Checkpoint 5: Natural Language Processing Findings

The Enchanted Badgers
Alexander Einarsson, Sergio Servantez, Marko Sterbentz
November 25, 2020

The theme of our project is to identify meaningful categories for uncategorized complaints by analyzing the relationship that exists between the complaint category and the complaint report narrative, and to use this new information to explore various aspects of complaint investigations and outcomes. To this end, we sought to answer the following question by applying natural language processing techniques to the CPDP dataset, and generating visualizations with the classification results:

1. Can we train a transformer language model (e.g. BERT) on a dataset of complaints with assigned categories (the input being the report narrative, and the label being the category) and use this trained model to predict the categories of complaints which are currently not categorized, or categorized as “unknown”?

The analysis of the results of using this model are discussed below. Additionally, we have provided a discussion on our use of transformer models to classify uncategorized complaints and our process for cleaning and integrating additional complaint narratives that were not already in the CPDP database.

We wanted to answer this question in order to determine which complaint types were most frequently miscategorized by the Chicago Police Department, which would allow us to determine if there were any types of complaints that were consistently being miscategorized by the reporting officer, either intentionally or unintentionally. The rest of this document is divided into three parts: a section detailing how we cleaned and integrated additional narrative summaries, a section detailing how we built and trained language models to classify uncategorized complaints, and a section analyzing the classification results.

Cleaning and Integrating Additional Narratives

To improve our chances of training an accurate language model classifier, we first sought to increase the amount of training data. Within the CPDP database, there are roughly 1100 complaints that have narrative summaries associated with them. Unfortunately, this was not enough raw data to train these kinds of models. When fine-tuning a BERT model on this set of complaint narratives, we were able to achieve a classification accuracy of 66.8% on the test set. This number, while better than one might expect for a multi-class classification problem such as this with so few data points, was still far lower than we would like.

Fortunately, in addition to the narrative summaries that are associated with allegations in the CPDP database, there was a set of roughly 45,400 narratives external to the database that

were available. However, before being ready for usage, these summaries needed to be cleaned and integrated with the summaries and complaints already in the database.

In order to do this, we first needed to gather the data together. This included downloading the new set of narratives from The Invisible Institute's GitHub document analysis repository. Additionally, a dump of the data_allegation table was downloaded in order to make it easier to clean and integrate these datasets using Python and Pandas.

First, this script ("narrative_cleaning_integration.py") processes the "narratives.csv" file. We first take all rows that have a column_name of either "Initial / Intake Allegation" or "Allegation". Then, in the result set of 31,572 rows, there are a variety of complaints that have summary text that consists entirely of either "(None entered)" or "NO AFFIDAVIT". These are replaced with an empty string in order to match the format of the complaints already in the data_allegation table. The complaints texts also have a bunch of seemingly random carriages returns sprinkled throughout the summaries which are likely the result of the optical character recognition software. These carriages returns are replaced with spaces. We also strip white space from the front and end of the text, and then remove any rows where the summary text is an empty string. Lastly, there are duplicate summaries contained in the set, some of which are different and have varying lengths. Upon further examination of the data, it appears duplicate summaries for a single complaint would have increasing amounts of detail the longer the summary was. So, as a heuristic, we remove duplicates and keep the row that had the longest summary for that complaint. This will maximize the amount of information that the language model will have to train and perform the classification with.

Next, we had to integrate this with the data and narrative summaries from the data_allegation table. This first entailed loading the data into a Pandas data frame and dropping the unnecessary columns. This resulted in a data frame with columns of "crid" and "summary". The columns of the new narratives were renamed to match these two columns, and the data frames were concatenated. The same duplicate removal procedure we used when cleaning the new narratives was adopted here as well, and we also make sure to remove any rows that do not have a summary. The end result is a set of narratives associated with the proper complaint report ID. There were 16010 rows of data here.

The resulting cleaned and integrated set of complaints and their narratives was written out to a CSV file called "all_narratives.csv". In order to generate a dataset with the complaints, their summaries and other associated data, we need to create a new table in the database in order to join these summaries with the other data. To this end, we ran the following SQL table creation query:

```
CREATE TABLE data_narratives (
    id SERIAL,
```

```
    cr_id varchar(30),
    summary text,
    PRIMARY KEY (id)
);
```

Then, we loaded in the integrated narrative dataset into this table with the following SQL query:

```
COPY data_narratives(cr_id, summary)
FROM '/full/path/to/all_narratives.csv'
DELIMITER ','
CSV HEADER;
```

The '/full/path/to/all_narratives.csv' obviously needs to be replaced with the actual full path to the “all_narratives.csv” file. After this, there was a table called data_narratives that contained all narrative summaries and the id of the complaint record with which they are associated. We used the following SQL query to join this data with a couple of other tables in order to produce a full set of narrative summaries, the complaint they are associated with, the final outcome of this complaint, and the category of this complaint.

```
SELECT oa.allegation_id as allegation_id,
       oa.allegation_category_id as category_id,
       ac.category as category,
       oa.final_outcome as final_outcome,
       n.summary as summary
FROM data_officerallegation oa
    INNER JOIN data_allegationcategory ac on oa.allegation_category_id
= ac.id
    INNER JOIN data_narratives n on oa.allegation_id = n.cr_id
WHERE n.summary > '';
```

The results of this query were then downloaded into a JSON file that is used as input into the final data processor that prepares the data for use in training the language models.

Using Language Models to Classify Uncategorized Complaints

The data processing script (“process_raw_data.py”) reads this JSON file into a Pandas data frame. We first count the number of complaints associated with each category. The class and data distribution is as follows:

Category Label	Number of Complaints

Bribery / Official Corruption	41
Conduct Unbecoming (Off-Duty)	292
Criminal Misconduct	42
Domestic	283
Drug / Alcohol Abuse	52
Excessive Force	11
False Arrest	1373
First Amendment	5
Illegal Search	2330
Lockup Procedures	966
Operation/Personnel Violations	4166
Racial Profiling	3
Supervisory Responsibilities	89
Traffic	280
Use Of Force	1575
Verbal Abuse	144

Any category that has less than 5 training samples associated with it lumped into a category called “other_category”. We do this since any category with less than 5 training samples likely does not have enough training data to properly classify narratives as coming from this category.

We then make use of Scikit-Learn’s train_test_split() function to divide the resulting set of samples into training, validation, and testing data sets with a train/validation/test split of 60/20/20. We then create three TSV files containing the set of samples and their class/category for each of the training, validation, and testing data sets.

With the final training and testing datasets put together, we can finally train the language model classifiers. For this task, we trained two models: a bag of embeddings models and a BERT transformer model. The bag of embeddings model is similar to a bag of words model except we use word embeddings rather than raw words or n-grams.

The training process makes use of the AllenNLP framework, which provides a wrapper around a variety of transformer implementations, including the BERT model. The BERT model has already been pretrained on a massive corpus of text, and all we need to do now is add a classification output layer and fine-tune it on the set of complaint narratives and their classes. The code for this is included in the src directory of our submission, as are the instructions for training and testing the model.

The result is a bag of embeddings classifier which achieves a classification accuracy of 83.2% on the test datasets, and a trained BERT classifier that achieves an accuracy of 83.9%. This is far better than the 63.6% and 66.8% classification accuracies achieved by the bag of embeddings and BERT classifiers respectively when trained on the original dataset derived from the roughly 1100 complaints narratives contained in the CPDP database.

With these two classifiers in hand, it is now possible to use these models to classify the complaints in the database that have no known category. This would be the complaints where either the category is “Unknown” or is NULL and where the complaint has a summary. These complaints were retrieved from the database using the following SQL query:

```
SELECT DISTINCT oa.allegation_id as allegation_id,
       oa.allegation_category_id as category_id,
       ac.category as category,
       oa.final_outcome as final_outcome,
       c.summary as summary
FROM data_officerallegation oa
    LEFT JOIN data_allegationcategory ac on oa.allegation_category_id =
ac.id
    LEFT JOIN data_narratives c on oa.allegation_id = c.cr_id
WHERE (ac.category = 'Unknown' OR ac.category IS NULL)
  AND c.summary > '';
```

The result was 101 distinct complaints and their narratives. Using the “classify_uncategorized_complaints.py” script, we used the two models to classify these 101 complaints. The results are described in the tables below.

BERT Classification Results

Category Label	Number of Complaints
Operation/Personnel Violations	28
Illegal Search	7

Use Of Force	45
False Arrest	1
Lockup Procedures	4
Conduct Unbecoming (Off-Duty)	4
Domestic	10
Traffic	1
Verbal Abuse	1
Supervisory Responsibilities	0
Drug / Alcohol Abuse	0
Criminal Misconduct	0
Bribery / Official Corruption	0
Excessive Force	0
First Amendment	0
other_category	0

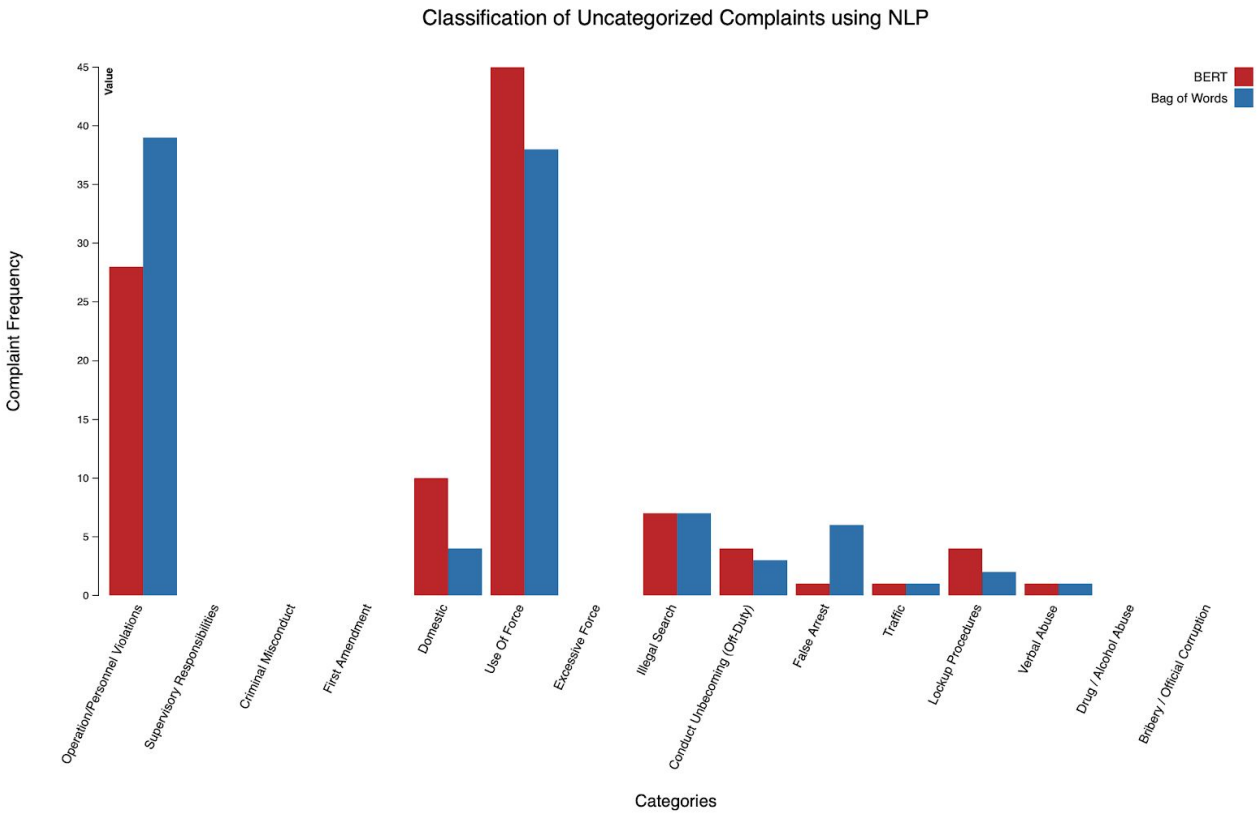
Bag of Embeddings Classification Results

Category Label	**Number of Complaints**
Operation/Personnel Violations	39
Illegal Search	7
Use Of Force	38
False Arrest	6
Lockup Procedures	2
Conduct Unbecoming (Off-Duty)	3
Domestic	4
Traffic	1
Verbal Abuse	1

Supervisory Responsibilities	0
Drug / Alcohol Abuse	0
Criminal Misconduct	0
Bribery / Official Corruption	0
Excessive Force	0
First Amendment	0
other_category	0

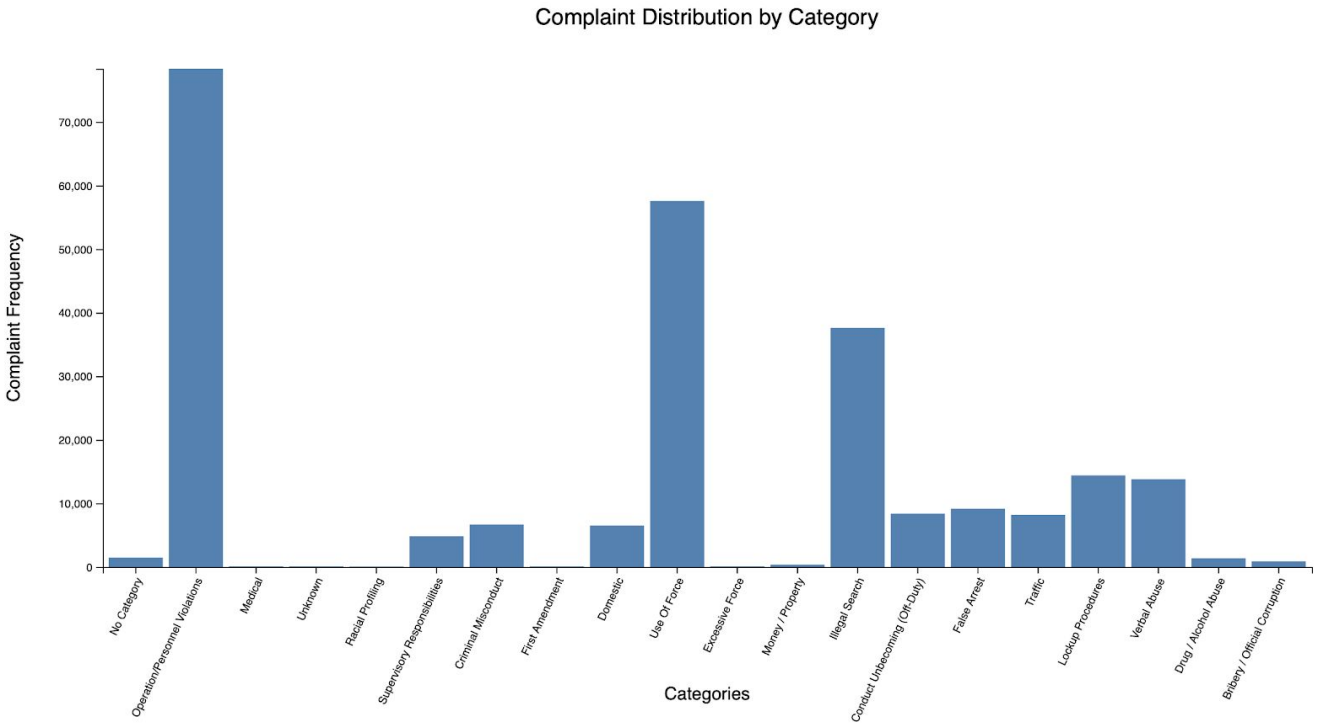
Analysis of Results

Previously, as part of Checkpoint 2, we had produced a grouped bar chart showing the classifications of uncategorized complaints in the database. Since it is easier to examine the classification results as a chart rather than a table of values, we include it here as well:



Looking at the classification results, there are a number of interesting observations that jump out. The first is the high number of uncategorized complaints that were classified as belonging to the “Operations/Personnel Violations” category. This category seems to be a sort of catch all category, and it stands to reason that a high number of uncategorized complaints would fit into this category. This could simply be the result of the person in charge of categorizing the complaint being unsure what to report as the category, but it could also be that these typically intra-department complaints are being intentionally obfuscated in order to obscure the problems occurring within the department. In order to test this, more qualitative analysis of these complaints will be required. It is also worrisome that so many uncategorized complaints were classified as “Use of Force” by both models. While this could just be a coincidence, the disproportionate number of “Use of Force” classifications suggests that these complaints might have been intentionally miscategorized.

The following visualization, also taken from Checkpoint 2, shows the distribution of all complaints according to their categories.



We can see that overall, the distribution of categories assigned to the uncategorized complaints follows roughly the same distribution as the original dataset. However, there is clearly a higher proportion of the uncategorized complaints belonging to the “Use of Force” category, which seems a little odd. Further qualitative analysis of the complaints that were classified into this category would be needed to determine whether these were intentionally miscategorized, if it was an honest mistake (i.e. multiple categories make sense), or simply a mistake by the

language model classifiers. However, if it is indeed the case that these complaints were obviously miscategorized by the Chicago Police Department, this would provide compelling supporting evidence for a need to improve the reporting system citizens use to hold officers accountable. A definitive answer to this requires more work.

Additionally, the discrepancies in the classifications of the two models warrants further investigation. It could be the case that a single complaint viably fits into two categories, but the different model architectures are picking up on different signals in the text. This warrants further investigation as part of some future work as well.