Predicting How Online Communities Interpret Information

Antone Christianson-Galina[1]

[1]Affiliation not available

February 15, 2022

Introduction

The overarching goal of this research is to build a series of models that predict how an online community shares, reacts to, and interprets the news, using a dataset harvested from the social networking site Reddit. https://www.reddit.com I used Reddit because it has a publicly available API, and clearly structured data. On reddit, users form into communities called subreddits- these online communities were the focus of my study research.

I divided this document into three sections based upon methods and technologies.

Section 1 outlines how I used web APIs to collect data and scripting languages to sterilize and model the data.

Section 2 outlines my experiments with statistical natural language processing models (Statistical NLP) specifically sentiment analysis and latent dirchilet allocations.

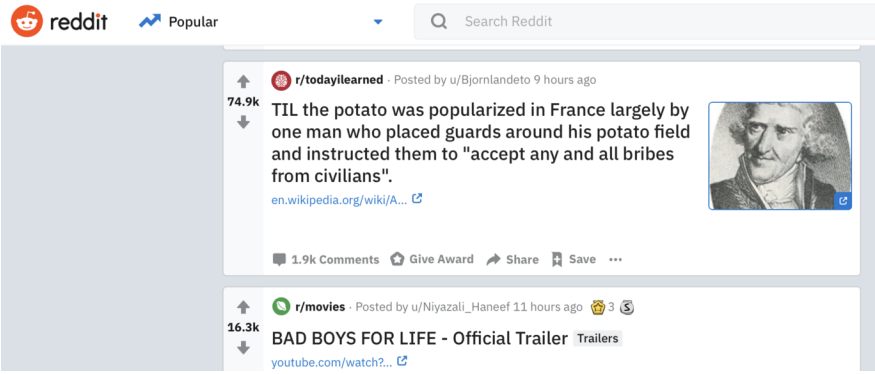
Section 3 outlines my neural language processing models (Neural NLP), including recurrent neural networks and deep learning.

Throughout this body of work, I developed skills that prepared me to undertake broader questions, beginning with simple sentiment analysis (tying words to connotation scores) and progressing to using a mixture of deep and recurrent neural networks to tackle more open-ended questions.

I conclude that Sentiment Analysis, Topic Modelling, and Word Embedding models do not have the representational capacity to model how people will interpret the news. On the other hand, recurrent neural networks and deep learning have representational capacity but require large datasets. A promising research direction for the future would be optimizing deep learning models for smaller data sets. My future interests in this body of work aim to inquire about the volume of data necessary to ensure the fidelity of the results.

Harvesting the Data

Harvesting data from Reddit using python is straightforward with their well-documented API, called PRAW:[https://praw.readthedocs.io/en/stable/getting_started/quick_start.html]. For documentation on how API calls work in general, you may consult the following: [https://tray.io/blog/how-do-apis-work].



To call the Reddit API, you must first set up a connection. The most straightforward way is to register an application with Reddit via OAuth. I registered a web application and stated "Research" as the purpose.

As part of this registration, I was granted a client id and a client secret. The following expression shows how it was implemented in python:

reddit = praw.Reddit(client_id='xxxxx,

client_secret=xxxx

user_agent=xxxx

Then I needed to specify the thread to call for the API. Each reddit thread has a unique key that you can pull from the URL. That key becomes the logical entity I named "submission"

submission = reddit.submission(key from the url)

This uses the Praw API to pull data related to a thread.

This returns an unstructured forest of comments. At the top, is the first post in the thread, which can have multiple comments, which then can each have other comments. Therefore, it was necessary to structure the data.

I sorted the comments to get the oldest first with the specification:

submission.comment_sort = 'old' _

Then, I created a for loop to iterate through each comment. For each comment, it harvests the time that the comment was created and appends it to an array.

for comment in submission.comments.list():

time = comment.created

ltime.append(comment.created_utc)_

In the for loop, I implemented either tokenisation or stemming depending on the experiment.

Tokenisation simply turns the sentence string into a list of words. In natural language processing, this is called the "bag of words" technique. This allows for easy processing but totally ignores sentence structure.

Stemming removes word endings. For example, "carefully" and "careful" both become the word "careful". I used a python implementation of the Porter stemmer for my code, developed by Martin Porter in 1979 and still maintained by him. [https://tartarus.org/martin/PorterStemmer/]

stemmer = PorterStemmer()

for w in words(entry):

w= stemmer.stem(w).lower()

Statistical Natural Language Processing

My first experiment with statistical natural language processing implemented Udny Yule's K characteristic(Yule, 1944) . This formed the basis for my master's dissertation. [https://www.authorea.com/users/107755/articles/277345-measuring-online-feedback-loops?commit=d79a10bf2e22949ba350e2a858b73f430b42aee5%5D This metric measures the probability of two words being the same in a text. The assumption is that less repetition = complexity. This agrees with Kolmogorov complexity theory(Kolmogorov, 25AD). The Kolmogorov complexity of an object is the shortest possible program that it would take to specify an object. In this case, many unique words would require a complex program.

Below is my implementation of Yule's K characteristic, based on this blog post http://swizec.com/blog/measuring-vocabulary-richness-with-python/swizec/2528

M1 = float(len(d))

M2 = sum([len(list(g))*(freq**2) for freq,g in groupby(sorted(d.values()))]

I scored each Reddit post by its complexity and then plotted the complexity across time in a Reddit thread. I then charted the average complexity to measure whether or not a post was increasing or decreasing in complexity through time.

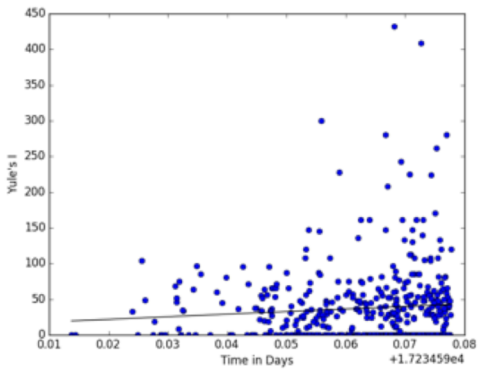


Figure 5: A map of Yule I scores on thread- EPA chief Scott Pruitt says carbon dioxide is not a primary contributor to global warming

To see my full implementation, you can visit my GitHub repo.

https://github.com/antonecg/SuperYule

Ultimately, I found the utility of this metric limited and had trouble finding clear drivers of complexity. One measure did not give me enough information to model human behavior.

My following approach was to obtain more ways to measure behavior. Researching ways to measure personality, I came across the research of the Cambridge psychometrics institute.(Kosinski et al., 2016)Researchers at the institute used latent Dirichlet allocation (which I will examine further below) to create a dataset that scored words with associations to big five personality profiles.

Using my Reddit dataset and the Cambridge psychometrics institute study results, I created an algorithm that scored each word on its association with the personality traits openness, conscientiousness, extraversion, agreeableness, and neuroticism.

```
def neuroscore(entry):
    d = {}
    stemmer = PorterStemmer()
    totalscorepost = 0
    scoreword = 0
    for w in words(entry):
        w = stemmer.stem(w).lower()
        try:
            d[w] += 1
        except KeyError:
            d[w] = 1
        scoreword = dict.get(w)
        try:
            totalscorepost += float(neurodict[w])
        except:
            continue
```

I was able to create personality scores for the posts, which could be a potential area for further study. I would be interested in seeing a potential convergence of personality in posts on a thread or subreddit. The big issue I found with this approach was context-dependence. Different subreddits had different language and unique vocabularies. For example, "That's Bad" could mean many different things depending on context.

I then decided to try my implementation of Latent Dirichlet Allocation. Rather than relying on an association with arbitrary personality types, my implementation created word groups called topics and sorted the words into this topic.

The algorithm would first break the document up into topics and randomly assign each word to a topic. It would next take the proportion of words in a document assigned to a topic and find the proportion of assignments to the topic that come from a word.

For more details, see : https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2

Here is my implementation: [https://github.com/antonecg/RedditLDA/blob/master/Reddit-%20LDA.ipynb] With it, I was able to break down any Reddit threads into different topics. However, I was underwhelmed with the results. The topics were an interesting tool for explanation and could be interesting for social science research but did not have the predictive power that I was looking for.

I realized that if I wanted to predict how an online community shares, reacts to and interprets the news, I would need to move past statistical NLP and into Neural NLP.

Neural NLP

Seeing the limitations of Statistical NLP methods, I decided to explore neural NLP methods and create my implementation.

This research was extremely computationally intensive, but fortunately, I had access to a rack-mounted server with a Tesla K80 and could run days-long jobs.

I researched three areas of Neural NLP; word embeddings, recurrent neural networks, and transformers. I implemented a transformer architecture based on google's BERT with my Reddit training set.

The word embedding technique I researched was Word2Vec, created by Thomas Mikolov at google. Word2Vec uses two-layer neural networks to generate vectors out of massive blocks of text.

For example, a vector would represent the word "Help". It could have a neuroticism score of 90/100 and an extraversion score of 40/100. Using cosine similarity, one is then able to find similar words.

For an excellent introduction to word2vec, see: [https://jalammar.github.io/illustrated-word2vec/]

This seemed like the next step after my research into Latent Dirichlet Allocations, but I found that word embeddings are generally pre-trained on very large datasets, so are not context-specific. I would not be able to create a word embedding on a Reddit thread and thus would be unable to use word embeddings to predict how online communities would predict information.

All of the models I had researched or implemented up to this point relied on the "bag of words" notion and looked only at words rather than building a notion of context.

However, we are living in a time of rapid progress with deep learning, and new algorithms are able to develop an understanding of context and have a memory.

While a traditional neural network only considers the immediately preceding state, a recurrent neural network implements back-propagation. This means that each layer feeds forward to the next layer, keeping a memory of recent layers.

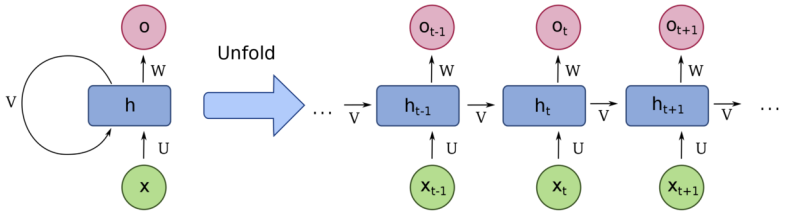


Figure 1: [http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/]

In the context of my research, this means that the RNN can understand the whole sentence, looking at multiple words at once and how they come together.

However, as the layers are stacked, the last layers gradually lose the ability to consider earlier layers and end up ignoring them. This is commonly known as the "Vanishing Gradient Problem" [https://en.wikipedia.org/wiki/Vanishing_gradient_problem]

Luckily this is a problem that Google has invested time and money into solving, and they have created a more complex architecture called a Transformer. This adds another layer to an RNN. This layer looks at preceding

words or sentences and can determine what words or concepts are essential. For more detail, I recommend the article "The Illustrated Transformer" by Jay Alamar. [https://jalammar.github.io/illustrated-transformer/]

Google's BERT transformer is an exciting and powerful tool that has been used to greatly enhance the quality of google's search results and is open source. This means that I can train models with the BERT Architecture.

BERT allows you to predict a word or series of words that belong in a sentence based on the context. A trained BERT model could fill out the blank with the example sentence: _____ like to bark and are man's best friend.

If you could train a transformer on a Reddit thread or subreddit, you could predict how the group posting would answer a new, unique question.

You can see my attempts here:

[https://github.com/antonecg/Deep-Learning/blob/main/SmallThreadBert.ipynb]

Initially, I used full-length posts but ran into time constraints (it would take weeks to run). I then created a model that ran on only a subset of posts, but the results thus far have not been satisfactory. The dataset that I was using, my harvested Reddit data, was far too small to create an accurate BERT Model.

Fortunately, there is cutting-edge (last few weeks) research trying to create transformer architectures that can be trained on smaller data sets. I am very excited about Deep Mind's Retrieval Transformer architecture and home to start my implementation in the coming weeks.

https://deepmind.com/research/publications/2021/improving-language-models-by-retrieving-from-trillions-of-tokens

References

Statistical Study of Literary Vocabulary. (1944).

On Tables of Random Numbers. (25AD). *Sankhyā Ser, 1963.*

Mining big data to extract patterns and predict real-life outcomes.. (2016). *Psychol Methods*, *21*, 493–506.