Police Union Contract Misconduct Complaint Detection

Zian Wang†
University of Pittsburgh
Pittsburgh, PA 15260
ziw42@pitt.edu

Sonal Gupta
University of Pittsburgh
Pittsburgh, PA 15260
sog26@pitt.edu

Shuo Zheng
University of Pittsburgh
Pittsburgh, PA 15260
shz113@pitt.edu

ABSTRACT

Tons of contracts and sessions are dispersed all over different websites and are very hard to navigate. Many citizens are left in the dark about how to approach police reforms. Finding the problematic and confusing sentences across all the contracts is not easy, we therefore, aim to provide a user-friendly way that helps to predict the problematic sentences, based on data mining tools and model training techniques. We applied supervised learning, classified sentences, trained different models, evaluated our results based on the performance of different models, and returned the most appropriate categories of every sentence. Additionally, a user-friendly decision support system can also give users a way to find problematic sentences.

KEYWORDS

SLDA, SVM, Supervised learning, Labeling, Contract, text mining, classification

ACM Reference format:

Zian Wang, Sonal Gupta and Shuo Zheng. 2021. Police Union Contract Misconduct Complaint Detection. In *Proceedings of ACM Woodstock conference (WOODSTOCK 18). ACM, New York, NY, USA, 7 pages.*

DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Police Union Contract Misconduct Complaint Detection System aims to give users a way to search for a sentence or select the sentence they feel confused about. It analyzes the police contract from police departments, discovers problematic sentences and displays the category of the sentence, helping to know which part of the contract is misused by the police to take away citizens' rights. Hence, it decrypts the encrypted information behind the text.

We chose this project because it is socially-meaningful, solves the real-world problem; it is interesting to be aware of how inappropriate some contracts are. The idea of discovering encrypted and manipulated information is powerful. It is convenient, giving users a way to just select or search, thereby preventing the citizens' rights and giving them strength to fight against the unjust police.

2 RELATED WORKS

DeRay et al. [1] provide the police union contracts in Campaign Zero, and analyze the police bill of contracts. Their findings have demonstrated that police officers have access to an alternate justice system that allows them to act with impunity, one that is at odds with justice and simply unacceptable.[1] This review gives us an insight into what is misconduct behavior in the contracts, for detail, which category should the clauses of contracts be in, as well as the impact and whether they need negotiating in the contracts. Our experts have worked on the ground truth of the training model to detect the appropriate category and labels.

In addition, Campaign zero research[2] talks about the significance and necessity of text mining on the police union contracts. Since the police disorder and police misconduct behavior is getting prevalent and people want legal rights to limit the misuse of the police force, therefore, some already existing methods to reduce police force did not seem useful, and some police union contracts allowed the misuse of the police force, which is the origin of the misconduct behavior, Our project, police union contract misconduct behavior detection is useful and significant to the society. Also, the Campaign Zero project gives us solutions in the comprehensive package[3] by ending the minor "Broken Windows" offenses, and by profiling to prevent the potential police misconduct behavior.

Another paper written by Mohamed et al[4]. discussed that contract interpretation and analysis is significant to reduce time and save energy assisting users to figure out what they need help with.

3 DATASET

Our dataset includes Police contract data files from the POLICE CONTRACTS DATABASE, which is originally from the police union contracts in different states and different websites. There are about 88 contract files from 100 different cities in the United States..
We also use Human annotated data, provided by Dr. Lin, as the ground truth to train the data.

4 METHODOLOGY

Our project's goal is to build a system that can predict whether a police contract sentence is problematic or not, and if it is, the system can predict which kind of problem it has. We transfer this goal into a classification task. By training classification models, we can use the trained models to predict the classes of the input sentences. In our system, each class corresponds to a label of the sentences. There are seven different kinds of labels, six of them are different problems categories that appear in clauses of police contract, the last one is "non-problematic" for the sentences that do not have a problem.

In this project, we will use several supervised classification methods like SLDA, SVM, Boosting, Bagging, Random Forest, Neural Network, and Decision Tree. We choose to use the human-annotated data as the ground truth labels of the sentences. The whole process is divided into three steps: data preprocessing, building the models, and evaluating the models.

4.1 DATA PREPROCESSING

The raw data includes two parts, the first is police contracts. There are 88 police contracts of the top 100 largest cities in the U.S. from the "Police Union Contract Project". Another one is the human-annotated data. This file includes 601 problematic sentences which are labeled by experts.

The police contracts are PDF files. The first step we need to do is to transform the contracts to the form that the R language can process these text files. Firstly, we use adobe acrobat to convert all files to a doc file. Then, we use the online converter tool to convert it to a text file. Here we get the .txt files that R can easily read and store them. We are aware that we can use other methods to get the text files, so you may try the tools that work for you best. Because our goal is to label the sentence, not the whole contract, we will here read the contracts one line at a time to divide the contracts into sentences.

The next step is to filter out the sentences whose length is smaller than 500 but bigger than 120, so as to exclude some "useless" sentences, including titles, indexes, signatures, and so on. We keep the filtered out sentences as our `text` vector. The `text` vector of sentences are highly likely to be the legal provisions we need, and we will manually check if there are still some sentences that we do not need. The reason we choose to filter the sentences only by the length rather than other attributes is that the format of the police contracts of different cities is different. For example, in the police contracts of Albuquerque, the legal provisions start with Arabic numerals indexes, like "1.2.1", but in Indiana's police contract, some sentences start with an English letter or Roman numerals indexes. These differences make it hard to match them exactly by using regular expressions or some other ways. Therefore, we just filter them by length and do a manual check wherever required.

Then we will label the sentences as problematic or non-problematic. This is because we use supervised learning algorithms, so we need to have the true classes of training data. We have contract files, which contain all the police contract sentences of these 88 cities, but no labels included. We use a human-annotated data file, which only contains the problematic sentences with their category labels. Further, we compare the sentences in the human-annotated data file and the contract files (`text` vector), assigning the corresponding labels to the problematic sentences in the contract file.

We use a function "stringsim" of the R library named "stringdist" to find the corresponding labels for the problematic sentences in the contract files by comparing the sentence in the contract file and the sentence in the human-annotated data one by one. Based on the similarity score of the two sentences (0.85 or higher), it will assign the sentence in the contract file it's corresponding label. We do not use the exact match because when we do the transformation of the PDF files, there will inevitably be some conversion errors, for example, some upper-lower case confusion or some letters are transformed to numbers that look like, such as l and 1. Therefore, the exact match will ignore some sentences that contain some small transformation errors. These small errors won't influence our model because we will do some other steps later, for example, removing punctuation and changing all the letters to lowercase. We will take the sentences which have small transformation errors, which means we will set a threshold similarity (0.85 or higher), and all sentences that have a

similarity higher than that will be assigned the category labels.

The next step is removing the punctuations, numbers, and stopwords, and stemming. These steps are standard steps for text data preprocessing. Because punctuations, numbers, and stopwords have a very small influence on the modeling of police contract sentences, and removing them can make the model focus more on the important information, reduce the size of the training and testing data, and reduce the training time, we will get rid of them. We will further stem the `text` corpus. Stemming in text pre-processing, as we know, is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming also reduces the size of the data, and it can reduce the influence of the different forms of the same word.

The last step is to turn the sentences into a term-document matrix. We use the "create_matrix" function in the "RTextTools"[5] library to do that. This function takes a character string vector as input and outputs the term-document matrix that is built from the strings in the `text` vector.

There is a problem we encountered with the term-document matrix. When we train the models, even if we remove stopwords and do the stemming, the matrix is still too big for R to handle. There are 21905 sentences in the `text` vector, each one of them is a document in the term-document matrix, and the matrix we build will have 14395 terms. Therefore, the term-document matrix is quite huge. To handle this problem, we will do an extra step, which is removing sparse terms. The sparsity of a word means the proportion of this word's document frequency by the number of the terms in the term-document matrix. Formula 1 is the formula of sparsity, where N is the number of the terms in the matrix, and df_i is the document frequency of the term. We will set the threshold sparsity to 0.002, which means removing the terms whose sparsity is smaller than 0.002. After this step, the term-document matrix now has 1644 terms, which is much less than the number of terms in the previous term-document matrix.

$$\text{Sparsity}_i = \frac{df_i}{N} \qquad (1)$$

After going through the steps above, we now have the data ready for training.

4.2 MODEL TRAINING

In this part, we will train the models. The models we trained are- SLDA, SVM, Boosting, Bagging, Random Forest, Neural Network, and Decision Tree. After training the models, we will combine all these models together to predict the label of new sentences.

4.2.1 Oversampling

Firstly, we will use the term-document matrix to build a container object for RTextTool to handle. We will assign the index of the training and testing data here. However, we encountered another problem here- Our data was found to be highly imbalanced. Among 21905 police contract sentences, 20952 sentences are non-problematic, which means 95.6% of sentences in the data belong to one class. If we train the models using this data, the model will always predict the new sentences as non-problematic ones. The following table shows the result of the model with imbalanced data. To be sure of this problem, we carried out a little experiment on the test data. In the table, "NUM_MANUALLY_CODED" is the true number of category predictions and "NUM_CONSENSUS_CODED" shows the result of the system.

CLASS	NUM_MANUALLY_CODED	NUM_CONSENSUS_CODED
1	63	0
2	55	0
3	143	0
4	303	0
5	3817	4381
6	0	0
7	0	0

Table 1: Predictions on imbalanced dataset: Actual Vs Predicted

We can find that if we use the imbalance data here, the system will predict all test data to be non-problematic. Therefore, we need a solution to solve this problem. In this research study, we use oversampling techniques to make the training data balanced. Oversampling is a method that can increase the number of data points in small classes. Here we use oversampling to multiply the number of problematic sentences. We will simply copy and paste the problematic sentences 50 times in the training data, which means for each problematic sentence, there will be 49 other sentences in the training data. Here we need to notice that we have to first divide the data into training data and test data, then do oversampling. If we do oversampling first, because for each problematic sentence there are 49 other sentences in the data set, when we divide the data there will be the same sentences divided into both training and test data sets. Therefore, when we test the models, they are actually predicting the training data, which will make the performance incorrect.

Also, there is another method called undersampling that can solve the imbalance problem. On the contrary, undersampling will delete some data points in the classes that have too many points. The reason we do not use undersampling is that it will delete many non-problematic sentences, which is a deletion of the needed information. Instead, oversampling can be seen as an emphasis on problematic sentences in our study. The other reason is that it will make our data set too small. If we use oversampling, there will be only about 1100 police contract sentences left.

4.2.2 Coverage

Now we have the correct way to prepare the data ready for training and testing, the next step is to use "RTextTools", which is a library in R language, to train the models.

The training process is simple, we just need to use the term-document matrix we developed to build a container for the training process, then input the container into the training function.

Here is another problem, which is because after oversampling, we have 59390 police contract sentences in the training data set, so if we want to use K-fold cross validation, the device cannot handle the cross validation process. We tried several times, from 5-fold to 10-fold cross validation, the program crashed after running for about an hour when training most of the models. The error shown in R studio is "Error: cannot allocate a vector of size 720.2 Mb", so the reason for this crash should be memory overflow. Even though we can run 5-fold cross validation for some simple models, for example, the SLDA model, without crash, it will take about 2-3 hours to finish. We made a comparison between the SLDA models trained with 5-fold cross validation and without cross validation, and we find the improvement is tiny. Therefore, we traded that insignificant performance improvement by training models without cross-validation. We will see that our systems perform well without cross validation. Table 2 shows the performance of the SLDA models with and without 5-fold cross validation.

We trained 7 models in this step, and table 3 shows the performances of these 7 models. We can see if we pick any of these models, the performance is not good. Except for the overwhelming class, the highest precision is from the Random Forest model, whose precision is 0.75. The highest recall is 0.45 from the Boosting model. The highest F-Score is 0.47 from the SVM model. We can see even the highest performance still cannot satisfy our requirements. Therefore, we need to combine all these models, to improve the performance of the system.

Therefore, here we introduce Coverage. Coverage is the proportion of the test data that has at least N models' agreement. Here the N is set by ourselves and is mostly equal to the number of models which are appropriate for a given task. For example, if N equals 5, and we have 10 police contract sentences in the test data, let 2 of 10 sentences have a predicted label which is agreed by at least 5 models, so our coverage would be 20%. Here is the formula for coverage[6], in which k is the percent of sentences whose label is agreed by at least N models, and n represents the total number of sentences.

$$Coverage = \frac{k}{n} \qquad (2)$$

We will combine all our models by using the basic concept of coverage, which is only classifying sentences with labels that are agreed by at least N models. If there is an input sentence that cannot be agreed by at least N models, we will classify it as a sentence which we cannot process. It is obvious that if we increase the threshold value N, there will be fewer sentences that meet the requirement, but the performance will be better. Therefore, there is a tradeoff between the number of sentences that we can classify and the performance of our system. We can choose a reasonable value of N between 0 and 7 to make our system's performance relatively good and which can also predict most of the sentences. We will discuss this choice in the evaluation section.

	1	2	3	4	5	6	7
PRECISION	0.09	0.09	0.01	0.03	0.97	0.07	0.03
CV_PRECISION	0.11	0.12	0.03	0.05	0.97	0.07	0.05
RECALL	0.33	0.36	0.07	0.18	0.75	0.42	0.22
CV_ RECALL	0.32	0.38	0.10	0.18	0.75	0.43	0.23
FSCORE	0.14	0.14	0.02	0.05	0.85	0.12	0.05
CV_ FSCORE	0.15	0.16	0.05	0.06	0.85	0.13	0.07

Table 2: Performance of SLDA model trained with and without K-Fold Cross Validation

	1	2	3	4	5	6	7
SVM_PRECISION	0.6	0.67	0.15	0.21	0.97	0.44	0.21
SVM_RECALL	0.25	0.36	0.07	0.17	0.98	0.21	0.21
SVM_FSCORE	0.35	0.47	0.1	0.19	0.97	0.28	0.21
SLDA_PRECISION	0.09	0.09	0.01	0.03	0.97	0.07	0.03
SLDA_RECALL	0.33	0.36	0.07	0.18	0.75	0.42	0.22
SLDA_FSCORE	0.14	0.14	0.02	0.05	0.85	0.12	0.05
LOGITBOOST_PRECISION	0.01	0.02	0	0.03	0.97	0	0.03
LOGITBOOST_RECALL	0.25	0.45	0.07	0.28	0.61	0	0.09
LOGITBOOST_FSCORE	0.02	0.04	0	0.05	0.75	NA	0.04
BAGGING_PRECISION	0.17	0.5	0.12	0.11	0.97	0.42	0.09
BAGGING_RECALL	0.17	0.36	0.07	0.22	0.95	0.26	0.17
BAGGING_FSCORE	0.17	0.42	0.09	0.15	0.96	0.32	0.12
FORESTS_PRECISION	0.75	0.57	0.33	0.75	0.96	1	0.53
FORESTS_RECALL	0.25	0.36	0.11	0.15	1	0.26	0.16
FORESTS_FSCORE	0.38	0.44	0.16	0.25	0.98	0.41	0.25
TREE_PRECISION	0.01	NA	NA	NA	0.96	NA	0.04
TREE_RECALL	0.08	0	0	0	0.91	0	0.16
TREE_FSCORE	0.02	NA	NA	NA	0.93	NA	0.06
NNETWORK_PRECISION	NA	NA	NA	0.05	0.96	NA	NA
NNETWORK_RECALL	0	0	0	0.22	0.95	0	0

Table 3: Performances of the models

5 EVALUATION

In this section, we will evaluate the performance of all the models, and the aggregate performance after combining all the models.

5.1 MODEL PERFORMANCES

Firstly we will see the performances of all models. Figure 1-3 shows the performances of all models. We can see that if we use F-Score as our metric, the Random Forest model performs the best, followed by SVM and Bagging models. However, Boosting and Decision Tree models perform poorly. Also, we can see the recall and precision of the models. If we use recall as a metric, SLDA, Random Forest, SVM, and Bagging models perform similarly. Their recall values are all around 0.32. Then if we look at the precision of different models, the forest still performs the best, and its performance is much better than the second best performance. Boosting, Decision Tree and Neural Network perform poorly this time.

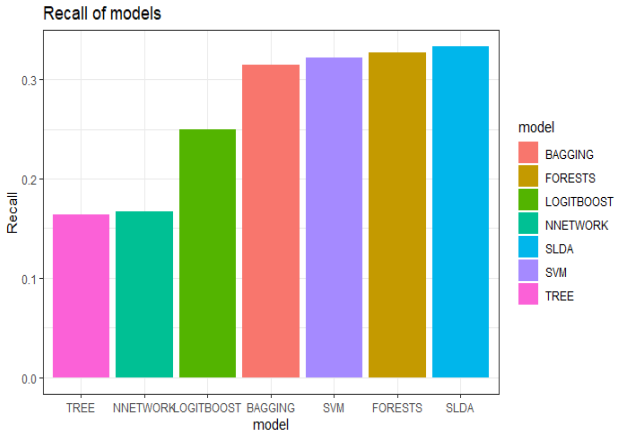

Figure 1: Recall of the models

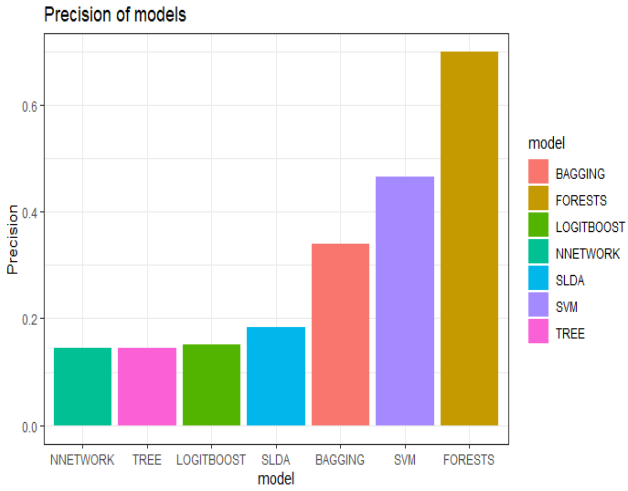

Figure 2: Precision of the models

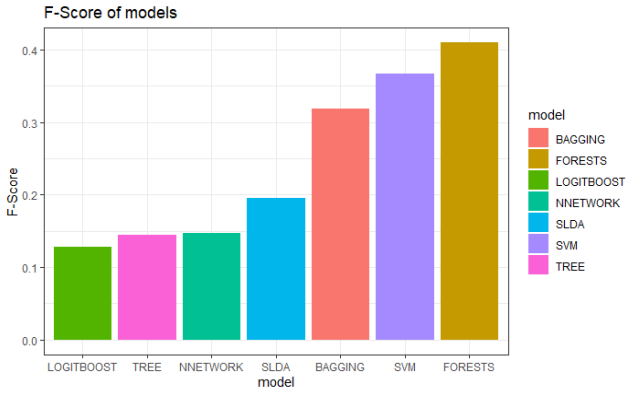

Figure 3: F-Score of the models

5.2 CRITERIONS

Now, we have to consider which criteria we want to use. Here is an important point which is that our original data set is imbalanced. As we discussed in 4.2.1, 95.6% of the test data are non-problematic sentences. Therefore, we will not use precision as our criterion. This is because if we use precision as the criterion, it is simple for a model to perform well, all it needs to do is to predict all sentences as non-problematic, and the precision of that model will reach 95.6%. Also because precision has a big influence on F-Score, we will also not use F-Score as the criterion.

We, therefore, introduce the concept of recall and see how it works for our problem statement. In the classification task which has more than 2 classes, we will use a different recall formula. We found this formula in the "RTextTools" library's source code. The way that RTextTools calculates the recall for the classification with more than 2 classes is to calculate the recall of each class. Figure 4 shows how "RTextTools" calculate the recall. The recall for class M is the proportion of the sentences of class M which are correctly predicted among all the sentences that are in class M. In figure 4, it is $n_{m,m}$ divided by the sum of the n^{th} row.

We can see that recall is the most proper criterion among these metrics. Firstly, if a model has a high recall, it means among all the sentences of class M, the model has to predict most of these sentences correctly, and for all the class M, it has to meet this requirement. This is very close to our goal. When we predict the sentences in a new police contract, we want most of the sentences of each class to be correctly labeled. We can accept some "false positives" because we can ask some law experts for future determination. However, let us go back to the precision, if we have a high precision model, its performance could be very poor. For example, this model can easily achieve high precision by only predicting the sentences that it is one hundred percent sure of, and if the sentence's class is not sure, it will simply classify it as non-problematic. Because we have over 95% of the non-problematic sentences, by doing this, the precision could be very high. For the problematic classes, its precision would be 100%, and for the non-problematic class, its precision will be at least 95%. This is why we use recall as our criterion and do not use precision and F-Score, which are influenced by precision.

Classified / Actual	1	2	3	4	5	6	7
1	n1,1	n1,2	n1,3	n1,4	n1,5	n1,6	n1,7
2	n2,1	n2,2	n2,3	n2,4	n2,5	n2,6	n2,7
3	n3,1	n3,2	n3,3	n3,4	n3,5	n3,6	n3,7
4	n4,1	n4,2	n4,3	n4,4	n4,5	n4,6	n4,7
5	n5,1	n5,2	n5,3	n5,4	n5,5	n5,6	n5,7
6	n6,1	n6,2	n6,3	n6,4	n6,5	n6,6	n6,7
7	n7,1	n7,2	n7,3	n7,4	n7,5	n7,6	n7,7

Figure 4: A table shows how to calculate the recall

Further, we discuss Coverage and Recall Accuracy and tradeoffs between them. An ensemble (consensus) agreement to enhance accuracy is recommended for our modeling task. Ensemble agreement refers to the fact that multiple different algorithms make the same prediction for the class of an event. Using a four-ensemble agreement approach, Collingwood and Wilkerson (2012) found that when four of their algorithms agree on the label of a textual document, the machine label matches the human label over 90% of the time. The rate is just 45% when only two algorithms agree on the text label.[6]

RTextTools has create_ensembleSummary() function to calculate coverage and the recall accuracy. As we previously mentioned, Coverage simply refers to the percentage of documents that meet the recall accuracy threshold. For instance, say we find that when seven algorithms agree on the label of a sentence, our overall accuracy is 98% (when checked against our true values). Then, let's say, we find that only 20% of our sentences meet that criterion. If we have 10 sentences and only two sentences meet the seven ensemble agreement threshold, then our coverage is 20%.

	n-ENSEMBLE COVERAGE	n-ENSEMBLE RECALL
n >= 1	1.00	0.95
n >= 2	1.00	0.95
n >= 3	1.00	0.95
n >= 4	0.99	0.96
n >= 5	0.93	0.97
n >= 6	0.76	0.98
n >= 7	0.44	0.98

Table 4: Ensemble Coverage and Recall

From our evaluation of Coverage and Recall (Table 4), we see that with n >= 1, coverage is 100% and recall is 95% and the same is for n >= 2 and n >= 3. From n >= 4 till n >= 7, coverage decreases gradually, and counterintuitive recall increases.

Generally, the trend followed is as the coverage decreases, the recall will increase meaning that you can classify fewer sentences more accurately. For example, just 44% of the sentences in our data have seven algorithms that agree. However, recall accuracy is 98% for these sentences when the 7 algorithms do agree. Figure 5 shows the tradeoff between coverage and recall. Considering that 95% is often the most suitable inter-coder reliability standard, and the coverage decreases much quicker than the recall increases, we agree upon 4 or 5 ensemble agreement with these data sentences because we label 99 -100% of the data sentences with an accuracy of 95%. [6] Therefore, here we trade off between the accuracy recall and the coverage. We try to choose those values for coverage and recall where we cover most of the data sentences and also maintain the accuracy recall at the same time. We try to get a model which has high coverage and high recall accuracy, thereby doing a negotiation between the two.

There is something about recall we would like to reconsider. Considering the high recall accuracy of 98%, there are some influences from the imbalance data. Because we have about 95% of the data in class 5, the recall of class 5 of all models will be high, and we can find this in table 3. The high recall of class 5 will make the overall recall also a higher value. However, we have to also notice that even though the class 5's recall is high, it cannot make the overall recall reach 0.98. Therefore, our system's good performance is not because the system only performs well on class 5, or another specific class, it is because our system has a good performance overall.

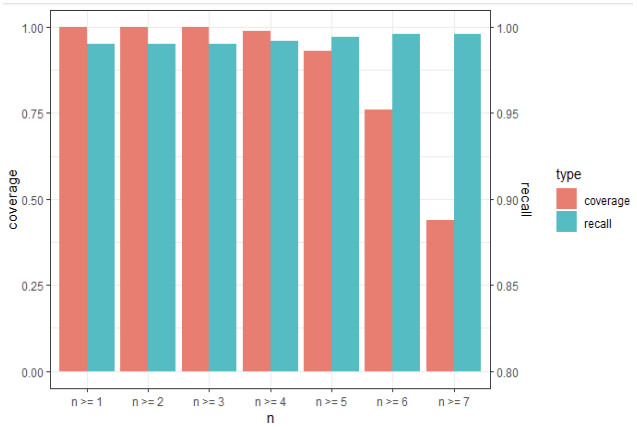

Figure 5: Coverage and Recall

6 CONCLUSION

Our Police Union Contract misconduct behavior system deals with real-world problems, providing users a way to find help based on the misconduct behavior they think they have encountered. The Data modeling parts can get a high recall score, meaning we can get good output and appropriate category labels from the model. Large amounts of data can be handled by using different techniques. We used data preprocessing, oversampling techniques and supervised classification algorithms to label the sentences in every pdf contract correctly.

7 FUTURE WORK

The code for our current research is available at the URL https://github.com/class-data-mining-master/2021-fall-dm-project-angrynerds. We have categorized the problematic and non-problematic sentences into different labels, but we believe that there is more we can do. We further plan to build an intelligent user-friendly system that will answer the citizen's queries' based on the contract related problems of a city.

8 ACKNOWLEDGMENTS

We extend our sincere gratitude towards Professor Dr. Yuru Lin who helped us with the project idea and the data. Her encouraging words have helped us to deal with this project, as we all did for the first time the text classification task. Her ongoing research in this field has enticed us to take up this as our research project.

9 MEMBER CONTRIBUTION

Zian Wang is responsible for data oversampling and assigning the test and training data, also for building the term-document matrix, and the Data Modeling and Evaluation, including making the graphs, of the research project. Sonal worked on the Data Preprocessing part - handled a large amount of the contracts and made them ready for modeling. She suggested a way to get the label for the sentences from the human-annotated data, helped Zian Wang understand the complexities of the data and the model and further in training and evaluating different models. Shuo helped us with the dashboard built in R Shiny [7] to show our results on a UI Interface using R Shiny gallery [8], which is the creative and interactive part we want to achieve. We are still yet to configure it with our prediction results. Moreover, She helped us out with the logistics of the project. Lastly, we met frequently and helped each other out to finish this project.

REFERENCES

[1] DeRay McKesson, Samuel Sinyangwe, Johnetta Elzie and Brittany Packnett. 2016. Police Union Contract Review. https://www.checkthepolice.org/review

[2] Campaign zero research. https://campaignzero.org/research

[3] Solutions in Campaign Zero. https://campaignzero.org/solutions.html#solutionsoverview

[4] Mohamed M. Marzouk, Mohamed Enaba. 2018. *Automation in Construction*, Volume 98:265-274. DOI:https://doi.org/10.1016/j.autcon.2018.11.018

[5] Timothy P. Jurka, Loren Collingwood, Amber E. Boydstun, Emiliano Grossman, Wouter van Atteveldt. 2012. RTextTools: Automatic Text Classification via Supervised Learning. *R package version 1.3.9.* http://CRAN.R-project.org/package=RTextTools

[6] Timothy P. Jurka, Loren Collingwood, Amber E. Boydstun, Emiliano Grossman, and Wouter van Atteveldt. 2013. RTextTools: A Supervised Learning Package for Text Classification. *CONTRIBUTED RESEARCH ARTICLES*, *The R Journal Vol. 5/1, ISSN 2073-4859.* *https://journal.r-project.org/archive/2013/RJ-2013-001/RJ-2013-001.pdf*

[7] Winston Chang et al. 2021. shiny: Web Application Framework for R. *R package version 1.7.1.* *https://cran.r-project.org/web/packages/shiny/index.html*

[8] R shiny gallery, https://shiny.rstudio.com/gallery/career-pathfinder.html