CSC411H1S Project 3

Hao Hui Tan(999741711, tanstev1)
Kyle Zhou (1000959732, zhoukyle)

March 19, 2018

1. The Real headline data set seems to be larger than the Fake headline data set. Most of the headlines for both data sets are in English, but there are some French and Spanish headlines, as well as possibly other languages.

   Fake headlines seem to use "Trump" to refer to Donald Trump, while real headlines tend to use "Donald Trump." Fake headlines also tend to use more sensational or inflammatory terms such as declaring something "an hilarious fail," or have grammatical mistakes like the aforementioned example. Headlines in general are all lowercase with no punctuation. However, it seems that the real headlines tend to be truncated, while the fake headlines seem to all have the full text. Some of the headlines are also misspelled (e.g. "x jinpingi" instead of "xi jinping").

   It is difficult to categorize headlines solely based on keywords, since the same word in different contexts could be either sensational, or factual. Some useful keywords could be "racist" (5 occurrences in fake, 2 occurrences in real), "hillary," (18 occurrences in real, 97 occurrences in fake), and "rigged" (3 occurrences in real, 15 occurrences in fake).

2. The Naive Bayes algorithm was computed by computing the conditional probability

   $$P(x_i|c) = \frac{count(x_i = 1, c)}{count(c)}$$

   for all x_i in the training set, and for each class c (i.e. "real" or "fake"). The actual formula used involves using m and $\hat{p}$ as priors in order to improve the accuracy of our model, since many words only occur once, or a few times. Thus, the formula we trained with was

   $$P(x_i|c) = \frac{count(x_i = 1, c) + m\hat{p}}{count(c) + m}$$

   To predict whether a headline was real or fake, we computed the conditional probability

   $$P(x_1, ..., x_n|c)P(c)$$

   by computing

   $$\prod_{i=1}^{n} p(x_i|y = c)$$

   However, since for the less frequent words, $p(x_i|y = c)$ is very small, and multiplying them together may result in underflow, we computed the exponential of the sum of the log probabilities instead. Thus, our formula becomes

   $$P(x_1, ..., x_n|c)P(c) = \exp\left(\sum_{i=1}^{n} \log(P(x_i|y = c))\right) P(c)$$

   We then return the class with the highest probability as our prediction.

   In order to tune the m and $\hat{p}$ parameters, we trained the model with varying values, with $m \in [1, 20]$ and $p \in [0.1, 1.0]$, with a step of 1 and 0.1, respectively, and found the values that performed best on the validation set.

   The performance of the classifier on the training set is 96%, the performance on the validation set is 85%, and the performance on the test set is 85%

   The best params are as follows:

   $$m = 2, \hat{p} = 0.2$$

   The code is included below:

Listing 1:

```
def train_model(real_headlines, fake_headlines, m, p):
    word_list = get_wordlist(real_headlines, fake_headlines)
    real_counts = count_word_occurrance(real_headlines)
    fake_counts = count_word_occurrance(fake_headlines)
    probabilities_real = {}
    probabilities_fake = {}
    for word in word_list:
        # if word in ENGLISH_STOP_WORDS: continue
        if word in real_counts:
            probabilities_real[word] = (real_counts[word] + m * p) / float
                (len(real_headlines) + m)
        else:
            probabilities_real[word] = (0 + m * p) / float(len(
                real_headlines) + m)
        if word in fake_counts:
            probabilities_fake[word] = (fake_counts[word] + m * p) / float
                (len(fake_headlines) + m)
        else:
            probabilities_fake[word] = (0 + m * p) / float(len(
                fake_headlines) + m)

    return probabilities_real, probabilities_fake, m, p, len(
        real_headlines), len(fake_headlines), word_list

def predict_model(model, headline):
    probabilities_real, probabilities_fake, m, p, real_count, fake_count,
        word_list = model
    logprob_real = 0.0
    logprob_fake = 0.0
    real_class_prob = float(real_count) / (real_count + fake_count)
    fake_class_prob = float(fake_count) / (real_count + fake_count)
    headline_split = headline.split(' ')
    for word in word_list:
        # if word in ENGLISH_STOP_WORDS: continue
        if word in headline_split:
            logprob_real += math.log(probabilities_real[word])
            logprob_fake += math.log(probabilities_fake[word])
        else:
            logprob_real += math.log(1 - probabilities_real[word])
            logprob_fake += math.log(1 - probabilities_fake[word])
    real_prob = math.exp(logprob_real) * real_class_prob
    fake_prob = math.exp(logprob_fake) * fake_class_prob
    # print real_prob, fake_prob
    return real_prob > fake_prob

def tune_model(real_training, fake_training, real_validation,
    fake_validation):
    performance_report = {}
    m = 1
    while m <= 20:
        p = 0.1
        while p <= 1:
            model = train_model(real_training, fake_training, m, p)
            performance = get_performance(model, real_validation,
                fake_validation)
            print m, p, performance
            performance_report[(m, p)] = performance
            p += 0.1
        m += 1

    print "The m and p value is", max(performance_report, key=
        performance_report.get)

    return performance_report

def get_performance(model, real, fake):
    correct = 0

    for hl in real:
        if predict_model(model, hl):
            correct += 1

    for hl in fake:
        if not predict_model(model, hl):
            correct += 1

    return float(correct) / (len(real) + len(fake))
```

3. (a) The 10 words whose presence most strongly predicts that the news is real

   i. japan
   ii. daniel
   iii. refugee
   iv. denies
   v. zoe
   vi. charlottesville
   vii. korea
   viii. business
   ix. nfl
   x. south

   The 10 words whose absence most strongly predicts that the news is real

   i. trump
   ii. the
   iii. to
   iv. hillary
   v. a
   vi. is
   vii. and
   viii. for
   ix. clinton
   x. in

   The 10 words whose presence most strongly predicts that the news is fake

   i. hats
   ii. debunks

iii. sleep
iv. hating
v. battleground
vi. haters
vii. pointing
viii. dazu
ix. pantano
x. amtsantritt

The 10 words whose absence most strongly predicts that the news is fake

i. donald
ii. trumps
iii. us
iv. says
v. ban
vi. korea
vii. north
viii. turnbull
ix. travel
x. australia

These lists were obtained by finding the words with the max values for the conditional probabilities

$$P(\text{class}|\text{word}) = \frac{P(\text{word}|\text{class})P(\text{class})}{P(\text{word})}$$

where

$$P(\text{word}) = \frac{\text{count}(\text{word}) + m\hat{p}}{\text{count}(\text{headlines}) + m}$$

and

$$\begin{aligned} P(\text{class}|\neg\text{word}) &= \frac{P(\neg\text{word}|\text{class})P(\text{class})}{P(\neg\text{word})} \\ &= \frac{(1 - P(\text{word}|\text{class}))P(\text{class})}{1 - P(\text{word})} \end{aligned}$$

The words that have the biggest sway on whether a headline is real or fake tend to be words that only appear in one set and not the other. However, the presence of a word in one class's top 10 doesn't imply that it will be as strong when absent from the other class.

For the code, please see `get_top_bottom_word_occurrences` in fake.py.

(b) The following lists have had the stopwords from scikit_learn excluded

The 10 words whose presence most strongly predicts that the news is real

i. japan
ii. daniel
iii. refugee
iv. denies
v. zoe
vi. charlottesville
vii. korea
viii. business
ix. nfl
x. south

The 10 words whose absence most strongly predicts that the news is real

i. trump
ii. hillary
iii. clinton
iv. just
v. america
vi. supporters
vii. vote
viii. black
ix. win
x. watch

The 10 words whose presence most strongly predicts that the news is fake

i. hats
ii. debunks
iii. sleep
iv. hating
v. battleground
vi. haters
vii. pointing
viii. dazu
ix. pantano
x. amtsantritt

The 10 words whose absence most strongly predicts that the news is fake

i. donald
ii. trumps
iii. says
iv. ban
v. korea
vi. north
vii. turnbull
viii. travel
ix. australia
x. climate

(c) It might make sense to remove stopwords when looking at the ratio of word frequency between the real and fake headlines, because these words are common, and would likely appear in most if not all headlines. These words are also uninteresting on their own, and so they crowd up the word frequency rankings, possibly pushing more interesting words down the list. Another reason for removing stopwords would be when the input set includes headlines of different languages. This is because stopwords in one language are usually uncommon in other languages, and once again skew the results.

It would make sense to keep stopwords because their presence could still help distinguish between classes, because one class might use certain stopwords in a different way, or use them more (or less) frequently than the other class. Furthermore, in testing, we have found that training while excluding stopwords does not necessarily increase the performance of our model.

4. The learning curves of Logistic Regression can be seen in Figure 3.

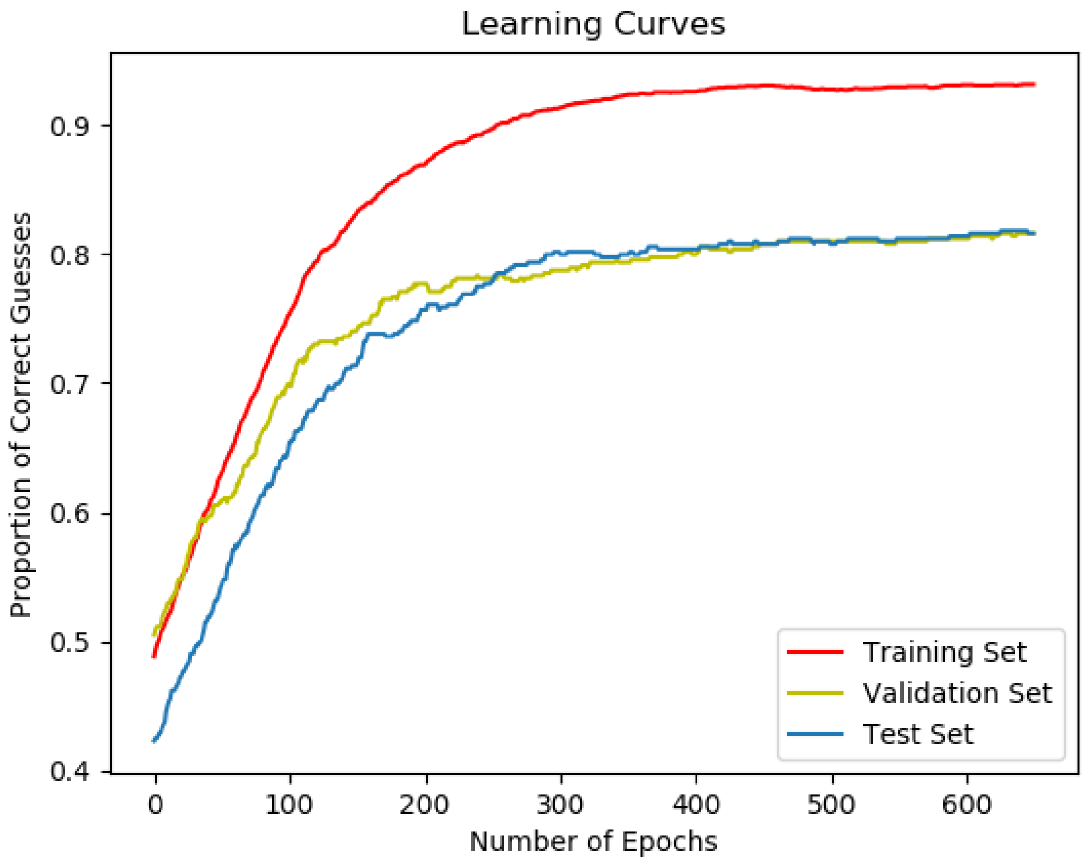

Figure 1: Performance of Logistic Regression vs. Number of Epochs

We used L2 regularization to prevent overfitting. We chose the λ value by training the model on the training set with varying λ values, and choosing the value that had the highest performance on the validation set. The value that seemed to perform the best was 0.007.

When searching for the λ value, we started off with small values and a small step size. This is because our performance on the validation set without L2 regularization was already quite good, and an excessively large lambda value would penalize the data too much.

5.

6. (a) The top positive words in the logistic regression model are as follows:
   i. nfl
   ii. australia
   iii. ban
   iv. defends
   v. over
   vi. week
   vii. appears
   viii. law
   ix. back
   x. pence

This list contains a combination of the 10 words whose presence most strongly predicts that the news is real, and the 10 words whose absence most strongly predicts that the news is fake. For example, "nfl" can be found in the former list from 3(a), and "australia," and "ban" can be found in the latter list.

The top negative words in the logistic regression model are as follows:

i. this
ii. reason
iii. paid
iv. hillary
v. have
vi. it
vii. 5
viii. votes
ix. are
x. need

This list should contain a combination of the 10 words whose presence most strongly predicts that the news is fake, and the 10 words whose absence most strongly predicts that the news is real. For example, "hillary" can be found in the former list from 3(a). However, no words can be found in the latter list. This is likely due to stop words pushing the other words down the list.

Notably, the stopwords contained in these lists are not the same.

(b) The top positive words in the logistic regression model excluding stopwords are as follows:

i. nfl
ii. australia
iii. ban
iv. defends
v. week
vi. appears
vii. law
viii. pence
ix. senate
x. push

Again, this is a combination of the top 10 words whose presence most strongly predicts that the news is real, and the top 10 words whose absence most strongly predicts that the news is fake. The similarities are the same as in 6(a).

The top negative words in the logistic regression model excluding stopwords are as follows:

i. reason
ii. paid
iii. hillary
iv. 5
v. votes
vi. need
vii. goes
viii. f
ix. won
x. riots

Again, this is a combination of the top 10 words whose presence most strongly predicts that the news is fake, and the top 10 words whose absence most strongly predicts that the news is real. The similarities are the same as in 6(a). There seem to be some additional synonyms that correspond (e.g. "votes" vs. "vote"), but not additional exact matches.

(c) It might not be the best idea to read the logistic regression parameters directly because logistic regression does not require normalized input. If there was a value that was abnormally high or low in the training set, the weight corresponding to that value would be small to compensate. That value could be quite significant, but since the data is not normalized, those weights would be lower than one would expect. This is not a concern in our case since our data sets are all normalized (the only values the features could take are 0 and 1).

$$\log \frac{p(\text{real}|x,\theta)}{p(\text{fake}|x,\theta)} = \theta_0 + \theta_1 x_1 + ... + \theta_d x_d$$

7. (a) The following graph shows the relationship between **max_depth** values and the accuracy of the training and validation sets.

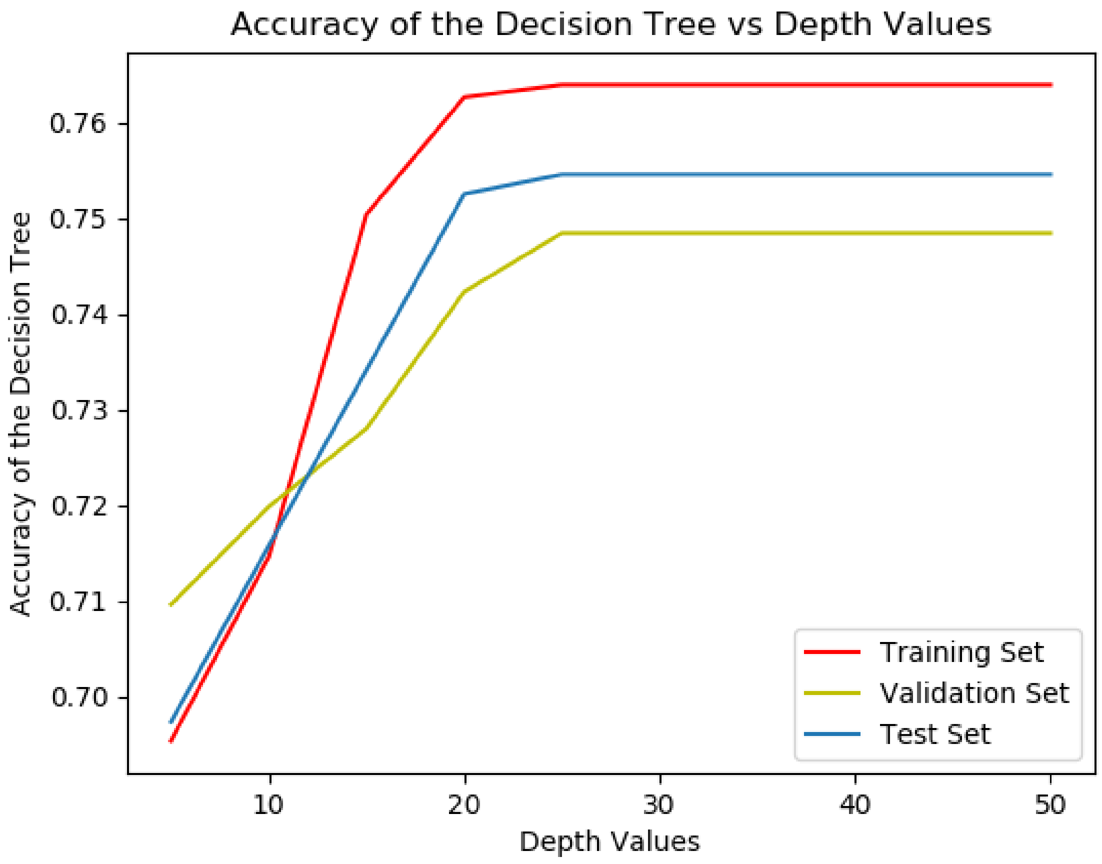

Figure 2: Performance of Training and Validations sets and Max_depth values

As one can see from the graph, the best max_depth values to use is 25 as the Decision Tree classifier reaches the best performance at that value for the validation set. In addition to max depth values, I have also tried out different criterion values. I have tested out Gini and Entropy criterion and found that the Entropy criterion has better performance than the former. I have also tried out different min_sample_leaf values and settled on 15 because it gives better values than the rest. If the minimum sample leaf value is too small, it will likely to cause over-fitting as each branch may not have enough data for the algorithm to generalize well. If the value is too large, then it will prevent the model from learning at all.

(b) Here is the visualization of the first two layers of the decision tree.

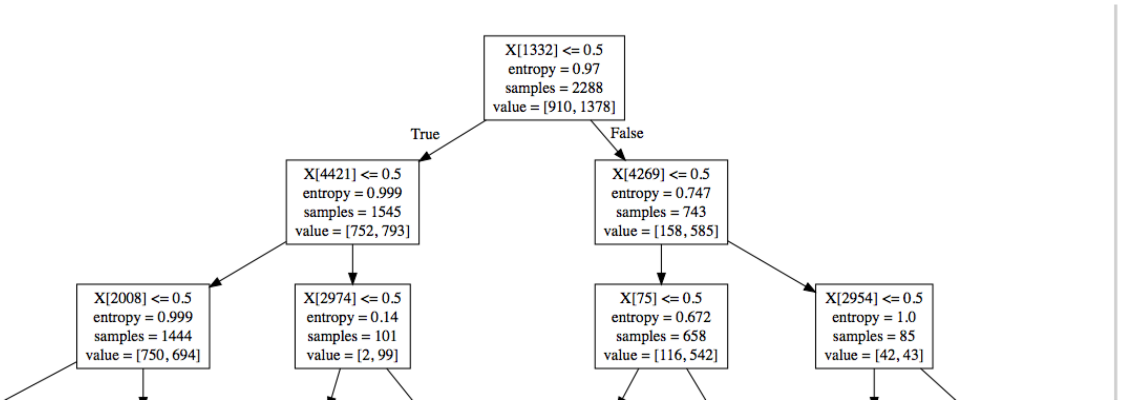

Figure 3: Visualization of the first 2 layers of the Decision Tree

The most important keyword features in the Naive Bayes from 3a are
Positive indication that headline is real:

i. hats
ii. debunks
iii. sleep
iv. hating
v. battleground
vi. haters
vii. pointing
viii. dazu
ix. pantano
x. amtsantritt

Negative indicating that headline is real:

i. donald
ii. trumps
iii. us
iv. says
v. ban
vi. korea
vii. north
viii. turnbull
ix. travel
x. australia

The most important keyword features for the logistic regression model are:

Positive indication that it's real:

i. nfl
ii. australia
iii. ban
iv. defends

v. over
vi. week
vii. appears
viii. law
ix. back
x. pence

Negative indication that it's real:

i. this
ii. reason
iii. paid
iv. hillary
v. have
vi. it
vii. 5
viii. votes
ix. are
x. need

The most important keyword features for the decision tree are 'donald', 'trumps', 'the', 'hillary', 'on', 'a', 'of'. The keywords donald and trumps are in the set of important features in naive bayes. The keyword hillary is in the set of important features in logistic regression. It is interesting that all these words are negative indications that the headline is real, i.e they are fake news predictors.

(c) The performance of the Naive Bayes is 96% on the training set, 85% on the validation set, 85% on the test set.

The performance of the Logistic Regression is 94% on the training set, 80% on the validation set 80% on the test set.

The performance of the Decision Tree is 76.2% on the training set, 74.8% on the validation set 75.2% on the test set.

Naive Bayes performed the best according to the data. However, it is a close between Naive Bayes and Logistic Regression. The Decision Tree algorithm performed the worst. Surprising, logistic regression over-fitted the most. In theory, it should be decision that is the most likely to over-fit and Naive Bayes that is the least likely to over-fit.

8. (a) $I(Y, x_1) = H(Y) - \sum_k P(X = x_k)H(Y|X = x_k) = 0.97 - ((.67)(0.999) + (0.33)(0.747)) = 0.054$ Thus the mutual information of the first split is 0.054.

(b) $I(Y, x_2) = H(Y) - \sum_k P(X = x_k)H(Y|X = x_k) = 0.999 - ((.93)(0.999) + (0.07)(0.14)) = 0.06$ I chose the first child of the first split as the variable for this question. The mutual information on this variable is slightly larger than the first split. This indicates that the this variable tells us more information about whether the headline is real or fake than the first headline.