Final Project

Luca Buchoux

2024-12-12

Introduction

One of the most talked-about terms in recent years is "fake news." This refers to the spread of misinformation, either by individuals who unknowingly share false information or by those who are aware of its falsity but have ulterior motives. In both cases, uncovering the truth amidst all the noise can be incredibly difficult, if not impossible. Moreover, in today's world, an increasing number of people turn to the internet for their news, where it's all too easy to publish anything and present it as fact. Even those who still rely on traditional news sources, like television, aren't immune, as these outlets may be repeating false or misleading information from their own sources. This creates a significant problem, as people can be easily influenced by what they hear or read, especially when it appears to come from a trusted source. Misinformation often surrounds controversial issues and is strategically crafted to deceive. As more false information circulates, distinguishing the truth becomes even more challenging. Fortunately, advancements in machine learning have led to the development of models that can help assess the credibility of information. Additionally, as this issue has gained prominence, a wealth of data has been collected to train these models, enhancing their accuracy. The paper I have selected for analysis, titled *Fake News Detection Using Machine Learning Algorithms* by Sharma et al., addresses this issue. The authors explore relevant works, apply four different classifiers to significant datasets, and present their findings. In the end, the authors came up with a Naive-Bayes model, a Random Forest model, and a Logistic Regression model. Their Logistic Regression model had the best performance. In this report, using the publicly available LIAR dataset, I will attempt to recreate the results from the paper, critique their methodology, and finally consider the danger that such models can pose despite any good intentions.

Analysis of Methods

The first challenge was to preprocess the data in the same way the researchers did. The LIAR data consists of pre-split datasets with roughly 10,000 observations in the training data and about 1,200 in the testing data. Each dataset has 14 variables, including an ID, a label, the statement, and other things that give context to the statements such as speaker, subject, political affiliation, state, and speaker job title. However, in the paper, the authors specifically state that they only used the statement itself and the label as the classifier. The authors also state that the label variable contains 2 labels: True and False. However, this is not true as the raw LIAR dataset has 6 levels: 'pants-fire', 'FALSE', 'barely-true', 'half-true', 'mostly-true', and 'TRUE'. This is one of my first sources of confusion for my recreation that stems from the vagueness of the paper. The authors do not specify if they exclude every statement that is not TRUE or FALSE, or if they lump the other labels into the TRUE or FALSE category. My initial idea was the second option, but then comes the question of when a statement becomes true. For example, is a 'half-true' statement true or false? Thus I opted to focus solely on the statements explicitly labeled TRUE or FALSE. This also has the added benefit of reducing the training data to about 3,600 observations and the testing data to about 500 observations, which as you will soon see, greatly reduces the amount of variables and thus computing time. Below is an example of a FALSE statement from the raw data, just to visualize where I started from. But again, the important variables are the label and statement variables.

```
## [1] "ID"        "2635.json"
## [1] "Label" "FALSE"
## [1] "Statement"
## [2] "Says the Annies List political group supports third-trimester abortions on demand."
## [1] "Subjects" "abortion"
## [1] "Speaker"      "dwayne-bohac"
## [1] "Speaker.Job.Title"    "State representative"
## [1] "State" "Texas"
## [1] "Affiliation" "republican"
## [1] "barely.true.counts" "0"
## [1] "false.counts" "1"
## [1] "half.true.counts" "0"
## [1] "mostly.true.counts" "0"
## [1] "pants.on.fire.counts" "0"
## [1] "Context"  "a mailer"
```

The next step was to vectorize this data. The authors outline the 3 methods they used to vectorize the data: bag-of-words, ngrams, and TF-IDF (Term Frequency - Inverse Document Frequency). Bag-of-words is a method in which each word in a statement becomes its own variable, then is given a 1 or 0 if it appears in its corresponding statement. Ngrams is a method where n adjacent words are grouped together as variables, giving more context to words. Finally, TF-IDF is a score given to words based on their frequency as it relates to how common a word is. Essentially, it boosts the importance of rarer words and diminished more common words. I chose to focus on bag-of-words for simplicity, but I will address the other two methods. Now, it may become clear to see why reducing the number of observations was a benefit to me. These vectorization methods significantly increase the number of variables in the data. For example, for my training data with 10,000 initial observations, each unique word in all those observations would become its own variable. This is an insane amount of variables. It also doesn't even take into account that while I train my models only on the training observations, I need to include all the variables from the testing data so that I can actually obtain testing results. Anyways, I found a useful package called 'tm' that I used to bag-of-words my data. First, I converted the false labels to 0 and the trues to 1. Then, I converted all the statements to lower case, removed punctuation and numbers, removed stop words, and stripped the whitespace. Stop words are insignificant words such as "a" or "the" that don't contribute anything to the statements. This is the main reason I decided not to pursue the TF-IDF method, since insignificant words were already removed and I wanted to reduce the complexity of the data as much as possible. Then, with the new transformed data, I was able to separate each unique word out and detect that word in each statement, completing the bag-of-words method. Now, all I have left is the binary true/false label, and 7405 words acting as the variables with a total of 4120 observations. Below is the first ten observations with the first ten variables.

```
##    LabelsTF abortions annies demand group list political supports
## 1         0         1      1      1     1    1         1        1
## 2         0         0      0      0     0    0         0        0
## 3         1         0      0      0     0    0         0        0
## 4         0         0      0      0     0    0         0        0
## 5         1         0      0      0     0    0         0        0
## 6         0         0      0      0     0    0         0        0
## 7         0         0      0      0     0    0         0        0
## 8         0         0      0      0     0    0         0        0
## 9         1         0      0      0     0    0         0        0
## 10        0         0      0      0     0    0         0        0
##    thirdtrimester care
## 1               1    0
## 2               0    1
## 3               0    0
## 4               0    0
## 5                    0     0
## 6                    0     0
## 7                    0     0
## 8                    0     0
## 9                    0     0
## 10                   0     0
```

Here, I noticed a few glaring issues that are hard to assess on a grand scale due to the sheer size of the data. First of all, you may notice that one of the variables is named 'thirdtrimester'. These are two words that have been mashed together due to the fact that they were previously connected with a hyphen but were morphed into one word during transformation. This made me realize that there could be a lot of typos that create basically useless variables but are practically impossible to deal with. Another issue I began to suspect is that words might be too unique. You can see where the first statement ends at the variable 'thirdtrimester' and the next one starts at 'care'. You can also see that there are only ones showing up for the first statement and everything else is 0. Obviously, this is a tiny sample of the data so I can't say this is an issue, but it made me worry that each statement only has ones for its specific words which would not be very informative. I decided to investigate this further by looking at words that show up in multiple observations, as well as the amount of completely unqique words.

```
##     percent       state       obama         tax       years      health  president
##         453         351         260         258         257         255        248
##      states      people        year        care     million        jobs government
##         214         206         205         200         173         170        151
##     federal       texas        bill      barack     billion      budget
##         150         149         146         143         142         141
```

```
## [1] 3826
```

```
## [1] 0.5166779
```

Above, you can see the top 20 most common words in the data. For example, the most common is 'percent' which appears in 453 different statements. The next value is the number of unique words, i.e. the number of words that appear only in one statement, which is 3826. There are 7405 words in this data, making over 50% of the words completely unique. More specifically, 51.67% of the words are unique. To me, this seems significant since it essentially means over half of our variables only give us information about only one particular statement. I considered simply removing these unique words, but then surely we would be left with a considerable number of statements which had been made up of only unique words that would be left with no information about them, making it impossible to predict for them. I believe some of these unique words are due to some special cases, like typos as discussed earlier, but due to the number of variables present here it's not very feasible to go through and eliminate these. Even if this process was done, I still believe we'd be left with a significant amount of unique words. The authors of the paper make no mention of any of these concerns, so I really had no recourse to try and deal with them. They simply lay out the idea of the bag-of-words method with no details on how they implement it and deal with issues. Therefore, I decided to just move forward with the data as is and see what happens. Next, I wanted to try implementing the ngrams method, but this proved to vastly increase the complexity of the data. With ngrams, you can choose your n to include that many words in the groups. The authors again do not specify what n they choose or how they implement it. They mention n=2 and n=3 when explaining the concept, but they do not say if they chose one or both of these. I decided to try both, but you can see below that 2-grams would add around 30,000 variables and 3-grams would add almost 35,000. I also included some examples of the 2-gram just to visualize what is happening here. Remember, at this point we are sitting at 7405 variables, so adding these would literally increase the number of variables ten-fold. Thus, I decided to move ahead with only the bag-of-words data and see if we could get some results, even if is less accurate.

```
## An ngram object with 29518 2-grams
```

```
## An ngram object with 34686 3-grams
```

```
##  [1] "lower score"             "countries stay"
##  [3] "green jobs"              "rooms gov"
##  [5] "smallpox america"        "waterboarding philippine"
##  [7] "reduced average"         "arizona florida"
##  [9] "supporting work"         "prolife beliefs"
## [11] "rated mccain"            "adam putnam"
## [13] "laws americas"           "julie lassa"
## [15] "land eminent"            "marriage trap"
## [17] "virginias righttowork"   "committing suicide"
## [19] "mack wrote"              "gas phones"
```

Now, as I mentioned before, the authors four classifiers were Naive-Bayes, Random Forest, and Logistic Regression. Since I don't have as much experience implementing random forest models, and we studied Support Vector Machines in this course, I decided to try an SVM instead. Below are the confusion matrices for my SVM and naive-bayes predictors respectively. You can clearly see that none of my results are informative at all. The default SVM model predicted every observation in the testing data as FALSE, and the naive-bayes predictor predicted every observation as TRUE. I have no explanation for why this happens. The training data is balanced, with both labels having between 2,000 and 2,500 observations. Even if it was unbalanced, I do not know why one model would classify as all FALSE while the other classifies as all TRUE. There are no logistic regression results because I was unable to actually run the function. When I did, it would simply load forever and never actually create the model. I am not sure if the data is just too large or if there is an issue with the all binary data, but I see no reason why it should not work. This leads me into some critiques of the paper itself.

```
##                  true
## binary_preds_svm   0   1
##                0 245 204
```

```
##         true
## preds_nb   0   1
##        0   0   0
##        1 245 204
```

Overall, the paper is extremely vague in its explanation of the implementation of the methods. The authors give basic descriptions of each of their methods, but do not elaborate at all on how they tailor it to the specific data. I though that since the LIAR dataset was publicly available, some sort of comparable results would be possible to achieve, but clearly something went wrong with my analysis. I wish that there had been more detail in the paper that could have guided my recreation of the methods, because at this point I really have no way of knowing if I vectorized the data incorrectly, oversimplified the data, or went wrong training my models somewhere. I am also sure that the authors of the paper have access to a lot more processing power than I do, and maybe I underestimated the complexity of the data itself. However, I had hoped by reducing the observations in the data and simplifying to just the bag-of-words method would get me some kind of results, but clearly this is not the case. Again, it really comes down to the lack of detail from the authors that hindered my recreation, which is my biggest gripe with the paper itself.

Analysis of Normative Considertation

The main normative consideration I'd like to address is the danger of labeling the truth. First of all, the truth is not something that is cut and dry. The LIAR dataset even takes this into account. In its raw form it has a range of truth values. In this analysis, the authors and I simply focus on completely true and completely false statements. However, by simple removing all the in-between labels, well over half of the actual statements are not considered at all. Thus, truth-telling models seem to be very circumstantial. In the rare case that a statement is a polar truth or lie, it may be useful. However, in the majority case where a statement falls in between, labeling such a statement can be very dangerous. If something is mostly true but gets labeled as false for example, a lot of truth was just lost. Similarly, a mostly false statement being labeled as the truth could just further perpetuate the spread of misinformation. Models like these vastly oversimplify the idea of truthfulness.

Another glaring danger I see with models like this is how false positives and false negatives could really be harnessed with models such as this. Even the best model the authors of the paper chose, which was logistic regression, was claimed to be about 80% accurate after optimization. So let's say that 20% of the time, this model is wrong and assigns a true statement to be false or a false statement to be true. This 20% is very powerful. Of course, every classification model has some room for error and you may wonder why a model like this shouldn't get the same leeway. It's because every observation in data like this is very case-by-case and hard to compare. In other classification problems, you may be able to actually look at outliers for example and explain why they may have been misclassified. With true or false statements, hot topic items may be more likely to be misclassified, but if they are, it would be hard to argue why. It would be much easier to just accept the model's ruling, especially if a false positive or negative is the agenda you want to push. For example, if a news outlet existed solely to spread misinformation and used the model from this paper to guide what information it put out, it would be so easy to just pick and choose which classifications it wants to misinform. Essentially, overreliance on such a model may produce the truth 80% of the time, but would misinform the other 20%. Thus a model like this does not even solve the problem it set out to solve if used incorrectly. Even if trained with the intention of spreading the truth, it is inevitable that the model itself will create some misinformation, and those who want to propagate that misinformation can harness these results and justify it with the fact that an algorithm produced it.

Conclusion

While the term "fake news" may have become more prevalent in recent years, misinformation has always been a part of our society and it certainly will continue to be an issue in the future. Advancements in technology make it easier and easier to create deliberately misleading information, and with how connected our world has become, it is easier than ever to spread it to more and more people. But along with such advancements also comes ways to combat the spread of misinformation. As we have discussed, researcher such as Sharma et al. have created algorithms that are able to correctly classify news as true or false with 80% accuracy. While this is certainly a valuable tool, the drawbacks and limitations of such a model must be considered. The main drawback being that for all the good a model such as this can do, it also has the potential to cause harm and defeat its own purpose. The false positives and negatives produced can be propped up as a correct results in the same way the accurate predictions can, which can further perpetuate the spreading of misinformation. Going forward, it would be interesting to see a model incorporate the varying levels of truthfulness from the original LIAR dataset. This could lead to a broader approach as to how we classify statements, instead of the very cut and dry method of completely true or completely false. This could also remedy some of the dangers outlined by being more descriptive when labeling information, making it more challenging to generalize and lump whatever controversial topics into the category someone may want. Overall, fake news detection is an important issue that deserves further research, but the inherent dangers must also be addressed.