs201608[1:42,],
                        mapping = aes(y = Relative_frequency, x = bigram))

BigramPlot2013 + geom_bar(stat="identity") + coord_flip() + ggtitle(
                                                    "42 most frequent bigrams, August 2013")

BigramPlot2016 + geom_bar(stat="identity") + coord_flip() + ggtitle(
                                                    "42 most frequent bigrams, August 2016")
```

Fig. 20. Script to generate the output.

The computation of the bigrams was by far the most time-consuming part of the process, taking about 15-20 minutes each. Still, this is far better than the attempts that were performed with the tm equivalent, where the computation of just one of the two sets went on for 30 minutes before it was interrupted, after estimating that the total computation time would not be worth the wait.

3.4 Drawing up conclusions

3.4.1 Character frequency analyses

Let us compare the numerical results of the character analysis with broader analyses conducted upon the English language:

Reddit August 2013, Relative Frequencies	Reddit August 2016, Relative Frequencies	Relative Frequencies in English [17]	Relative Frequencies in German [18]
e – 10.021560%	**e** – 9.779287%	**e** – 12.702%	**e** – 16.396%
o – 7.208927%	**o** – 6.997390%	**t** – 9.056%	**n** – 9.776%
t – 7.094280%	**a** – 6.939962%	**a** – 8.167%	**s** – 7.270%
a – 6.902828%	**t** – 6.807039%	**o** – 7.507%	**r** – 7.003%
i – 6.227559%	**i** – 6.129486%	**i** – 6.966%	**i** – 6.550%
n – 5.962507%	**n** – 5.957482%	**n** – 6.749%	**a** – 6.516%
s – 5.551534%	**s** – 5.372399%	**s** – 6.327%	**t** – 6.154%
r – 5.449756%	**r** – 5.329489%	**h** – 6.094%	**d** – 5.076%
l – 3.641640%	**l** – 3.690390%	**r** – 5.987%	**h** – 4.577%
h – 3.519061%	**h** – 3.360108%	**d** – 4.253%	**u** – 4.166%
d – 3.075086%	**d** – 2.983715%	**l** – 4.025%	**l** – 3.437%
u – 2.600138%	**u** – 2.628805%	**c** – 2.782%	**g** – 3.009%
m – 2.362098%	**m** – 2.261663%	**u** – 2.758%	**c** – 2.732%
c – 2.288873%	**c** – 2.257496%	**m** – 2.406%	**o** – 2.594%
g – 2.040262%	**g** – 2.005898%	**w** – 2.360%	**m** – 2.534%
y – 2.004611%	**y** – 1.907668%	**f** – 2.228%	**w** – 1.921%
p – 1.824265%	**p** – 1.817927%	**g** – 2.015%	**b** – 1.886%
f – 1.767252%	**f** – 1.577810%	**y** – 1.974%	**f** – 1.656%
w – 1.446336%	**w** – 1.388331%	**p** – 1.929%	**k** – 1.417%
(dot) – 1.287480%	**b** – 1.162955%	**b** – 1.492%	**z** – 1.134%

Table 1. Comparison of the results of the character analyses with external ones

The results show a clear pattern: that both Reddit datasets have almost all the letters in the same order of presence to the English one, and the few that are out of order are just one or two places up or down the list with respect to it. This points out to the fact that the titles were, as of the moment when the datasets were made, mostly written in English.

3.4.2 Word frequency analyses

Let us now compare the exact results of the word frequency analyses.

24 most frequent words August 2013		24 most frequent words August 2016	
new	5.6635700‰	**can**	5.4950457‰
just	5.4180910‰	**new**	5.4230902‰
post	5.3580662‰	**get**	4.9742688‰
can	5.0500850‰	**game**	4.2191177‰
get	4.6897944‰	**just**	4.2056328‰
like	4.6706827‰	**like**	3.9964267‰
one	4.5185679‰	**look**	3.8789393‰
first	4.0064456‰	**one**	3.8441626‰
time	3.7405101‰	**help**	3.6787413‰
look	3.6051711‰	**time**	3.3426575‰
year	3.5835819‰	**first**	3.2427496‰
game	3.5440137‰	**trump**	3.0450629‰
day	3.3859530‰	**need**	3.0087576‰
make	3.3616033‰	**make**	2.9993673‰
help	3.1110279‰	**question**	2.9900862‰
guy	3.0694777‰	**2016**	2.9360923‰
think	3.0533389‰	**anyon**	2.8809518‰
use	3.0468268‰	**will**	2.8792048‰
want	2.6813690‰	**use**	2.7916352‰
made	2.6517813‰	**day**	2.7520542‰
now	2.5831208‰	**year**	2.7381872‰
know	2.5763255‰	**post**	2.5383713‰
will	2.5692471‰	**now**	2.5207919‰
need	2.5541701‰	**people**	2.4936584‰

Table 2. Comparison of relative word frequencies.

Some interesting trends can be noticed here. Many words seem to have dropped in popularity, while others have risen. It is especially remarkable how the expressions “2016” and “Trump” rose to the top 20 ranks. From an insider’s perspective, this can be understood as a result of two very specific circumstances: 2016 being a very eventful year compared to previous one, thus making the people tending to reference the year itself in the title; and the American electoral campaign, which got so much media attention that it boosted the prevalence of the candidates’ names themselves among the titles of the posted content.

3.4.3 Bigram frequency analyses

Now, let us compare the exact results of the bigram frequency analyses.

Most common combinations of two words in Reddit's titles, August 2013			
Places 1-33		***Places 34-66***	
x-post_r	2.46692942‰	ever_seen	0.17569505‰
xpost_r	0.73719145‰	r_gaming	0.17411293‰
first_time	0.67627961‰	thought_share	0.16596498‰
anyone_else	0.63656826‰	hey_r	0.16098128‰
last_night	0.52914192‰	finally_got	0.15900363‰
r_pics	0.45209240‰	might_like	0.15860810‰
need_help	0.45130134‰	united_states	0.15789614‰
year_old	0.40344204‰	first_attempt	0.14919445‰
r_funny	0.40344204‰	lt_3	0.14033455‰
just_got	0.38588045‰	front_page	0.13732851‰
looks_like	0.35843057‰	last_year	0.13709119‰
right_now	0.32757912‰	one_best	0.13448068‰
anyone_know	0.32512682‰	need_advice	0.13131643‰
years_ago	0.31642513‰	best_way	0.13123732‰
â_â	0.29767694‰	r_wtf	0.12854771‰
every_time	0.26975242‰	best_friend	0.12831039‰
can_anyone	0.25345653‰	can_help	0.12783575‰
can_get	0.24570411‰	last_week	0.12657005‰
feel_like	0.24317271‰	years_old	0.12601631‰
please_help	0.23961293‰	hey_guys	0.12593720‰
thought_guys	0.23771438‰	free_shipping	0.12593720‰
new_york	0.23486655‰	star_wars	0.12459239‰
high_school	0.22837984‰	let_know	0.12348491‰
post_r	0.22315882‰	can_make	0.12340580‰
guys_think	0.22110206‰	cake_day	0.12245653‰
can_someone	0.21991547‰	happy_birthday	0.12126993‰
ì_ì	0.21627658‰	just_wanted	0.12071619‰
just_finished	0.21516909‰	1024_x	0.11897585‰
look_like	0.20694204‰	first_post	0.11739372‰
one_favorite	0.18423853‰	north_korea	0.11660266‰
just_found	0.18321015‰	í_ì	0.11509964‰
guys_might	0.18233998‰	guys_like	0.11462500‰
x_post	0.18218177‰	r_atheism	0.11462500‰

Table 3. Comparison of relative bigram frequencies, part 1.

Most common combinations of two words in Reddit's titles, August 2016			
Places 1-33		***Places 34-66***	
ð_ð	3.93159126‰	please_help	0.23912489‰
ã_ã	3.77675290‰	ðµð_ð	0.23560317‰
ï_ï	1.91810565‰	last_night	0.23120102‰
ð_ñ	1.57685083‰	ã_ªã	0.23020320‰
ñ_ð	1.40610603‰	can_anyone	0.22662278‰
ãƒ_ãƒ	0.69806391‰	new_york	0.22227933‰
ãƒ_ã	0.68479876‰	w_paypal	0.21629240‰
anyone_else	0.63889899‰	ã_è	0.21588153‰
donald_trump	0.63637509‰	î_î	0.21523588‰
ã_ãƒ	0.63326423‰	ç_ã	0.21347502‰
need_help	0.61988169‰	years_ago	0.21065765‰
x-post_r	0.56130372‰	â_â	0.21048156‰
ñ_ñ	0.55877982‰	pokã_mon	0.20813374‰
hillary_clinton	0.54510380‰	ð_ðµð	0.20760549‰
first_time	0.53283647‰	ã_ç	0.20537506‰
å_ã	0.46856505‰	need_advice	0.20314464‰
ã_å	0.45618033‰	ªã_ã	0.19023166‰
anyone_know	0.44414779‰	é_ã	0.18952732‰
æ_ã	0.37336118‰	look_like	0.18265996‰
right_now	0.36796121‰	œã_ã	0.18265996‰
ã_æ	0.30597891‰	discussion_thread	0.17984258‰
à_à	0.30374849‰	xbox_one	0.17438391‰
looks_like	0.29969851‰	ã_é	0.17127306‰
can_someone	0.29153985‰	ã_ÿã	0.17109697‰
can_get	0.28572901‰	ðºð_ð	0.17109697‰
pokemon_go	0.28027034‰	best_way	0.16968828‰
feel_like	0.26647693‰	ð_ðµñ	0.16839699‰
man's_sky	0.25162701‰	windows_10	0.16358397‰
year_old	0.25086397‰	ã_ä	0.16129485‰
just_got	0.24839876‰	suicide_squad	0.16123615‰
å_å	0.24587486‰	thread_august	0.16017964‰
ã_œã	0.24446617‰	clinton_foundation	0.15830138‰
è_ã	0.24047489‰	match_thread	0.14937969‰

Table 4. Comparison of relative bigram frequencies, part 2.

Note of the author: *The character composition of some bigrams suggests that the encoding of the text may have been broken at some of the previous steps in the process of importation. Still, despite the best attempts at clearing up the issue, the weird characters still appeared.*

With this information in hand, lots of patterns can be noticed, especially when reading it along with the results of the word analyses.

In both cases, there is an important prevalence of:

- Common jargon among Reddit users, i.e. "x-post", "cake day", "match thread", etc.
- Popular themes at the time, i.e. "new york", "united states", "star wars", "north korea" or "lt_3" (reference to a smartphone brand) in 2013; or "donald trump", "hillary clinton", "pokemon go" or "windows 10" in 2016.
- Expressions transpiring the mood/situation of the author, i.e. "first time", "looks like", "need help", "can anyone", "need advice", "look like", etc.

Nonetheless, there is a striking difference between the two. While literal names of subreddits, such as "r/pics", "r/funny", "r/gaming" or "r/atheism" are rather prevalent in 2013, they seem to be completely gone by 2016. Depending on other circumstances, this could be a sign of general changes in the tone of the posts' titles, maybe towards a less pronounced sense of having to direct a post towards the subreddit as a monolithic community, and more towards directing it to the people themselves.

4. Final statement

A textual analysis of the titles of the most popular Reddit posts for August of 2013 and 2016 has been performed. The top most recurrent characters, words and bigrams among them have been determined and listed. The shortcomings of using a mere home laptop to compute these amounts of data has been evident; the original plan of analysing three different months had to be cut down to just two in order to get all the analyses done in time. Also, many meaningful variables from the imported datasets had to be sacrificed in exchange for being able to have reasonable computation times, with a reduction from 32 to 6 columns. Nonetheless, some interesting trends have been noticed in the resulting ranks of individual characters, words and word combinations, which signals for much more interesting information to be extracted if more complex analyses were to be undertaken on a larger scale, such as analysing the top words for every month available and with tools specifically designed for doing it, comparing the frequencies of words for languages other than English, analysing separately the titles of NSFW content, comparing the frequencies for each individual subreddit through time, etc. This adds to the mounting evidence of data mining being a field full of opportunities, and ripe for the taking by anyone who finds a way to process these huge amounts of data and getting strategically important information from it.

5. Exporting the results

The results have been exported into CSVs with the write.csv() command from the core R functionality. They are publicly available for download at:

https://www.dropbox.com/s/7emt1xrg436lxcg/TFGResults.zip?dl=0

6. References.

[1] FEINERER, Ingo; HORNIK, Kurt; MEYER, David. *Text Mining Infrastructure in R.* Journal of Statistical Software, [S.l.], v. 25, Issue 5, p. 1 - 54, mar. 2008. ISSN 1548-7660. Available at: <https://www.jstatsoft.org/v025/i05>. doi:http://dx.doi.org/10.18637/jss.v025.i05.

[2] Jeffrey L. Solka, *Text Data Mining: Theory and Methods*, 2008. Statistics Surveys Vol. 2 (2008) 94–112 **ISSN:** 1935-7516 Available online at: https://arxiv.org/pdf/0807.2569.pdf

[3] *Text mining*, Wikipedia. [Online]. Available at: https://en.wikipedia.org/wiki/Text_mining

[4] S. Munzert, C. Rubba, P. Meissner, D. Nyhuis, *Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining,* Wiley, 2015. **ISBN-13:** 978-1118834817

[5] Hearst, Marti A. (1999). *"Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics": 3–10.* doi:10.3115/1034678.1034679. **ISBN** 1-55860-609-2. [Online]. Available at: http://people.ischool.berkeley.edu/~hearst/papers/acl99/acl99-tdm.html

[6] *R (programming language)*, Wikipedia. [Online]. Available at: https://en.wikipedia.org/wiki/R_(programming_language)

[7]:

- *Vance, Ashlee (2009-01-06). "Data Analysts Captivated by R's Power". New York Times.* [Online].
  Available at: https://www.nytimes.com/2009/01/07/technology/business-computing/07program.html
- David Smith (2012); *R Tops Data Mining Software Poll*, Java Developers Journal, May 31, 2012. [Online]. Available at: http://java.sys-con.com/node/2288420
- Karl Rexer, Heather Allen, & Paul Gearan (2011); *2011 Data Miner Survey Summary*, presented at Predictive Analytics World, Oct. 2011. [Online].
  Available at: http://www.rexeranalytics.com/Data-Miner-Survey-Results-2011.html
- *Robert A. Muenchen (2012). "The Popularity of Data Analysis Software".* [Online].
  Available at: http://r4stats.com/articles/popularity/
- *Tippmann, Sylvia (29 December 2014). "Programming tools: Adventures with R". Nature. **517**: 109–110. doi:10.1038/517109a.* [Online]. Available at:
  http://www.nature.com/news/programming-tools-adventures-with-r-1.16609

[8] *"R: What is R?". R-Project. Retrieved 7 February 2016.* [Online].
Available at: https://www.r-project.org/about.html

[9] *Hornik, Kurt (November 26, 2015). "R FAQ". The Comprehensive R Archive Network. 2.1 What is R?.* [Online].
Available at: https://cran.r-project.org/doc/FAQ/R-FAQ.html#What-is-R_003f

[10] https://tools.ietf.org/html/rfc4180

[11] http://www.alexa.com/siteinfo/reddit.com

[12] https://www.reddit.com/gold/about

[13] https://github.com/umbrae/reddit-top-2.5-million/blob/master/LICENSE

[14] https://cloud.google.com/bigquery/

[15] https://cloud.google.com/products/

[16]
- http://minimaxir.com/2015/10/reddit-bigquery/
- https://www.reddit.com/r/datasets/comments/6607j2/reddit_march_submissions_and_comments_are_now/
- https://www.reddit.com/r/bigquery/comments/663uky/reddit_comments_and_posts_updated_in_bigquery_up/
- https://www.reddit.com/r/datasets/comments/5z34a8/february_2017_reddit_comments_are_now_available/
- https://www.reddit.com/r/bigquery/comments/5z957b/more_than_3_billion_reddit_comments_loaded_on/
- https://pushshift.io/

[17] http://en.algoritmy.net/article/40379/Letter-frequency-English

[18] https://en.wikipedia.org/wiki/Letter_frequency#Relative_frequencies_of_letters_in_other_languages

7. Bibliography

- FEINERER, Ingo; HORNIK, Kurt; MEYER, David. *Text Mining Infrastructure in R.* **Journal of Statistical Software**, [S.l.], v. 25, Issue 5, p. 1 - 54, mar. 2008. ISSN 1548-7660. Available at: <https://www.jstatsoft.org/v025/i05>. doi:http://dx.doi.org/10.18637/jss.v025.i05.

- Jeffrey L. Solka, *Text Data Mining: Theory and Methods*, 2008. Statistics Surveys Vol. 2 (2008) 94–112 **ISSN:** 1935-7516

- S. Munzert, C. Rubba, P. Meissner, D. Nyhuis, *Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining,* Wiley, 2015. **ISBN-13:** 978-1118834817

Escola Universitària Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Grau en Enginyeria Informàtica

Text Mining Reddit's Top Posts: The Potential Information In Internet-Based Communities

Economical annex

ALEX FERNÁNDEZ PAWLUKOJC – Researcher
XAVIER FONT ARAGONES – Tutor

SPRING 2017

TecnoCampus Mataró-Maresme

A1. Costs assessment

1.1 Human resource costs

Type	Man-hours	Price/hour (€)	Total
Project developer	200	4	800 €

1.2 Equipment amortization

Equipment	Hours of use	Price/hour (€)	Total
Laptop	366	0,006	2,2 €

1.3 Estimated total cost

Human resource cost	800 €
Equipment amortization	2,2 €
Subtotal	802,2 €
Indirect costs (+25%)	200,55 €
Total cost	**1002,75€**

A2. Intellectual rights