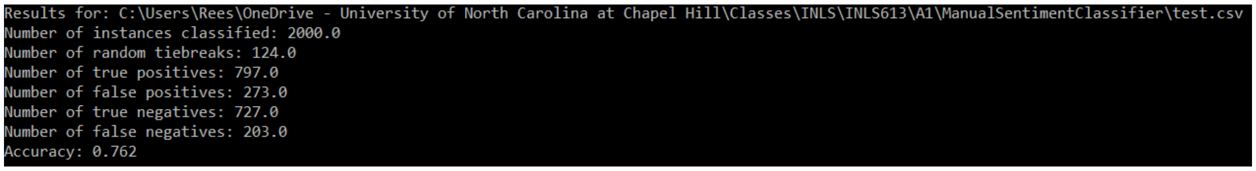
Rees Braam

INLS 613

1/29/2019

# Assignment 1

*Peak classifier accuracy: 0.762*



```
Results for: C:\Users\Rees\OneDrive - University of North Carolina at Chapel Hill\Classes\INLS\INLS613\A1\ManualSentimentClassifier\test.csv
Number of instances classified: 2000.0
Number of random tiebreaks: 124.0
Number of true positives: 797.0
Number of false positives: 273.0
Number of true negatives: 727.0
Number of false negatives: 203.0
Accuracy: 0.762
```

```
cd C:\Users\Rees\OneDrive - University of North Carolina at Chapel Hill\Classes\INLS\INLS613\A1\ManualSentimentClassifier
```

```
java -jar ManualSentimentClassifier.jar train.csv test.csv positive.txt negative.txt
```

**Q1**

I used three different approaches when trying to improve the classifier: through intuition, the internet, and programming. At first, I just tried to add words by intuition to see if there would be some easy, obvious words. I did this by looking at misclassified reviews from train.pred and picking words that seemed like they should be on the lists to add. Doing this had a pretty small effect (only a few percent) and felt like very slow going. After a couple hours, I decided I would take the suggestion to look online. Looking online, I initially found what I thought was a decent wordlist.[1] However, it required some manipulation to get it into a the same format as positive.txt and negative.txt. I reformatted this list and separated the words into the appropriate lists and found that the accuracy boosted to around 68% here. I had hoped to get the accuracy higher than that, so decided to delve into train.pred.csv again to check where predictions were going wrong. I wrote a small python script to quickly count the number of positive and negative words in each review so that I could easily see which words could be causing poor results, or if words were not being included that should have been. Spending some time on this, I was able to bring the accuracy up by a few percentage points, but it was difficult to tell what was truly being caused by my changes and what was just random chance. In addition, it felt like my changes were having almost no impact since I was only able to look at one review at a time, and the words I added likely only occurred no more than 3-5 times in the whole dataset. After failing to increase the success rate as much as I'd hoped (i.e I didn't get lucky), combined with the number of missing/wrong words in the list I'd downloaded, I eventually decided to look for another list. This list, created by a CS professor at UIC, instantly shot my prediction accuracy up to 76% with no changes required. [2] At this point I felt that the accuracy was high enough and had learned that any changes I made would have a minimal effect, and so I left it as it was.

**Q2**

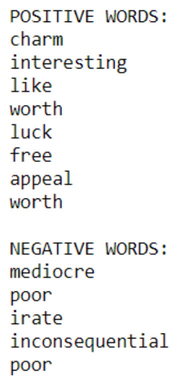
With an even distribution of positive and negative reviews, it would be expected that merely guessing should result in a 50% accuracy rate. This is because for each review, no matter what you guess, you always have a 50% chance of being correct because there are only 2 possible outcomes. As such, any accuracy lower than 50% should be an immediate red flag that essentially says you've trained your model in the opposite direction, or at least that something somewhere is going wrong.

If I knew that test.csv were split 80-20 between positive and negative reviews, this would not affect my strategy for improving the classifier whatsoever. When creating a model, you should never look at the test data, and even if you did, you should not modify your classifier based on it. If you did, then your test data would really just become alternative training data, and your classifier would not perform as well on data it had never seen before.

**Q3**

*"Since its release in 1983 A Christmas Story the Jean Shepherd-narrated story of his alter-ego Ralphie has become a true classic. My Summer Story however still has Shepherd as the narrator but it has absolutely none of the charm and the characters are nowhere near the caliber of the original film. My Summer Story is basically a mishmash of mediocre and just plain not very interesting stories which include hillbilly neighbors and battling tops. Charles Grodin who I normally like is extremely unlikeable in the role of the father (more aptly handled by Darren McGavin in the original) and his character never seems anything but forced. Kiernan Culkin is a poor substitute for Ralphie and the little brother is all but forgotten here. Only mom seems to have any worth here and perhaps that's because she beans a cinema manager with a gravy boat when he pushes his luck too far with irate housewives on free dish night . The stories in this are mostly inconsequential and stretched paper-thin. May appeal to the extremely undemanding but as a sequel to A Christmas Story it's a very poor one and not worth most people's time. 2 out of 10."*
– row 1405

I find this example interesting because this review is extremely negative, and a human would probably say that this review could probably not be much more negative than it already is. Despite this, the classifier found 3 more positive words than negative, and predicted positive. One would think that there should be no way this review could be predicted as positive considering that the reviewer literally spells it out at then end, saying "2 out of 10." However, the classifier does not know that this is by far the most important, or at least the most telling part of the review. It misses the fact that almost all the positive words it finds are being negated, and so misses the fact that this review is pretty much nothing but negative sentiment. To have caught this, the classifier would have had to look at the other words in the sentence, or at least nearby to find negation ("*none* of the charm", "*not* very interesting", "*not* worth most people's time").

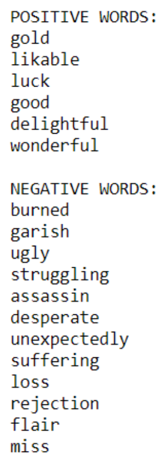

```
POSITIVE WORDS:
charm
interesting
like
worth
luck
free
appeal
worth

NEGATIVE WORDS:
mediocre
poor
irate
inconsequential
poor
```

**Q4**

*"The Matador stars Pierce Brosnan as a burned out assassin. He's James Bond gone to seed in too-tight garish clothes gold chains and an ugly haircut. Our struggling assassin Julian Noble is in Mexico trying to regain his nerve. Staying at the same hotel is a likable down-on-his luck businessman Danny Wright (Greg Kinnear) also trying to regain his equilibrium. Danny is desperate to close a deal and return to his wife in Denver (Hope Davis) with good news. Noble and Wright unexpectedly become friends. Wright*

*convinces Noble to reveal certain techniques which he demonstrates at a bullfight. Noble is eventually targeted by his employers and shows up in Denver. Writer and director Richard Shepard did the Q&A after this delightful movie at the Austin Film Festival. Shepard was also down on his luck. After suffering the loss of his agent and rejection of recent scripts he decided to write a story no one would buy and create a character no one would want to play. Then Pierce Brosnan called. Brosnan regains his equilibrium in this movie. (There is life after Bond!) He has a wonderful flair for self-deprecating comedy. Don't miss it. Stay for the closing credits to read what the filmmakers say about bullfighting. I look forward to more of Richard Shepard's projects."*
- row 5



```
POSITIVE WORDS:
gold
likable
luck
good
delightful
wonderful

NEGATIVE WORDS:
burned
garish
ugly
struggling
assassin
desperate
unexpectedly
suffering
loss
rejection
flair
miss
```

I find this example of a false-negative to be interesting because the classifier failed to recognize that the negative words in this review were being used merely to describe something in the movie as a summery, and were not aimed at the movie itself. In addition, like the last question, the classifier again failed to identify descriptive words that changed the meaning of or negated a word ("*wonderful* flair", "*Don't* miss it"). In order to avoid the major mistakes here, the classifier would have had to identify the subject of the words it was marking as negative and decide if that made them pertinent to the classification or not.

**Q5**

I certainly did find myself wishing that the program did something different. The algorithm used in the program made an assumption that is absolutely incorrect and was the hallmark of most of the failures found in the resulting classification: it assumed that each word only has one sentiment, positive or negative. This is obviously not true, as most of the words it was looking for could easily be negated, resulting in an incorrect classification. For example, the words "good" and "bad" are two of the easiest words to separate, "good" is positive and "bad" is negative. However, if you add the word "not" in front of these words, suddenly they become swapped. In this fashion, many of the incorrect classifications were for reviews that used sentence structures like this. As I discussed in the previous two questions, I think that the program would need to examine other words in the sentence to deem when a word is positive or negative, not rely solely on a set lookup. Essentially, I am saying that the classifier needs to consider the context of a word, and not just the word itself. Even just a rudimentary consideration for the context of a word would probably increase the classification rate significantly (i.e reverse the sentiment of the word if the sentence contains words like "not", "none", etc).

**Q6**

I have little doubt that a classifier trained on movie reviews would do worse on a test set of reviews from another domain like hotels. I would even go so far as to say this is why the prediction accuracy I reached in this assignment was not higher: I am using a list of words created with another purpose from another training set. As such, my classifier was only able to predict so accurately. However, had I used ML techniques myself to create a comprehensive list from the given training data, I wouldn't be surprised to learn that the test accuracy could have gone significantly higher – over 80% at least, maybe even to 85% or 90% if the list I used was really that different. In terms of specific differences between the hotel review and movie review domain, there are some easily identifiable words that would not translate well between them. For example, a “hot” movie means it is good, but if your hotel room was too hot, the opposite is true. Similarly, saying that a movie was “sick” or “wild” means it was cool, but saying the hotel got you sick would be bad. A list trained on movie reviews would also likely miss many words that could identify a hotel review as positive or negative. For example, you wouldn’t hear a lot about noise complaints in a movie review (aside from music), but you would in a hotel review. Therefore, words like “noisy”, “loud”, “quiet”, etc would be missing.

[1] http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
[2] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon