Fake News Detection

Carson Hanel
Mohammed Habibullah

Explanation of our program

During the last year, one of the issues that has plagued the global political spectrum has been the prevalence of unsubstantiated news reporting. Though there has been a wide push for journalistic integrity and rigor to intensify, still we've seen that unsubstantiated or misleading news has made its way into social circles including online media locations, such as Twitter and Facebook. This project is our naive approach to understanding patterns in news which would be indicative of a false article, which builds upon features extracted from the corpus to essentially draw a line in the sand between news articles that have been deemed substantiated. We hope that you like what we've done, but we would like to remind you all that this is a closed environment test, and as such is biased due to some data being orthogonal between real vs. fake datasets.

Program flow

- Scraping and cleaning of data
- Feature engineering
- Maximum entropy model learning of body textual content
- Maximum entropy model learning of head textual content
- Maximum entropy Frankenstein model (original)
- Feature extraction
- Psuedo maximum entropic classifiers run through neural network
- Gradient Boost scoring of final extracted features across all documents
- Evaluation of real vs. fake accuracy
- Data insight gathering

Scraping the data and data cleaning

Extracted the Fake News data from Kaggle and the real news data from TheGuardian API.

Dropped the irrelevant News sections and retained news articles on US news, Business, Politics & World News and converted it to .csv format.

Data preprocessing:
1. dropped irrelevant columns such as urls, likes and shares info etc.
2. removed stop words and high bias word like guardian and the guardian etc.
3. Performed lemmatization to bring the word to their basic form

Merged the Fake News and Real News into a csv with an additional Fakeness column.

Feature Engineering

Jaccard Similarity: It calculates the overlap between the headline and body of the news article.

Sentiment Feature: It calculates the polarity scores of headline and body using Vader(Valence Aware Dictionary and Sentiment Reasoner) sentiment analyzer.

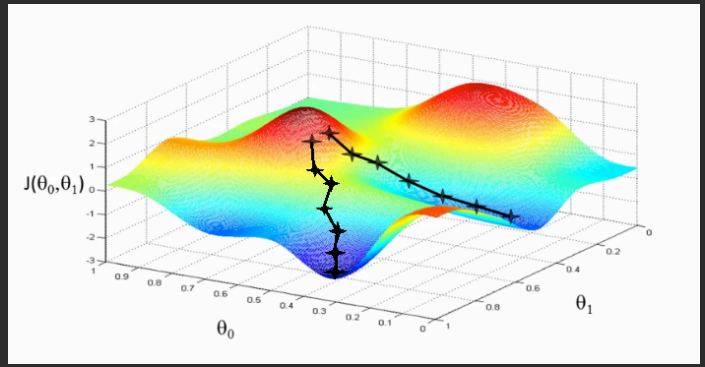
Named Entity similarity feature: It calculates the cosine similarity between the counts of the named entities from the Headline and Body.

Polarity feature: It indicates the polarity of headline and body towards refuting words like ‘fraud’, ‘hoax’, ‘debunks’, ‘denies’.

Miscellaneous features: it’s the counts of various chargrams and n-grams that are occurring in the headline and body.

Maximum Entropy Classifier

```
while(change > eta):
    changes += 1
    if change == 99:
        change = 0
    sum_weights = 0
    for w in set(words):
        w = w.lower()
        sum_weights += self.weights[int(self.words[w])]
    for w in set(words):
        w = w.lower()
        prev_weight = self.weights[int(self.words[w])]
        x_i_j       = occurrence[int(self.words[w])]
        y_i         = klass_int
        lamb_weight = -1 * (lambdaa * prev_weight)
        document_p  = 1 / (1 + np.exp(-sum_weights))

        new_weight  = prev_weight + eta*(lamb_weight + x_i_j*(y_i - document_p))
        self.weights[int(self.words[w])] = new_weight

        change     += new_weight - prev_weight
        parse      += 1

    change          = change / len(set(words))
```



Maximum Entropy Results

```
[INFO]      Fold 0 Accuracy: 0.928150
The vocab length is: 146966

[INFO]      Fold 1 Accuracy: 0.925197
The vocab length is: 147240

[INFO]      Fold 2 Accuracy: 0.918307
The vocab length is: 145156

[INFO]      Fold 3 Accuracy: 0.922736
The vocab length is: 146935

[INFO]      Fold 4 Accuracy: 0.927165
The vocab length is: 148125

[INFO]      Fold 5 Accuracy: 0.934547
The vocab length is: 147194

[INFO]      Fold 6 Accuracy: 0.928642
The vocab length is: 147844

[INFO]      Fold 7 Accuracy: 0.934547
The vocab length is: 147895

[INFO]      Fold 8 Accuracy: 0.930576
The vocab length is: 147840

[INFO]      Fold 9 Accuracy: 0.923191
[INFO]      Accuracy: 0.927306
```

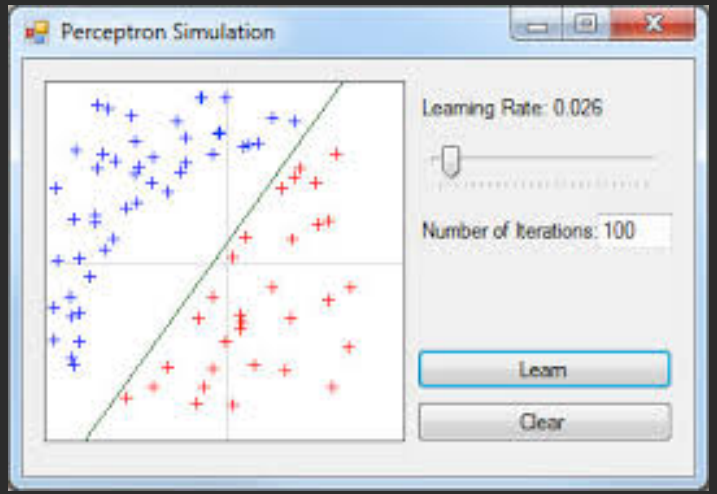
To the left you'll see body content

To the right, you'll see headline content

```
[INFO]      Fold 0 Accuracy: 0.786417
The vocab length is: 17445

[INFO]      Fold 1 Accuracy: 0.774606
The vocab length is: 17357

[INFO]      Fold 2 Accuracy: 0.784941
The vocab length is: 17402

[INFO]      Fold 3 Accuracy: 0.779035
The vocab length is: 17406

[INFO]      Fold 4 Accuracy: 0.782972
The vocab length is: 17423

[INFO]      Fold 5 Accuracy: 0.781988
The vocab length is: 17406

[INFO]      Fold 6 Accuracy: 0.785925
The vocab length is: 17469

[INFO]      Fold 7 Accuracy: 0.790846
The vocab length is: 17387

[INFO]      Fold 8 Accuracy: 0.788774
The vocab length is: 17408

[INFO]      Fold 9 Accuracy: 0.780404
[INFO]      Accuracy: 0.783591
```

Perceptron Classifier

```
klass_int = 1
if(klass == 1):
    klass_int = -1
occurrence = np.zeros(self.vocab_length)

for i in range(0, iterations):
    sum_weights = 0.
    #Set(words) gets the average of the individual words, not of all words in doc.
    for w in set(words):
        sum_weights += self.weights[int(self.words[w])]
        occurrence[int(self.words[w])] += 1
    #Changing set(words) to words for feature engineering. More features = more weight
    for w in set(words):
        occur       = occurrence[int(self.words[w])]
        if np.sign(sum_weights * occur) != klass_int:
            self.weights[int(self.words[w])] += (klass_int - np.sign(sum_weights * occur)) * occur

pass
```



Perceptron Results

```
[INFO]      Fold 0 Accuracy: 0.909941
The vocab length is: 146966

[INFO]      Fold 1 Accuracy: 0.938484
The vocab length is: 147240

[INFO]      Fold 2 Accuracy: 0.933071
The vocab length is: 145156

[INFO]      Fold 3 Accuracy: 0.922244
The vocab length is: 146935

[INFO]      Fold 4 Accuracy: 0.923228
The vocab length is: 148125

[INFO]      Fold 5 Accuracy: 0.924705
The vocab length is: 147194

[INFO]      Fold 6 Accuracy: 0.917323
The vocab length is: 147844

[INFO]      Fold 7 Accuracy: 0.935531
The vocab length is: 147895

[INFO]      Fold 8 Accuracy: 0.929591
The vocab length is: 147840

[INFO]      Fold 9 Accuracy: 0.928607
[INFO]      Accuracy: 0.926273
```

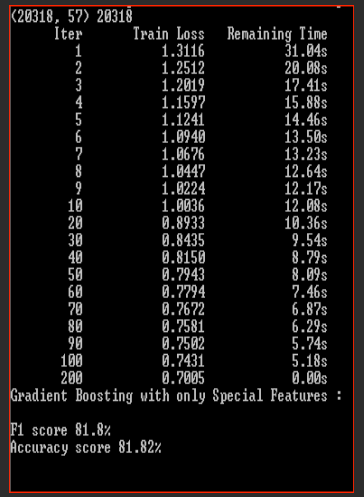
To the left, you'll see the body content

To the right, you'll see the headline content

```
[INFO]      Fold 0 Accuracy: 0.775591
The vocab length is: 17445

[INFO]      Fold 1 Accuracy: 0.788386
The vocab length is: 17357

[INFO]      Fold 2 Accuracy: 0.768701
The vocab length is: 17402

[INFO]      Fold 3 Accuracy: 0.777067
The vocab length is: 17406

[INFO]      Fold 4 Accuracy: 0.771654
The vocab length is: 17423

[INFO]      Fold 5 Accuracy: 0.794291
The vocab length is: 17406

[INFO]      Fold 6 Accuracy: 0.776575
The vocab length is: 17469

[INFO]      Fold 7 Accuracy: 0.787894
The vocab length is: 17387

[INFO]      Fold 8 Accuracy: 0.768587
The vocab length is: 17408

[INFO]      Fold 9 Accuracy: 0.763171
[INFO]      Accuracy: 0.777192
```

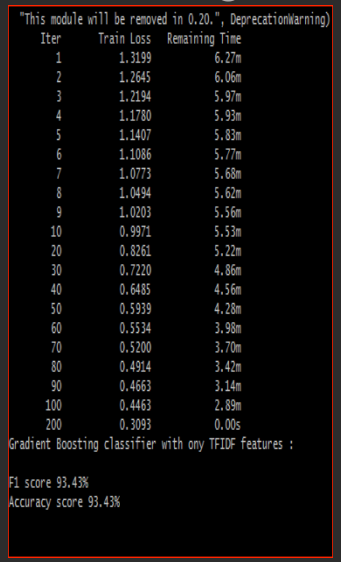
Baselines and Improvements

Baseline is running Gradient Boosting Classifiers with just the features extracted in Feature Engineering.
The Baseline has F1 Score= 81.8% and Accuracy= 81.82%.
We found

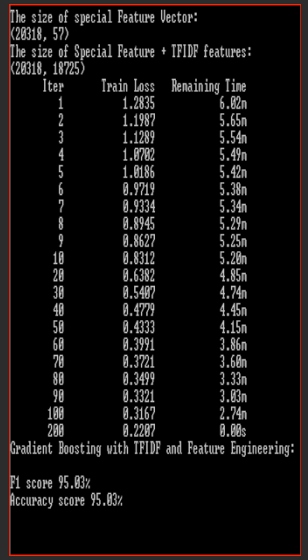


Few More Approaches with TF-IDF Features:

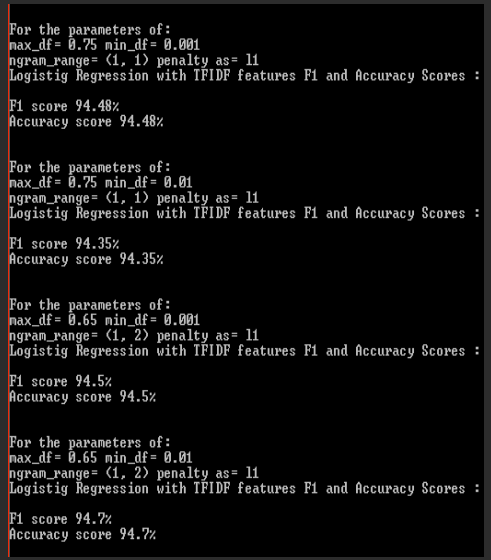
We then tried with TF-IDF as Feature Vectors for logistic regression classifier and Gradient Boosting Classifier. And then combined them with extracted features from Feature Engineering. Below are the results for the same



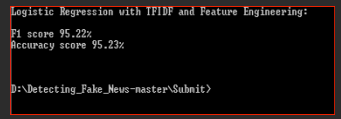
GB With only TF-IDF



GB With TF-IDF + extracted Features

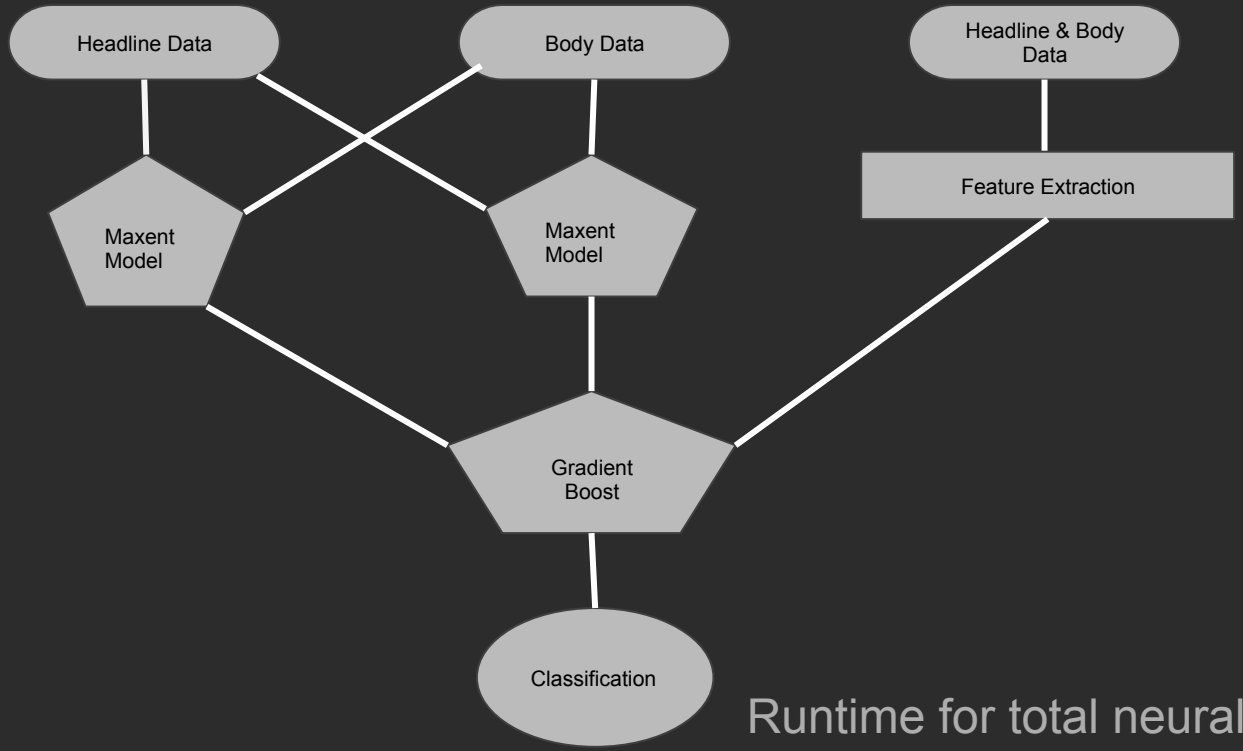


Logistic Regression with only TF-IDF



Logistic Regression with TFIDF + extracted Features

Neural Network design



Runtime for total neural network to train:

Approximately 2 to 2.5 hours

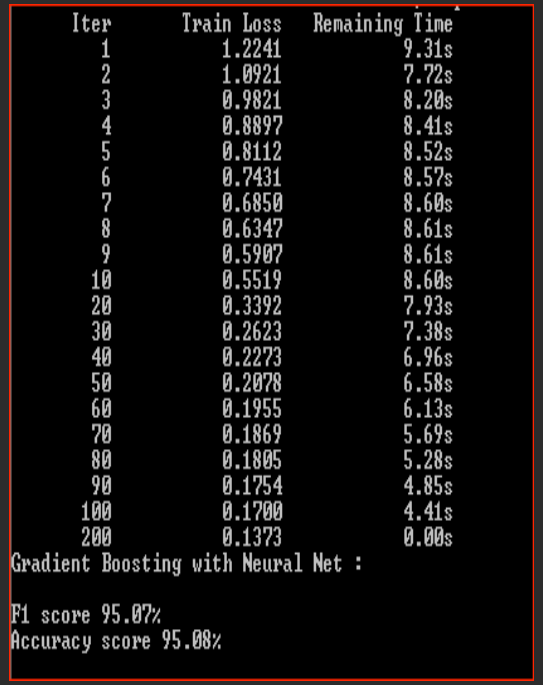
Gradient Boosting Classifier

It is a classifier which produces a prediction model in the form of an ensemble of weak prediction models in a forward stage-wise fashion.

At each stage
1. It computes the residual error (loss function e.g Mean Square error)
2. Learn to predict the residual
3. Combines (adds) the new model with the predecessor weak model

The classifier can be tuned by parameters like
n_estimators : The number of boosting stages to perform.
loss : loss function to be optimized.
learning_rate : There is a trade-off between learning_rate and n_estimators.

Gradient Boosting Classifier results

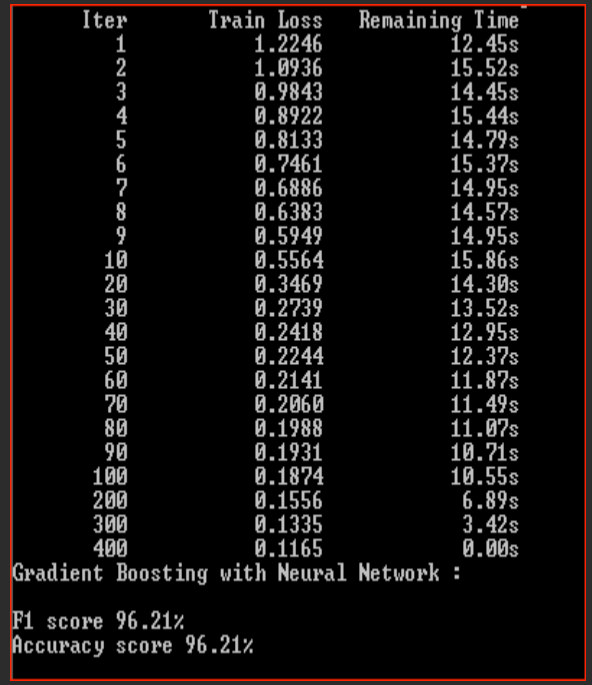


Iter	Train Loss	Remaining Time
1	1.2241	9.31s
2	1.0921	7.72s
3	0.9821	8.20s
4	0.8897	8.41s
5	0.8112	8.52s
6	0.7431	8.57s
7	0.6850	8.60s
8	0.6347	8.61s
9	0.5907	8.61s
10	0.5519	8.60s
20	0.3392	7.93s
30	0.2623	7.38s
40	0.2273	6.96s
50	0.2078	6.58s
60	0.1955	6.13s
70	0.1869	5.69s
80	0.1805	5.28s
90	0.1754	4.85s
100	0.1700	4.41s
200	0.1373	0.00s

Gradient Boosting with Neural Net :

F1 score 95.07%
Accuracy score 95.08%

With One Fold



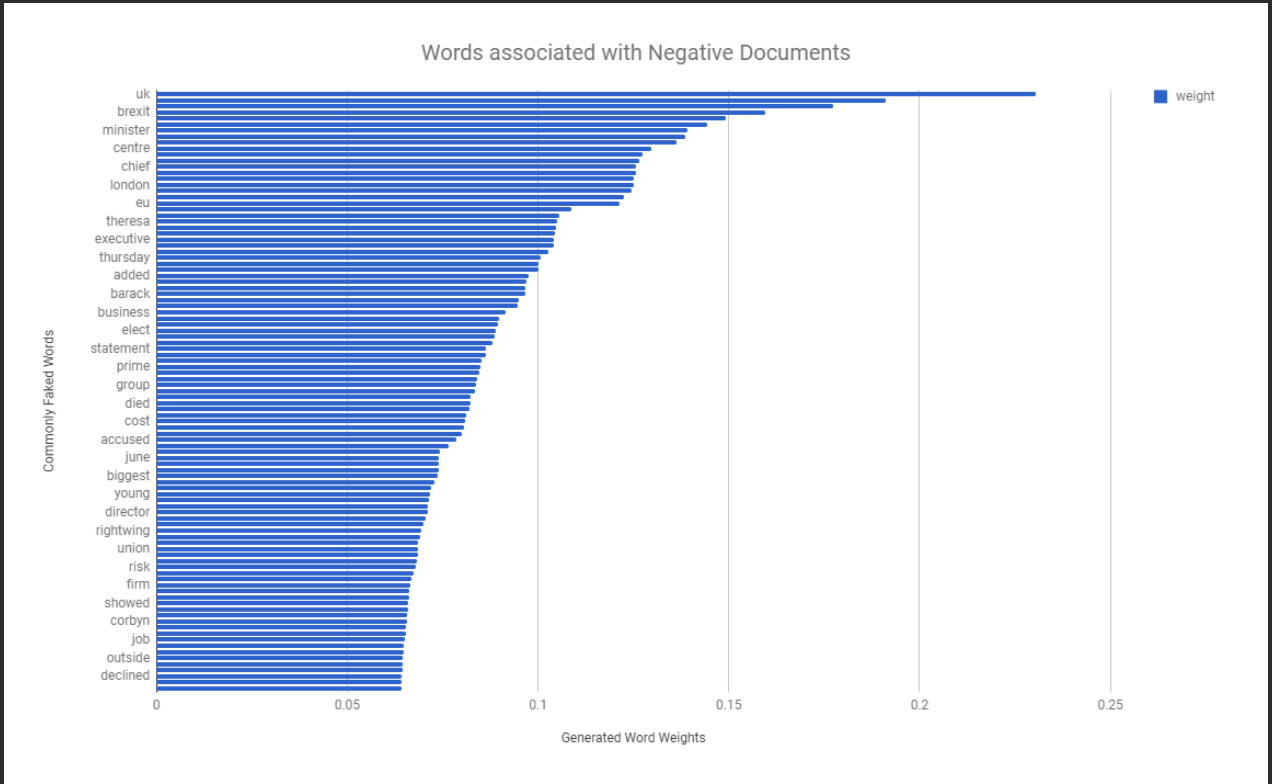
Iter	Train Loss	Remaining Time
1	1.2246	12.45s
2	1.0936	15.52s
3	0.9843	14.45s
4	0.8922	15.44s
5	0.8133	14.79s
6	0.7461	15.37s
7	0.6886	14.95s
8	0.6383	14.57s
9	0.5949	14.95s
10	0.5564	15.86s
20	0.3469	14.30s
30	0.2739	13.52s
40	0.2418	12.95s
50	0.2244	12.37s
60	0.2141	11.87s
70	0.2060	11.49s
80	0.1988	11.07s
90	0.1931	10.71s
100	0.1874	10.55s
200	0.1556	6.89s
300	0.1335	3.42s
400	0.1165	0.00s

Gradient Boosting with Neural Network :

F1 score 96.21%
Accuracy score 96.21%

For Complete Dataset

Real News Weights for Body



Real News Weights Overview

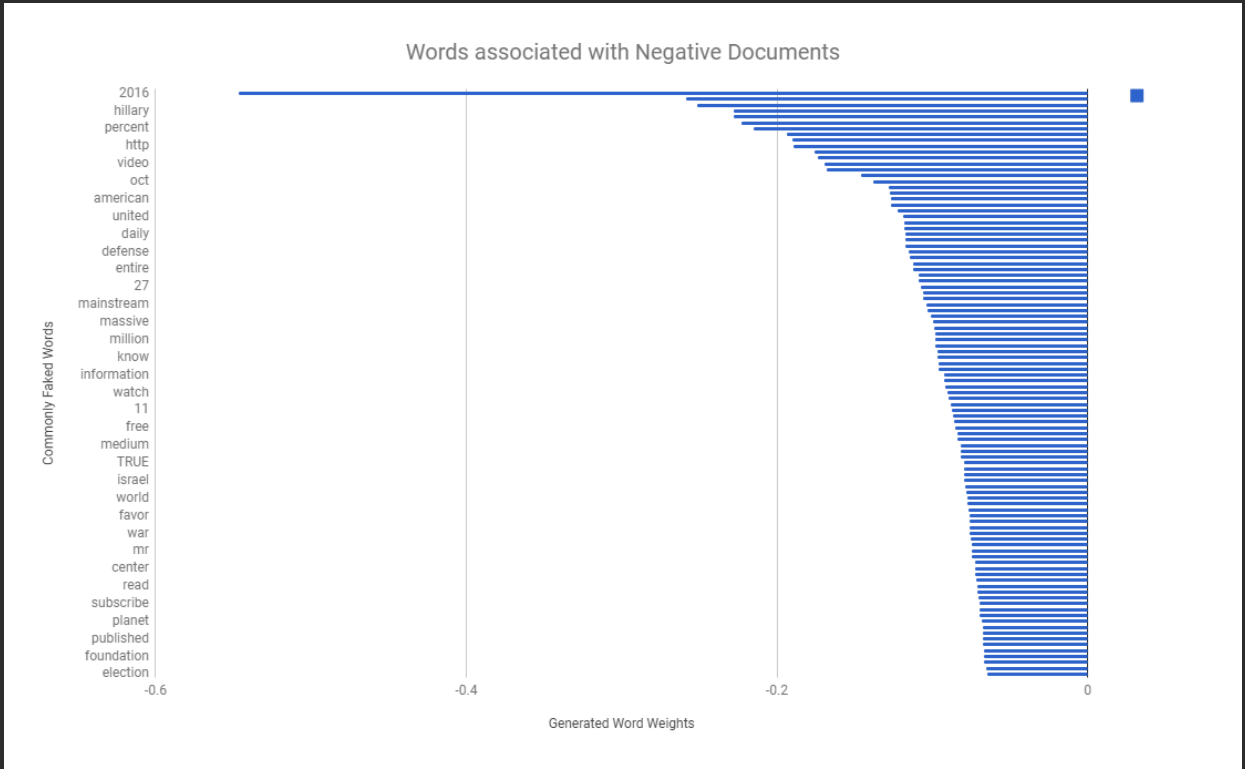
uk	0.2304523645
said	0.191139028
debate	0.1773307965
brexit	0.1595648109
labour	0.1491834285
britain	0.1442656238
minister	0.1391415865
monday	0.1385187117
month	0.1364134753
centre	0.129734339
told	0.127470513
mp	0.1264412825
chief	0.1255605275
organisation	0.1255348329
week	0.1250984511
london	0.1250680793
british	0.1245329114
amp	0.1226411874
eu	0.1212557917
programme	0.1088099823
tuesday	0.1056857545
theresa	0.1049138341
spokesman	0.1047617226
sunday	0.1045576247

sunday	0.1045576247
executive	0.1041613018
friday	0.1040331944
referendum	0.1026565721
thursday	0.1005768822
leader	0.1001224757
year	0.1000644836
added	0.09740772242
company	0.09701823054
expected	0.09679307677
barack	0.09660608323
suggested	0.09497332406
conservative	0.09470215639
business	0.09150318767
criticised	0.08977945198
letter	0.08950102492
elect	0.08895509
including	0.0886614078
described	0.08814813707
statement	0.08634022604
0	0.08619787281
august	0.08519643977
prime	0.08489776501
city	0.08453473784
figure	0.08408558411
group	0.08361559643

group	0.08361559643
december	0.08353842937
wednesday	0.08232457211
died	0.08219123099
party	0.08199422449
saying	0.08107482378
cost	0.08084950455
senior	0.08060254001
asked	0.08006399208
accused	0.07867601853
favour	0.07655141549
tweeted	0.07427962954
june	0.07413851403
night	0.07404306952
nbsp	0.07403316278
biggest	0.0737153277
defence	0.07288146218
senator	0.07198371114
young	0.07164215915
remain	0.07148765634
low	0.0712243293
director	0.07116902832
jeremy	0.07043389605
rival	0.06992035585
rightwing	0.06932859911
remark	0.06913917784

remark	0.06913917784
amid	0.06852537501
union	0.06851821866
saturday	0.06844103931
conference	0.06820633665
risk	0.06807347672
warned	0.06748143682
decision	0.06683659897
firm	0.06639997152
ahead	0.06638102556
raised	0.06616892645
showed	0.06609124317
tory	0.06597991531
concern	0.06579102038
corbyn	0.06557139932
second	0.06539014791
july	0.06525588762
job	0.06523014955
christmas	0.06484553332
parliament	0.06467659497
outside	0.06458180844
adviser	0.06453551411
secretary	0.06451869863
declined	0.06434120828
governor	0.06429042112
hour	0.06415323134

Fake News Weights for Body



Fake News Weights Overview

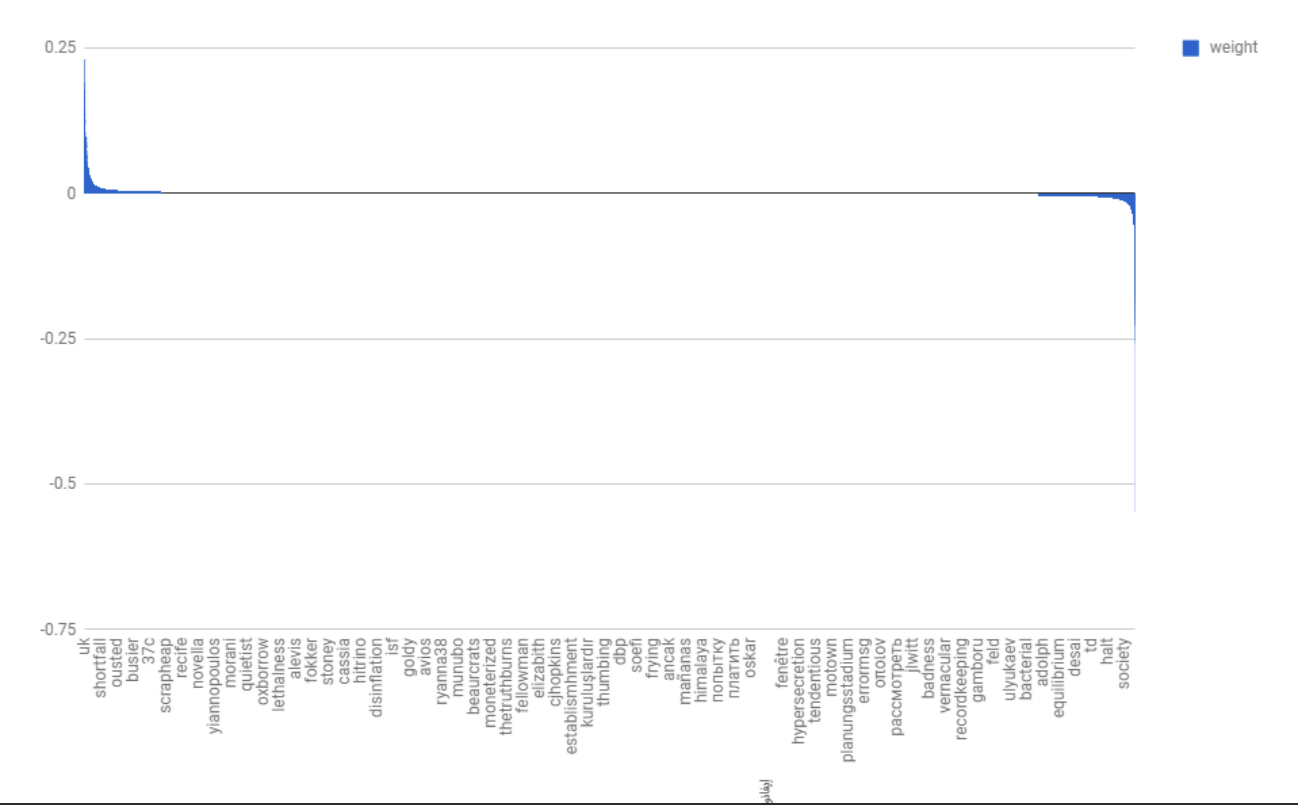
2016	-0.5465530145
october	-0.2582369477
com	-0.2510103496
hillary	-0.227988887
source	-0.2276518261
clinton	-0.2227701836
percent	-0.2151590714
november	-0.1935076921
share	-0.1900641032
http	-0.1892323488
posted	-0.1760477544
article	-0.1735624564
video	-0.1691586234
news	-0.1677348449
today	-0.1458657728
oct	-0.1380025297
post	-0.1278402894
fact	-0.1274498363
american	-0.1264814711
email	-0.1263399876
follow	-0.1222963467
united	-0.1189784178
billion	-0.1181693064
nov	-0.1178560603
daily	-0.1176511745

daily	-0.1176511745
wikileaks	-0.1176379722
facebook	-0.1173323595
defense	-0.1149877521
organization	-0.1147765805
twitter	-0.1125610263
entire	-0.1123355791
fbi	-0.1090123534
related	-0.1084040573
27	-0.107155848
actually	-0.1061837938
www	-0.1057524752
mainstream	-0.1039767337
america	-0.1034013475
course	-0.1011050844
massive	-0.09930458761
image	-0.09883804539
click	-0.09810685791
million	-0.0979222946
program	-0.09773277068
comment	-0.09644709145
know	-0.09636966282
print	-0.09621653547
author	-0.09613448517
information	-0.0926216764
nation	-0.09228276062

nation	-0.09228276062
pic	-0.09184357077
watch	-0.0903484341
note	-0.08927293762
reason	-0.08783437183
11	-0.08715431173
like	-0.08697139073
truth	-0.0859191111
free	-0.08538521559
28	-0.08386533313
just	-0.08345858377
medium	-0.08186035902
mail	-0.08166113938
al	-0.08144808431
TRUE	-0.07977473598
state	-0.07976071667
russia	-0.07950001717
israel	-0.07946371544
add	-0.07880506704
corruption	-0.07799798057
world	-0.07736618346
stated	-0.07736374312
order	-0.07661173035
favor	-0.07630319357
various	-0.07630279281
pm	-0.07578820307

pm	-0.07578820307
war	-0.07566862689
elite	-0.07558085973
trump	-0.07480382313
mr	-0.07461463904
story	-0.07433974075
26	-0.07258345326
center	-0.07256656177
podesta	-0.07216809369
editor	-0.07158789787
read	-0.07128013462
explained	-0.07107440487
corrupt	-0.07012743366
subscribe	-0.06979443844
secret	-0.0694570452
isn	-0.06933839091
planet	-0.06833019426
military	-0.06760055131
example	-0.06756858498
published	-0.06717534638
youtube	-0.06706256701
earth	-0.06697400506
foundation	-0.06692479884
rt	-0.06673267703
breaking	-0.06561320411
election	-0.06490913688

Entire Corpus Weights



Resources

Vader Feature: http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf
http://www.nltk.org/api/nltk.sentiment.html

Jaccard Similarity:
http://www.iaeng.org/publication/IMECS2013/IMECS2013_pp380-384.pdf
https://pdfs.semanticscholar.org/1554/884e643d811c0a3933ea22d6995afc63e912.pdf

Gradient Boosting Classifier:
http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

Other References:
A simple but tough-to-beat baseline for the Fake News Challenge stance detection task
https://arxiv.org/pdf/1707.03264v1.pdf

Continued..

RNNs for Stance Detection between News Articles
https://web.stanford.edu/class/cs224n/reports/2757443.pdf

Stance Detection for the Fake News Challenge: Identifying Textual Relationships with Deep Neural Nets
https://web.stanford.edu/class/cs224n/reports/2757443.pdf

Social Media Writing Style Fingerprint

Himank Yadav, Juliang Li

1.
Overview

Overview

- Humans have the cognitive ability to differentiate between writing styles of various authors.
- Previous work has been done on authorship attrition for books and long texts.
- We focus on shorter writing samples gathered through social media

2.
Motivation

Sahil Dhanju
September 22 ·

I'm happy to announce that I have accpeted a full-time job at McDonalds to flip burgers and will start immediately after finishing my Computer Science Degree at Texas A&M. I know my family and friends are proud, and honestly now that I think about it, I could not have found a better job. Actually, based on my grade in Operating Systems and Calculus, I'm really suprised that I even made it this far. Drop by to say hi if you are in the area!

Love Comment

You, Juliang Li, Denise Irvin and 489 others

Tyler Durden I think you are right about how you couldn't have done any better. Well done Sahil.
Like · Reply · September 22 at 6:32pm

Jay Khatri lol did you get hacked again man?
Like · Reply · September 22 at 9:18pm

Write a comment...

Sahil Dhanju
September 24 ·

My account got hacked by someone that obviously knows me but I have no idea who they are. I luckily am not going to work at McDonald's.

Sad Comment

You, Juliang Li, Victoria Wei and 35 others

Alexandra Neikirk I was so proud
Like · Reply · September 24 at 9:52pm

Danny Byrne Rip the profile pic
Like · Reply · September 25 at 8:17am

Write a comment...

Applications

- Detect social media hacking activity.
- Establish the credibility of a source.
- Identifying anonymous negative phenomenon (bullying)

3.
Data

Collection & Cleaning

- Comments v.s. Posts bias
- VRCkid (aka Sahil) and other top redditors that he interacts with
- PRAW - Python Reddit API Wrapper
- Removed empty comments and edge cases

4.
Method

Word Frequency

- Counts the usage of vocabulary
- Decide features (bag of word)
- Use logistic regression to train and test
- 76% accuracy

Lexical KMeans

- Using unsupervised learning to capture distinctiveness in sentence structure.
- Tokenize words and sentences, focus on lexical features and punctuation style of text
- Word density, vocabulary diversity and punctuation placement
- Use clustering to find natural groupings, predict using a KMeans cluster ~ 69%

Character N-gram

- Views the text as a sequence of characters
- Use N-gram to extract data
- Select top N-gram
- 77% Accuracy

Author Verification for Short Messages

- Supervised learning combined with n-gram stylometric analysis
- Split data into training and verification, compute a specific threshold for the given user, derive user profile by extracting n-grams.
- Divide the verification user data into p blocks of characters of the same size
- Calculate the percentage of unique n-gram shared by blocks and the training set
- Block p is said to be a genuine sample of user if the percentage of unique n-grams shared by a block is greater than threshold value specified for the user

Parts of Speech

- Analyze Syntactic information
- Author may subconsciously make similar phrase structure.
- Pick the most frequently used structures as features

Master Classifier

- Multi-layered classifier simulating a neural net
- 5 input nodes where the final layer uses majority vote from the middle layer vector to classify
- Focus on strengths and weakness for each of the 5 classifiers
- Accuracy ~80%

Fold 1	0.819
Fold 2	0.810
Fold 3	0.824
Fold 4	0.819
Fold 5	0.794
Fold 6	0.813
Fold 7	0.794
Fold 8	0.789
Fold 9	0.826
Fold 10	0.778
Average	0.807

5.
Future Work

Extension

- Expand features to include semantic analysis and other external non-language data points
- Explore different social media datasets
- Build tooling to measure real world success

Questions?

Capture Discriminative Attributes

SemEval 2018 Task 10

By Yucheng Jiang

Background

- State of art semantic models do an excellent job at detecting semantic similarity. For example, calculating cosine similarity between two word vectors.
- Such model will be able to tell that cappuccino, espresso and americano are similar to each other. However, it cannot predict semantic differences between words.
- If you can tell that americano is similar to cappuccino and espresso but you can't tell the difference between them, you don't know what americano is. As a consequence, any semantic model that is only good at similarity detection will be of limited practical use.

Introduction

- The goal of my research is to build a model to extract sematic difference. To be more specific, the model is able to predict whether a word is a discriminative attribute between two other words.
- The model takes a triple like apple, banana, red and then determine whether it exemplifies a semantic difference or not.

word1	word2	word3	output
apple	banana	red	True
teeth	tongue	white	True
banana	cucumber	long	False

Data

Training data: contains 17782 automatically generated training examples with a total of 1292 distinct features.

Validation data: contains 2722 manually curated examples with a total of 576 distinct features.

17783 lines (17782 sloc) | 397 KB

```
coconut,rice,infested,0
ostrich,emu,cloves,0
bathtub,toilet,barrier,0
pan,skillet,lid,1
pumpkin,spinach,snake,0
crow,crane,flies,1
jeans,shirt,bug,0
raisin,cranberry,clasp,0
pillow,bag,shows,0
tie,blouse,jazz,0
parakeet,owl,reeds,0
dog,cow,chases,1
toad,donkey,barks,0
horse,lion,repelling,0
coyote,raccoon,eating,0
caterpillar,jeep,ballrooms,0
gate,church,hay,0
tuna,avocado,teachers,0
thimble,comb,dimples,1
olive,coconut,rope,0
crocodile,zebra,breakable,0
tangerine,raspberry,barks,0
cucumber,asparagus,seeds,1
otter,toad,freezing,0
```

Approach

- Build a binary classifier
- Sources of features:
  1. Google word2vec
  2. Wordnet
  3. Corpus in NLTK
- Machine learning algorithms:
  1. SVM
  2. Logistic Regression
  3. Decision Tree

Features
cosine similarity

- If one word is the feature of another word, they are more likely to share same context.

	word1	word2	feature	label	cos(w1,w2)	cos(w1,f)	cos(w2,f)	cos(w1,f)/cos(w2,f)
0	sandwiches	breakfast	lunch	1	0.401286	0.330694	0.220761	1.497905
1	surfboard	raft	water	0	0.640837	0.836909	0.613705	1.363677
2	banana	cucumber	long	0	0.428193	0.749014	0.848541	0.882698
3	lambs	cattle	wool	1	0.391495	0.566834	0.702309	0.807090
4	shrimp	spinach	pink	1	0.355456	0.721406	0.727034	0.992245
5	bus	taxi	passengers	0	0.274754	0.473767	0.424060	1.117190
6	garage	brick	large	1	0.472511	0.633485	0.577068	1.097745
7	walkway	alley	walk	0	0.508481	0.488866	0.527899	0.926042
8	polyurethane	polyester	material	0	0.339968	0.548181	0.531949	1.030495
9	spoon	potato	wood	1	0.495251	0.705570	0.739728	0.953811

Features
cosine similarity

- I also tried to use Euclidean distance and Manhattan distance as features.

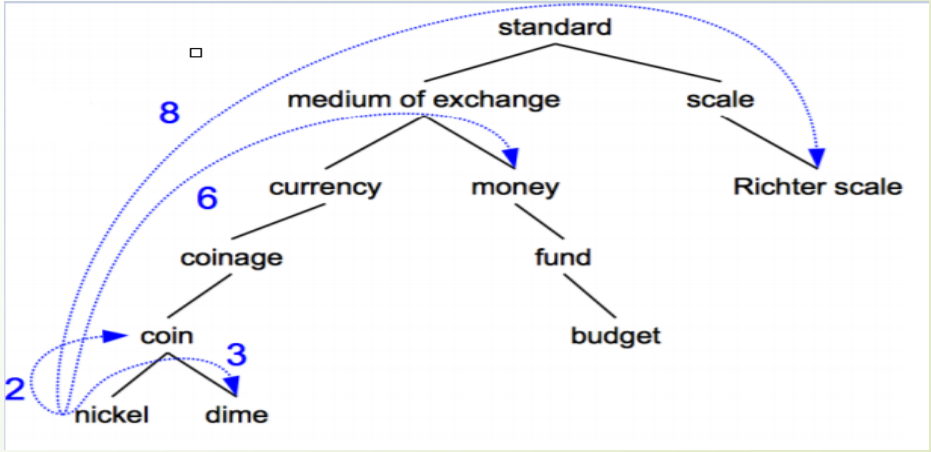
	SVM	Logistic Regression	Decision Tree
Cosine similarity	55.9%	48.7%	51.9%
Euclidean	53.4%	49.2%	50.3%
Manhattan	53.2%	48.2%	50.2%

Features
Path similarity in wordnet

- Each word has different senses in wordnet. Banana has two senses:

```
Synset('banana.n.01') any of several tropical and subtropical treelike herbs of the genus Musa having a terminal crown of large
entire leaves and usually bearing hanging clusters of elongated fruits
Synset('banana.n.02') elongated crescent-shaped yellow fruit with soft sweet flesh
```

- Each pair of senses have a path similarity in wordnet
- I tried to use first, maximum, minimum and average path similarities as features. Accuracy is around 50%.



Features
Definitions in wordnet

- There are definitions for some of senses in wordnet.

```
Synset('banana.n.01') any of several tropical and subtropical treelike herbs of the genus Musa having a terminal crown of large
entire leaves and usually bearing hanging clusters of elongated fruits
Synset('banana.n.02') elongated crescent-shaped yellow fruit with soft sweet flesh
```

- Check if the attribute exist in the definitions of first two words.
- E.g. for triple banana, apple, yellow, “yellow” exists in the definitions of banana, but not in the definitions of apple.
- However, only a small subset of attributes of a word will appear in its definitions
- Slightly improve the performance.

Features
PMI

- Using Brown corpus and Gutenberg Corpus in NLTK to calculate 3 PMI scores for each triple.

$$\mathrm{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

- We also want to know given a word, the chances that feature appears in its context.

  e.g. P(red | apple)= P(apple, red)/P(apple)

- Slightly improved the performance

Evaluation

- After plenty of experiments, I chose 9 features:
  cos(word1, word2), cos(word1, word3), cos(word2, word3), cos(word1,word3)/cos(word1,word2), count_in_definitions(word1, word3), count_in_definitions(word2, word3), P(word1 | word2), P(word3 | word1), P(word3 | word2).

	Accuracy	Precision	Recall	F1 score
SVM	56.06%	55.53%	61.80%	58.50%
Decision Tree	54.52%	55.01%	50.73%	52.78%
Logistic	51.10%	51.25%	49.49%	50.35%

Reference

- Cree, George S and McRae, Ken, 2003, Analyzing the factors underlying the structure and computation of the meaning of chipmunk, cherry, chisel, cheese, and cello (and many other such concrete nouns)., American Psychological Association
- Lazaridou, Angeliki and Pham, Nghia The and Baroni, Marco, 2016, The red one!: On learning to refer to things based on their discriminative properties
- McRae, Ken and Cree, George S and Seidenberg, Mark S and McNorgan, Chris, 2005, Semantic feature production norms for a large set of living and nonliving things, Behavior research methods, Springer

CSCE-489-500
Natural Language Processing
Final Project Showcase

Micheal Peterson

Yang Yang

Task: SemEval 2018-Affect in Tweets

- Subtask 1: Detecting Emotion Intensity (regression)
  - Given a tweet and an emotion E (anger, fear, joy, or sadness)
  - Determine the intensity of emotion E represented in the tweets
- Subtask 2: Ordinal classification
  - Similar to subtask 1 but return intensity as ordinal-level of [0,1,2,3]

Training Data

- Training set from organizer of SemEval 2018
  - 7k labeled tweets for each task, including overlaps
  - Around 2k labeled tweets for each emotion
- Training set from crawling tweets
  - 180k unlabeled tweets
  - Later labeled by bootstrap regressor to increase dataset size

Iteration 1: BOW, Ngram, Word2Vec Features

- The model in iteration 1 only contained bag of word, ngram or word embedding features, mutually exclusive.
- Word embedding feature from word2vec implementation in modeling toolkit Gensim
- Using mean of the word embedding vector as feature of word2vec
- Accuracy is low, around 25% (slightly better than random)
- At this point, we were thinking about using three classifier(regressor), each for one feature.

Iteration 2: Using Glove, Only Considering Emotion Lexicons

- Replacing Gensim's Word2Vec with Glove, a pre-trained word embedding model
- Using NRC-Emotion-Lexicon, a dataset of existence of emotions(We used anger, fear, sadness, joy, positive and negative) in around 13000 English words
- Only consider word embeddings of emotional lexicons in our tweets during training
- Using LinearSVC classifier and LinearSVR regressor
- Average Pearson Correlation score of around 0.17 for all emotions

Iteration 3: Feature Union, Emotion Lexicon Features

- Maybe having too little features is why our accuracy is low
- Using ngram, word embedding of all words in the tweet, and feature matrix of emotion lexicon as features
- Increased correlation drastically to around 0.32, with the best correlation of 0.37 for emotion joy

Iteration 4: Spacy Stopwords, Emotional Lexicon for Hashtags and Neural Networks

- Recalled preprocessing is also an important step we are missing
- Used SpacyNLP's solution to remove all stop words from the tweets, before feature extraction
- Included feature matrix of emotion lexicon for hashtags, which has intensity data of the lexicons
- Used neural network as our classifier/regressor instead of LinearSVC/LinearSVR
- Preprocessing boosted the average by around 4~5 percent
- Changing to neural network boosted the average to over 0.45, with the max correlation being over 0.60

Iteration 5: Additional Tweets

- Starting from iteration 3, we also considered having too little a dataset(less than 2k tweets for each emotion) is a flaw of our system
- Created a multi-thread tweet crawler that uses multiple instances of Twitter Streaming API to collect tweets with different keywords
- Collected around 180k tweets with obvious emotion lexicons as keywords
- Using our previous model in iteration 4 to label our new tweets and retrain the model with new tweets and old tweets
- Boosted Correlation only slightly, around 3 percent

Conclusion/Future Works

Conclusion:

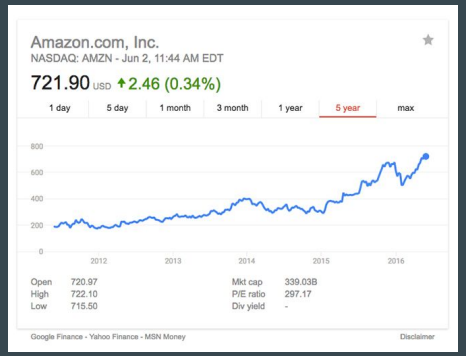
- It is often good to have more features
- Bring more noisy data in may not increase accuracy

Future Work:

- We attempted a mechanism that learns lexicon from the inputted tweets by including words that has close distance to known emotional lexicons in the word embedding vector space
- Could improve accuracy if completed, since our emotional lexicon dataset is pre-trained
- Maybe other hyperparameters for classifiers/regressor would work better

Analyzing Twitter Sentiment of President Trump to Predict the Stock Market.

Young Sur, Vibhor Mishra



Introduction

- President Trump loves to make public announcements through his twitter account
- Whole world is paying attention
- Trump uses his tweets to manipulate or scare companies
- Companies fear his tweets would negatively affect their stock price
- We want to investigate using sentiment classifiers

Components

- Data Collection
  - Twitter API add-on on Google Sheets
  - Archive of Trump's Tweets at www.trumptwitterarchive.com
  - Historical Stock Price archive www.nasdaq.com
- Programming
  - Python
  - Google Sheets Regex

Training Data

- Total of 10,000 random tweets written in English
- 5,000 Positive 5,000 Negative
- Emojis:
    - Positive Tweet:
    - Negative Tweet:
- Cleaned the data by removing
    - Stop words
    - URLs
    - Hashtags
    - Account name
    - Punctuations

Testing Data

- President Trump's tweets from Nov/08/2016 to Nov/08/2017
- 30 tweets containing company names listed on the stock market
- Stock prices of companies mentioned in Trump's tweets
- Considered difference in Opening and Closing price of the day company was mentioned

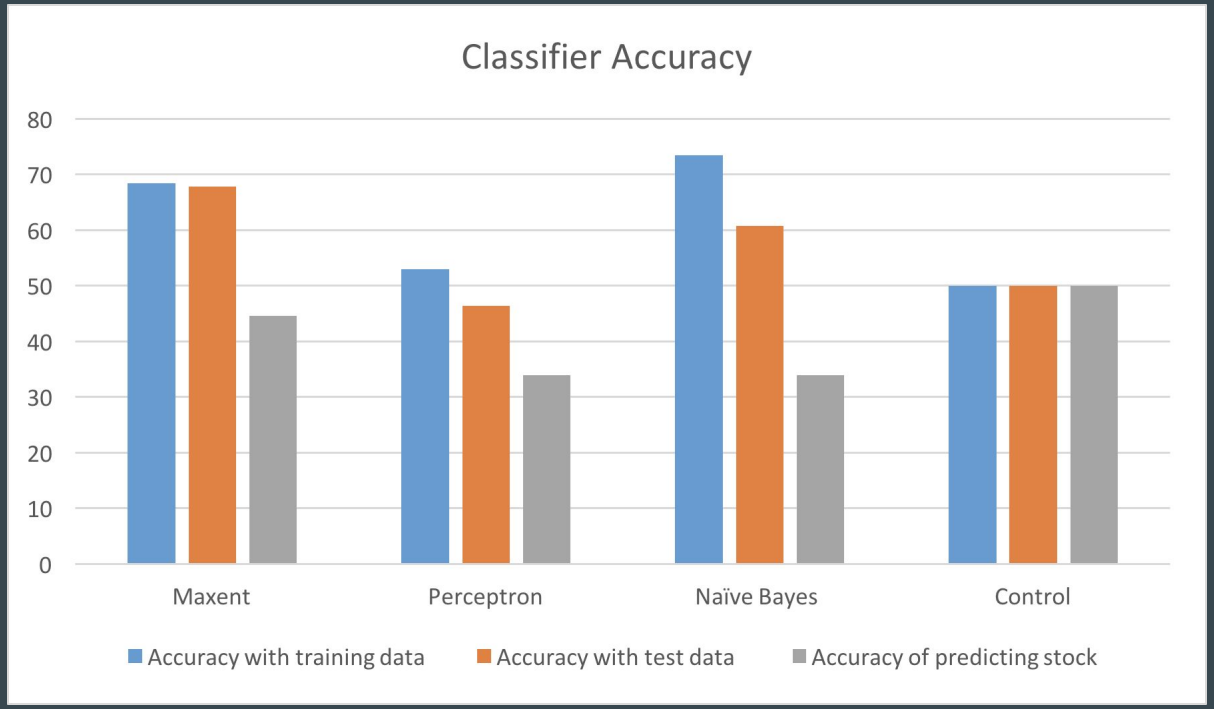
Classifiers

- Maximum Entropy
  - Feature: bag-of-word
  - Binary Classification: positive or negative
- Perceptron
  - Feature: bag-of-word
  - Binary Classification: positive or negative
- Naive-Bayes
  - Feature: bag-of-word
  - Binary Classification: positive or negative

Analysis

Classifier	Naive Bayes	Maxent	Perceptron
Training Data	74%	69%	54%
Test Data	60%	68%	47%
Stock Data	34%	45%	34%

Analysis


Classifier Accuracy
80
70
60
50
40
30
20
10
0
Maxent
Perceptron
Naïve Bayes
Control
Accuracy with training data
Accuracy with test data
Accuracy of predicting stock

Findings

- There seems to be no correlation between Trump's tweets and stock prices
- Good example is company named Carrier
- From Nov/24/2016 to Dec/01/2016, Trump tweeted positively about the Carrier almost every day.

Probable reasons why Maxent outperformed others

- Considers interlinking of features.
- Non Linear.
- It maximizes the log likelihood of training data.
- Best for in-domain testing.

❖ Possible Drawback- It may overfit training data, accuracy in training data way more than that in testing data.

Little more on Analysis -

- Does ignoring punctuation improve our classification?

  Yes , But only slightly

- Does ignoring URLs and usernames improve our classification?

  Again not a drastic improvement

- Does ignoring common words unrelated to sentiment improve our classification?

  Reasonable improvement in performance

Reasons for low accuracy on tweets -

- Length of tweet too small, average tweet length is 28 characters., our data set had 5.4 words per tweet.
- Words are usually misspelled, ex- nice written as niceeeee .
- Many tweets don’t convey sentiment like “just tried chicken tikka today “.
- Informal way of communication, high use of english slangs.
- Sarcasm used more often here than in other social media.

Whats Next?

- Same study with tweets of all politicians or even all twitter users.
- Need Named Entity Recognition (NER) for finding company names in tweets.
- Due to the informal nature of tweets, it is very difficult to perform NER to find company names.
- Would need to build a custom NER for the task.
- Would need to investigate better classifiers or training data to improve accuracy.

Reference

- Robinson, D. 2017. “Text analysis of Trump's tweets confirms he writes only the (angrier) Android half,” Retrieved from: http://varianceexplained.org/r/trump-tweets/
- Chin, D., A. Zappone and J. Zhao (2016) “Analyzing Twitter Sentiment of the 2016 Presidential Candidates.” Available at: https://web.stanford.edu/~jesszhao/files/twitterSentiment.pdf
- Rau, L.F. (1991). Extracting company names from text. 29 - 32. 10.1109/CAIA.1991.120841.