Natural Language Processing Fundamentals: A Beginner-Friendly Guide to Analyzing Text Data

An Honors Capstone

Written By

Alexander McWilliams

Advised By

Reagan Mozer

Bentley University

Waltham, Massachusetts

April, 2021

Abstract

reddit.com is the 19th most popular website in the world as of March 2021. The website hosts over 2 million unique communities, each devoted to unique topics. The 'subreddits' have a wide ranging scope – popular favorites include /r/science and /r/pics, each boasting communities of ~25,000,000 unique users. Some subreddits have smaller, devoted followings, like /r/DarkSky and /r/linguistics. Reddit enables image and video sharing, but the character of the site shines through in the comment sections. Users are encouraged to engage with one another, and many users do exactly that – any given comment section might include anything from heated debate to fervent pun-offs, and everything in between. It is, naturally, a treasure trove of text data. This project focuses on text data that was mined from /r/movies. Programming can be intimidating to beginners, and the goal of this project is to help reduce some of that intimidation. It does so by walking through and breaking down, step-by-step, the methods used to construct text analyses with RStudio. It includes engaging data visualizations and annotated code output, again, to make the process of data analysis more accessible. If people realize that they can, with a bit of time and effort, use and benefit from using the powerful tools that are freely available through RStudio/CRAN, the world of data will be better off. Ever abundant is the world of data, but all too scarce are the data literate.

Introduction

This paper is intended to provide its readers with an easy to understand introduction to Natural Language Processing using the open source software RStudio. When I entered Bentley, I had no familiarity with programming. After several courses, countless hours spent reading documentation, and more than a few visits to stackoverflow, I feel somewhat confident in my programing ability. I'm certainly not an expert – I'm probably not even intermediate - but I know that I have the ability to learn more. Certainly, some concepts are harder and more time consuming than others, but I've acquired the tools necessary to tackle those challenging concepts. I wanted to create something that would allow me to share those tools, and this desire coincided with my capstone project.

I've always enjoyed talking to my non-technical friends about the concepts I've been exposed to in my courses, and Natural Language Processing (NLP) in particular is a field that seems to consistently fascinate and mystify. The idea that a machine can process text data seems to baffle many in the same way that it baffled (and sometimes still baffles) me. The interest exhibited by my non-programmer peers combined with guidance from my advisor, a specialist in the NLP domain, geared me towards creating some sort of easily digestible NLP content. So, in the spirit of this paper being geared towards general consumption, it's important that I start with "why." That is, *why* should you be interested in programming, RStudio, or Natural Language Processing? I'm running the risk of sounding dramatic, but you should care because it affects you, your parents, your friends, their parents, and nearly everyone else.

When people can use intuitive and powerful tools, there's a decrease in friction – life moves faster and simpler. To see this in real time, one need look no further than one's own pocket. A smartphone has the potential to make life easier by several orders of magnitude. A city-living executive in need of a car can use her phone to call an Uber. No more subways, no more waiting for taxis. In this case, life moves faster. A farmer in rural India whose child has fallen sick after eating an unknown plant can turn on his phone and use Google to search his child's symptoms, find out whether the plant was poisonous, and look up ways to teach his child not to eat mysterious plants. Here, life is easier. Rewind the clock ten years and these tasks become far more complicated. Our executive would need to get in touch with a driving agency or run outside to hail a cab in the potentially arduous weather. Friction. Our farmer would have needed to travel to the nearest doctor for advice, which might be several miles away. Again - friction. Technology, when wielded correctly, can reduce friction, and, in turn, change the world for the better. Fortunately, the cell phone isn't alone in this wonderous capacity for change! There are plenty of technologies that, when combined with intuitive design and competent users, drastically improve lives. So, why should you care about programming? Without programming, these technologies don't exist!

Computers understand code – that's how they operate. To create computer-readable code, programming is necessary. When computers get readable code, magic (software) happens. If you live in an area with electricity, you're probably using some sort of software to make your life easier. Modern mobile phones are just computers with five-inch screens. TV's, too! The thermostat, the oven, the microwave – computers! Programming, code, and software are all fundamental parts of the modern world. Without them, we'd have no Facebook, Wikipedia, or E-Mail. Yikes! And yet programming is still neglected! Is there any other idea so baked into the

modern human experience, yet so utterly ignored? Take food, for example. Nutrition is a central tenant of our lives – some people don't care about what they intake, but it's safe to say that most people want to know what it is they're eating. What about our cars? Sure, we don't *quite* understand what horsepower is (to be fair, the term is confusing – couldn't they have dropped the 'horse'? I digress), but we generally know when our car is in good shape, and why! There's oil to be checked, gas and tires to be filled, and a battery to keep charged. Of course, not everyone has a car, but humans are now outnumbered by smart devices! Is there any such degree of familiarity with an iPhone? Can we tell when our phone's LEDs are worn out? What about our CPUs and GPUs? Do we know if our Wi-Fi driver is in need of an update? No! Of course, you could argue that we might just take our devices to the proverbial mechanics, but doing so treats the symptoms, not the disease. Until programming is as widespread and accessible a subject as world history, companies will always have the ability to charge $700 for a $20 task (Rossmann, 2015). Of course, there are non-financial incentives to this knowledge, too. Herein lies the importance of familiarizing oneself with Natural Language Processing.

In a social word, the efficient exchange of information is crucial, and language is the wonderful way by which humans complete this task. However, the nuances of human language that make it such a powerful tool for information transmission are also what make it so difficult to operationalize in the context of computing. Supercomputers are unrivaled when it comes to mathematical processing, but they tend to fall behind humans in the realm of abstract communication. Yelling at the computer to "make a grilled cheese," isn't particularly productive – there's not going to be any sandwich sitting on your desk, no matter how red your face gets. As such, scientists tend to opt for language that computers *can* understand. Binary code. 0's and 1's. In recent years, however, there have been many exciting developments in the field of computing

known as “Natural Language Processing” It is of the opinion of the author that NLP is an immensely powerful tool that, if utilized correctly, can fundamentally change the way humans and computers interact. Fortunately, it’s also a lot less complicated than it sounds! In the spirit of the title of this paper (NLP for dummies), it’s important to define some key terms, before things start to get scary. So, first off, here are some common questions:

What’s a “natural” language?

- Well, the words on this paper are a great example! Natural language is defined by John Lyons, one of the fathers of the field of linguistics, as “Any language that has developed that has evolved naturally through use or repetition without conscious planning or premeditation” (Lyons, 1991). What counts? Spoken word, written word, or signed words are all good, so long as they’ve developed naturally. This “developed naturally” bit might bring about some confusion, though.

Are there “unnatural” languages?

- Kind of. They’re commonly referred to as “constructed” or “formal” languages. They’re arduously designed and have very strict rules. These languages can be very useful – think of coding! It wouldn’t make sense for computer scientists to convey to their computers what they wanted done by speaking English, because computers don’t know what English is. Instead, computer scientists develop programming languages, which are constructed language that follow a precise set of rules.

Is one better than the other? “Unnatural” sounds… well, unnatural!

- Not at all! They’re both extremely handy when used correctly. For example: a constructed language, like Python, guarantees users consistent reproduction, something that natural languages sometimes fail at. If someone inputs print(“Hello, world!”) into their Python terminal, their computer will output “Hello World!” one hundred times out of one hundred. Take, on the other hand, Homer, a human. Homer’s friends love his jokes! They think he’s a riot, and he loves to make them laugh. Think of Homer’s jokes as inputs. When he gives these inputs to his friends, he receives a consistent output – laughter! If Homer decides to try to share his inputs with others, though, he might not receive the same output - imagine making a fart joke at a funeral! Alternatively, computers aren’t always too smart either. They might understand a user input like “print,” but, unless they’re guided in very specific ways, they can actually be quite clueless. Ex: Homer would know what to do if, after leaving his bed unmade, he got a condescending look from his mother, and a “really, Homer?”– he’d make his bed. A computer, on the other hand, has a hard time reading social cues. While it might be very obvious to 32 year-old Rita that she would want her bedside lamp to turn off at nighttime, no amount of talking, signing, or writing letters to her Macintosh would make it do any such thing! Homer, on the other hand, would know that people like to sleep with the lights out, and he’d probably shut off Rita’s lamp (if he were in a good mood, after his jokes at the funeral flopped.) Hopefully it’s

clear – both natural and constructed languages have myriad uses when used in proper context.

Well, is there any way to make the computers better at the bits that they're not good at?

- Yes! Just like it's possible to program a computer to automatically save an essay every five minutes, it's theoretically possible to program to do other, equally cool things! It's just a matter of feeding it the proper inputs. As of now, those inputs primarily arrive to the computer in the form of code.Inan ideal world, a computer would be able to understand the intentions of its programmer with absolute clarity. Andy tells his computer that he wants to see how rich he is compared to Jeff Bezos, and his computer calculates Andy's net worth next to Jeff's, and automatically generates an appropriate data visualization. As of now, that's not quite how it works. Andy would have to go and find, download, and clean the data manually and explicitly direct his computer to create the visualization. Clearly, computers can't quite understand the meaning of the phrases they find so easy to decode. Fortunately, there's hope on this front! Programmers have a few techniques they use to help computers understand natural language. Herein lies the good bit – there's an entire field dedicated to the study of helping computers understand natural language! Natural Language Processing is important, and not as complicated as it sounds, and the purpose of this paper is to illustrate just that – anyone can use it, and to great benefit.

The technical lingo and mathematical syntax of computer science can sometimes scare away people who would be otherwise interested in technology. The purpose of this paper is to

provide an accessible introduction to and illustration of NLP method. It's exciting, useful, intuitive, and fun! The literature review section of this research paper will attempt to summarize the history of and current developments in the NLP field and enable widespread understanding of these on goings. Additionally, the paper will include a text analysis of Reddit.com text data. A csv file containing Reddit post text data will be analyzed. Questions will be posed and addressed, such as: "Which actors or movies are mentioned most frequently," "can we estimate comment sentiment by genre," and "how can we visualize text data?" This specific methodology was chosen for its broad appeal – many people have favorite movies, and it's exciting to find more data about those films! Reddit is a massively popular website (several hundred million monthly users), and, as such, many people would feel comfortable reading analyses of data that come from it. Again, the intention of this paper is to create a beginner-friendly guide to natural language processing using RStudio, and Reddit text data is a sufficient launch pad for that kind of guide.

Background

The life story of Natural Language Processing is exciting, filled with disheartening hiccups and hopeful breakthroughs. As is the case with most amazing technology, NLP arose out of necessity. One can imagine a mid-20th America, the world's leading country in a world with war on its mind. Translators were employed to decode enemy country correspondence, but there were only so many military men who spoke both German and English. NLP's first phase, Machine Translation (MT), was born in response to this challenge. The idea made sense – if it were possible to teach computers to translate between languages, any random Joe on the street would become a certified polyglot. Unsurprisingly, machine translation proved more difficult than anticipate. There were two main culprits behind delays in MT development. First; mid-20th century hardware couldn't yet efficiently compute complex actions like those required for MT. Additionally, the synthesis of linguistics and computer science was an undeveloped field, that needed time to grow. MT research started as early as the 1940's, but even by the late 1960's the fastest machines using the most sophisticated algorithms could take as long as seven minutes to process long sentences (Plath, 1967). Any reader familiar with 21st century computers will likely cringe imagining lag to this degree. For contrast, China's Sunway Taihulight was the world's fastest computer in 2016. If it were able to do so, the Sunway Taihulight would probably chuckle if it were asked to translate a sentence from English to Russian. For a detailed breakdown of the specifications (1.31 petabytes of storage, 10,649,000 cores) and abilities of the Sunway Taihulight, read Dongarra's "Report on the Sunway TaihuLight System" (Dongarra, 2016). Developers have come a long way from the days of punch card computing and seven-minute processing times. Computer hardware and software have developed significantly in the years

since NLP's inception, and so, too, has the synthesis of linguistics and computer science. A granular review of the decades of NLP research is presented below.

The first phase (1940-1960) of NLP tasked researchers with the task of translating natural language into a format that a computer could evaluate. They were, in a way, starting from scratch. Thus, a starting point was selected; linguistics, the academic study of language, served as the point of inception. Many of the foundational research efforts in this phase were led by established linguists and language experts who worked to establish a preliminary framework for systematic analysis of text data. Early on, an idea adopted by many researchers was that an understanding of sentence syntax -- the arrangement of the components of a sentence -- would be crucial for the development of a functioning NLP system (Jones, 2001). Programming a computer to understand the structure of a sentence might be a step in the right direction, they thought. If a sentence could be broken down into its component parts, perhaps computers could learn to read and subsequently construct their own sentences by following the syntactical rules provided. Another challenge in this domain concerned the operationalizing of semantics –- the "meaning" of the sentences – within a computer system. Leading researchers like Silvio Ceccato and Margaret Masterman attempted to tackle this avenue by using semantic pattern matching techniques and "world knowledge" to develop a framework by which sentence semantics could be applied to computer science (Ceccato, 1967, pp. 77-135).

Another exciting note in this discussion is Alan Turing's involvement in the early stages of NLP. Turing was a legendary computer scientist who is now widely known as "the father of artificial intelligence." In 1950, Turing published his now-famous "imitation game." The test determines whether a computer has a level of intelligence equal to or indistinguishable from that of a human. The test is performed as follows: two humans and a computer are recruited. Human

one, referred to as X, will be a performer. Human two, Y, will serve as a judge. Computer one, Z, will serve as the second performer. X will respond to questions posed by Y(judge). Y will read the responses X gives. Z (computer) will answer the same set questions and provide its responses to Y. Y will review the answers submitted to him, and attempt to discern machine from man. If Y fails to judge correctly, then the machine "wins" the game, and is deemed "intelligent." (Turing, 1950, pp 433-434). To clarify – one of the most prominent computer scientists of all time believed that the best way to determine whether a computer is as smart as a man would be to test its natural language processing capabilities. Hopefully this paints a somewhat compelling picture – NLP is important. Simultaneously, Turing's declaration paints another picture –solving the problems of NLP is hard. If programming a computer to process and produce language were simple, the world would be filled with automated therapists, authors, and musicians. Apple's Siri and Amazon's Alexa can "talk," but these systems are far from passing Turing's test.

Machine Translation research continued on into the 1960's, but some researchers were dubious of the efficiency by which the field was moving forward. John Pierce, a Bell Laboratories Engineer, wrote a now famous report and subsequent paper, entitled "Language and Machines: Computers in Translation and Linguistics" and "Whither Speech Recognition?" respectively. These articles illustrated the attitude of great concern that Pierce and other members of the Automatic Language Processing Advisory Committee shared towards the study of MT and speech recognition: "General-purpose speech recognition seems far away. Special-purpose speech recognition is severely limited. It would seem appropriate for people to ask themselves why they are working in the field and what they can expect to accomplish" (Pierce, 1969, pp. 1049). To Pierce's credit, computers, as previously mentioned, were far less advanced

in 1969 than they are today, some 50 years later. Even still, a possible interpretation of Pierce's writings may be "MT research is not efficient at this point in time. Why are we spending money on this again?" His letter hit home – funding dried up, and it did not return until Pierce retired in 1971 and a new director stepped in.

The next wave of NLP research dubbed "Phase 2" by Karen Jones, computer scientist responsible for the concept of inverse document frequency, spanned from 1960 to 1979, and brought energy and momentum back into the research community. Artificial Intelligence (AI) integration excited researchers greatly, and for good reason. The utility of AI in NLP research was evident early on in work like Green et. al's BASEBALL question-answering system. BASEBALL was fed baseball data (duh); player names, team names, dates of games played. Researchers would then provide the machine with queries like "did the Red Sox play vs the Yankees on September 3rd" and the machine, using AI techniques, would output a response. Asking for the answers to such 'basic' questions may seem ridiculous anyone with access to Google, but at the time such an accomplishment was remarkable. Inspired by BASEBALL, other AI/NLP projects began to spring up. LUNAR was programmed to answer questions about NASA space data. It was able to handle more complex questions than BASEBALL – many saw it as a direct upgrade (Woods, 1973). SHRDLU, on the other hand, was Terry Winograd's take on AI/NLP integration. SHRDLU was like BASEBALL and LUNAR, but it had a unique twist -it was graphics-based. Users, upon booting SHRDLU up, were greeted with an internal world that displayed a robotic hand, and some random items. From there, users were prompted for inputs, and SHRDLU would alter the graphical world accordingly. For example: if a user were to tell SHRDLU to "pick up a big red block," users would see the robot hand on their screen pick up the biggest, reddest block available. SHRDLU was a momentous step forward in the AI and NLP

fields (Winograd, 1971). This second phase of NLP, reliant on AI, also focused on semantics (a consistent trend in NLP research) and world knowledge.

Post 1970's research saw the birth of the grammatico-logical phase of NLP research (Jones, 2001). AI researchers had begun to adopt logic as a framework for knowledge representation and reasoning, and some NLP researchers drew from these approaches, and applied AI techniques to their own work. The 1980's saw the publication of one of the first publicly available toolkits for researchers: "Alvey Natural Language Tools" (Grover, 1993). The United Kingdom's Alvey program developed a morphological analyser, a parser, and a grammar and lexicon which allowed for a wide-ranging analysis of a considerable amount of English language (Briscoe). This period was "one of growing confidence and consolidation, and also an expanding community" (Jones, 2001).

Since the turn of the century, the field of NLP has seen an explosion of new ideas and developments. However, it is important to recognize that the tools and techniques at use today have been built upon decades of work. For half a century, incremental improvements in NLP research helped the field grow into what it has now become. Many of the world's leading tech companies have speech recognition departments, and some of the problems they face are to those faced by researchers in the 50's, 60's and 70's. Fortunately, advancements in technology has helped to bypass at least some of these issues. NLP research has also shifted radically in terms of scale, due in large part to the rapid development of information technology. The 1990's and 2000's saw the development of new data processing techniques, allowing for complex analyses of massive datasets. Software better equipped to handle massive tables of data has been developed in recent years (SQL, Tableau, Oracle). Additionally, programming languages like

Python have entire publicly available libraries devoted to NLP/text analysis. NLP research has been greatly enabled by computers in recent years.

There has also been a dramatic shift in methodology since the mid-to-late 1900's, As of 2020, there are many large text corpora (digitized collections of text documents) that have been made publicly available. The Natural Language Toolkit (NLTK) is a popular, beginner-friendly resource that offers several such data sets. The NLTK team hosts a Python package, which, upon being imported, provides access to a wide swath of text analysis tools, as well as a 100 MB set of 30 text files, including U.S. Presidential Speeches, the Universal Declaration of Human Rights, the Bible, and many others. The package also includes a variety of text manipulation tools. For example, users can search for specific terms or phrases, evaluate term frequencies, or generate random word clusters to find terms that commonly co-occur within a given text document (or collection of documents). The NLTK is a useful resource for beginners looking to actually "do" NLP. A review of the literature mentioned and referenced in the NLTK is valuable – many of these techniques are powerful and intuitive.

A corpus, in an NLP context, is a set of structured data texts. Corpora can take on a wide range of styles - a text document containing all U.S. presidential is a corpus, and so is an Excel file containing millions of Whatsapp/text messages. Both are prime candidates for text analysis, albeit with differing styles and methods of analysis. The entirety of the history, benefits, and detriments of corpus analysis are beyond the scope of this paper. For further reading, see McEnery & Gabrielatos' (2006) chapter in *The Handbook of English Linguistics* entitled "English Corpus Linguistics." The authors provide an outline of the major impacts that corpora have had on the study of the English language. Obtaining pre-structured, cleaned corpora is easy, thanks to the internet (NLTK), but, as aforementioned, text analysis is not limited to the Bible.

With the help of Python, users can extract text from nearly any crevice of the internet. *Natural Language Processing with Python* (Bird et. al., 2009) walks through a wide range of NLP activities that those with basic Python knowledge can conduct. The topics covered by the book include:

- How to extract text from PDFs, blogs, and MSWord documents
- How to perform corpora normalization techniques; tokenizing, stemming, lemmatization
- N-gram tagging
- Naïve Bayes classifiers

and many more. *Natural Language Processing with Python* is a useful resource for NLP beginners. The authors mostly avoid complex language and maintain a sense of accessibility for novice users. *Natural Language Processing with Python* brings NLP research into the realm of understanding for non-academics. Maintaining such a sense of accessibility while simultaneously providing insight into the underpinnings of academic research is a challenging task. The work that 21st century researchers conduct is astounding but can at times lie beyond the grasp of a broad, non-academic audience. This gap is natural, and not at all illogical– the audience that academic scholars have in mind when writing complex analytical papers is not the same audience that Stephen King has when writing his novels – but it may create a barrier to entry for those who lack a strong technical background. A deeper understanding of current scientific on goings should be a valuable and widespread thing for any person. As such, the ability to translate academic-text to plain-text is important, and many such works have garnered mainstream appeal and applause. For two popular examples, see Naked Economics and *Life 3.0*, two works that attempt to introduce economics and artificial intelligence to non-academic audiences.

Alternatively, one can read Stephen Hawking's' books for a greater appreciation of the study of physics. People love to read about science, as proven by the mainstream success of novels that successfully translate the works of academia into pop-science. NLP is a field that, when translated, should deeply excite readers.

Google performs extensive NLP research in the development of its Voice Assistant (they're not the only ones! Apple works hard on Siri, as does Amazon with its Alexa.) As such, it seems prudent to examine some of the awe-inspiring work going on at Google. Pulling the curtains of jargon and confusing lingo back to reveal the truth behind some of the research going on at these companies will prove useful.

Johan Schalkwyk, a member of Google's speech team dubbed "The Man Behind Google's Voice Tech," wrote a blog post in March of 2019 titled "An All-Neural On-Device Speech Recognizer," detailing some of the leaps and bounds that Google's voice search capabilities have made in recent years. The post details the exciting results of the Speech Team's 2019 paper entitled "Streaming End-to-End Speech Recognition for Mobile Devices." Google was able to develop a speech recognition app that's twice as fast, reduces word error rate by 20%, and does so without any sort of internet connection. This article serves a wonderful illustration of the widespread applicability of NLP, while simultaneously showcasing some of the shortcomings of current NLP research. The results of this research are substantial and quite simple in reality - Google was able to make its personal assistant app faster, less prone to error, and offline-ready by using some sophisticated NLP and optimization techniques. This is exciting, and relevant to every single one of Google Assistant's 500+ million daily users! Unfortunately, as previously mentioned, academic research of this nature may fail to garner attention or appreciation from non-academic audiences.

So, in review, are there any beginner-friendly tools that are powerful and intuitive? Yes! Two of the 'friendliest' resources out there for beginner text analysis are the NLTK and Bird et. al.'s *Natural Language Processing with Python.* Developing an appreciation and understanding of the field of NLP is important from an academic perspective, but for someone just looking to perform text analysis themselves, the above two sources are the most valuable. They, at least initially, avoid the potentially confusing language that may scare NLP beginners off. By using these beginner friendly resources, novice programmers can develop an appreciation for NLP and a familiarity with some of the lower level language of the field, which can lead to a developed interest in more complex NLP research.

Methods

The methods section of this paper will describe the process that I went through in familiarizing myself with RStudio and text analysis and create a general and accessible workflow for people new to programming and NLP.

RStudio

An open source, free to use software, RStudio is a great place to begin working with text data. Download instructions can be found at https://www.rstudio.com/products/rstudio/download/ and the program is relatively easy to setup. The open source version of the app is free, but working professionals can opt for the Desktop Pro version, which costs $995 per year. Pro version includes priority support, RStudio professional drivers, and remote RStudio Server Pro access, but those things aren't necessary for a beginner. For the purpose of this paper, and, realistically, any sort of analysis that a beginner is interested in, the free version works perfectly well. Once RStudio is installed, the fun can begin.

dplyr is an R package that enables elegant data frame manipulation. When dealing with large datasets, the old-fashioned "look for the columns you want" fails. Scrolling by hand through 20,000 rows isn't feasible – herein lies the power of filtering. With dplyr's filter(), users can provide search criterion:

"Show me records that have more than 400 votes, less than 100 comments, and were posted between december and march!"

The mutate() function is just as useful:

"Create a new column that fills with a 0 if column A is positive, and a 1 if column A is negative."

The landing page for the package achieves a level of brevity and eloquence, find it here: https://dplyr.tidyverse.org/

In summation: incredibly easy to learn, and equally powerful, dplyr is a tool that any aspiring RStudio user should familiarize his or herself with.

R has built-in tools that allow for data visualization, but ggplot2 allows users to take their visualizations to the next level. Ggplot2 allows users to design bar charts, stacked bar charts, box plots, scatter plots, violin plots… the list goes on! There are some incredible resources out there for learning to create strong and meaningful visualizations, and ggplot2 enables a level of customization that's, as far as I can tell, unrivaled. Additionally, ggplot2tor (https://ggplot2tutor.com/) is filled to the brim with useful advice on how to utilize ggplot2 efficiently. I've spent hours trying to tweak margins, add photos, or adjust orientation, only to find a quick 2-minute tutorial on this website. Don't make the same mistakes I did - consult this resource!

The above tools are general purpose RStudio essentials. For NLP specifics, I'll focus on two specific resources:

quanteda is the first NLP tool introduced in this paper. It's an R package built by Kenneht Benoit and Kohei Watanabe that enables the managing and analysis of textual data. There are alternative packages available for text analysis (tidytext is a popular choice), but quanteda is, in my experience, more inviting to beginners. Following the vignette/quick-start guide allows for near immediate analysis – you can jump right in!

Unfortunately, this resource isn't available in R. The Natural Language Toolkit (NLTK) is a python-native platform that contains over 50 corpora/lexical resources and a host of text analysis tools. The only reason I've included it in this methods section is that I think it's too important to pass up – additionally, it's entirely possible that beginner programmers might lose interest in R.

Python is a “sexy” language – nearly every analyst job wants some sort of Python experience, and Python resources outnumber R resources by a wide margin. So, if in the market for a Python-based set of NLP resources, the NLTK is the place to go. There’s even a free book!

http://www.nltk.org/book/

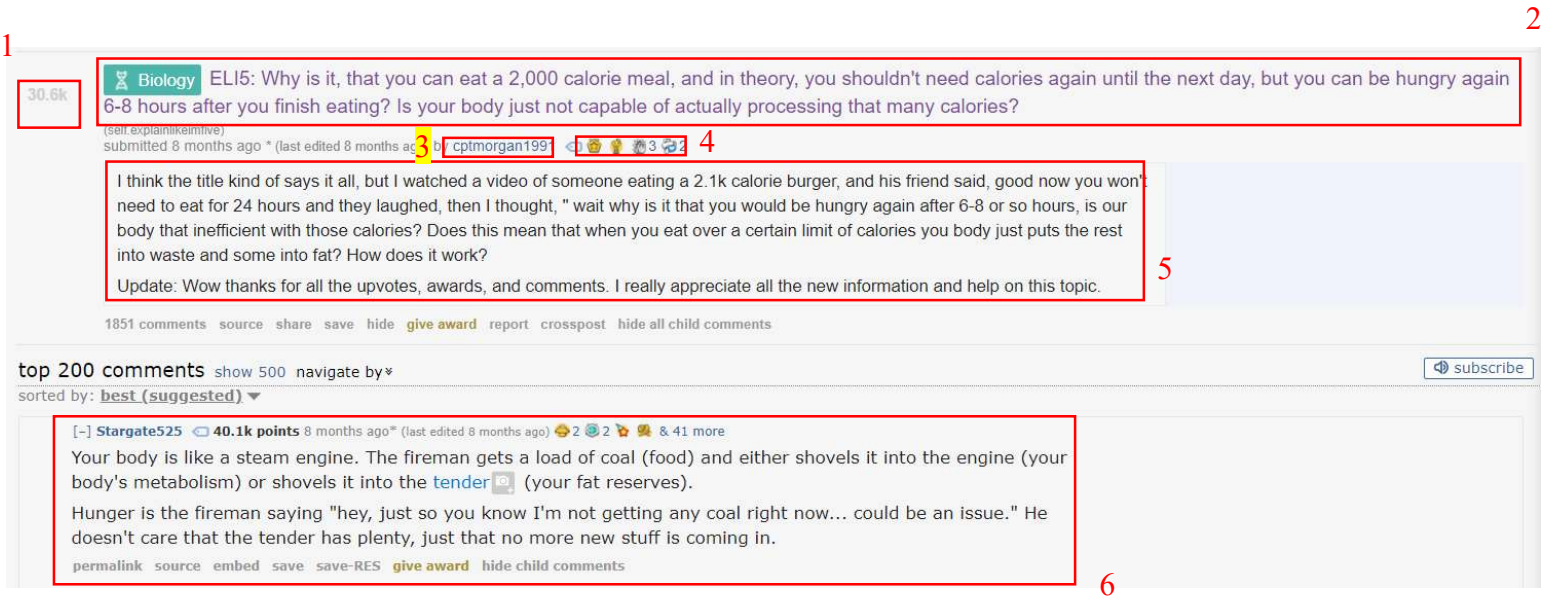
Applied, Beginner Friendly Analysis Using Reddit Data

Data

The data for this paper was pulled from a post by u/Stuck_In_The_Matrix. A direct link to the original post can be found in the works cited page.

From the original dataset uploaded by /u/Stuck_In_The_Matrix, a subset was pulled by Professor Reagan Mozer. This subset contains data from the /r/movies subreddit only.

To describe the data most effectively, an annotated image is included below:



A Reddit thread, pulled from /r/ELI5

1. Post Votes
    a. This value corresponds to the sum total of up/downvotes received by this post. Reddit users with registered accounts can elect to up/downvote a post, although there is no requirement to do so. The higher the total of this value, the more people upvoted the post, and presumably enjoyed it. This value is captured in column “score”
2. Post Title

   a. Corresponds to the title of the post that the author wrote. Titles are mandatory – no post can exist without a title. Represented in column “title”

3. Post author
   a. Corresponds to author of the post. Each post can have only one author. Column “author”
4. Rewards
   a. Relatively new, these rewards are not captured in our dataset. Reddit used to sell “Reddit Gold,” which one could pay real world cash for. Reddit Gold allowed users to “gild” their favorite comments and posts. The new award system has expanded far beyond Reddit Gold, and the dataset does not reflect these updates. Reddit Gold is distinguished, though, in column “gilded”
5. Post Content
   a. This is the heart of post – this field is optional. Users can leave the post blank, opting instead for a title-only post. If not, users are asked to describe in further detail the title of their post. This is represented in column “selftext”
6. Comment
   a. Users can comment on their own and other users’ posts. This section includes text data, comment score data (valued the same as aforementioned score) and username of commenter.

The subset of data used in this paper excludes user comment data – instead, the analysis focuses on post content. Where comment sections can run rife with spam, bots, and general noise, posts are typically more fleshed out and precise.

The entire dataset includes 30 columns, but the only variables relevant to this paper are author, num_comments, score, title, selftext, and gilded. An updated csv file with the restricted dataset can be found here:

https://github.com/xndrmcw/capstone_data

Research Questions

This paper is written for individuals with little to no programming or statistical background. Thus, the following research questions are aimed at that same audience.

1. Is there a 'most popular' movie, and if so, can we find it using the tools available to us?

2. Can we search the dataframe for specific phrases? That is, can we find all the data associated with comments that mention specific actors, directors, etc?

3. Can we create an aesthetic text-based visualization that provides some sort of insight we otherwise wouldn't have?

These questions, while far from being statistically comprehensive, are the kinds of questions that I had when I first starting working with this kind of data, and I think they're the kind of questions that are easily accessible to people unfamiliar with experimental design.

Results

As with any data analysis, it's important to examine, clean and process the data, if necessary.

```
setwd("C:/Users/Alexander/Desktop/capstone_data")
library(readxl)
library(tidyverse)
library(gridExtra)
# read in data ----
dat <- read_excel('movies_subset.xlsx')
```

From the top, moving down:

Setwd() supplies RStudio with a 'working directory,' which, to simplify, is just where the files that you want to use are going to come from. If there's any confusion about finding and setting a working directory, refer to http://rfunction.com/archives/1001

Library() imports packages that aren't native to R. Individuals can develop their own packages, and you can download them! They're incredibly useful. Tidyverse should be familiar (see Methods), but readxl and gridExtra are both particularly useful as well. Readxl allows users to import Excel files into R as dataframes, and gridExtra allows users to customize the layout of their ggplot outputs.

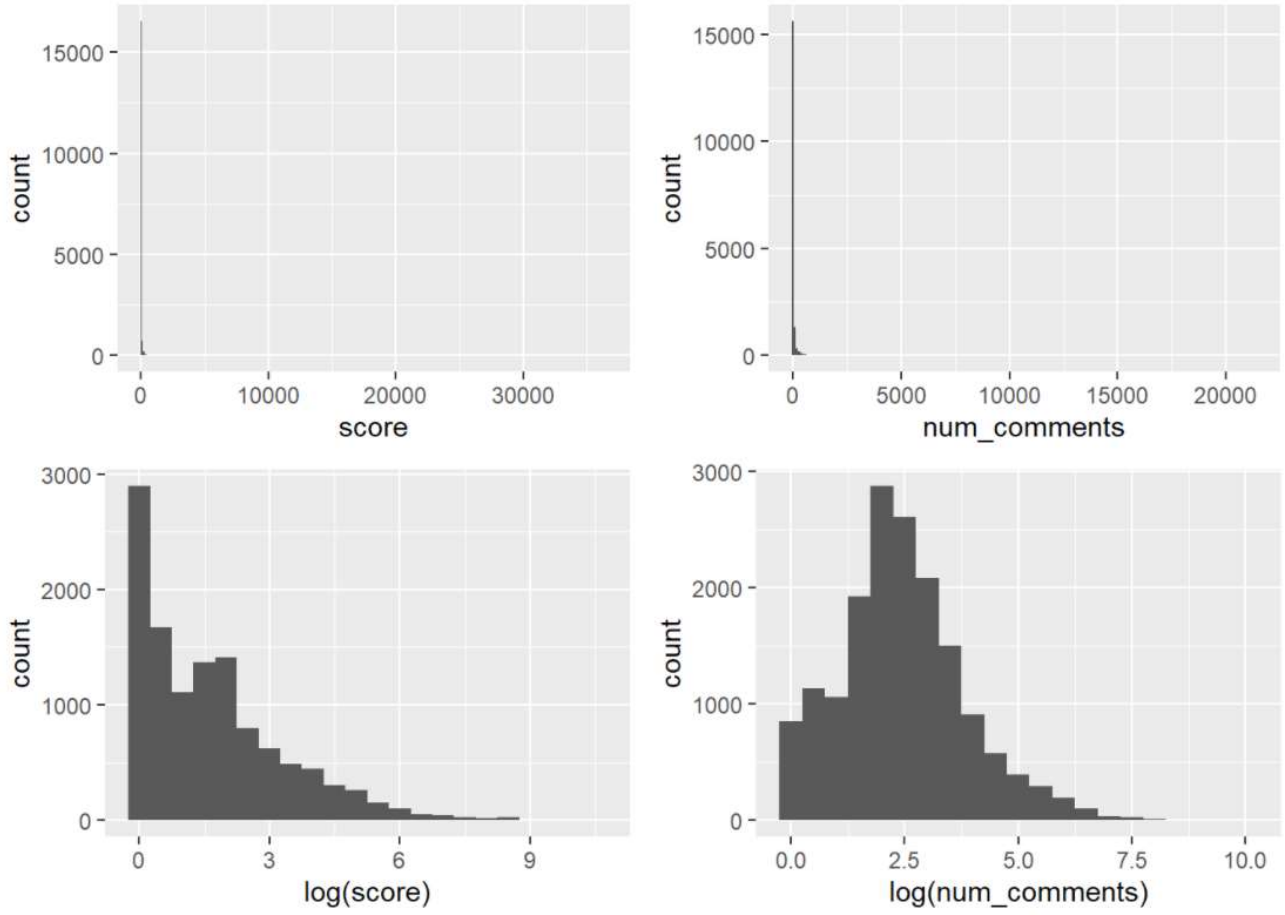
Read_excel() is where you specify which file you want to read in. If you've set your working directory, all you need to put here is the name of your file!

```
# remove useless columns, clean data
dat2 <- dat[,c(3, 7, 8, 11, 12, 15)]

# check for na values, turns out 3 usernames have been deleted
isna <- dat2 %>% filter(is.na(dat2))

# filter out those usernames
dat2 <- dat2 %>% filter(!is.na(dat2))
write.csv((dat2), "dat2.csv")

# time for some non-text EDA ----
score_plot <- ggplot(aes(x = score), data = dat2) +
  geom_histogram(binwidth = 100)
comments_plot <- ggplot(aes(x = num_comments), data = dat2) +
  geom_histogram(binwidth = 100)
log_score_plot <- ggplot(aes(x = log(score)), data = dat2) +
  geom_histogram(binwidth = .5)
log_comments_plot <- ggplot(aes(x = log(num_comments)), data = dat2) +
  geom_histogram(binwidth = .5)

grid.arrange(score_plot, comments_plot, log_score_plot, log_comments_plot, ncol=2)
```

Our first line of code is a simple dataframe manipulation. As specified in the data section, we don't need *all* of the data stored in the excel sheet that we imported. So, we specify which columns we want to keep! This format, dat[,c(x,y,z)] can be applied to any dataframe. Just substitute x,y, and z with the number of the column that you'd like to retain! For more information on dataframe manipulation, see https://ourcodingclub.github.io/tutorials/data-manip-intro/ for a step-by-step walkthrough on getting your data in a friendly format.

Next, we create a new dataframe that searches for NA values. This step is important, because it gives us some familiarity with the data– why is username the only field with NA values? Because users have the ability to delete their accounts, and their usernames go into the void with them! From there, we can filter our original dataframe with dplyr to remove those NA values. Dat2 is where we're going to be working from, after this!

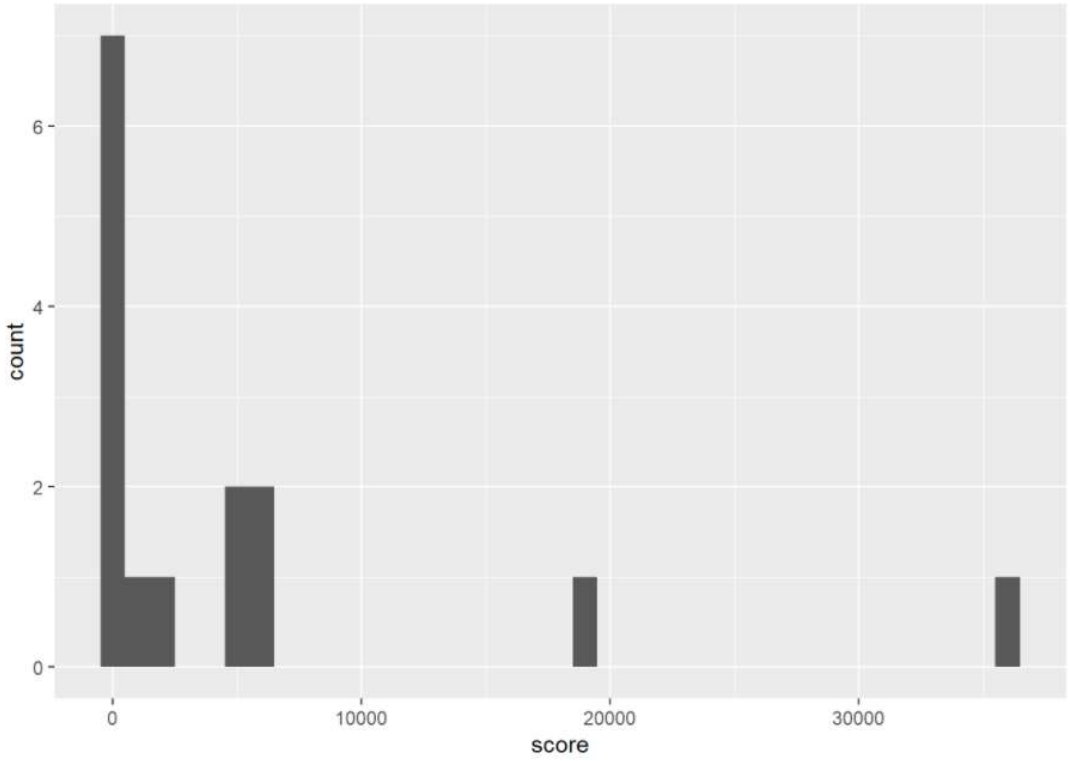
Write.csv() exports an excel file with a specified dataframe. Here, I clarified that I wanted dat2 to be exported as dat2.csv. It saves the file to your current working directory! Access to this file can be found on GitHub – link is in the Data section.



Now, we move into EDA visualizations! We create four plots to give us a better sense of the shape of our data. Ggplot2tor is a great resource for learning the syntax of this package, but the resultant graphs are the real focus of this chunk of code. In the first two graphs, we see that the data is right skewed. We attempt a log-transformation (which may be a bit beyond the scope of a beginner-level stats background) and see that doing so provides a much neater distribution! These subsequent graphs indicate that the majority of posts receive nearly zero points.

Interestingly, even posts that have zero points still receive comments! The num_comments variable follows a seemingly regular distribution – most posts receive ~log(2.5) comments. Again, domain knowledge is important when we work with data. Individuals with no experience using Reddit might see the num_comments distribution as an indication that even low quality posts gain interaction. Those familiar with the site realize that all posts, regardless of content, receive anywhere from 1-3 "auto-mod" comments. This is a key takeaway – data, by itself, doesn't paint a picture. It's only through the synthesis of data analysis and personal insight that we can achieve useful results.

```
# some context specific exploration. gilded awards aren't given out often in /r/movies,
# and when they are, they're not always given out where you'd expect. of the 15
# gildings, 8 were given to posts with less than 1,000 comments. Cool!
gilded <- dat2 %>% filter(gilded == 1)
ggplot(aes(x = score), data = gilded) + geom_histogram(binwidth = 1000)
```



On the topic of context specific exploration, let's say our novice programmer *is* familiar with reddit, and wants to do some extra sleuthing. He or she might notice the "gilded" column in the dataset, and take note that, at first glance, there's an overwhelming amount of 0's, and not very many 1's. So, let's use the filter function again! We filter specifically for posts that have a "1" in the gilded column and put together a graph using the same methods from before. Here, we see that of the 17,000 rows in the dataset, only 15 have received gold. Wow! From this, our novice user might realize, "huh. Doesn't seem like I'm very likely to get gold… maybe this dataset can't teach me how to do it." Another important takeaway – data isn't magical. Poor questions receive poor answers, and data can't change that. Regardless, because it might interest our reader, let's see who's received gold!

	author	num_comments	score	title
1	hitrecordjoe_	3587	4863	I am Joseph Gordon-Levitt. AMAA: Ask Me Anything Again...
2	MasterLawlz	3313	6120	I just realized that "Men in Black" is one of the very, very fe...
3	tommythehandsome	252	2274	Stardust is the wonderful fantasy film nobody talks of.
4	OfficialValKilmer	2330	19064	Hello Reddit! I am actor and artist Val Kilmer. I have been a ...
5	doswillrule	22	9	Hey reddit. You asked me to go and watch the new Ghostbu...
6	fantomknight1	1544	6180	[SPOILERS] Arrival: Some Easter Eggs and explanations of so...
7	JakeEllz	1534	5194	After Tarantino releases his final film, he should release all of...
8	OfficialValKilmer	2942	36484	Hello Reddit! I am actor and artist Val Kilmer. I have been D...
9	felixjmorgan	99	484	In the last year I've moved from a casual IMDB movie fan to ...
10	The_Real_Kyle_Butler	38	14	Most historically accurate movie?
11	oldboyFX	76	181	We created a place for browsing and discovering movies. N...
12	Drew_bacca	52	75	Can we just take a minute to appreciate Michael Giacchino ...
13	JohanHegg	218	546	I Am Johan Hegg from metal band Amon Amarth and an ac...
14	Bennylegend	139	266	Just watched Contact for the first time...
15	LegitIAmjack	332	197	What are some words that /r/movies gets wrong?

Well, a couple celebrities seem to crop up! Joseph Gordon-Levitt and Val Kilmer are both members of the exclusive gilded club. Both of Val's posts are also major outliers with 19,000 and 36,000 upvotes respectively. Cool!

So far, the results we've obtained are interesting, but aren't directly related to the text features of the dataset. For that, we're going to employ…

Quanteda

```
library(quanteda)
library(quanteda.textstats)

#create corpus, document feature matrix for title
corpus_title <- corpus(dat2$title)
head(corpus_title)
```

```
## Corpus consisting of 6 documents.
## text1 :
## "What is with the hate for Sean Penn?"
##
## text2 :
## "So about Mel Gibson"
##
## text3 :
## "Let's talk about yoga hosers."
##
## text4 :
## "What's your Top 3 TV Shows and Top 3 Movies?"
##
## text5 :
## "Which "What!? You've never seen _____!?" movie have you nev..."
##
## text6 :
## "What are your favorite examples of songs being used in movie..."
```

More specifically, we're going to employ quanteda's corpus() function. Without any real familiarity with NLP standards or methods, we're able to construct a corpus of the rows of title column text data. A beginner analyst might be thinking "okay, cool, so what? Isn't that just our title column, but with a fancy name?" Sort of, but quanteda likes corpora! With quanteda's backing, we can run some exciting functions. Using the **dfm()** function, we can find a

satisfactory answer to our first research question – **what's the most commonly mentioned movie on /r/movies?**

```
dfm_title <- dfm(corpus_title, remove = stopwords("english"), stem = TRUE, remove_punct = TRUE, remove_numbers = TRUE)
topfeatures(dfm_title, 100)
```

```
##      movi      film     watch      help   spoiler      just      best      like
##      7352      2328      1487      1153      1107       890       779       768
##       can     think  question      good       one      look      find      need
##       743       719       717       645       629       623       621       613
##     anyon      time       see   favorit      seen      year     scene       saw
##       609       570       519       513       476       459       454       450
##      make      name   thought      star     actor         r       get       new
##       409       397       395       394       387       387       384       372
##       end      love   discuss      ever     world      know     great     peopl
##       366       357       352       345       334       334       328       323
##    horror     pleas   trailer       war     first recommend    realli    review
##       319       315       311       309       308       307       307       300
##    rememb        go       day    cinema   charact      last  director      made
##       300       295       290       289       288       285       284       284
##       tri      want       guy      titl       els    reddit       bad      hate
##       275       266       252       244       243       239       233       226
##     oscar         s      list     night   suggest    someth       man    releas
##       226       223       222       219       217       216       214       212
##   opinion      talk      plot      show      dark       old    someon       now
##       207       198       196       193       193       190       189       189
##    action    origin     never      much      feel       top       use      idea
##       187       184       183       182       180       179       177       176
##      anim       big    sequel    batman  favourit      back     thing       amp
##       173       170       170       169       168       168       166       166
##   theater      play    happen    better
##       166       166       164       163
```

Without any real knowledge of the specifics of document feature matrices, users can put get themselves an easy-to-read output which displays the most common 'features' (in this case, words, or, more specifically, stems) inherent to the corpus we've analyzed. So, with a bit of rudimentary eye-scanning… we find **Batman**! Based on this approach, we're led to believe that the Batman series is the most frequently mentioned on reddit. Looking at the table, though, we see a limitation of this simplified kind of analysis – it's entirely possible that **Star**, mentioned 394 times, could be in a sentence referring to the film **A Star is Born.** So, how can we tackle this

problem? I'm no linguist, and the reader of this paper likely aren't either, so we'll need help. Fortunately, quanteda has a function specifically for this situation!

```
toks1 <- tokens(char_tolower(corpus_body), remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE)
toks2 <- tokens_remove(toks1, stopwords("english"))
toks3 <- tokens_ngrams(toks2, 3)
head(toks3)
```

```
## Tokens consisting of 6 documents.
## text1 :
##  [1] "probably_living_rock"       "living_rock_something"
##  [3] "rock_something_everyone"    "something_everyone_ripping"
##  [5] "everyone_ripping_apart"     "ripping_apart_can"
##  [7] "apart_can_anyone"           "can_anyone_tell"
##  [9] "anyone_tell_also"           "tell_also_bad"
## [11] "also_bad_person"            "bad_person_think"
## [ ... and 22 more ]
##
## text2 :
##  [1] "just_polanski_something"     "polanski_something_horrible"
##  [3] "something_horrible_mean"     "horrible_mean_everyone"
##  [5] "mean_everyone_else"          "everyone_else_gets"
##  [7] "else_gets_amnesty"           "gets_amnesty_stop"
##  [9] "amnesty_stop_excusing"       "stop_excusing_mel"
## [11] "excusing_mel_gibson's"       "mel_gibson's_horrible"
## [ ... and 29 more ]
##
## text3 :
##  [1] "hey_guys_gals"          "guys_gals_just"         "gals_just_finished"
##  [4] "just_finished_watching" "finished_watching_part" "watching_part_two"
##  [7] "part_two_kevin"         "two_kevin_smith's"      "kevin_smith's_great"
## [10] "smith's_great_white"    "great_white_north"      "white_north_trilogy"
## [ ... and 23 more ]
##
```

Using the **tokens()** function, we can split our corpora into 3 word chunks. Ta-da! We can look at a frequency table again, and with it, gain a bit more context. There are likely tuning parameters and functions available in quanteda that allow for more effective presentation and analysis of these trigrams, but this portion is intended simply to showcase a beginner-friendly method of text analysis. Already, the novice programmer can get a feel for the complexity of NLP – if it weren't for quanteda and decades of academic work, how would we ever have conducted this sort of multi-word analysis? Tools like quanteda make life and learning easier for experts and beginners alike.

movie_already_posted	gt_gt_gt	movie_really_like	one_movie_per	please_post_one
327	275	172	166	165
post_one_movie	another_movie_really	already_posted_upvote	movie_country_current	country_current_thread
165	164	164	163	163

Now we can move on to our second research question: **"can we search the dataframe for specific phrases? That is, can we find all the data associated with comments that mention specific actors, directors, etc?"**

```
# now for some light text analysis ----
quent_comments <- dat2 %>%
  filter(str_detect(title, "Tarantino")) %>% glimpse()
```

```
## Rows: 40
## Columns: 6
## $ author       <chr> "deliaprod", "JasonWilley777", "marianitten", "LeJavie...
## $ num_comments <dbl> 17, 17, 16, 23, 6, 1, 8, 5, 23, 29, 18, 25, 10, 20, 15...
## $ score        <dbl> 0, 0, 2, 1, 4, 2, 0, 0, 0, 4, 3, 3, 0, 0, 5194, 0, 11,...
## $ title        <chr> "Would Quentin Tarantino be a perfect tonal match to w...
## $ selftext     <chr> "Do the characteristics of the dialogue driven \"Merc ...
## $ gilded       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ...
```

Again, quanteda supplies an intuitive function for this exact purpose. Using **str_detect()** we can search for specific terms. Quentin Tarantino is a popular film director on Reddit, so let's search for his unique last name. We yield 40 rows, meaning there are 40 post titles that include Tarantino in them! We can try the same for the body text:

```
# now for some light text analysis ----
quent_comments <- dat2 %>%
  filter(str_detect(selftext, "Tarantino")) %>% glimpse()
```

```
## Rows: 190
## Columns: 6
## $ author       <chr> "Baby-exDannyBoy", "johncosta", "deliaprod", "epicluca...
## $ num_comments <dbl> 16, 71, 17, 4, 47, 5, 1, 32, 13, 16, 14, 198, 11, 3, 1...
## $ score        <dbl> 0, 227, 0, 1, 0, 2, 4, 0, 3, 2, 0, 29, 0, 0, 1, 1, 1, ...
## $ title        <chr> "Anybody get this weird feeling about Woody Allen?", "...
## $ selftext     <chr> "I was looking for a film to watch this week and I fou...
## $ gilded       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

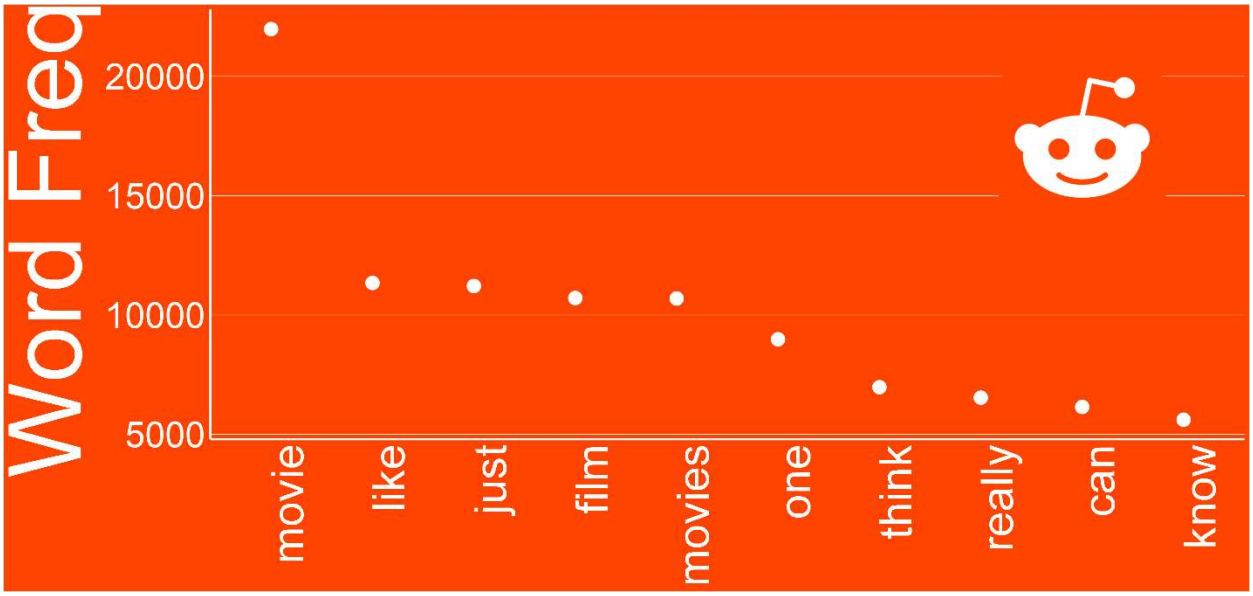
190 rows that detect "Tarantino" in the selftext column! Beginners can swap "Tarantino" for anything they'd like, and run wild.

Lastly, we'll try to tackle our third research question: **"Can we create a good looking text-based visualization?"**

```
s1 <- png::readPNG('reddit-logo.png')
s1 <-  grid::rasterGrob(s1, interpolate=TRUE)
# ggplot of dfm title frequency
features_dfm_body <- textstat_frequency(dfm_body, n = 10)

# Sort by reverse frequency order
features_dfm_body$feature <- with(features_dfm_body, reorder(feature, -frequency))

theme_set(theme_dark())
theme_update(legend.position = "none",
             panel.background = element_rect((fill = "#FF4300")),
             plot.background = element_rect((fill = "#FF4300")),
             # library shadowtext to add black outline to title
             plot.title = element_text(family = "Twitch", color = "white",
                         size = rel(13), vjust = 1.3, hjust = .5),
             # got rid of the title, self explanatory
             axis.title.x.bottom = element_blank(),
             # basic aesthetic updating
             axis.title.y.left = element_text(size = rel(9), family = "Sans", colour = "white"),
             axis.text.x.bottom = element_text(size = rel(5), lineheight = .3, family = "Twitch", colour = "whit
e"),
             axis.text.y.left = element_text(size = rel(4), family = "Twitch", colour = "white"),
             panel.grid.major.x = element_blank(),
             panel.grid.major.y = element_line(colour = "white"),
             panel.grid.minor = element_blank(),
             axis.line = element_line(size = 1, colour = "white"),
             axis.ticks = element_blank(),
             axis.text = element_text(colour = "white"))
ggplot(features_dfm_body, aes(x = feature, y = frequency)) +
    geom_point(color = "white", size = 6) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
annotation_custom(grob = s1, xmin = 8,
                  xmax = 10, ymax = 30000) +
labs(y = "Word Freq.") +
ggsave(here::here("reddit.png"), device = "png",
       type = "cairo", width = 17, height = 8, dpi = 300)
```

This chunk of code is admittedly less accessible than the rest. GGplot's syntax is confusing at first. Again, beginners can refer to ggplot2tor for guides and resources. With this code chunk, though, we get a nice looking, easy to interpret graph! The parameters in the above chunk that we tweaked aren't particularly hard to play around with, and with a bit of time and fiddling,

anyone can make a visualization comparable to this one.



Discussion

The aim of this paper was to provide an intuitive, beginner-friendly guide to using RStudio for data and text analysis. In doing so, I hoped to foster an interest in text analysis, and subsequently, in NLP. To this end, I feel as though my goal is partially accomplished. The paper (and its author) is limited in its ability, but the elements of the paper should be easily understood by most non-technical authors. The analysis section, despite remaining relatively informal, will hopefully bring NLP-related questions to readers' minds. Anyone who follows along with the code chunks will have a degree of familiarity with NLP concepts (corpora, n-grams, DFM's,) and a relative amount of programming knowledge. With this, they will be able to delve into more complex text analysis and have the tools at their disposal to do so.

This paper has limitations, of course. Following the examples above can only extend one's skill so far. It's my opinion that beginning a self-motivated project is immensely influential on retention and enjoyment. Lastly, in writing this paper, I realized that I'm not as easily able to explain the function of code chunks as I had imagined. Being able to do something and explaining why and how you do it are altogether different things. Still, anyone who reads the code's annotation should be able to perform the analyses.

Working Bibliography

Bird, Steven, et al. *Natural Language Processing with Python*. O'Reilly, 2009. http://www.nltk.org/book_1ed/

Briscoe, Ted. "A formalism and environment for the development of a large grammar of English" *IJCAI*, 1987, pp. 703-708

Ceccato, Silvio. "Correlational Analysis and Mechanical Translation" *Machine Translation,* 1967, pp 77-135

Dongarra, Jack. "Report on the Sunway TaihuLight System," *University of Tennessee Department of Electrical Engineering and Computer Science*, 2016.

Grover, C. et al. "The Alvey Natural Language Tools Grammar." (1993).

He, Yanzhang, et al. "STREAMING END-TO-END SPEECH RECOGNITION FOR MOBILE DEVICES" *Arxiv,* 15 Nov, 2019. https://arxiv.org/pdf/1811.06621.pdf

Jones, Karen. "Natural Language Processing: A Historical Review" *Artificial Intelligence Review*, pp 1-12. 10 Dec, 2001. http://www.cl.cam.ac.uk/archive/ ksj21/histdw4.pdf

Lyons, John. *Natural Language and Universal Grammar: Essays in Linguistic Theory*. Cambridge, Cambridge University Press, 1991. https://archive.org/details/naturallanguageu0000lyon/mode/2up

McEnery, T., & Gabrielatos, C. "English corpus linguistics." In B. Aarts & A. McMahon

(Eds.), The Handbook of English linguistics (pp. 33–71).

Mozer, Regan, et. al. “Matching with Text Data: An Experimental Evaluation of Methods for Matching Documents and of Measuring Match Quality.” *Political Analysis,* 17 Mar. 2020, pp. 445-468

“Natural Language Toolkit” NLTK, Apr. 13 2020. http://www.nltk.org/

Pierce, John. “Whither Speech Recognition?” *Journal of the Acoustical Society of America*, Oct. 1969, pp. 1049-1051.

---. “Language and Machines – Computers in Translation and Linguistics.” *ALPAC Report,* National Academy of Sciences, 1966.

Plath, W. “Multiple Path Analysis and Automatic Translation” *Booth,* 1967, pp 267-315

Rossmann, Louis. “Apple uses spite to force planned obsolescence.” 16 Sep, 2015. https://www.youtube.com/watch?v=NVAmnV65_zw

Schalkywk, Johan. “An All-Neural On-Device Speech Recognizer.” Googleblog, 12 Mar, 2019. https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

Stuck_In_The_Matrix. “ I have every publicly available Reddit comment for research. ~ 1.7 billion comments @ 250 GB compressed. Any interest in this?” 11 Jul, 2015. https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/

Turing, Alan. “Computing Machinery and Intelligence.” *Mind,* Oct. 1950, pp 433-460

Winograd, Terry. “Procedures as a Representation for Data in a Computer Program for Understanding Natural Language” *MIT AI*, Jan 1971.

Woods, William. “Progress in natural language understanding: an application to lunar geology” *AFIPS*, 1973. https://dl.acm.org/doi/pdf/10.1145/1499586.1499695