9/24: Sentiment Analysis

Discord: https://discord.gg/68VpV6

A Quick Development Tip

Development Tip: Github!

- Whenever you do any project of whatever size, it's always a good idea to have it somewhere on your GitHub

- For example, if you complete any of our notebooks, create a repository and upload it to your GitHub!

Steps to do this for SIGNLL

1) Create a new repository (title it something like SIGNLL 2020)

2) Create a folder inside the repository for each project

3) Organize your repository!!!

Extra help: https://reproducible-science-curriculum.github.io/sharing-RR-Jupyter/01-sharing-github/

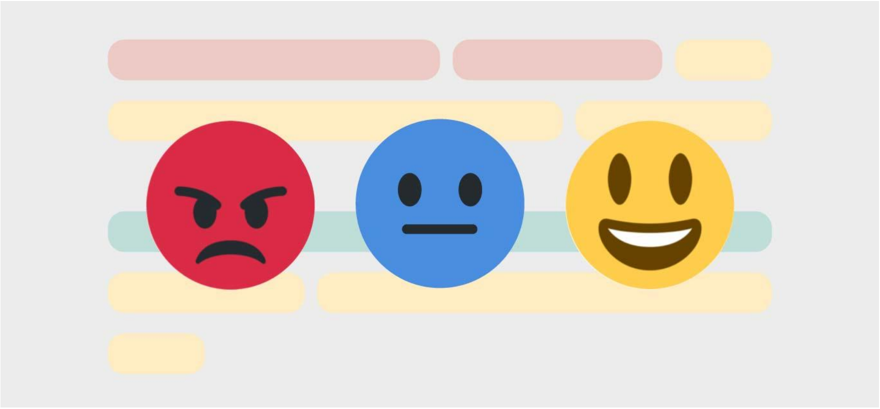
Sentiment Analysis

How can we extract emotion from text?

What is Sentiment Analysis?

"

Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data using text analysis techniques



How do <u>we</u> know?

SIGNLL
@signll

I hate stupid Purdue

12:00 PM · Jun 1, 2020

SIGNLL
@signll

UIUC is the best!

12:00 PM · Jun 1, 2020

Which tweet is positive? Which is negative? Why?

Classifying Words

negative tweet

positive tweet

Organizing These Words

We can organize our tweets using this data table:

Word	Negative Tweet Count	Positive Tweet Count
I	1	0
hate	1	0
stupid	1	0
Purdue	1	0
UIUC	0	1
is	0	1
the	0	1
best!	0	1

Sentiment Analysis Steps

1) Process and clean up our tweets

2) Organize them into a dictionary

3) Create and train a model using logistic regression

4) Verify our results with test data

Multiple Word Definitions

Do these tweets basically mean the same thing?

SIGNLL
@signll

I am very happy today!

12:00 PM · Jun 1, 2020

SIGNLL
@signll

I am happier today!

12:00 PM · Jun 1, 2020

SIGNLL
@signll

I am the happiest today!

12:00 PM · Jun 1, 2020

Word Tokenization

Fortunately, we can run a built-in algorithm to make sure that these words all mean the same thing

["happy", "happier", "happiest"]

"happy"

Stopwords

There are also some words that add no meaning to the sentence, such as:

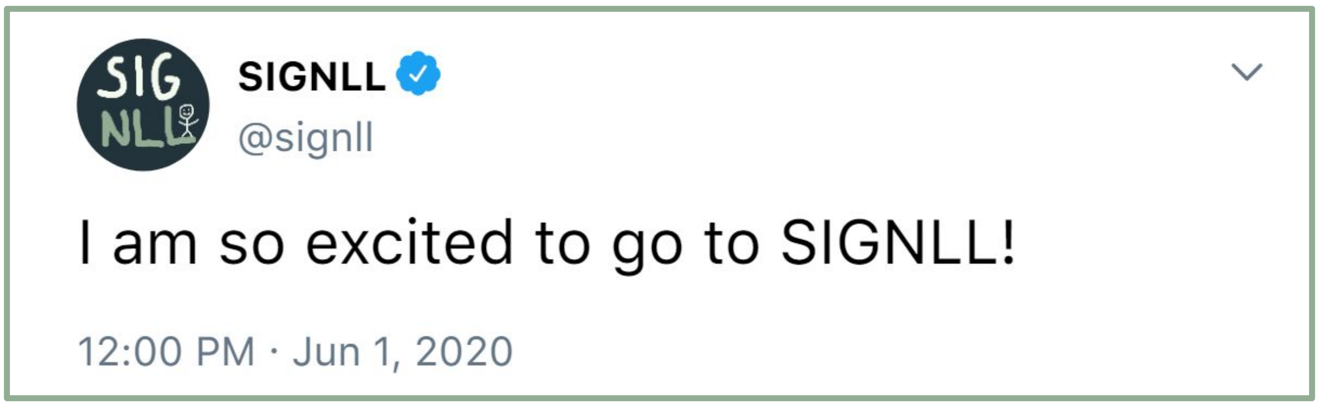
```
["the", "a", "an", "I", ...]
```

Since these words don't help us that much, we can remove them from our tweets

Note: We'll also being doing this with punctuation and twitter symbols (@, #, etc.)

Step 1: Processing our tweet

After tokenizing, removing stop words, and some other steps (like removing punctuation), we can convert our tweet to useful data for the computer:



`[“excit”, “go”, “signll”]`

Step 2: Building our dictionary

Recall our data table from earlier:

Word	Negative Tweet Count	Positive Tweet Count
I	1	0
hate	1	0
stupid	1	0
Purdue	1	0
UIUC	0	1
is	0	1
the	0	1
best!	0	1

Dictionary Conversion

Instead of having separate columns for positive tweet count and negative tweet count, we can represent them as a single list of length 2:

$$\{n_{neg} \;\; n_{pos}\}$$

This is useful because we can represent our dictionary with the following key/value pair:

$$word : \{n_{neg} \;\; n_{pos}\}$$

Example Dictionary

A common dictionary of tweets might look like:

```
Sample Tweet Dictionary:
-----
{'excit': [4, 28], 'aw': [30, 2], 'angri': [5, 3],

 'want': [185, 69], 'wish': [91, 113]}
```

(This is for the words exciting, awful, angry, want, and wish. The real dictionary would have many many more words)

Step 3: Training our model

Since we are predicting categorical data (whether the tweet is positive or negative), we can use **logistic regression**!

Reminder: logistic gradient descent equation:

```
def log_grad_descent(x, actual_y, thetas, learning_rate, m, num_iterations):
    # perform the algorithm for the specified number of iterations
    for iteration in range(num_iterations):
        # calculate our sigmoided predicted output
        pred_output = sigmoid(x @ thetas)
        # get the derivative of this value
        gradients = lin_reg_derivative(pred_output, actual_y, x, m)
        # adjust our thetas
        thetas = thetas - learning_rate * gradients
    return thetas
```

Representing our tweets as numbers

To use logistic regression, we need to represent our tweets as numbers

We can do this by using three parameters from our dictionary:

$$\{1 \; \sum count_{neg} \; \sum count_{pos}\}$$

The first parameter is used for the intercept, while the second and third parameters add all of negative and positive counts for each word in the tweet, respectively

Tweet representation example

Tokens: ["wish", "friend", "like"]

"wish" : [63, 29]
+
"friend" : [30, 40]
+
"like" : [182, 187]

---

tweet_val = [1, 275, 256]

Logistic Gradient Descent

Now that we have all of our tweets represented as numbers, we can run logistic gradient descent as normal!

$$y = \begin{Bmatrix} 1 & neg_0 & pos_0 \\ 1 & neg_1 & pos_1 \\ \dots & \dots & \dots \\ 1 & neg_{m-1} & pos_{m-1} \end{Bmatrix} \cdot \{\Theta_0 \ \Theta_1 \ \Theta_2\}$$

$$y = sigmoid(y)$$

Step 4: Testing our model

There's two ways we can test our model: with the training set or our own custom inputs

It's always a good idea to verify your data with the test set before using custom inputs

One common metric for an evenly distributed dataset is accuracy, defined as:

$$accuracy = \frac{n_{correct}}{n_{total}}$$

Tasks to Complete

1) Work on the notebook (**twitter_code.zip**) in the google drive folder https://drive.google.com/drive/folders/1XinmyYoyhzVSNvzz7Djzoxkvg3rnekD1?usp=sharing

Try to work on it collaboratively! You might meet some people you could do a project with in the future

As always, let us know if you need any help!

SIG
NLL