WHY SO SERIOUS

Introduction

What is natural language processing? Very simply put, it's turning words into numbers so we can "teach" our computers what language is. Computers don't understand words, but they understand numbers exceptionally well. When we turn words into numbers, the computer can do things that would take us meat machines years to do.

To better understand why we need to turn words into numbers, let's look at a real life example. Let's say I want to know how viewers feel about the new Joker movie. I go to the theater with a recording device and I ask 100 people what they thought about the movie and record their answers. I then go to the office and hand these recordings to my interns who laboriously transcribe everything that was said. We now have a text document of 100 reviews.

Unfortunately, we didn't think to ask the interviewee outright if s/he liked the movie, but we have all the words they used and my interns are fairly confident they can determine if someone liked/disliked the movie simply by reading the reviews. The interns get to work labeling each review as "liked" or "disliked." Since things like sentiment, or perceived sentiment, can be somewhat subjective, I have three interns look at each review. They blindly label the review as "liked" or "disliked" and if each review was agreed upon, it gets labeled and set aside. If the review wasn't agreed upon, it goes to three more interns. Eventually, we end up with an excel sheet for these reviews. The excel sheet is two columns of 100 rows. The first column is the label (assigned by the intern system) and the second column is a gigantic column containing the entire transcribed review.

This is great, but what happens when we want 100 more reviews? 1000 more reviews? What happens when our data is challenged by people living in a different demographic? What happens when we realized we inadvertently only polled white males ages 12-24? Do we send interns out to every single movie theater with recorders? We most certainly do not. We take advantage of the great world wide web!

We scrape IMDB for reviews use Natural Language Processing, of course!

Analysis & Models

ABOUT THE DATA

The review data was scraped from IMDB using `Beautiful Soup`. The code can be found in Appendix C (C for Code). The reviews were scraped by getting ~25 'most helpful' reviews for each star rating, 1-10, and exporting that text into a txt document. The total number of reviews ended up at 246. Reviews 5 stars and below (123 reviews) were transferred into a

“NEG” folder to become our “Negative corpus.” Reviews 6 stars and above (also 123 reviews) were transferred into a “POS” folder to become our “Positive corpus.”

Preliminary Analysis

Text Blob

	Kendra’s Data	Ami’s Data	Cornell Data	Dirty Data	Joker Data
CORRECT NEG	5	1	229	227	64
CORRECT POS	0	4	971	972	114

VADER

	Kendra’s Data	Ami’s Data	Cornell Data	Dirty Data	Joker Data
CORRECT NEG	2	3	445	454	64
CORRECT POS	5	3	828	824	114

NLTK

	Kendra’s Data	Ami’s Data	Cornell Data	Dirty Data	Joker Data
CORRECT NEG	--	--	89%	86%	81%
CORRECT POS	--	--	74%	70%	35%
ACCURACY	--	--	81%	77%	58%

K-NLTK

	Kendra’s Data	Ami’s Data	Cornell Data	Dirty Data	Joker Data
CORRECT NEG	--	--	89%	86%	81%
CORRECT POS	--	--	74%	70%	35%
ACCURACY	--	--	81%	77%	58%

Secondary Analysis

1. Started fresh using column 1
2. Created a “clean_review” column where \n was turned into spaces and titles counted for double
3. Used casual tokenizer and Counter to get a bow

Results

Conclusion

APPENDIX C:

```
https://danielcaraway.github.io/html/HW3_JOKER_IMDB_reviews.html
```

```
import re
import urllib
from bs4 import BeautifulSoup

def get_reviews(rating):
    url = "https://www.imdb.com/title/tt7286456/reviews?sort=helpfulnessScore&dir=desc&ratingFilter=" + str(rating)
    html = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(html, 'html.parser')
    text = soup.findAll("div", {"class": "imdb-user-review"})
    for num,t in enumerate(text):
        scale = t.find("span", {"class": "point-scale"})
        title = t.find("a", {"class": "title"})
        text = t.find("div", {"class": "text show-more__control"})
        review = title.text + "==" + text.text
        try:
            print_to_file(scale.previous_sibling.text, review, num)
        except:
            print('nope')

def print_to_file(rating, review, num):
    both = rating + '**' + review
    output_filename = str(rating) + '_jk_' + str(num) + '.txt'
    outfile = open(output_filename, 'w')
    outfile.write(both)
    outfile.close()

for rating in range(1,11):
    get_reviews(rating)
```